

Angerhausen/Brückmann
Englisch/Gerits

64 intern

***Het grote boek
voor de Commodore 64
met uitvoerige ROM-listing.***

Commodore Bibliotheek 14

**DATA BECKER
NEDERLANDS***

Angerhausen/Brückmann

Englisch/Gerits

64 intern

***Het grote boek
voor de Commodore 64
met uitvoerige ROM-listing.***

Commodore Bibliotheek 14

**DATA BECKER
NEDERLANDS ***

Oorspronkelijke titel: 64 Intern.
Copyright (c) 1984 DATA BECKER GmbH, Düsseldorf.
Copyright (c) 1985 A.W.Bruna & Zoon, Utrecht
Vertaling: Jaap Bloem
Redactie: ProCread, Groningen
Productie: ProCread, Groningen
Druk: Vonk, Zeist
ISBN 90 229 3339 3
D/1985/0939/162

Dé inhoud én de werking van de programma's zijn gecontroleerd door de

Stichting Computerhobbyisten Nederland

Postbus 4666

1009 AR Amsterdam

Voor vragen en opmerkingen betreffende de inhoud en programma's van deze uitgave kunt u zich schriftelijk wenden tot bovenstaand adres.

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 dan ook zonder voorafgaande 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.

Antwerpesteenweg 29a, 2630 Aartselaar.

1 DE COMMODORE 64 ONDER DE LOEP

1.1	Wat u over de machine moet weten	7
1.2	Organisatie van de hardware	9
1.3	De processor 6510 en zijn bijzonderheden	10
1.4	Organisatie van het geheugen	11
1.5	De uitbreidingspoort	23
1.6	De gebruikerspoort	24
1.6.1	De programmering van de gebruikerspoort in BASIC	25

2 DE SYNTHESIZER EN ZIJN PROGRAMMERING

2.1	De sound controller 6581	28
2.1.1	Algemene opmerkingen over de SID 6581	28
2.1.2	Registerbeschrijvingen van de SID	31
2.1.3	De analoog/digitaal omzetter	34
2.1.3.1	Het gebruik van de analoog/digitaal omzetter	34
2.1.3.2	Het gebruik van paddles	35
2.2	De programmering van de SID 6581	36
2.3	SYNTHIMAT	39

3 DE GRAFIEK

3.1	De videocontroller 6569 VIC	41
3.1.1	Algemene opmerkingen	42
3.1.2	Registerbeschrijving van de VIC	42
3.1.3	De modes van de VIC	45
3.2	De aansluiting op de processor	47
3.3	De aansluiting op het RAM	48
3.4	De aansluiting op de karaktergenerator	50
3.5	De aansluiting op het kleur-RAM	50
3.6	Programmering van kleur en grafiek	52
3.7	Sprites - het toverwoord van de Commodore 64	68
3.7.1	Programmering van sprites	70

4 STURING VAN IN- EN UITVOER -CIA 6526

4.1	Algemene opmerkingen over de 6526	79
4.2	Registerbeschrijving van de CIA	81
4.3	I/O-ports	84
4.4	Timer	85
4.5	Real-time clock	86
4.5.1	De juiste tijd met een trucje	86
4.6	De CIA's van de Commodore 64	88
4.7	Het gebruik van joysticks	89
4.8	De IEC-bus	
4.8.1	Wat geschiedenis	89
4.8.2	De grote IEC-bus	90
4.8.2.1	Definities	91
4.8.2.1.1	Gebruik van de connector en namen van signalen	91
4.8.2.1.2	Adressen van de apparatuur	92
4.8.2.1.3	Secundaire adressen	93
4.8.2.1.4	De systeemvariabele ST	94
4.8.2.2	De adressering	95
4.8.2.3	Gegevens overdragen	98
4.8.2.4	Adressering beëindigen	100
4.8.3	De seriële IEC-bus van de Commodore 64	102
4.8.3.1	Basiskennis	102
4.8.3.2	Adressering	103
4.8.3.3	Overdracht van gegevens	103

5 BASIC VANUIT EEN ANDER PERSPECTIEF

5.1	De verwerking van de BASIC-interpreter	108
5.2	Van invoer tot verwerking	109
5.3	Doe meer met BASIC	110
5.3.1	Het uitbreiden van BASIC	110
5.3.2	HARDCOPY - RENEW - PRINT USING	111
5.3.3	Mathematische routines naar eigen ontwerp	117
5.3.4	SQR- ,SUM- en PROD-functie	118
5.3.5	Verandering van formaten van gegevens	122

6 PROGRAMMEREN IN MACHINETAAL OP DE COMMODORE 64

6.1	De monitor	125
6.2	Nuttige adressen van het bedrijfssysteem van de Commodore 64	129
6.3	In- en uitvoer in machinetaalprogramma's	131
6.3.1	In- en uitvoer van afzonderlijke bytes	132
6.3.2	In- en uitvoer via de randapparatuur	133
6.3.3	De techniek van het opslaan van gegevens - LOAD en SAVE	135

6.3.4	De programmering van de RS-232 interface	140
6.3.5	De seriële IEC-bus van de Commodore 64	146

7 COMMODORE 64 - VIC 20 - CBM

7.1	De zeropage en andere belangrijke geheugenlocaties	150
7.2	De adressen van BASIC-routines	156
7.3	Tabel ter vergelijking van de VIC 20 en de Commodore 64	158
7.4	Tabel ter vergelijking van de CBM 8000 en de Commodore	161
7.5	Aanpassing van programma's voor de VIC 20	164
7.6	Aanpassing van programma's voor de grote CBM-computers	165

8 ROM-LISTINGS VAN DE COMMODORE 64

8.1	Gebruik van de ROM-listing	167
8.2	Alfabetische lijst van ROM-routines	168
8.3	Commodore 64 ROM-listing	184

DOCUMENTATIE BIJ HET SCHEMA VAN DE COMMODORE 64

9.1	Opmerking vooraf	356
9.2	De stroomvoorziening	357
9.3	De produktie van de klokfrequentie	358
9.4	De processor	363
9.5	Decodering van adressen	365
9.6	De videocontroller 6569	366
9.6.1	De aansluiting tussen RAM en VIC	368
9.6.2	Het kleur-RAM: 16 kleuren met 4 bits	370
9.7	De karaktergenerator	371
9.7.1	De processor en het RAM	371
9.8	De SID 6581	375
9.9	De CIA's 6526	377
9.9.1	De modulator	382
9.10	Een defecte Commodore	382
9.10.1	Geen beeld	383
9.10.2	Defecten van andere aard	384
9.11	Het binnenwerk	387
9.12	Lijst van halfgeleiders	391
9.13	De draagbare SX-64	382

VOORWOORD

Vanaf de eerste dag was de Commodore 64 een absoluut kassucces. Zijn overweldigende capaciteiten zorgden voor geheel nieuwe maatstaven in de verhouding tussen prijs en prestatie. Er was slechts een enkel probleem: de uiterst gebrekkige fabrieksinformatie over programmering en gebruik. Over belangrijke dingen als 'sounds' en 'graphics' werd bijna niets vermeld. Bovendien was het handboek in het Engels geschreven. Daarom besloten we dit boek, "64 INTERN", op te nemen in de reeks van DATA BECKER/NEDERLANDS*.

"64 INTERN" had evenveel succes als de Commodore 64 en geldt inmiddels als standaardwerk. Natuurlijk hebben we sinds de eerste druk iets meer over de Commodore 64 geleerd. Deze bewerkte vertaling is daarom aanzienlijk uitgebreider dan het origineel uit 1983. Dat een dergelijk boek door verschillende auteurs werd geschreven, heeft zijn nut bewezen. Iedereen kon zich op die manier aan zijn specialisme wijden. Michael Angerhausen, lid van het software team van DATA BECKER, hield zich vooral bezig met sprites en graphics. Rolf Brückmann, een van onze meest ervaren technici, ontleedde het binnenwerk van de Commodore 64. Commodore was zo vriendelijk hiervoor de schema's ter beschikking te stellen. Lothar Englisch, een specialist op het gebied van systeemprogrammering, bewerkte voor de vierde druk de ROM-listing. Die is voor machinetaalprogrammeurs van groot belang. Daarnaast hield hij zich vooral bezig met de uitbreidingen van BASIC. Klaus Gerits, leider van de researchafdeling van DATA BECKER, hield zich diepgaand bezig met de hardware van de Commodore 64. Nieuw is zijn bijdrage over de seriële IEC-bus. Dankzij "64 INTERN" hebben talloze gebruikers een beter begrip van hun Commodore 64 gekregen. Bovendien vergemakkelijkt dit boek het gebruik. Hopelijk draagt ook de vierde druk van deze bestseller ertoe bij, dat zoveel mogelijk lezers hun Commodore 64 optimaal gaan benutten.

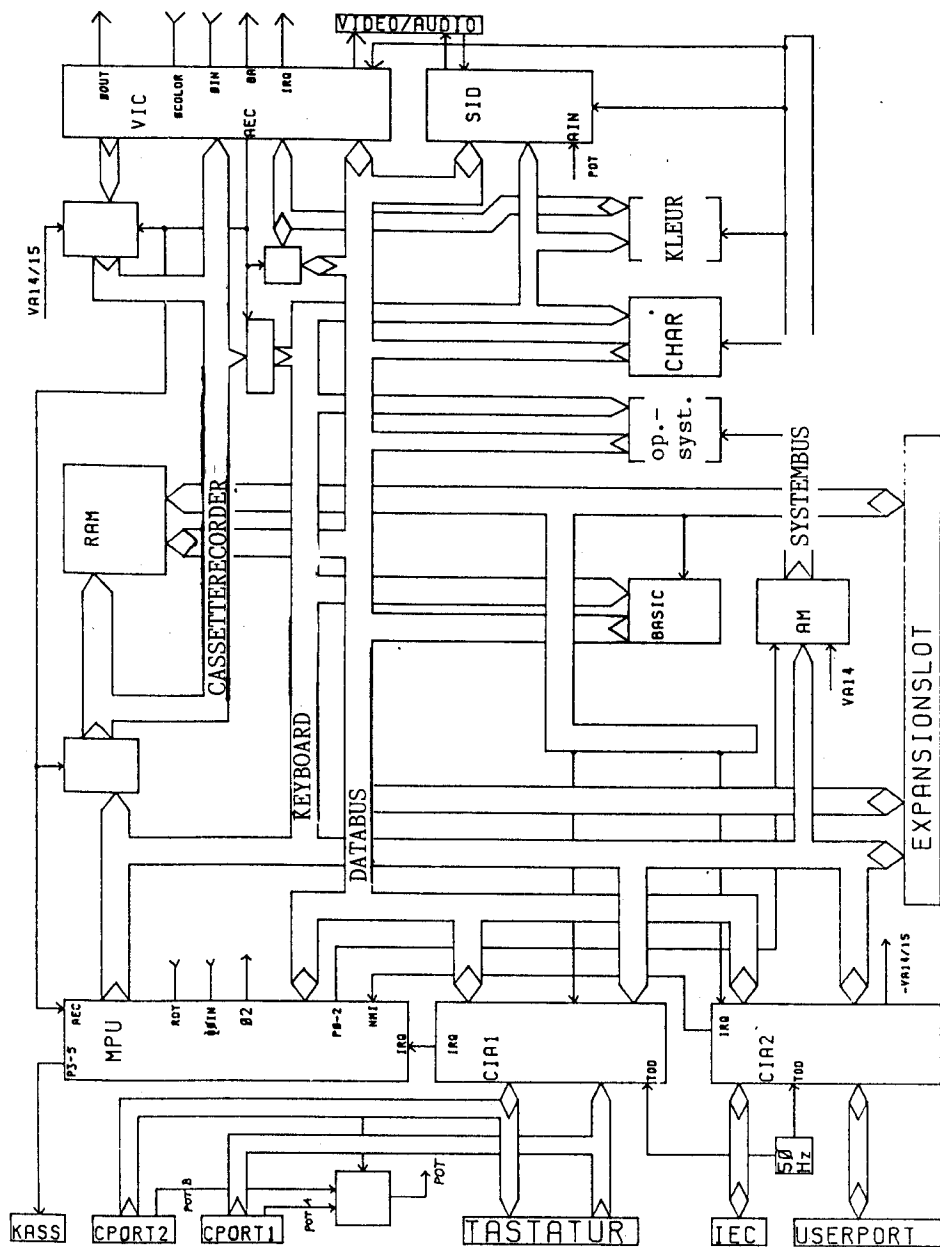
1 DE COMMODORE 64 ONDER DE LOEP

1.1 Wat u over de machine moet weten

Het is de makers van de Commodore 64 gelukt een uitstekend produkt op de markt te brengen, dat relatief eenvoudig is. Als u ook een VIC-20 bezit, moet u het binnenwerk van beide apparaten eens met elkaar vergelijken. Ondanks zijn grotere mogelijkheden heeft de Commodore 64 minder IC's. Dat komt door de verwerking van een aantal nieuwe, complexere IC's (Integrated Circuits). Commodore maakt deze gedeeltelijk zelf. Het is ongelooflijk dat het binnenwerk van de Commodore 64 een complete single board ++computer is, met alle elementen die voor communicatie met de buitenwereld nodig zijn (vergelijk de foto's op bladzijde 341-344). De makers hebben kans gezien om in een adresbereik van 64K (dat is het normale maximum voor een 8-bits processor) de volgende eenheden onder te brengen.

- * 64K dynamische RAM
- * 1K kleur-RAM
- * 4K karaktergenerator
- * kleurenvideocontroller met HIRES-graphics
- * synthesizer met drie onafhankelijke stemmen
- * 8K BASIC
- * 8K bedrijfssysteem
- * 2 parallel-I/O

Het blokschema van de Commodore 64 staat op de volgende bladzijde. In hoofdstuk 3 en paragraaf 1.4 wordt op enkele bijzonderheden ingegaan. U zult zich misschien afvragen hoe alles kon worden ondergebracht, zodat het ook nog functioneert. Alleen het RAM neemt immers al het gehele adresbereik in beslag. De details, die in de volgende paragrafen aan de orde komen, zijn noodzakelijkerwijs nogal technisch. Het verdient daarom aanbeveling een inleiding in de microcomputertechniek te lezen. Vooral hardware- en programmeerhandboeken, die speciaal op IC's uit de 65xx familie betrekking hebben komen hiervoor in aanmerking.



1.2 Organisatie van de hardware

Hoewel de processor 6510 slechts een adresbereik van 64K heeft, konden door multiplexen alle genoemde voorzieningen worden ondergebracht. Dat betekent dat dezelfde lijnen op verschillende tijdstippen voor verschillende componenten worden gebruikt. Niet alle hardwarecomponenten worden immers op hetzelfde tijdstip aangesproken. Het is de kunst om ze zo aan te sluiten, dat alle eenheden die tot een logisch blok behoren (bijvoorbeeld I/O-blok, videoblok), zonder omschakelen te bereiken zijn. Het hart van deze complexe functie is een speciaal IC, de AdresManager (AM), dat de adressen beheert. Meer hierover in paragraaf 1.4. Een goed voorbeeld voor de overlapping van componenten die hetzelfde adresbereik innemen, zijn de karaktergenerator en het I/O-blok. Hoewel ze dezelfde adressen gebruiken, rijden ze elkaar niet in de wielen. Als de VIC de karakterset aanspreekt, maakt hij handig van die ogenblikken gebruik, waarop de processor de adresbus toch niet benut.

Bij zoveel overlappingen kunnen natuurlijk problemen optreden, maar door de juiste software te gebruiken zijn die goed te vermijden. Daarom is dit hoofdstuk vooral voor machinetaalprogrammeurs van belang. Het gehele ROM en delen van het RAM vertonen overlappingen in een gebied van 20K. In paragraaf 1.4 is te lezen hoe dit wordt opgelost. Een andere overlapping betreft niet de adresbus, maar het perifere gebied. Daar zijn enkele I/O-lijnen van de CIA's dubbel bezet, namelijk zowel door het toetsenbord, als door de controlports aan de buitenkant van de computer. In paragraaf 4.7 wordt uitvoerig behandeld hoe dit probleem kan worden omzeild. Het ligt voor de hand, dat in zo'n complex systeem de afzonderlijke componenten taken moeten vervullen, waarvoor ze niet zijn gemaakt. Een goed voorbeeld is de videocontroller 6569 (VIC). Het RAM bestaat uit 8 dynamische RAM-IC's 4164 van 64K*1 bit, georganiseerd tot een bereik van 64Kbyte. Toch zitten er in de hardware geen componenten die de tamelijk gecompliceerde - vergeleken met statische RAM's althans - adressering voor hun rekening nemen of op tijd zorgen voor een refresh. Een refresh is het opfrissen van de geheugeninhoud door pulsen met kleine tijdsintervallen. Al deze aangelegenheden worden automatisch door de videocontroller afgehandeld. Daarnaast vervult deze zijn eigenlijke taak en zorgt voor een goed beeld.

1.3 De processor 6510 en zijn bijzonderheden

De processor van de Commodore 64 is de 8-bits MPU 6510. Dit nieuwe IC uit de familie 65xx werkt in de Commodore 64 op een klokfrequentie van 980KHz. Het wezenlijke verschil met de bekende 6502 is, dat de 6510 over zes I/O-lijnen beschikt. De 6502 heeft er daarentegen geen enkele. Het voordeel van de 6510 is, dat in minimumconfiguraties zonder extra perifere IC's afzonderlijk programmeerbare verbindingen ter beschikking staan. Deze kunnen zorgen voor de verbinding met de buitenwereld of voor interne controlefuncties. In de Commodore 64 ondersteunen deze lijnen aan de ene kant het cassettegebruik en aan de andere kant de geheugenorganisatie. De instructie is dezelfde als bij de 6502. Voor programmeurs is misschien interessant, dat de I/O-lijnen de adressen 0 en 1 gebruiken, namelijk 0 als datarichtingsregister en 1 als dataregister. De 'stack' kan zich in het gebied \$0100 tot \$01FF bewegen. Onder 'stack' verstaat men een stapelgeheugen, waarin bijvoorbeeld bij het springen naar subroutines de processorregisters kunnen worden weggezet. Bij terugkeer naar het hoofdprogramma kunnen ze weer tevoorschijn worden gehaald. Er kan vervolgens met de oorspronkelijke gegevens verder worden gewerkt.

Bij de nu volgende pin-configuratie nog een opmerking. Het is ons niet gelukt een datasheet te pakken te krijgen, waarop de werkelijke stand van zaken staat aangegeven. We hebben er weliswaar verschillende, maar iedere keer is de pin-configuratie anders weergegeven. Daarom hebben we de empirische weg bewandeld en gecontroleerd waar de lijnen naar toe gaan.

- | | |
|-------|---|
| 1 OIN | Systeemklokingang; in de Commodore 64 ongeveer 980KHz. |
| 2 RDY | Ready; 0=processor stopt bij de volgende leescyclus totdat RDY=1. Van deze mogelijkheid wordt bijvoorbeeld bij langzame geheugens gebruik gemaakt. |
| 3 IRQ | Interrupt Request; 0=processor haalt het volgende instructie-adres uit \$FFFE en gaat daar verder. Deze toestand treedt alleen op, indien de interrupt was toegestaan (bit 2 in het flag-register=0). |
| 4 NMI | Non-Maskable Interrupt; 0=processor haalt het volgende instructie-adres uit \$FFFA en gaat daar verder. |

5 AEC	Adress Enable Control; 0=processor brengt de data-, adres- en systeembus in de hoogohmige toestand (tri-state). De bus kan nu door andere eenheden worden gebruikt, bijvoorbeeld door een tweede processor.
6 VCC	voedingsspanning
7-20 A0-A13	adresbus
21 GND	signaalaarde
22-23 A14-A15	adresbus
24-29 P5-P0	I/O-pinnen
30-37 D7-D0	databus
38 R/W	0=schrijfgebruik, 1=leesgebruik. Het gebruik vindt alleen plaats, wanneer 02=1.
39 02OUT	Systeemklokuitgang voor de voeding van andere bouwstenen.
40 RES	Reset; 0= processor gaat over naar rusttoestand. Bij de overgang van 0 naar 1 haalt de processor een adres uit \$FFC en begint daar met het programma.

1.4 Organisatie van het geheugen

De meervoudige overlappingsen en de daarmee samenhangende noodzakelijke organisatie van het adresbereik leiden tot een aantal keuzes. Het hart van het hele complex is zoals gezegd de AdresManager (AM) die het adresbeheer voor zijn rekening neemt. Dit is een FPLA-IC (Field Programmable Logic Array). Zulke bouwstenen bevatten veel logische schakelingen die door fabrikanten van het eindproduct in de gewenste combinatie kunnen worden geprogrammeerd. Bij de Commodore 64 gaat het om een IC met 16 ingangen en 8 uitgangen. Door de programmering kreeg iedere ingangscombinatie (van de 65536 mogelijkheden) een uitgangscombinatie (van de 256 mogelijkheden) toegewezen. Omdat dit IC een belangrijk bestanddeel van de Commodore 64 is, worden

hieronder de in- en uitgangssignalen verklaard en hun doel (uitgangen) en herkomst (ingangen) aangegeven. Allereerst de ingangen:

- CAS Signaal van de VIC om de RAM-adressen te sturen
- LORAM signaal van de processorport bit 0
- HIRAM als boven, maar dan bit 1
- CHARN als boven, maar dan bit 2. Dit signaal moet het uitlezen van de karaktergenerator mogelijk maken. Doet u dat nooit in BASIC, want dan treft de door de timer aangewezen interruptroutine de karaktergenerator aan in plaats van de verwachte timer (karaktergenerator en timer in de CIA hebben namelijk hetzelfde adres). De interruptroutine vertakt dan op een verkeerde manier, waardoor het bedrijfsysteem vastloopt.
- VA14 komt van CIA 2, port A, bit 0; is een deel van het basisadres van karaktergenerator en video-RAM.
- A13-A15 adresbussignalen, dienen hier voor de decodering van het I/O-bereik.
- BA signaal van de VIC, werkt op de RDY-lijn van de processor.
- AEC 0= processor stuurt de bus, 1= VIC stuurt de bus. Het signaal wordt door inverteren van de AEC van de VIC verkregen.
- R/W stuursignaal van de processor
- EXROM ingangssignaal van het expansion-slot
- GAME als boven
- VA12-13 adresbus van de VIC

Nu de uitgangen:

- CASRAM 0=RAM neemt de MSB (Most Significant Byte = de hoogstwaardige byte) van het geheugenadres over
- BASIC 0=BASIC-ROM is geselecteerd

-KERNAL	0=bedrijfssysteem-ROM is geselecteerd
-CHAROM	het kiezen van de karaktergenerator
GR/W	0=Er kan in het kleur-RAM worden geschreven.
-I/O	0=selecteert de I/O-decoder
-ROML	signaal naar uitbreidingsslot
-ROMH	idem

Op de volgende bladzijde vindt u een (bijna) volledige lijst van de koppelingscombinaties van in- en uitgangsmogelijkheden. Het was heel gemakkelijk geweest dit boek met alle uitgangsmogelijkheden te vullen, die bij de 65536 ingangsmogelijkheden horen. Dit soort IC's heeft een groot aantal redundante (extra) ingangsbits, dat wil zeggen bits, waarbij in een bepaalde constellatie niets aan de uitgang verandert. In de tabel zijn ze door X gekenmerkt. Om ze op het spoor te komen is een programma ontwikkeld dat vier weken lang zonder onderbreking liep, en toen nog niet helemaal klaar was. Daardoor ontbreken er aan het eind van de tabel misschien een aantal combinaties. Het zijn er een paar van de uitgangsmogelijkheid \$FE en alle combinaties van de uitgangsmogelijkheid \$FF. Deze uitgangsmogelijkheid is echter toch niet van betekenis, omdat zoals te zien is alle uitgangen 0-actief moeten zijn, willen ze enig effect hebben. Als u zich niet door de tabel heen wilt worstelen, vindt u op de volgende bladzijden in verkorte vorm de belangrijkste veranderingen, die de AdresManager (AM) in de geheugenstructuur aanbrengt.

- C H - - V A R R E A M S													- C K - - G H E R S A S R A M													
- - - - R R - - R O O M H L													- - - - R R - - R O O M H L													
- - - - R R - - R O O M H L													- - - - R R - - R O O M H L													
V A 1 3 2	V A 1 3 2	- G A R M E	- X A R M E	- R A R M E	- A E A	A 1	A 2	A 3	A 4	A 5	A 6	A 7	A 8	A 9	A 10	A 11	A 12	A 13	A 14	A 15	A 16	A 17	A 18	A 19	A 20	
0	0	X	X	1	0	X	X	1	0	1	X	X	1	X	X	1	X	X	0	1	1	1	1	1	1	1
1	0	X	X	1	0	X	X	1	0	1	X	X	1	X	X	1	X	X	0	1	1	1	1	1	1	1
1	0	1	1	X	1	X	X	1	0	1	X	X	1	X	X	1	X	X	0	1	1	1	1	1	1	1
X	0	X	X	1	0	X	X	1	0	0	1	X	X	1	X	1	1	1	X	1	1	1	1	1	1	1
0	X	X	X	1	0	X	X	1	0	0	1	X	X	1	X	1	1	1	X	1	1	1	1	1	1	1
1	0	X	X	0	X	X	0	0	0	1	X	X	1	X	X	1	X	1	X	0	1	1	1	1	1	1
X	X	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	0	1	1	1	1	1	1	1
X	X	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	0	1	1	1	1	1	1	1
1	0	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
X	X	X	X	1	0	1	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
X	X	X	X	1	0	1	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
1	0	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
1	0	X	X	1	0	1	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
0	X	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
0	X	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
X	1	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
X	1	X	X	0	0	X	1	0	1	1	1	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1
X	1	0	1	X	1	X	X	X	X	X	1	X	X	X	1	X	X	X	1	X	X	X	X	X	X	X
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
X	1	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1	X	X	1	1	1	1	1
0	X	X	X	1	0	X	X	1	1	1	X	X	1	X	X	1	X	X	1							

\$E000	BEDRIJFSSYSTEEM
\$DC00	CIA1/2 & 1/O 0-1
\$D800	KLEUR-RAM
\$D000	VIC II/SID
\$C000	4K RAM
\$A000	8K BASIC
\$8000	
\$0000	

Wanneer er geen apparatuur op het expansion slot is aangesloten, is deze configuratie na het inschakelen van de computer ingesteld.

-LORAM =1
-HIRAM =1
-EXROM =1
-GAME =1
-CHAREN =1

\$E000

BEDRIJFSSYSTEEM

\$DC00

CIA1/2 & 1/O 0-1

\$D800

KLEUR-RAM

\$D000

VIC II/SID

Deze configuratie is voorstelbaar, nadat
een gemodificeerd BASIC in het
overlappend RAM-bereik wordt gekopieerd.

- LORAM =0
- HIRAM =1
- EXROM =X
- GAME =1
- CHAREN =1

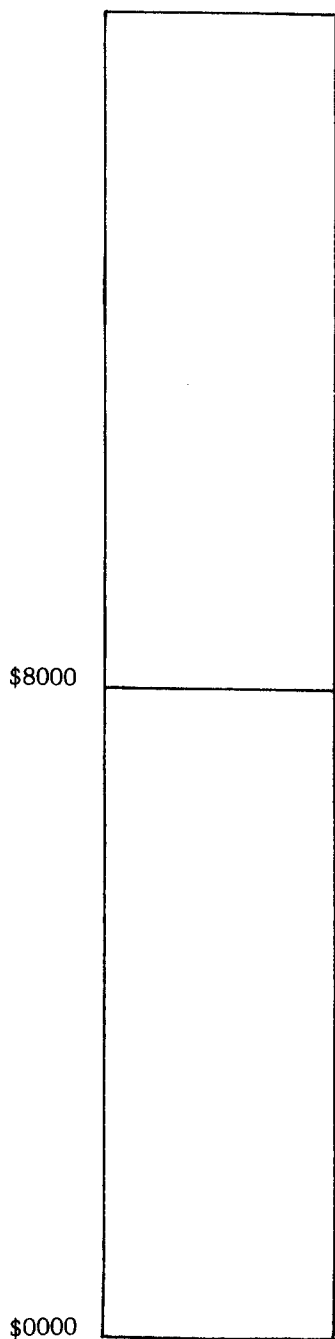
\$8000

\$0000

\$E000	8K RAM
\$DC00	CIA1/2 & 1/O 0-1
\$D800	KLEUR-RAM
\$D000	VIC II/SID
\$8000	
\$0000	

In deze configuratie kan het geheugen door een externe processor worden gebruikt, waarbij de perifere bouwstenen ook worden benut.

-LORAM =1
-HIRAM =0
-EXROM =X
-GAME =1
-CHAREN =1



Deze vorm lijkt op de vorige. Het I/O-bereik kan hier echter niet worden benut.

- LORAM =0
- HIRAM =0
- EXROM =1
- GAME =1

\$E000	BEDRIJFSSYSTEEM
\$DC00	CIA1/2 & 1/0 0-1
\$D800	KLEUR-RAM
\$D000	VIC II/SID
\$C000	4K RAM
\$8000	16K INSTEEL-ROM
\$0000	

Deze configuratie is typisch voor de situatie, waarin een andere taal dan BASIC moet worden gebruikt. De andere taal (bijvoorbeeld Pascal) zit in een externe ROM-cassette.

-LORAM =1
-HIRAM =1
-EXROM =0
-GAME =0
-CHAREN =1

\$E000	BEDRIJFSSYSTEEM
\$DC00	CIA1/2 & 1/O 0-1
\$D800	KLEUR-RAM
\$D000	VIC II/SID
\$C000	4K RAM

Deze configuratie is typisch voor spelletjes.

- LORAM =0
- HIRAM =1
- EXROM =0
- GAME =0
- CHAREN =1

\$E000	BEDRIJFSSYSTEEM
\$DC00	CIA1/2 & I/O 0-1
\$D800	KLEUR-RAM
\$D000	VIC II/SID
\$C000	4K RAM
\$A000	8K BASIC
\$8000	BASIC- UITBREIDING
\$0000	

In deze configuratie is BASIC door een ROM-cassette uitgebreid (bijvoorbeeld toolkit).

-LORAM =1
-HIRAM =1
-EXROM =0
-GAME =1
-CHAREN =1

\$DF00	I/O 1
\$DE00	I/O 0
\$DD00	I/O 2
\$DC00	I/O 3
\$D800	KLEUR-RAM
\$D400	SID-CHIP
\$D000	VIDEO- CONTROLLER

De ontsluiting van het I/O-bereik,
wanneer -CHAREN=1

Bij -CHAREN=0 gebruikt de
karaktergenerator het hele bereik.

1.5 De uitbreidingspoort

Aan de achterkant van de Commodore 64 zit een 44-polige aansluitmogelijkheid. Hier worden bijvoorbeeld uitbreidingen van BASIC of van het bedrijfssysteem aangesloten (toolkit, machinetaalmonitor enzovoort), maar ook spelletjesprogramma's en uitbreidingen van de in- en uitgang. Door een externe processor aan te sluiten kunnen zelfs de componenten binnen de computer worden aangesproken. Door de juiste aanpassingen kunnen op de uitbreidingspoort meer uitbreidingen tegelijkertijd worden aangesloten. Modules voor de VIC 20 kunnen echter logisch, noch fysiek op het expansion slot worden aangesloten. Wilt u het toch proberen, dan moeten de voor de VIC 20 typische blokkeuzesignalen worden verkregen door decoding van de adresbits 13-15. Veel signalen zijn niet op het eerste gezicht duidelijk. Daarom volgt hier de pinconfiguratie.

1	GND	
2-3	+5V	
4	-IRQ	met IRQ van de processor verbonden
5	CR/W	met R/W van de processor verbonden
6	DOT CLOCK	klok van het puntraster voor de VIC, ongeveer 7,83Mhz
7	-I/O 1	gewoonlijk=0 in het adresbereik \$DE00 tot \$DEFF
8	-GAME	ingang voor de MM (Memory Manager = geheugenbeheer); zie paragraaf 1.4 voor het effect als boven
9	-EXROM	
10	-I/O 2	gewoonlijk=0 in het bereik \$DF00 tot \$DFFF
11	-ROML	uitgang van de MM; zie paragraaf 1.4
12	BA	signaal van de VIC, dat de geldigheid van leesdata aangeeft
13	-DMA	ingang; 0=bussysteem voor extern gebruik reserveren
14-21	CD7-CDO	databus
22	GND	
A	GND	
B	-ROMH	uitgang van de MM; zie paragraaf 1.4
C	-RESET	
D	-NMI	
E	O2	uitgang van de systeemklok
F-Y	CA15-CA0	adresbus
Z	GND	

Vergeleken met het expansion slot van de VIC 20 zijn er belangrijke verschillen. De onderlinge afstand van de contacten

is bij de Commodore 64 2,54 mm, bij de VIC 20 3,96 mm. De blokken RAM-keuzesignalen -RAM1-3 en -BLK1-5 ontbreken volkomen, omdat in de Commodore 64 alle 16 adreslijnen zijn aangebracht, terwijl de VIC 20 slechts de lijnen 0-12 heeft. Het I/O-bereik heeft in de Commodore 64 een omvang van 256bytes, in de VIC 20 1Kbyte. Naar aanleiding van deze en de vorige paragraaf zal duidelijk zijn, dat men zelfs van buitenaf een enorme invloed heeft op de geheugenconfiguratie. Dat geldt vooral voor de signalen -EXROM en -GAME. Hiermee kan de geheugenstructuur zo worden veranderd, dat na het inschakelen zelfs het verwachte READY niet meer op het beeldscherm verschijnt.

1.6 De gebruikerspoort

Via de userport wordt de randapparatuur van de gebruiker gestuurd. In het eenvoudigste geval zijn dat lampen die via stuurtransistoren zijn aangesloten. Randapparatuur kan echter ook bestaan uit een printer met een 8-bits parallele Centronics interface. Een geschikte stekker voor de userport is de Cardconnector 251-12-50-170 uit de serie cinch-connectors van de firma TRW. Deze is direkt op de printplaat aan te sluiten en bezit in twee rijen de gewenste 2x12 contacten met een onderlinge afstand van 3,96 mm. In principe bestaat de userport uit een 8-bits poort en diverse stuurlijnen, waarover hieronder meer.

1	GND	
2	+5V	belastbaar met maximaal 100mA
3	-RESET	met de gelijknamige processorlijn verbonden
4	CNT 1	verbonden met CNT van CIA 1
5	SP 1	met SP van CIA 1 verbonden
6	CNT 2	lijn CNT van CIA 2
7	SP 2	verbonden met SP van CIA 2
8	-PC 2	handshake-uitgang van CIA 2
9	ATN OUT	stuurlijn van de seriële IEC-bus; begint bij PA3 van CIA 2
10	9V AC	wisselspanning; met ten hoogste 100mA belastbaar
11		Tegenpool van 10
12	GND	
A	GND	
B	-FLAG 2	handshake-ingang van CIA 2
C-L	PB0-PB7	I/O-lijnen van CIA 2
M	PA2	I/O-lijn van CIA 2. Deze lijn vervangt de CB2 van

de VIA 6522. De CB2 is bekend van andere
CBM-machines.

N GND

Enkele van deze verbindingen hebben in de Commodore 64 al vaste functies. Hierover en over het gebruik van de CIA's informeert hoofdstuk 4. Wilt u aan de userport apparaten aansluiten die eigenlijk zijn bedoeld voor aansluiting op de VIC 20 of de CBM 8032 of andere, dan is dat in beginsel slechts bij die apparaten mogelijk die de pinnen A-N, 1 en 12 niet nodig hebben. In ieder geval moet u letten op de programmatische te onderscheiden pinnen B en M. In hoofdstuk 4 meer hierover.

1.6.1 De programmering van de gebruikerspoort in BASIC

Deze paragraaf is bedoeld voor mensen, die goed in BASIC thuis zijn, maar alleen met standaardaansluitingen vertrouwd zijn. We willen laten zien, hoe een eenvoudige, in BASIC te bedienen schakeling op de userport kan worden aangesloten. De schakeling bestaat uit slechts 4 schakelaars, 4 LED's, 8 weerstanden en een IC. Hiermee kunt u de elementaire begrippen van de data-invoer en -uitvoer makkelijk begrijpen en voor eigen projecten gebruiken. De schakeling staat aan het eind van de paragraaf. Ze is zo eenvoudig, dat ze geen commentaar behoeft. Vanwege het grote aantal aansluitlijnen op de userport (het zijn er meer dan bijvoorbeeld bij de computers uit de PET reeks) moet allereerst worden duidelijk gemaakt wat nu eigenlijk voor de gebruiker bestemd is. Als u geen RS-232-cartridge gebruikt, kunt u de volgende aansluitingen gebruiken zonder dat de overige functies van de computer worden beïnvloed:

1 - 2
4 - 8
10 - 12
A - N

In de vorige paragraaf is de betekenis van de pinnen vermeld.

Nu het voorbeeld. De datalijnen zijn afzonderlijk op in- of uitvoer te programmeren. In dit voorbeeld gebruiken we PB0-PB3 als invoer, PB4-PB7 als uitvoer. De datarichting wordt bepaald door het datarichtingsregister voor de dataport B op adres 56579 te laden. Een ingesteld bit (=1) betekent uitvoer op het

corresponderende bit van dataport B (adres 56577). Een teruggezet bit betekent invoer. Om de datarichtingen voor het voorbeeld te bepalen, dat wil zeggen bits 0-3 als invoer en 4-7 als uitvoer te definiëren, gebruikt u de volgende instructie.

POKE 56579,240

De genoemde 4 bits zijn nu ingesteld en dus staan de corresponderende databits van poort B op uitvoer en de rest op invoer. Het verdere gebruik van de schakeling verloopt als volgt.

PRINT PEEK(56577)AND15

geeft de stand van de 4 schakelaars aan en met

POKE 56577,X

kunt u de LED's in- of uitschakelen. De waarde X mag alleen bestaan uit de getallen 16, 32, 64, 128. Als alle lampjes uit moeten zijn, is de waarde van X 0. Mocht u eigen projecten op de userport willen aansluiten, dan moet u - om beschadiging van de computer te voorkomen- het volgende goed in acht nemen. Als de userport als ingang wordt gebruikt, mag de voedingsspanning 0-5V zijn. Een spanning van 0-0,6V wordt bij het uitlezen van de datapoort als 0 geïnterpreteerd, een spanning van 1,6-5V als 1. Het bereik van 0,7-1,5V is indifferent, dat wil zeggen, dat zo'n spanning toevallig als 0 of als 1 kan worden herkend. Als de userport als uitgang wordt gebruikt, moet u er rekening mee houden, dat de uitgangen slechts de belasting van een TTL-ingang kunnen verdragen. In geen geval kan dus een LED direct worden aangesloten. Op den duur zou daardoor namelijk de CIA kapot gaan. Let er daarom goed op, welke waarde u in het richtingsregister van de databus laadt, opdat niet per ongeluk een bit, dat voor invoer is bestemd, op uitvoer wordt geprogrammeerd. Als u de userport de stroomtoevoer voor het project wilt laten verzorgen, bedenk dan, dat de beide spanningen, die ter beschikking staan elk met niet meer dan 100mA mogen worden belast. Bij een lichte overbelasting stopt eerst de cassette recorder, daarna springt de zekering van de Commodore 64 en eventueel ook de hoofdzekering bij de trafo. Vitale delen worden echter niet aangetast. Dit was slechts een kleine gebruiksaanwijzing voor een eenvoudig geval. Als u voor complexere doeleinden ook de andere verbindingen wilt gebruiken, kunt u zich ter oriëntatie in hoofdstuk 4 verdiepen. Wat hierboven gezegd werd, geldt ook in dat verband onverkort.

2 DE SYNTHESIZER EN ZIJN PROGRAMMERING

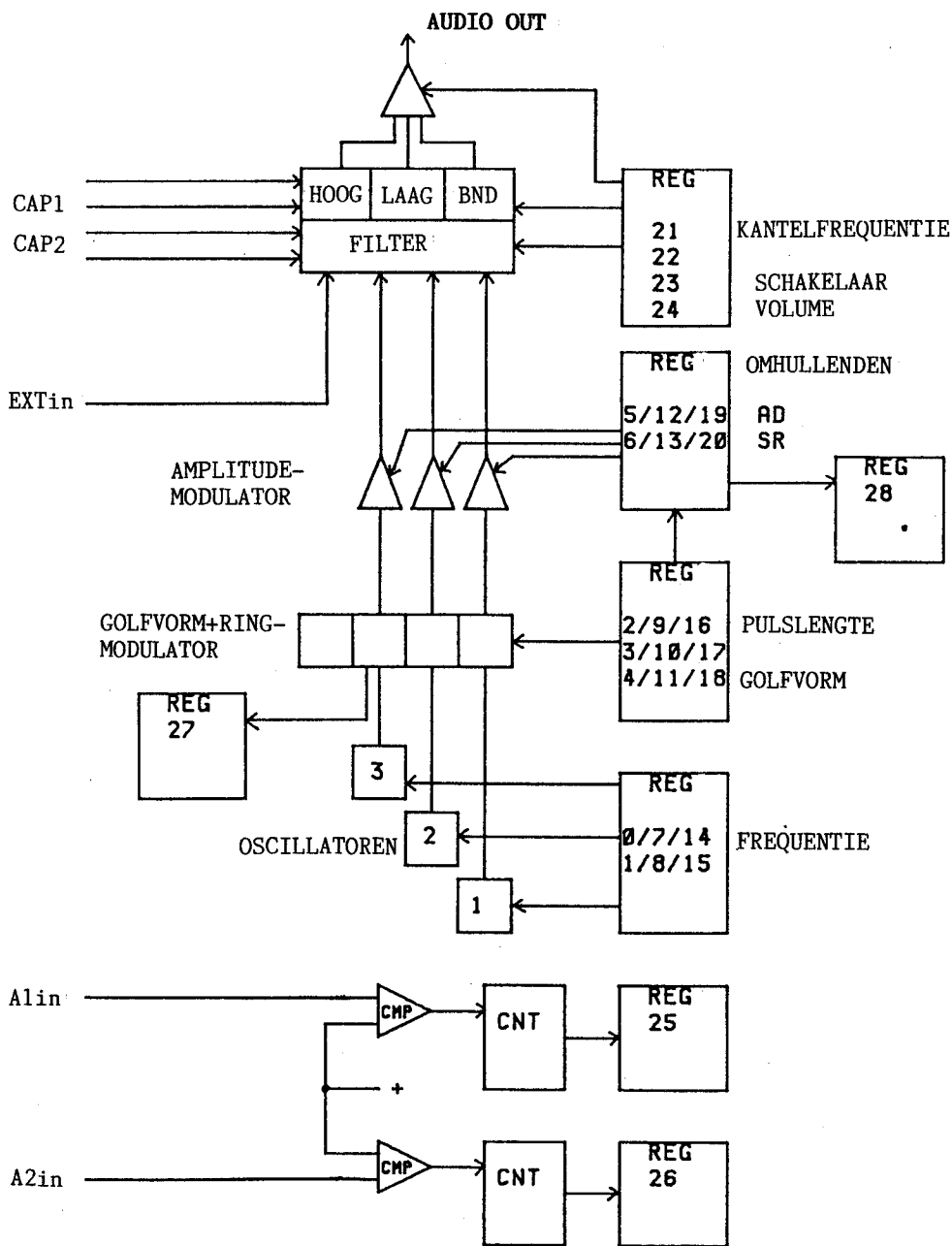
2.1 De sound controller 6581

2.1.1 Algemene opmerkingen over de SID 6581

De synthesizer, die in de Commodore 64 zit, biedt de mogelijkheid om alle denkbare geluiden te produceren. Synthesizers die normaal in de handel verkrijgbaar zijn, beschikken over 1 stem (monofoon); de synthesizer van de Commodore 64 heeft er daarentegen 3 (polyfoon). Alle elementen die voor het voortbrengen van geluid noodzakelijk zijn, bevinden zich hier op een enkel IC, de Sound Interface Device (SID) 6581. Dit is een nieuwe perifere bouwsteen uit de familie 65xx. Hij beschikt over:

- * 3 programmeerbare, onafhankelijke oscillatoren (stemmen)
- * 4 mengbare golfsoorten per stem
- * 3 mengbare filters (hoog-, laag- en bandfilter)
- * per stem een envelope generator (ADSR controll)
(=generator voor omhullende curven)
- * 2 cascadeerbare ringmodulatoren
- * vervormingsmogelijkheid voor externe signaalbronnen
- * 2 8-bits analoog-digitaal omzetters

Het blokschema van de SID staat op de volgende bladzijde.



De SID 6581 bestaat uit drie synthesizerstemmen. Ze kunnen onafhankelijk en in verbinding met elkaar of met een extern audiosignaal worden gebruikt om uiterst complexe golfvormen te creëren. Iedere stem bestaat uit een oscillator en een golfvormgenerator die vier verschillende golfvormen kan produceren. Verder een envelopegenerator (= generator voor omhullende curven) en een amplitudemodulator. De oscillator produceert de basisfrequentie in het bereik van 0-8200Hz (bij een systeemklok van ongeveer 1 MHz) met een nauwkeurigheid van 16bits. Hij kan vier golfvormen voortbrengen: driehoeksgolf, zaagtand, blokpuls (met variabele puls/pauze verhouding) en roze ruis. Iedere golfvorm heeft een typisch aantal boventonen. Alleen al door de keuze van de golfvorm kan de klankkleur dus worden beïnvloed. Een driehoeksgolf heeft ongeveer hetzelfde aantal boventonen als een sinus. Daardoor ontstaat een zachte klank, zoals van een blokfluit. De zaagtand heeft een volledig spectrum van boventonen dat gelijkmatig is verdeeld en daardoor scherp klinkt, ongeveer als een trompet. Bij de blokpuls ontbreekt een aantal boventoonbereiken. Ze klinkt enigszins hol, zoals een klarinet. De roze ruis is een toevallige volgorde van verschillende frequenties binnen vastgestelde grenzen.

Het volumeverloop van een bepaalde toon wordt door een amplitudemodulator beïnvloed, die door een envelopegenerator gestuurd wordt. Als die laatste door een triggerbit (te vergelijken met de aanslag van een pianotoets) wordt aangesproken, begint het geprogrammeerde verloop vanaf de aanzet via de volle toon tot en met het wegsterven ervan. Bovendien kunnen de uitgangen van alle stemmen nog door programmeerbare filters worden gestuurd om de klankkleur in een groot gebied te beïnvloeden. Tenslotte kunnen de tonen 1 en 2 met behulp van toon 3 worden bewerkt door de ringmodulator. Zo kunnen behalve de basisstem ook nog de som en het verschil met stem 3 ontstaan.

Als tijdens het verloop van een bepaalde toon de klankkleur veranderd moet worden (bijvoorbeeld WAH-WAH effect), kan dat bij stem 3 worden bewerkstelligd door de momentele waarde van de envelope-generator uit te lezen. Het effect ontstaat doordat vervolgens de filter- of oscillatorfrequentie wordt veranderd. Bovendien kan van stem 3 nog de ruisgeneratorstand op een willekeurig moment worden uitgelezen. Zo staat er dus een uitstekende toevalsgenerator ter beschikking.

De SID heeft twee 8bits A/D omzetter. Daarmee kan de relatieve waarde worden opgevraagd van twee potentiometers (bijvoorbeeld paddles), die op de controlport worden aangesloten. In de meeste gevallen zal het daarbij om spelletjes gaan, maar er zijn ook andere gebruiksmogelijkheden denkbaar, zoals bij de synthesizer de sturing van toonhoogte en volume of andere parameters.

2.1.2 Registerbeschrijving van de SID

Het basisadres van de SID in de Commodore 64 is \$D400(54272)

- REG 0** oscillatorfrequentie, LSB (Least Significant Byte = byte met de laagste waarde) voor stem 1
- REG 1** oscillatorfrequentie, MSB (Most Significant Byte = byte met de hoogste waarde) voor stem 1
- REG 2** pulsbreedte, LSB voor stem 1
- REG 3** pulsbreedte, MSB voor stem 1
De registers 2 en 3 bepalen de puls/pause-verhouding van de blokpulsuitgang van stem 1
- REG 4** stuurregister voor stem 1
-Bit 0 KEY: stuurbit voor het verloop van de envelopegenerator (zorgt voor de produktie van omhullende curven). Bij de overgang van 0 naar 1 stijgt het volume binnen de in REG 5 geprogrammeerde attacktijd van nul naar het maximum. Vervolgens daalt het volume gedurende de decaytijd (eveneens in REG 5 ingesteld) naar het sustainniveau, dat in REG 6 gekozen is. Totdat de stuurbit weer 0 wordt, blijft het op dat peil. Dan gaat het volume binnen de in REG 6 geprogrammeerde releasetijd terug naar 0.
-Bit 1 SYNC: 1=oscillator 1 wordt met oscillator 3 gesynchroniseerd. Deze bit heeft ook effect als stem 3 uitgeschakeld is.
-Bit 2 RING: 1=de uitgang van de driehoeksgolf van oscillator 1 wordt vervangen door een mengeling van frequenties (som en verschil van de frequenties van oscillator 1 en 3). Dit effect treedt ook op, wanneer stem 3 uitgeschakeld is.
-Bit 3 TEST: wanneer samen met de ruisgenerator nog een andere golfvorm van dezelfde oscillator is gekozen, kan het voorkomen, dat de ruisgenerator blokkeert. De blokkade kan door deze bit weer worden opgeheven.
-Bit 4 TRI: 1=driehoeksgolf gekozen
-Bit 5 SAW: 1=zaagtandgolf gekozen
-Bit 6 PUL: 1=blokpuls gekozen. De puls/pause-verhouding van deze golf wordt in REG 2 en REG 3 ingesteld.
-Bit 7 NSE: 1=ruisgenerator gekozen
Opmerking bij de bits 4-7. Het is praktisch mogelijk gelijktijdig meer dan een golfvorm te kiezen. Behalve van wat er over bit 3 gezegd is, dient men zich ervan bewust te zijn, dat het resultaat niet de som van alle vormen is, maar veeleer een logische EN-verbinding van de componenten.

- REG 5** **ATTACK/DECAY**
 -Bit 0-3. Deze bits bepalen de tijd, waarin het volume van maximaal naar het sustainniveau daalt. Het bereik is instelbaar van 6millisec.- 24sec.
 -Bit 4-7. Hiermee wordt de tijd bepaald, waarin het volume van 0 naar het maximum stijgt, nadat het KEY-bit is ingesteld. Het bereik is instelbaar van 2millisec. tot 8sec.
- REG 6** **SUSTAIN/RELEASE**
 -Bit 0-3. Met deze bits wordt de tijd ingesteld, waarin het volume van het sustainniveau naar 0 daalt, nadat het KEY-bit is teruggezet. Het bereik is instelbaar van 6millisec.- 24sec.
 -Bit 4-7. Deze bits geven het sustainniveau aan, dat wil zeggen het volume, dat de toon na de decaytijd heeft.
- REG 7-13** Deze registers sturen stem 2 op dezelfde wijze als de registers 0-6. Uitzonderingen hierop zijn:
 -SYNC synchroniseert oscillator 2 met oscillator 1;
 -RING vervangt de driehoeksuitgang van oscillator 2 door de geringmoduleerde frequenties van de oscillatoren 2 en 1.
- REG 14-20** Deze registers sturen stem 2 op dezelfde manier als de registers 0-6. Uitzonderingen zijn:
 -SYNC synchroniseert oscillator 3 met oscillator 2;
 -RING vervangt de driehoeksuitgang van oscillator 3 door de menging van frequenties uit de oscillatoren 3 en 2;
- REG 21** filterfrequentie van het LSB.
 Alleen de bits 0-2 worden gebruikt.
- REG 22** filterfrequentie van het MSB.
 Het 11-bits getal van de registers 21 en 22 bepaalt de kantel- respectievelijk middenfrequentie van het filter. In de Commodore 64 wordt die als volgt uitgerekend: $F = (30 + W * 5.8) \text{Hz}$, waarbij W het 11-bits getal voorstelt.
- REG 23** filterresonantie en -schakelaars
 -Bit 0 l=stem 1 wordt door het filter heen gestuurd.
 -Bit 1 l=stem 2 wordt door het filter heen gestuurd.
 -Bit 2 l=stem 3 wordt door het filter heen gestuurd.
 -Bit 3 l=de externe signaalbron wordt gefilterd.
 -Bit 4-7 Deze bits bepalen de resonantiefrequentie van het filter. Die wordt gebruikt om bepaalde delen van het frequentiespectrum naar voren te halen. De werking kan bij de zaagtandgolf goed worden waargenomen.
- REG 24** Dit register heeft de volgende functies:
 -Bit 0-3 totaalvolume
 -Bit 4 schakelt het lage doorlaatgebied van het filter in.
 -Bit 5 schakelt de bandaftakking van het filter in.

-Bit 6 schakelt het hoge doorlaatgebied van het filter in.

Het hoge en het lage gedeelte van het filter hebben een flankstijlheid van 12dB per octaaf.

Er kunnen meer filters ingeschakeld zijn. Als bijvoorbeeld het hoog- en laagfilter ingeschakeld zijn, is de bandaftakking geblokkeerd. Om de invloed van het filter te laten horen moet tenminste 1 filter zijn ingeschakeld en tenminste 1 stem door het filter worden heen gestuurd. Meestal wordt het filter gebruikt om bepaalde gedeelten uit een frequentiespectrum weg te filteren. De filters beïnvloeden daardoor het klankbeeld op veel fijnere wijze dan door keuze van de golfvorm mogelijk is. Als de filterfrequentie wordt veranderd gedurende het verloop van een toon (bij korte tonen is dat alleen in machinetaal mogelijk), kunnen de meest verschillende instrumenten perfect worden nagebootst.

-Bit 7 = stem 3 niet hoorbaar. Men kan hiervan gebruik maken, als het verloop van deze stem alleen de parameters voor de andere stemmen moet leveren (zie REG 27 en 28). In alle genoemde registers kan slechts geschreven worden. Een leesinstructie heeft geen resultaat. Alle volgende registers kunnen slechts gelezen worden.

REG 25 A/D omzetter 1

REG 26 A/D omzetter 2

REG 27 ruisgenerator 3

Dit register geeft een willekeurig getal, dat overeenkomt met de momentele stand van ruisgenerator 3. De generator moet dan wel ingeschakeld zijn, stem 3 kan echter niet hoorbaar zijn (bit 7 in REG 24 = 1).

REG 28 envelopegenerator

Dit register geeft de momentele stand van het relatieve volume van stem 3. Overeenkomstig het volumeverloop kan de frequentie of de filterparameter worden veranderd. Een voorbeeld staat in paragraaf 2.2.

2.1.3 De analoog/digitaal omzetter

Een A/D converter verandert een analoog signaal (bijvoorbeeld spanning) in een digitale waarde. De principiële moeilijkheid is de omvorming van de vloeiende analoge waarde in een trapsgewijze digitale waarde met vaste intervallen. Noodzakelijkerwijs is de grootste fout +/- een kleinste digitale stap. De SID 6581 heeft twee A/D convertors. Het gaat daarbij om een schakeling met een intern geproduceerde referentiespanning van ongeveer 2,5V. Het meetproces bestaat uit de ontlading van een externe capaciteit, waarna een waarde wordt overgenomen in REG 25 of 26. Die waarde komt overeen met de tijd, die nodig is voor een nieuwe oplading van de capaciteit tot aan de referentiespanning. Dit proces herhaalt zich steeds.

2.1.3.1 Het gebruik van de analoog-digitaal omzetter

Uit wat boven werd gezegd, blijkt dat voor de convertor alleen een potmeterschakeling in aanmerking komt. Voor de registratie van meetwaarden zijn daarom alleen veranderlijke weerstanden geschikt, bijvoorbeeld licht- of warmtegevoelige weerstanden. Als er spanningen moeten worden gemeten, moeten ze van te voren in een geschikte vorm worden omgezet. Dat kan gebeuren door een uni-junction-transistor. De meetschakeling ziet er als volgt uit: aan het eind van de meetweerstand wordt via de controlport +5V aangesloten en aan het andere eind (ook via de controlport) de analoogingang van de SID (benamingen: POTX en POTY). De waarde uit de registers 26 en 27 is een maat voor de weerstand. Om alle 8 bits te kunnen gebruiken, moet de weerstand tussen de 200 Ohm (niet kleiner!) en de 200 KOhm liggen. Het programmeer-technische gebruik van de A/D convertor wordt in de volgende paragraaf behandeld.

2.1.3.2 Het gebruik van paddles

Alle paddles die normaal in de handel verkrijgbaar zijn, kunnen op de Commodore 64 worden aangesloten. Ze worden gewoon in controlport 1 of aan de rechterkant van het apparaat geplugd. Een 'paddle' is niets anders dan een potmeter, die met de SID is verbonden plus een extra toets die op de joystickpositie LINKS voor de ene paddle werkt en voor de andere op RECHTS. Problematischer is de programmeer-technische bepaling van de paddlewaarden. Enkele bits, die voor de sturing en het afchecken van de paddles dienen, moeten de CIA's 1 en 2 met het toetsenbord delen. Dit wordt in paragraaf 8.7 uiteengezet. Aan de ene kant zijn dit de paddletoetsen. Aan de andere kant is er de mogelijkheid om twee paren paddles aan te sluiten. Maar omdat de SID slechts twee A/D convertors heeft, zijn de paddles over een analoge switch geschakeld. Om die te bedienen worden twee bits van de CIA 1 gebruikt. Om de waarde van de A/D convertor te bepalen, moet dus ook in dit geval het toetsenbord worden uitgeschakeld. Het volgende kleine machinetaalprogramma biedt de oplossing. Het maakt als data-statement deel uit van een BASIC-programma. De parameters van vier aangesloten paddles kunnen langs deze weg gemakkelijk worden bepaald.

Het programma zit in een gedeelte van de computer dat niet door het bedrijfssysteem wordt gebruikt. De terugmeldingen lopen over enkele bytes die alleen een andere functie hebben wanneer de cassette-recorder wordt gebruikt.

CFBE	SEI		; gebruik van toetsenbord verhinderen
CFBF	LDA	#\$80	; parameters voor paddlestel A
CFC1	JSR	\$CFEC	; A/D waarden A1 en A2 halen
CFC4	STX	\$033C	; en veilig stellen
CFC7	STY	\$033D	
CFCA	LDA	\$DC00	; toetsen A uit CIA 1 halen
CFCD	AND	#\$0C	; benodigde bits filteren
CFCF	STA	\$029F	; en veilig stellen
CFD2	LDA	#\$40	; parameters voor paddlestel B halen
CFD4	JSR	\$CFEC	; A/D waarden B1 en B2 halen
CFD7	STX	\$033E	; en veilig stellen
CFDA	STY	\$033F	
CFDD	LDA	\$DC01	; toetsen B uit CIA 2 halen
CFE0	AND	#\$0C	; benodigde bits filteren
CFE2	STA	\$02A0	; en veilig stellen
CFE5	LDA	#\$FF	; alle bits uitgang in CIA 1
CFE7	STA	\$DC02	; om gebruik van toetsenbord weer
CFEA	CLI		; toe te staan


```

CFEB      RTS          ; terugkeer in het BASIC-programma
CFEC      STA  $DC00    ; paddlestel uitkiezen
CFEF      ORA  #$C0     ; en overeenkomstige bits
CFF2      STA  $DC02    ; op uitgang zetten
CFF4      LDX  #$0      ; vertragsingslus voor
CFF6      DEX          ; stabilisatie van de
CFF7      BNE  $CFF6    ; A/D ingang
CFF9      LDX  $D419    ; A/D 1 halen
CFFC      LDY  $D41A    ; A/D 2 halen
CFFF      RTS          ; terugkeer in het hoofdprogramma

```

Nu het BASIC-laadprogramma. Sluit de paddles aan, laad en start vervolgens het programma.

```

10 DATA 120,169,128, 32,236,207,142, 60,  3,140, 61,  3,173
20 DATA  0,220, 41, 12,141,159,  2,169, 64, 32,236,207,142
30 DATA 62,  3,140, 63,  3,173,  1,220, 41, 12,141,160,  2,169
40 DATA 255,141,  2,220, 88, 96,141,  0,220,  9,192,141,  2
50 DATA 220,162,  0,202,208,253,174, 25,212,172, 26,212, 96
60 FOR M=53182 TO 53247
70 READ A: POKE M,A: NEXT: REM; machinetaalprogramma inlezen
80 AX=830: REM; paddle 1 aan controlport 1
90 AY=831: REM; paddle 2 aan controlport 1
100 BA=672: REM; toetsen van paddlestel A
110 BX=828: REM; paddle 1 aan controlport 2
120 BY=829: REM; paddle 2 aan controlport 2
130 BB=830: REM; toetsen van paddlestel B
140 SYS 53182: REM; alle waarden halen
150 PRINT PEEK(AX)" "PEEK(AY)" "PEEK(BA)" ";
160 PRINT PEEK(BX)" "PEEK(BY)" "PEEK(BB)" "
170 GOTO 140

```

2.2 De programmering van de SID 6581

Deze paragraaf beperkt zich tot de programmering van de synthesizer, omdat het gebruik van de A/D convertor in de paragrafen 2.1.3.1 en 2.1.3.2 werd besproken. Om een toon te produceren moet eerst het register 24 worden geladen, dan de ADSR-registers (5,6,12,13,19,20), daarna de overige registers en wel het liefst in de volgorde van de stemmen, waartoe ze behoren. Uitzonderingen hierop zijn de registers 4, 11 en 18. Tot slot laadt u die drie (vergeet bit 0 niet) en de toon weerklinkt. Door het volgende, kleine programma maakt u kennis met de golfvormen en

de toonomvang van de synthesizer.

```
10 S1=54272: REM; stem 1
20 S2=54279: REM; stem 2
30 S3=54286: REM; stem 3
40 FL=54293: REM; filter LSB
50 FH=54294: REM; filter MSB
60 RS=54295: REM; resonantie + schakelaar
70 PL=54296: REM; doorlaatgebied + volume
80 POKE S1+4,0: POKE S2+4,0: POKE S3+4,0
100 POKE S1+2,0: POKE S1+3,8
110 POKE S1+5,0: POKE S1+6,240
120 POKE RS,0 : POKE PL,15
130 PRINT "DRIEHOEK"
140 T=16: GOSUB=300
150 PRINT "ZAAGTAND"
160 T=32: GOSUB 300
170 PRINT "BLOKGOLF"
180 T=64: GOSUB 300
190 PRINT "RUIS"
200 T=128: GOSUB 300
210 END
300 POKE S1,0: POKE S1+1,0
310 POKE S1+4,T+1: REM; toon inschakelen
320 FOR I=0 TO 255: FOR J=0 TO 255 STEP 50
330 POKE S1,J: POKE S1+1,I
340 NEXT J,I
350 POKE S1+4,T: REM; toon uitschakelen
360 RETURN
```

De regels 10-80 dienen voor ieder geluidsprogramma staan. Ze vergemakkelijken namelijk het gebruik. Het volgende programma verduidelijkt de functie van de envelope-generator. De regels 10-80 uit het vorige programma kunt u overnemen.

```
100 A=9: D=9: S=8: R=9: H=400
110 POKE S1+5,16*A+D: POKE S1+6,16*S+R
120 POKE RS,0: POKE PL,15
130 POKE S1,37: POKE S1+1,17
140 POKE S1+4,33
150 FOR I=0 TO H: NEXT
160 POKE S1+4,32
```

De waarden in regel 100 kunnen worden veranderd, zodat u gevoel krijgt voor de invloed die de afzonderlijke parameters op de toon uitoefenen. De waarden voor A, D, S en R mogen alleen tussen 0 en 15 liggen, anders begint u een illegal quantity error. A bepaalt de attack-tijd en D de decay-tijd, die nodig is om het sustain-niveau te bereiken. H bepaalt hoe lang een toon op het

sustain-niveau, dat door S wordt bepaald, blijft en R bepaalt de decay-tijd die nodig is om naar nul terug te keren, nadat het KEY-bit is teruggezet. Als de waarde voor R tenminste 1 is, moet u de toon uitschakelen door de waarde 0 in register 4. Zo staat het weliswaar in het CBM-handboek (in de programmavoorbeelden op de bladzijden 80-90), maar dat mag u nu juist niet doen. De toon breekt dan namelijk abrupt af en de parameter R heeft dan geen invloed. U kunt veel beter bit 0 doelbewust terugzetten door het register nog een keer met de oorspronkelijke waarde te laden, maar dan zonder bit 0. De waarde moet daarom een even getal zijn.

Nu een drieklank op een spinet. De regels 10-80 uit het eerste programma kunt u weer gewoon overnemen.

```

100 A=0: D=1: S=13: R=10: H=100
110 POKE S1+5,16*A+D: POKE S1+6,16*S+R
120 POKE S2+5,16*A+D: POKE S2+6,16*S+R
130 POKE S3+5,16*A+D: POKE S3+6,16*S+R
140 POKE RS,0: POKE PL,15
150 POKE S1,37: POKE S1+1,17
160 POKE S2,154: POKE S2+1,21
170 POKE S3,177: POKE S3+1,25
180 POKE S1+4,33: POKE S2+4,33: POKE S3+4,33
190 FOR I=0 TO H: NEXT
200 POKE S1+4,32: POKE S2+4,32: POKE S3+4,32

```

In het volgende voorbeeld wordt de frequentie van de toon veranderd afhankelijk van het verloop van de omhullende. Stem 3 wordt daarvoor gebruikt, omdat de omhullende alleen van die stem uitgelezen kan worden. Experimenteer ook eens hier met de waarden in regel 100.

```

100 A=9: D=9: S=9: R=9: H=30
110 POKE RS,0: POKE PL,15
120 POKE S3+5,16*A+D: POKE S3+6,16*S+R
130 POKE S3+4,33
140 FOR I=0 TO H: POKE S3+1,PEEK(54300): NEXT
150 POKE S3+4,32
160 FOR I=0 TO R*4: POKE S3+1,PEEK(54300): NEXT

```

Tot slot laten we het ruimteschip Enterprise de revue passeren:

```

100 A=15: D=0: S=8: R=13: H=3000
110 POKE RS,0: POKE PL,15
120 POKE S1,0: POKE S1+1,30
130 POKE S2,0: POKE S2+1,1
140 POKE S3,0: POKE S3+1,100
150 POKE S1+5,16*A+D: POKE S1+6,16*S+R
160 POKE S1+4,129: POKE S3+4,23

```

```
170 FOR I=0 TO H: NEXT  
180 POKE S1+4,128: POKE S3+4,16
```

Deze voorbeelden zijn zeker inspirerend genoeg om nu zelf eens iets op de synthesizer te proberen. Veel plezier!

2.3 SYNTHIMAT 64

Met de software-uitbreiding SYNTHIMAT 64 verandert uw computer in een echte synthesizer. Een synthesizer is een muziekinstrument, dat een precieze controle over de elektronische produktie van klanken mogelijk maakt. Hij heeft geen vastomlijnd karakter, zoals een gitaar of een fluit. Een synthesizer kan al dergelijke muziekinstrumenten nabootsen. Bovendien bestaat een synthesizer uit een aantal modulen, die ieder gedeelte van een toon kunnen beïnvloeden.

Programma's voor de synthesizer zijn er in twee soorten. Ten eerste is dat software die het standaard-BASIC ondersteunt. Daarvoor moet u echter kunnen programmeren. De tweede soort programmatuur wordt gestuurd door menu's die u keuzemogelijkheden geven waarmee u optimaal kunt inspelen op de mogelijkheden die de hardware u te bieden heeft. U krijgt door deze menu's en niet door een programmeertaal toegang tot de mogelijkheden van uw computer. Dat betekent, dat u in een dialoog met het programma uw computer kunt besturen zonder zelf over programmeerkennis te beschikken. Een goed voorbeeld van zo'n programma met een dialoogkarakter is SYNTHIMAT 64. Klank en volgorde van de tonen worden direkt via het toetsenbord gestuurd. De functies van SYNTHIMAT 64 zijn verdeeld in modulen. Alle modulen van SYNTHIMAT 64 hebben een andere kleur en zijn op het beeldscherm overzichtelijk geordend. Iedere ingestelde waarde is op die manier altijd af te lezen en te veranderen. Een instelling van de modulen verandert de klankkleur direkt. Als u van de voorbeeldinstellingen af wilt wijken, moet eerst de klankkleur worden ingevoerd. De uiteindelijke muziek wordt niet in de vorm van noten ingevoerd, maar u kunt net als bij een echte synthesizer op het toetsenbord spelen. SYNTHIMAT 64 is polyfoon. Door verschillende toetsen tegelijkertijd in te drukken ontstaan echte akkoorden. Zelfs voor ervaren musici is het spelen op het toetsenbord in het begin wat problematisch, maar na verloop van tijd bent u er zo aan gewend, dat u er niets meer van merkt. SYNTHIMAT 64 verdeelt het toetsenbord in tweeën: een gedeelte voor

solo's en een voor de begeleiding, waarop alle toongeneratoren naar keuze kunnen worden geschakeld. Zo kunt u bijvoorbeeld een pianosolo met een basloopje begeleiden.

Behalve de basismodulen heeft SYNTHIMAT 64 nog 8 laagfrequente golfgeneratoren. Daarmee kan bijvoorbeeld het volume worden gemoduleerd, zodat het TREMOLO-effect ontstaat. Als de filterfrequentie gemoduleerd wordt, ontstaat een geluid dat lijkt op het kwaken van kikkers. Dat wordt het WAH-WAH effect genoemd. Bovendien kan de toonfrequentie van de oscillatoren en de pulsbreedte van de blokgolf met deze extra generatoren worden gemoduleerd, zodat allerlei merkwaardige geluiden kunnen worden gemaakt.

SYNTHIMAT 64 gebruikt de geheugencapaciteit van de Commodore 64 optimaal. Alle klanken die u heeft gemaakt, kunnen in registers worden weggezet en naar believen opgeroepen. Die kunt u op dan weer opslaan op diskettes om te voorkomen dat ze na het uitschakelen van de computer verloren gaan. Voor wie geen kennis van programmatuur en synthesizers heeft en toch zijn COMMODORE 64 in een muziekmachine wil omtoveren, is SYNTHIMAT 64 een uitkomst.

3. DE GRAFIEK

3.1 De videocontroller 6569 VIC

3.1.1 Algemene opmerking

VIC is een nieuwe IC uit de 65xx familie. Hij zorgt niet alleen voor het beeld en alles wat daarbij komt kijken, maar voert ook de timing voor een deel van het dynamisch geheugen uit. De processor wordt daardoor ontlast.

Het belangrijkste in het kort:

- * 16 kleuren
- * 40 tekens/25 regels
- * HIRESGrafiek met 320x200 punten
- * 5 soorten gebruik
- * gebruik van 8 sprites met elk 24x21 punten
- * produktie van een standaard PAL-signaal
- * zelfstandige hantering van 16K dynamische RAM
- * verschuifbare karaktergenerator
- * verschuifbare video-RAM

Pinbeschrijving van de VIC 6569

- 1-7 D6-D0; databus van de processor
- 8 -IRQ; 0 als een bit van de IMR en de IRR overeenkomen
- 9 -LP; ingang, lightpen strobe
- 10 -CS; acties op de processorbus vinden alleen plaats bij CS=0
- 11 R/-W; 0=data-overname van de bus
- 12 BA; =0 als data bij het lezen nog niet klaar staan
- 13 VDD; +12 V
- 14 colour; uitgang van de kleurinformatie
- 15 SYNC; impulsen voor regel- en beeldsynchronisatie
- 16 AEC; 0=VIC gebruikt systeembus, 1=bus vrij

- 17 OOUT; uitgang van de systeemklok
- 18 -RAS; stuursignaal voor het dynamische RAM
- 19 -CAS; als boven
- 20 GND
- 21 Ocolour; ingang van de kleurfrequentie
- 22 OIN; ingang van de dotfrequentie
- 23 A11; adresbus van de processor
- 24-29 A0/A8-A5/A13; gemultiplexte adresbus van de (video)RAM
- 30-31 A6-A7; adresbus van de (video)RAM-adresbus
- 32-34 A8-A10; adresbus van de processor
- 35-38 D11-8; data uit kleur-RAM
- 39 D7; databus van de processor
- 40 VCC; +5V

3.1.2 Registerbeschrijving van de VIC

De VIC beschikt over 47 registers.

- REG 0 sprite-register 0 X-coördinaten
 In dit register zijn 8 bits opgeslagen van de
 beeldschermcoördinaat X, waarop de sprite wordt
 afgebeeld. Het 9de bit zit in REG 16.
- REG 1 sprite-register 0 Y-coördinaten
 Als boven, maar dan voor de Y-richting. Dit register
 heeft geen overdracht.

De registers 2-15 zijn allemaal op deze manier opgebouwd en wel paarsgewijs: registerpaar 2/3 is voor sprite 1, paar 4/5 voor sprite 2 enzovoort.

- REG 16 MSB van de X-coördinaten
In dit register bevindt zich de overloop van de
sprite-X-registers: bit 0 voor sprite 0, bit 1 voor
sprite 1 enzovoort.
- REG 17 stuurregister 1
bit 0-2 offset van de afbeelding vanaf de bovenste
beeldrand in rasterregels
bit 3 0=24 regels, 1=25 regels
bit 4 0=beeldscherm uit
bit 5 1=standard bitmap mode
bit 6 1=extended color mode
bit 7 overdracht uit REG 18
- REG 18 Hier wordt het nummer van de rasterregel aangegeven,
waarbij een IRQ kan worden gestart, als de straal de
regel doorloopt.
overdracht van dit register in REG 17
- REG 19 X-aandeel van de beeldschermpositie, waar de straal
zich bevond toen een strobe werd gestart. (Pin LP=0)
- REG 20 Als boven, maar dan het Y-aandeel.
- REG 21 sprite enable
Aan iedere sprite is een bit toegewezen
1=sprite aan, 0=sprite uit
- REG 22 stuurregister 2
bit 0-2 offset van de afbeelding vanaf de linker
beeldrand in rasterpunten
bit 3 0=38 tekens, 1=40 tekens
bit 4 1=multi color mode
- REG 23 sprite expand X
Aan iedere sprite is een bit toegewezen
1=sprite wordt twee keer zo breed
- REG 24 basisadres van karaktergenerator en video-RAM
bit 1-3 adresbits 11-13 voor de tekenbasis
bit 4-7 adresbits 10-13 voor de basis van de video-RAM

De configuratie van dit register die in het CBM-handboek op
bladzijde 158 wordt gegeven, is fout!

- REG 25 IRR (Interrupt Request Register)
 bit 0 trigger is REG 18
 bit 1 trigger is REG 31
 bit 2 trigger is REG 30
 bit 3 trigger is pin LP
 bit 7 =1 als tenminste een van de andere bits =1
- REG 26 IMR (Interrupt Mask Register)
 Configuratie als boven. Als tenminste 1 bit uit IRR en
 1 uit IMR met elkaar overeen komen, wordt pin IRQ=0.
- REG 27 Aan iedere sprite is een bit toegewezen.
 1=achtergrondteken heeft een hogere prioriteit, dan de
 sprite.
- REG 28 Aan iedere sprite is een bit toegewezen
 1=sprite MC-modus (Multi Color mode)
- REG 29 sprite expand Y
 Aan iedere sprite is een bit toegewezen
 1=sprite wordt twee keer zo hoog
- REG 30 sprite-sprite samentreffen
 Aan iedere sprite is een bit toegewezen. Raakt een
 sprite een andere, dan worden de overeenkomstige
 bits=1. Daarna moet dit register worden gewist, omdat
 de bits zich niet zelf terugzetten.
- REG 31 sprite background samentreffen
 als boven, maar nu het als een sprite een
 achtergrondteken raakt.
- REG 32 exterior color (kleur van de omtrek)
- REG 33-36 background color 0-3 (achtergrondkleuren)
- REG 37-38 sprite multi color 0-1
- REG 39-46 color sprite 0 - sprite 7

3.1.3 De modes van de VIC

VIC heeft drie mogelijkheden om afbeeldingen te maken: karakters uit een generator, afbeelding in punten en in een afbeelding in sprites.

Sprites

De positie van sprites kan worden veranderd. Daarvoor dient een registerpaar. De resolutie van de spritebeweging is 512x256 punten. Door dit grote bereik kunnen sprites buiten het beeldscherm komen. De kleuren van de sprites kunnen op twee manieren worden weergegeven: normale resolutie met twee kleuren en halve resolutie met vier kleuren. In het eerste geval heeft de sprite een resolutie van 24x21 punten. De ene kleur komt uit het sprite color register, de andere is transparant. Dat betekent dat de kleur van de achtergrond dezelfde is. In het tweede geval (multi color mode; het bit in REG 28, dat met de sprite overeenkomt, moet gezet zijn) heeft de sprite slechts een resolutie van 12x21 punten. Iedere keer vormen twee bits van het sprite-bitpatroon een punt op het beeldscherm. Op de volgende manier bepalen ze de kleur van de punt:

00=transparant

01=multi color register 0

11=multi color register 1

10=sprite color

VIC moet weten, waar hij het bitpatroon vandaan moet halen. Daarom is er voor iedere sprite een sprite-pointer. Er zijn er acht en ze gebruiken de bovenste acht bytes in het video-RAM. De vermenigvuldiging van de waarde van een pointerbyte met 64 levert het reële geheugenadres op waar het bitpatroon moet zijn weggezet.

Normale afbeelding van tekens

Bij deze gebruiksmogelijkheid haalt VIC een bit uit het video-RAM, waarmee hij een positie in de karaktergenerator aanwijst. Het bitpatroon dat daar is weggezet, verschijnt tenslotte op het beeldscherm. Op deze manier kunnen 256 verschillende tekens worden afgebeeld. De kleurinformatie voor het teken staat in een bijzonder kleur-RAM. Voor iedere positie van het beeldscherm zijn er vier bits in het kleur-RAM die een van de 16 kleuren uitkiezen. Deze kleur geldt echter alleen voor de punten die in het tekenpatroon gezet zijn. De andere punten halen hun kleur uit

het achtergrondkleurregister 0. Daarom is voor alle tekens de kleur van de achtergrond hetzelfde.

**Normale afbeelding van tekens in de multi color mode:
(REG 22 bit 4=1)**

Bij deze gebruiksmogelijkheid worden eerst de vier bits van het kleur-RAM onderzocht. Is het MSB=0, dan wordt het teken in de normale 8x8 matrix afgebeeld. De kleur voor de 1-bits van het patroon wordt bepaald met de overige 3 bits van het kleur-RAM. De kleur voor de 0-bits komt weer uit het achtergrondkleurregister. Is het MSB van het kleur-RAM echter =1, dan vormen steeds 2 bits van het tekenpatroon een punt. Het teken wordt dan als 4x8 matrix afgebeeld met twee keer zo brede punten. De bitparen van het patroon bepalen dan de kleur van een punt:

00=achtergrondkleurregister 0
01=achtergrondkleurregister 1
10=achtergrondkleurregister 2
11=overige drie bits uit het kleur-RAM

In dit geval kan een teken vier kleuren tegelijkertijd hebben.

Extended color mode: (REG 17 bit 6=1)

Deze gebruiksmogelijkheid lijkt op de standaard-modus, omdat ook hier ieder teken uit slechts 2 kleuren kan bestaan. De kleur van de achtergrond is in tegenstelling tot de standaardmodus niet voor ieder teken gelijk. Voor de 1-bits van het patroon komt de kleur nog steeds uit het kleur-RAM. Maar voor de 0-bits wordt ze uit de beide MSB's van het video-RAM afgeleid:

00=achtergrondkleurregister 0
01=achtergrondkleurregister 1
10=achtergrondkleurregister 2
11=achtergrondkleurregister 3

Omdat slechts 6 bits als pointer op het tekenpatroon overblijven, bestaat de aan te spreken set nog maar uit 64 tekens.

Normale enkelpuntsmodus: (REG 17 bit 5=1)

Er bestaat hier een directe samenhang tussen geheugen en beeldscherm. Een bit in het geheugen komt met een punt op het scherm overeen. Hiervoor is een geheugen van 8K nodig. Het normale video-RAM wordt als kleur-RAM gebruikt. Het normale kleur-RAM heeft geen functie. De resolutie is 320x200 punten. Iedere byte van het video-RAM de kleurinformatie voor een groep van 8x8 punten. Dat gebeurt op zo'n manier, dat het meest significante nibble (=een halve byte oftewel 4 bits) voor de kleur van de 1-bits zorgt en het minst significante voor de kleur van de 0-bits.

Enkelpunts-modus met meer kleuren: (REG 17 bit 5=1 en REG 22 bit 4=1)

De resolutie is slechts 160x200 punten, omdat iedere twee bits van het geheugen met een twee keer zo brede punt op het beeldscherm corresponderen.

00=achtergrondkleurregister 0

01=meest significante nibble van het video-RAM

10=minst significante nibble van het video-RAM

11=kleur-RAM

De mogelijkheden om video-RAM en karaktergenerator te verschuiven worden in de paragrafen 3.3 en 3.4 behandeld.

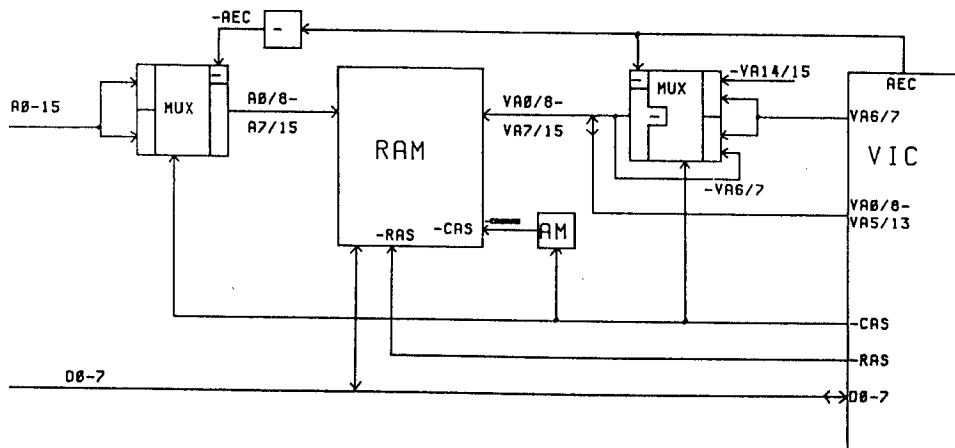
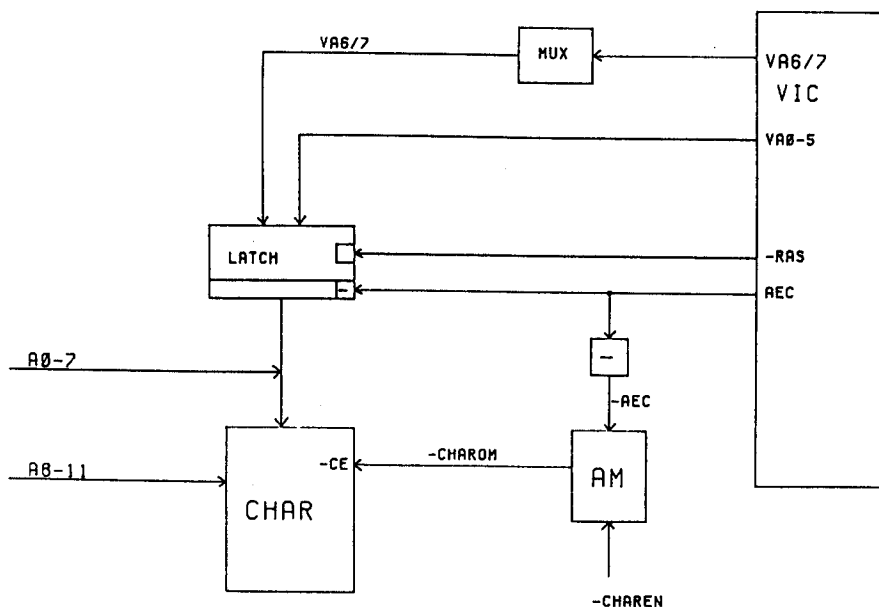
3.2 De aansluiting op de processor

VIC is als een normaal periferie-element op de processor aangesloten. Zonder beperking kan erin worden gelezen en geschreven. Toch zorgt hij ook nog voor de hele timing van de systeembus. De VIC maakt het gebruik van bepaalde elementen mogelijk of verhindert dat, bedient het RAM enzovoort. Dit is mogelijk, omdat de klokpulsen voor het systeem in de VIC worden geproduceerd. Daardoor weet hij altijd wanneer welke transacties kunnen plaatsvinden. Ook kan hij de besturing van de bus door de processor onderbreken. Dat is altijd nodig, als de VIC op

video-RAM, karaktergenerator of kleur-RAM moet aangrijpen. Door die cyclus verschijnt er doorlopend beeld op het scherm. Om de processor niet bij zijn werk te storen gebruikt hij de momenten, waarop de processor de bus niet nodig heeft. De afspraak is, dat de processor de bus gebruikt wanneer O2=1 en de VIC, wanneer O2=0. Om de bus vrij te maken zet de VIC de pin AEC op 0. Vervolgens brengt de processor de bus in de hoog-ohmige toestand (tri-state). Op die manier storen ze elkaar niet.

3.3 De aansluiting op het RAM

Het blokschema verduidelijkt deze paragraaf. De VIC zelf kan maar 16 RAM adreseren. Bij het verschuiven van video-RAM of karaktergenerator moet hij daarom de beide ontbrekende adresbits op een andere manier zien te krijgen. PA0-1 van CIA 2 stelt ze ter beschikking. Verandert men deze bits, dan verschuift het video-RAM direkt met stappen van 16K. Het zijn dus de adresbits 14 en 15 van het video-RAM. Op voorwaarde dat er genoeg vrij geheugen is, kunnen met deze bits verschillende beeldschermbladzijden worden omgeschakeld. Bedenk wel, dat deze bits L-actief zijn. Dat betekent dat om de onderste bladzijde van 16K te selekteren beide bits in de CIA 2 op 1 gezet moeten zijn. Voor de bovenste bladzijde moeten ze beide 0 zijn. De ene beeldschermbladzijde kan bijvoorbeeld in de HIRES-mode worden afgebeeld, terwijl een andere onzichtbaar alweer wordt geprepareerd. Met het VIC-register 24 kan het video-RAM in kleinere stappen van 1K worden verschoven. Hier corresponderen de bits 4-7 met de RAM-adressen 10-13.

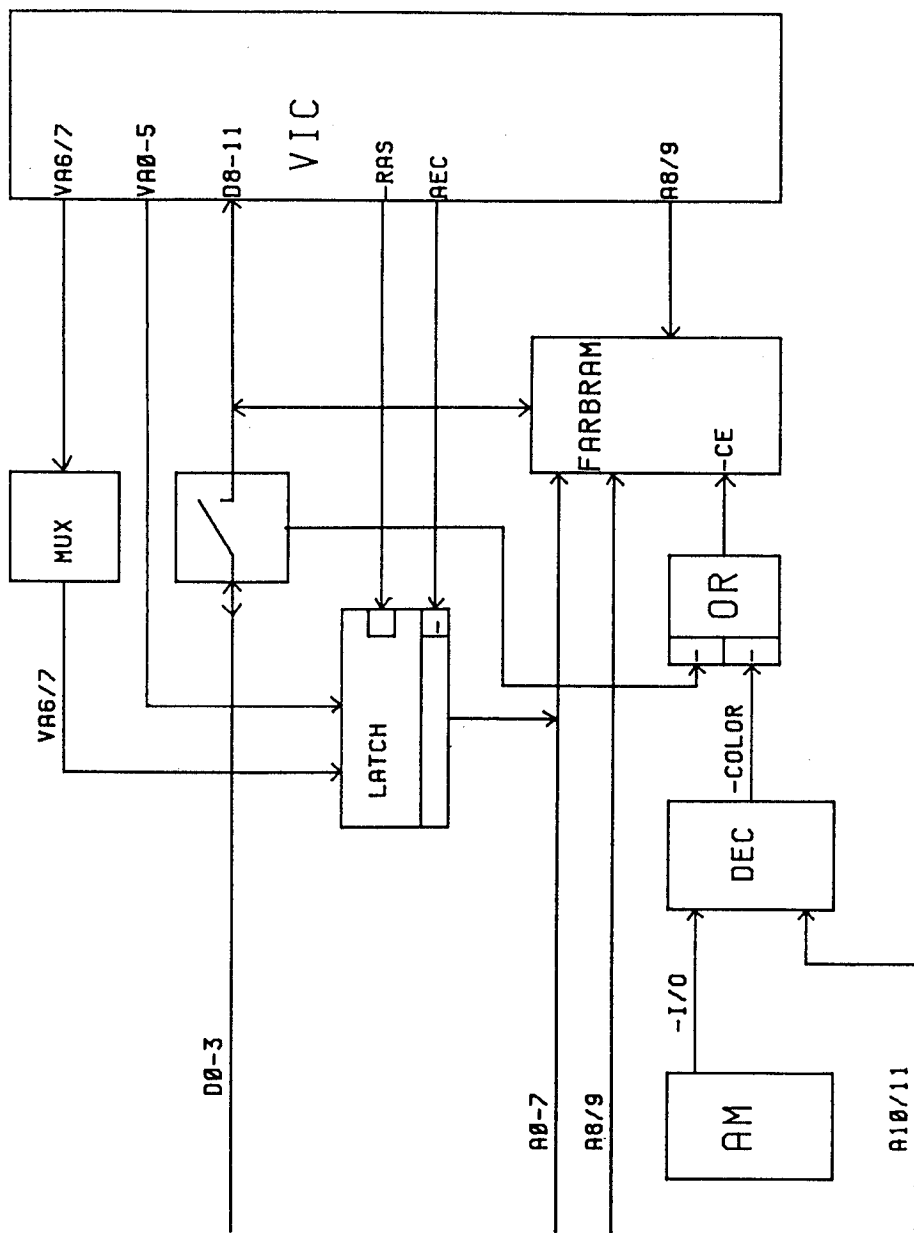


3.4 De aansluiting op de karaktergenerator

Het verloop is hier net zo als in de vorige paragraaf. De CIA-bits verschuiven tegelijk met het video-RAM ook de karaktergenerator in gelijke stappen. Hij kan met de bits 1-3 in REG 24 van de VIC in stappen van 2K worden verschoven. Deze bits worden gebruikt als adresbits 11-13. Dankzij de adresmanager heeft de karakter-ROM, die in de Commodore 64 is ingebouwd, een speciale status. De VIC gebruikt deze alleen als zijn relatieve adres tussen \$1000-\$1FFF of tussen \$9000-\$9FFF ligt. De karaktergenerator ligt in het adresbereik van de processor daadwerkelijk bij \$D000. Omdat het karakter-ROM niet in het gebruiksgebied van alle video-adressen ligt, moeten de ontbrekende bits in een bijzondere buffer worden opgeslagen. Het gaat om de adresbits 0-7. De basis van de karaktergenerator in REG 24 speelt nog een rol. Ze bepaalt namelijk niet alleen het beginadres van de generator, maar bit 3 bepaalt in de enkelbitsmodus ook de toestand van het beeldschermgeheugen. De bits 1 en 2 blijven dan buiten beschouwing.

3.5 De aansluiting op het kleur-RAM

In tegenstelling tot video-RAM en tekengenerator kan het kleur-RAM niet worden verschoven. De adressering is net als bij het karakter-ROM van twee kanten mogelijk: door de processor en door de VIC. Voor de adressering ontbreken ook hier weer de bits 0-7. Ze worden op dezelfde manier verkregen als bij het karakter-ROM. Het kleur-RAM heeft een eigen data-aansluiting voor de VIC. Door een schakelfunctie kan deze aansluiting van de databus van de processor worden afgetakt. Dat gaat allemaal heel eenvoudig. Bij AEC=0 staat de switch naar de VIC toe. Bij AEC=1 in de andere richting. Bovendien stuurt de adresmanager het schrijven en lezen van de processor in de kleur-RAM. Het blokschema op de volgende bladzijde maakt dit aanschouwelijk.



3.6 Programmering van kleur en grafiek

De mogelijkheden die de grafiek van de Commodore 64 biedt, zijn groter dan die van vergelijkbare machines of van de VIC 20. In de hardware van de Commodore 64 is de mogelijkheid ingebakken om ten hoogste acht vrij te kiezen sprites op het beeldscherm te bewegen. Ze kunnen allemaal een andere kleur hebben. In totaal staan er 16 kleuren ter beschikking. Bovendien zijn MultiColor (MC) sprites mogelijk die uit drie kleuren kunnen bestaan. Een sprite bestaat uit een matrix van 24x21 punten, te vergelijken met een blok van 3x3 letters. Altijd kan worden gecontroleerd of er sprites op het beeldscherm op elkaar zijn gebotst en waar ze zich bevinden. Op de programmering komen we later nog terug. Eerst is de programmering van kleur en grafiek aan de orde.

Grafiek en kleur

Op de Commodore 64 kunt u uit 16 kleuren kiezen. Op de achter- en voorgrond kunnen ze naar believen worden gewijzigd. De kleur van de karakters kan ook via het toetsenbord worden veranderd. Door tegelijkertijd de CTRL-toets of de C=toets (het CBM-symbool, links onder op het toetsenbord) en een toets tussen 1 en 8 in te drukken kunnen de karakters de volgende kleuren aannemen.

toets	kleur	getal
-----	-----	-----
CRTL - 1	ZWART	0
CRTL - 2	WIT	1
CRTL - 3	ROOD	2
CRTL - 4	CYAAN	3
CRTL - 5	VIOLET	4
CRTL - 6	GROEN	5
CRTL - 7	BLAUW	6
CRTL - 8	GEEL	7
C= - 1	ORANJE	8
C= - 2	BRUIN	9
C= - 3	LICHT ROOD	10
C= - 4	GRIJS 1	11
C= - 5	GRIJS 2	12
C= - 6	LICHTGROEN	13
C= - 7	LICHTBLAUW	14

Het kan voorkomen, dat het schrift op uw beeldscherm moeilijk of helemaal niet leesbaar is. Dat ligt aan het contrast tussen de kleur van de karakters en de voor- en achtergrond. Een groen letterbeeld doet het bijvoorbeeld goed tegen een zwarte voor- en achtergrond. De voor- en achtergrond kunnen worden gewijzigd door invoer van de kleurwaarden op de geheugenplaatsen 53280 en 53281. Op deze adressen staan altijd de actuele kleuren van het beeldscherm.

POKE 53280,0

POKE 53281,0

levert een volledig zwart beeldscherm op. Het schrift behoudt dan zijn normale kleur. Om de kleur van een beeldpunt te wijzigen is

POKE 55296,0

voldoende. Deze POKE-instructie maakt een teken in de linker bovenhoek zwart. Alle andere karakters behouden hun kleur. Dit adres is het eerste in het kleur-RAM van de Commodore 64. In dit bijzondere RAM staan de kleuren van alle tekens die op het scherm zijn afgebeeld. Het hele kleur-RAM loopt van 55296 tot 56295. Er is dus een byte gereserveerd voor elk teken van de 1000 die kunnen worden afgebeeld. Het adres 55296 correspondeert met regel 1, plaats 1; nummer 55297 met regel 1, plaats 2 enzovoort. Op adres 56295 staat de informatie voor de laatste plaats (40) op de laatste regel (25). Behalve het geheugen voor de afbeelding van tekens is er nog een voor het beeldscherm. Daarin zijn de tekens opgeslagen zonder hun kleurkenmerken. Dit RAM-bereik loopt van 1024 tot 2023. Het volgende programma laat alleen maar zwarte A's op het beeldscherm verschijnen.

10 PRINT CHR\$(147);: REM beeldscherm wissen

20 FOR I=1024 TO 2023: REM beeldschermgeheugen

30 POKE 54272+I,0: REM kleurgeheugen (54272+1024=55296)

40 POKE I,1: REM 1=getal voo A, 2 voor B enzovoort.

50 NEXT: REM vul beeldscherm en kleur-RAM

60 GOTO 60: REM om beeldschermgrenzen niet te overschrijden

Om hier wat meer kleur in te brengen kunt u regel 30 wijzigen in:

30 POKE 54272+I,INT(RND(1)*16)

Ieder teken wordt nu in een toevallig gekozen kleur afgebeeld. Nog iets over de random-getallen. Niet iedereen is daarmee vertrouwd, hoewel deze functie relatief vaak nodig is. Vooral bij het programmeren van spelletjes kunt u eigenlijk niet buiten de

RND-functie. Het gebruik is heel eenvoudig. Wilt u bijvoorbeeld een randomgetal tussen 0 en 15 genereren om een toevallige kleur te produceren, dan is $(\text{INT}(\text{RND}(1)*16))$ voldoende. De mogelijkheden die de Commodore 64 op het gebied van de grafiek in zijn mars heeft, zijn hiermee nog lang niet uitgeput. Het gebruik van kleurgrafiek met hoge resolutie (HIRES) laat zich het beste aan de hand van een PLOT-programma illustreren. Dat staat aan het eind van dit hoofdstuk. Het laat het gebruik en het effect zien van enkele interessante grafiekadressen. Eerst nog een paar feiten over de programmering van de grafiek.

Het programmeren van cirkels of lijnen is met de Commodore 64 helaas niet zo makkelijk als met sommige andere computers. Er is bijvoorbeeld geen instructie om een rechte lijn van punt naar punt te trekken. Ook ontbreekt een instructie om een cirkel met een bepaalde radius op een gewenste plaats te laten verschijnen. Maar dat betekent niet, dat de Commodore 64 dat niet kan. Het moet hem alleen worden geleerd. Aan het eind van dit hoofdstuk vindt u een uitvoerig hulpprogramma om grafieken te maken. Het heet HIRES GRAFIEKHULP. Voor gecompliceerde grafieken is er SUPERGRAFIEK 64, dat gewoon in de handel verkrijgbaar is. Met onze GRAFIEKHULP kunt u alvast een beetje oefenen. Informatie over de betekenis van de registers in dit programma staat in de paragraaf 3.1.2.

```

10 REM sinus plotprogramma
20 V=53248: REM startadres van de grafiekprocessor
30 AD=8192: REM startadres van de HIRES bit map
40 POKE V+17,59: REM grafiek inschakelen
50 POKE V+24,24: REM idem
60 FOR I=1024 TO 2023: REM zet HIRES kleur-RAM
70 : POKE I,16: REM kleurgetal
80 NEXT I
90 FOR I=8192 TO 16383: REM HIRES bit map wissen
100 : POKE I,0
110 NEXT I
120 FOR X=0 TO 319: REM teken X-as
130 : Y=100: REM positie van X-as
140 : GOSUB 1000: REM oproepen van tekenroutine
150 NEXT X
160 FOR Y=0 TO 199: REM teken Y-as
170 : X=160: REM positie van Y-as
180 : GOSUB 1000: REM oproepen van tekenroutine
190 NEXT Y
200 X=0
210 FOR I=-3,14159265 TO 3.14159265 STEP 0.0196349541
220 : REM intervalgrenzen
230 : Y=100+99*SIN(i): REM functie
240 : GOSUB 1000
250 : X=X+1

```

```

260 NEXT I
270 GOTO 270: REM zorgt dat beeldscherm niet verandert
1000 OY=320*INT(Y/8)+(Y AND 7): REM berekening van punt
1010 OX=8*INT(X/8)
1020 MA=2^(7-(X AND 7))
1030 AV=AD+OY+OX
1040 POKE AV,PEEK(AV) OR MA: REM plotten van de punt
1050 RETURN

```

Als u de opbouw van het programma heeft begrepen, zult u binnenkort ook uw eigen wiskundige functies op deze manier kunnen afbeelden. Schrijf ook zelf eens een functieplotter - dus een programma, dat na invoer van de functie en de grenzen daarvan de tekening op het beeldscherm laat verschijnen of over de printer stuurt. Het sturen van grafieken van het beeldscherm naar de printer is niet eenvoudig. In dit boek staan twee programma's, waarmee een HARDCOPY kan worden gemaakt. De eerste versie, bestemd voor een hardcopy op EPSON printers, maakt deel uit van het programma HIRES GRAFIEKHULP. De andere, voor een eenvoudige teksthardcopy, staat in hoofdstuk 6.3.2 onder de naam HARDCPY. In het plotprogramma gebeurt het volgende. In de regels 20 en 30 worden eerst de beide startadressen gedefinieerd. Daarna schakelen de POKE-instructies

POKE V+17,59 en POKE V+24,24

de grafiek in, of beter gezegd: ze zorgen ervoor dat van de tekstmodus op de grafiekmodus wordt omgeschakeld. Dat is nodig omdat de computer er anders van uitgaat, dat u in de normale modus, de tekstmodus, wilt doorgaan. Beeld en grafiek worden in het daarvoor bestemde beeldschermgeheugen opgeslagen. De Commodore 64 kan natuurlijk slechts een van beide geheugens in een keer afbeelden. Als u dus een grafiek bewerkt in de tekstmodus, zou de computer helemaal niets laten zien. Het omgekeerde, dus produktie van tekst in de grafiekmodus, leidt tot merkwaardige resultaten. Daarover straks meer. Als later weer op de tekstbladzijde moet worden omgeschakeld, moeten de beide adressen (V+17, V+24) op hun oude waarde terug worden gezet. Om dat te bewerkstelligen kan men het beste voorafgaand aan een wijziging de beide waarden in 2 variabelen opslaan. Dat kan er zo uitzien:

```

A1=PEEK(V+17)
en
A2=PEEK(V+24)

```

Om na bewerking van de grafiek weer in de tekstmodus te komen, moeten de opgeslagen variabelen (in ons voorbeeld A1 en A2) weer terug worden geschreven naar de adressen V+17 en V+24.

POKE V+17,A1
en
POKE V+24,A2

zorgen dat dat gebeurt. Nu bent u weer op de tekstbladzijde; er kan normaal verder worden gegaan. Nu moet de grafiekbladzijde de juiste kleur krijgen. Iedere byte moet worden gewijzigd. Dat gaat het beste in een FOR ... NEXT-lus. De eerste lus stelt de gewenste kleur van het grafiekbeeldscherm in, terwijl de tweede de geheugenpagina van het beeldscherm waarop de punten staan, volledig wist. Meteen maar wat informatie over de punten: iedere punt correspondeert met een bit. Een bit kan 0 of 1 zijn. 1 betekent dat op die plaats een punt is gezet en 0 dat dat niet gebeurd is. 8 punten vormen een byte en iedere lijn telt 40 bytes. Dat staat gelijk met $8 \times 40 = 320$ punten per lijn. Met de regels gaat dat net zo. Iedere regel is 8 lijnen dik. Bij een totaal van 25 regels is de resolutie dus $8 \times 25 = 200$ lijnen. In totaal staan er dus $200 \text{ lijnen} \times 320 \text{ punten} = 64.000$ punten ter beschikking. Nu gaan we verder. Er gaan $8 \times 8 = 64$ punten in een teken. Allemaal zijn ze afzonderlijk te zetten of te wissen. Alleen bij kleuren zijn er problemen. In het kleur-RAM worden de punten niet apart bekeken, maar is de hele 8×8 -matrix het onderwerp. Als men per punt een kleur wil kiezen, is alleen voor het kleur-RAM al een geheugen van 64KB RAM nodig. Dat is teveel verlangd. De 64 punten van een karakter moeten daarom dezelfde kleur hebben, hetgeen ook de afbeelding op het scherm ten goede komt. Een grotere oplossing van de kleuren zou een heel wazig beeld tot gevolg hebben. De informatie ziet er dus zo uit.

Een teken

```
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .  
. . . . .
```

telt 64 punten (van dezelfde kleur).

Dit karakter kan bijvoorbeeld op geheugenplaats 1024 een letter zijn of op dezelfde geheugenplaats de positie van de overeenkomstige punten in de HIRES-modus. Dat klinkt moeilijker dan het is. Zonder een hoop extra te programmeren is het niet mogelijk tekst en grafiek door elkaar te gebruiken. Dat is inderdaad een nadeel, maar aan de andere kant heeft geen enkele vergelijkbare computer deze faciliteit. Als dat mogelijk was, hoefde de computer ook geen gescheiden geheugen voor tekst en

grafiek te hebben. Zulke grafiekcomputers zijn veel en veel duurder dan een personal computer. Maar er is een klein trucje, waardoor schrifttekens in grafieken kunnen worden verwerkt. Een teken uit de tekentabel kan worden ingelezen (zie het voorbeeldprogramma in paragraaf 2.4) en in een grafisch symbool worden omgezet. Laten we aannemen dat op geheugenplaats 1024 inderdaad een teken staat. Tegelijkertijd bevindt zich daar ook de kleur van het grafische symbool, dat immers precies zo groot is als een normale letter (8 x 8). Als we dus eenmaal op de grafiekbladzijde zijn en we keren daarna naar de tekstpagina terug zonder eerst op de grafiekmodus te zijn overgegaan, zien we in plaats van een letter alleen een gekleurd hokje. Bij een foutmelding zou dus dat onleesbare gekleurde hokje verschijnen. Schakelen we terug naar de tekstmodus, dan kan er weer tekst worden gelezen, maar de grafiek verdwijnt. Deze tabel van de verschillende geheugengebieden geeft misschien een goed inzicht.

MODUS	BEELDSCHERM- RAM	KLEUR-RAM
Tekst	1024- 2023	55296-56295
Grafiek	8192-16383	1024- 2023

Wat in de tekstmodus de positie van het teken is, is in de grafiekmodus de kleur van het hokje. Daarom kunnen tekst en grafiek niet zonder problemen door elkaar worden gebruikt. Een ander probleem is het genereren van de punten. De 8x8-matrix die voor ieder teken geldt, dus ook voor een grafisch symbool, wordt namelijk op de volgende manier gevuld. De eerste punt wordt in de bovenste rij van ieder blok gezet. Een POKE-instructie stuurt de positie van de punt in deze rij. De volgende punt komt in de tweede rij van hetzelfde blok enzovoort. Na 8 rijen wordt er van blok gewisseld. Het volgende staat rechts naast het eerste.

8192->.....	8200->.....
8193->.....	8201->.....
8194->.....	
8195->.....	enzovoort
8196->.....	
8197->.....	
8198->.....	
8199->.....	

Nu eerst wat binair rekenen. De positie van de punt binnen de geheugenplaats is afhankelijk van een POKE-instructie. Die bestaat uit twee helften: de ene geeft het adres, de andere de waarde. Om in de eerste regel van het eerste blok op het beeldscherm een punt te laten verschijnen moet het eerste deel in ieder geval POKE 8192

zijn. En hier beginnen de moeilijkheden. Met deze POKE kunnen maximaal 8 punten tegelijkertijd worden gezet. De positie van de punt of punten moet dus duidelijk zijn. Als het patroon klaar is, moet het zijn binaire vorm nog krijgen.

VOORBEELD:

De linker en rechter punt op geheugenplaats 8192 moeten worden ingeschakeld. Dat betekent, dat de linker (b7) en rechter bit (b0) moeten worden gezet:

1 0 0 0 0 0 1

POKE 8192,10000001 kan niet. Het binaire getal moet in zijn decimale equivalent vertaald worden. Bit7 correspondeert met $1*2^7$ (=128) en bit0 betekent $1*2^0$ (=1). Het resultaat is dus 129. Dat wordt POKE 8192,129. Wanneer de grafiekmodus was ingeschakeld, staan er in de eerste regel nu 2 punten. Let bij grafische programmering steeds op deze eigenaardigheid van de Commodore 64, anders kunt u voor rare verrassingen komen te staan. Tot slot nog een opmerking over SUPERGRAFIEK 64. Dit snelle machinetaalprogramma neemt geen geheugenruimte voor BASIC in beslag. Het heeft in aangepaste vorm alle instructies van het bekende Supergraphic Board en bovendien nog wat speciale instructies.

SUPERGRAFIEK 64

SUPERGRAFIEK 64 is een handige instructie-uitbreiding voor de Commodore 64. Hiermee kan de HIRES kleurgrafiek in BASIC worden geprogrammeerd. Hieronder volgen de instructies uit dit programma.

PLOT x,y - punt zetten bij de coördinaten x,y

PLOT x1,y1 TO x2,y2
 - lijn van x1,y1 naar x2,y2

PLOT TO x2,y2 - lijn van grafiekcursor naar x2,y2

CIRCLE - cirkels, ellipsen en bogen

PAINT var FROM x,y
 - figuur vanaf x,y

FRAME d,x1,y1 TO x2,y2
 - kader van de dikte d

FILL x1,y1 TO x1,y2
 - veld

TEXT str,x,y,m - tekst in grafiek

GMODE - grafiekbladzijde en modus

GCLEAR - grafiek wissen

GMOVE - grafiek verschuiven

INVERS - grafiek inverteren

ROT= - PAINT-figuur draaien

SIZE= - PAINT-figuur vergroten

FCOL - kleur van kader

BCOL - kleur van achtergrond

SCOL - kleur van tekens

PCOL - kleur van punt

GSAVE - grafiek opslaan

GLOAD - grafiek laden

HCOPY - afdruk

SREAD - 63 bytes lezen

SDEFINE - definiëren

SMODE - spritemodus

SSET (TO) - sprite bewegen

IF# par THEN u,d,r,l,b,cn,c,cc
 - vertakking

IRETURN - onderbreking beëindigen

VOLUME - volume

SOUND - klank
 FILTER - filtermodus
 TUNE - toon spelen

SUPERGRAFIEK 64 wordt vanaf diskette geladen en kan dan direkt worden gebruikt.

Het programma HIRES GRAFIEKHULP vereenvoudigt het programmeren van grafiek. Hier volgt de assemblerlisting.

```

C000      CR          = 13      ; CARRIAGE RETURN
C000      ESC         = 27      ; ESCAPE
C000      XCOORD      = $14
C000      FLAG        = $97     ; PUNT ZETTEN/WISSEN
C000      MASKER      = flag    ; MASKER VOOR HARDCOPY
C000      SA          = $B9     ; SECONDAIR ADRES
C000      TMP         = $FD
C000      ADR         = TMP
C000      AV          = TMP
C000      CODE        = XCOORD
C000      KOLOM       = CODE+1
C000      COLLO       = $400    ; BEGIN HIRES KLEUR-RAM
C000      COLHI       = $800    ; EINDE HIRES KLEUR-RAM
C000      GRALO       = $2000   ; BEGIN HIRES BITMAP
C000      GRAHI       = $4000   ; EINDE HIRES BITMAP
C000      GETCOR      = $B7EB   ; HAALT X- EN Y-COÖRDINATEN
C000      CHKCOM      = $AEFD   ; KIJKT OF ER EEN KOMMA STAAT
C000      GETBYT      = $B79E   ; HAALT BYTE-WAARDE
C000      VIDEO       = $D000   ; VIDEOCONTROLLER
C000      GETPAR      = $ELD4   ; HAALT FIENAMEN EN NUMMER VAN APPARATUUR
C000      CLRSCR      = $E544   ; WIST BEELDSCHERM
C000      CHKOUT      = $FFC9   ; STELT UITVOERAPPARAAT IN
C000      CLRCH       = $FFCC   ; UITVOER OP DEFAULT
C000      PRINT       = $FFD2   ; UITVOER ROUTINE
C000      LOAD        = $FFD5   ; LOAD ROUTINE
C000      SAVE        = $FFD8   ; SAVE ROUTINE
;
C000      *= $C000      ; SPRONGTABEL VOOR FUNCTIES
;
C000 4C 1E C0      JMP INIT      ; GRAFIEKMODUS INSCHAKELEN
C003 4C 3C C0      JMP CLEAR     ; GRAFIEK WISSEN
C006 4C 51 C0      JMP COLOR     ; KLEUR INSTELLEN
C009 4C 6D C0      JMP REVERS    ; GRAFIEK INVERTEREN
C00C 4C 89 C0      JMP SET       ; GRAFIEKPUNT ZETTEN
C00F 4C 86 C0      JMP RESET     ; GRAFIEKPUNT WISSEN
C012 4C 52 C1      JMP GLOAD     ; GRAFIEK LADEN

```

C015	4C	39	C1	JMP	GSAVE	; GRAFIEK OPSLAAN
C018	4C	75	C1	JMP	HARD	; HARDCOPY
C01B	4C	61	C1	JMP	GOFF	; GRAFIEK UIT
C01E	20	3C	CO	JSR	CLEAR	
C021	AD	11	DO	LDA	VIDEO+17	; GRAFIEK INSCHAKELEN
C024	8D	70	C1	STA	STORE1	
C027	AD	18	DO	LDA	VIDEO+24	
C02A	8D	71	C1	STA	STORE2	
C02D	A9	3B		LDA	#27+32	
C02F	8D	11	DO	STA	VIDEO+17	
C032	A9	18		LDA	#16+8	
C034	8D	18	DO	STA	VIDEO+24	
C037	A2	10		LDX	#\$10	; PUNKLEUR WIT
C039	4C	57	CO	JMP	COL	; ACHTERGRONDKLEUR ZWART
C03C	A0	00	CLEAR	LDY	#\$00	; GRAFIEKGEHEUGEN WISSEN
C03E	A2	20		LDX	#>GRALO	
C040	84	FD		STY	TMP	
C042	86	FE		STX	TMP+1	
C044	98			TYA		
C045	EA			NOP		
C046	91	FD	CLR	STA	(TMP),Y	
C048	C8			INY		
C049	D0	FB		BNE	CLR	
C04B	E6	FE		INC	TMP+1	
C04D	CA			DEX		; VOLGENDE PAGINA
C04E	D0	F6		BNE	CLR	
C050	60			RTS		
C051	20	FD	AE COLOR	JSR	CHKCOM	; KLEUR INSTELLEN
C054	20	9E	B7	JSR	GETBYT	; KLEURCODE HALEN
C057	A0	00	COL	LDY	#\$00	
C059	A9	04		LDA	#>COLLO	
C05B	84	FD		STY	TMP	
C05D	85	FE		STA	TMP+1	
C05F	8A			TXA		; KLEURCODE
C060	A2	04		LDX	#>COLHI-COLLO	; AANTAL PAGINA'S
C062	91	FD	COL1	STA	(TMP),Y	
C064	C8			INY		
C065	D0	FB		BNE	COL1	
C067	E6	FE		INC	TMP+1	
C069	CA			DEX		; VOLGENDE PAGINA
C06A	D0	F6		BNE	COL1	
C06C	60			RTS		
C06D	A0	00	REVERS	LDY	#\$00	; GRAFIEK INVERTEREN
C06F	A9	20		LDA	#>GRALO	
C071	84	FD		STY	TMP	
C073	85	FE		STA	TMP+1	
C075	A2	20		LDX	#>GRAHI-GRALO	; AANTAL PAGINA'S
C077	B1	FD	REV1	LDA	(TMP),Y	
C079	49	FF		EOR	#\$FF	; ALLE BITS OMDRAAIEN

```

C07B 91 FD          STA (TMP),Y
C07D C8            INY
C07E D0 F7          BNE REV1
C080 E6 FE          INC TMP+1
C082 CA            DEX          ; VOLGENDE PAGINA
C083 D0 F2          BNE REV1
C085 60            ILL          ; VERKEERDE COORD. BIJ SET
C086 A9 80 RESET    LDA #$80    ; GRAFIEKPUNT WISSEN
C088 2C            .BIT $2C
C089 A9 00          SET          ; GRAFIEKPUNT ZETTEN
C08B 85 97          STA FLAG
C08D 20 FD AE       JSR CHKCOM   ; KOMMA
C090 20 EB B7       JSR GETCOR   ; X NAAR X-COORD. Y NAAR X-REG.
C093 E0 C8          CPX #200
C095 B0 EE          BCS ILL      ; Y-COORD. > 199?
C097 A5 15          LDA XCOORD+1
C099 C9 01          CMP #>320
C09B 90 08          BCC OK
C09D D0 E6          BNE ILL
C09F A5 14          LDA XCOORD
COA1 C9 40          CMP #<320    ; X-COORD. >319?
COA3 B0 E0          BCS ILL
COA5 8A            OK          TXA          ; Y-COORD. IN ACCU
COA6 4A            LSR
COA7 4A            LSR
COA8 4A            LSR
COA9 0A            ASL          ; DOOR 8*2
COAA A8            TAY
                ; OFFY=320*INT(Y/8)+(Y AND 7)
COAB B9 FF C0      LDA MULT,Y
COAE 8D 73 C1      STA OFFY
COB1 B9 00 C1      LDA MULT+1,Y  ; KEER 320
COB4 8D 74 C1      STA OFFY+1
COB7 8A            TXA          ; Y-COORD.
COB8 29 07          AND #$07
COBA 18            CLC
COBB 6D 73 C1      ADC OFFY
COBE 8D 73 C1      STA OFFY
                ; OFFX=8*INT(X/8)
COC1 A5 14          LDA XCOORD
COC3 29 F8          AND #%11111000
COC5 8D 72 C1      STA OFFX
COC8 18            CLC          ; AV=GRALO+OFFY+OFFX
COC9 A9 00          LDA #<GRALO
COCB 6D 73 C1      ADC OFFY
COCE 85 FD          STA AV
COD0 A9 20          LDA #>GRALO
COD2 6D 74 C1      ADC OFFY1
COD5 85 FE          STA AV+1

```

```

COD7 18          CLC
COD8 A5 FD       LDA AV
CODA 6D 72 C1    ADC OFFX
Codd 85 FD       STA AV
CODF A5 FE       LDA AV+1
COE1 65 15       ADC XCOORD+1
COE3 85 FE       STA AV+1
                ; MA=2^(7-(X AND 7)).
COE5 A5 14       LDA XCOORD
COE7 29 07       AND #$07
COE9 49 07       EOR #$07
COEB AA         TAX
COEC BD 31 C1    LDA GRBIT,X      ; TOT DE TWEEDE MACHT UIT TABEL
COEF A0 00       LDY #$00
COF1 24 97       BIT FLAG
COF3 10 05       BPL SET1
COF5 49 FF       EOR #$FF
COF7 31 FD       AND (AV),Y      ; PUNT WISSEN
COF9 2C         .BYT $2C
COFA 11 FD SET1  ORA (AV),Y      ; PUNT ZETTEN
COFC 91 FD       STA (AV),Y
COFE 60          RTS
;
                ; VERMENIGVULDIGINGSTABEL N*320, N=0,24
COFF          MULT = *
COFF 00 00 40 01 80 02 C0 03
C107 00 05 40 06 80 07 C0 08
C10F 00 0A 40 0B 80 0C C0 0D
C117 00 0F 40 10 80 11 C0 12
C11F 00 14 40 15 80 16 C0 17
C127 00 19 40 1A 80 1B C0 1C
C12F 00 1E
;
C131 01 02 04 GRBIT .BYT 1,2,4,8,$10,$20,$40,$80
C134 08 10 20
C137 40 80
;
C139 20 FD AE     JSR CHKCOM      ; GRAFIEK OPSLAAN
C13C 20 D4 E1     JSR GETPAR      ; FILENAAM EN DEVICENUMMER HALEN
C13F A2 00        LDX #<GRAHI
C141 A0 40        LDY #>GRAHI
C143 A9 00        LDA #<GGRALO
C145 85 FD        STA TMP
C147 A9 20        LDA #>GRALO
C149 85 FE        STA TMP+1
C14B A9 FD        LDA #TMP
C14D 85 B9        STA SA          ; ABSOLUUT OPSLAAN
C14F 4C D8 FF     JMP SAVE
C152 20 FD AE GLOAD JSR CHKCOM    ; GRAFIEK LADEN

```

```

C155 20 D4 E1      JSR GETPAR      ; FILENAAM EN DEVICENR. HALEN
C158 A9 01        LDA #1          ; ABSOLUUT LADEN
C15A 85 B9        STA SA
C15C A9 00        LDA #0          ; LOAD FLAG
C15E 4C D5 FF     JMP LOAD
C161 AD 70 C1 GOFF LDA STORE1      ; GRAFIEK UIT
C164 8D 11 D0     STA VIDEO+17
C167 AD 71 C1     LDA STORE2
C16A 8D 18 D0     STA VIDEO+24
C16D 4C 44 E5     JMP CLRSCR      ; BEELDSCHERM WISSEN
C171             STORE1 *= *+1
C172             STORE2 *= *+1
C173             OFFX  *= *+1
C175             OFFY  *= *+2
                    ; HARDCOPY VOOR EPSON FX80 MET VC-INTERFACE
;
C175 20 FD AE HARD JSR CHKCOM      ; KOMMA
C178 20 9E B7     JSR GETBYT      ; LOG. FILENUMMER
C17B 20 C9 FF     JSR CHKOUT      ; UITVOER OP PRINTER
C17E A9 00        LDA #<GRALO
C180 A0 20        LDY #>GRALO
C182 85 FD        STA ADR          ; WIJZER OP GRAFIEK
C184 84 FE        STY ADR+1
C186 A2 19        LDX #25          ; AANTAL REGELS
C188 A0 07        LDY #$07
C18A 2C           .BYT $2C
C18B A0 05 REGELS LDY #$05
C18D B9 D6 C1 GMD LDA GMD,Y
C190 20 D2 FF     JSR PRINT      ; PRINTER OP GRAF.MODUS
C193 88           DEY
C194 10 F7        BPL GMD
C196 A9 28        LDA #40
C198 85 15        STA KOLOM
C19A A9 80 KOLOMMEN LDA #$80
C19C 85 97        STA MASKER
C19E A9 00 BYTES  LDA #$00
C1A0 85 14        STA CODE
C1A2 A0 07        LDY #$07
C1A4 B1 FD BITS  LDA (ADR),Y      ; BITPATROON SAMENSTELLEN
C1A6 25 97        AND MASKER
C1A8 F0 07        BEQ TT2        ; BIT NIET GEZET
C1AA A5 14        LDA CODE
C1AC 19 DE C1     ORA GBIT,Y      ; BIT GEZET
C1AF 85 14        STA CODE
C1B1 88           TT2 DEY
C1B2 D0 F0        BNE
C1B2 10 F0        BPL BITS
C1B4 A5 14        LDA CODE
C1B6 20 D2 FF     JSR PRINT      ; CODE VOOR PRINTER

```

```

C1B9 46 97          LSR MASKER
C1BB 90 E1          BCC BYTES
C1BD A5 FD          LDA ADR
C1BF 69 07          ADC #$07          ; +CARRY=8
C1C1 85 FD          STA ADR          ; ADRES VERHOGEN
C1C3 90 02          BCC TT3
C1C5 E6 FE          INC ADR+1
C1C7 C6 15      TT3  DEC KOLOM          ; VOLGENDE KOLOM
C1C9 D0 CF          BNE KOLOMMEN
C1CB CA            DEX          ; VOLGENDE REGEL
C1CC D0 BD          BNE REGELS
C1CE A9 0D          LDA #CR
C1D0 20 D2 FF      JSR PRINT
C1D3 4C CC FF      JMP CLRCH          ; UITVOER WEER OP BEELDSCHERM
C1D6 01 40      GMOD .BYT >320,<320 ; AANTAL GRAFIEKPUNTEN
C1D8 06 2A 1B      .BYT 6,"*".ESC,CR ; GRAFIEKMODUS
C1DB 0D
C1DC 31 1B          .BYT "L",ESC ; 8 PUNTEN PER REGEL

C1DE 80 40 20 GBIT .BYT $80,$40,$20,$10,8,4,2,1

C1E1 10 08 04
C1E4 02 01
C1E6          .END

```

Er volgt weer een laadprogramma in BASIC.

```

100 FOR I = 49152 TO 49637
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 76, 30,192, 76, 60,192, 76,
      81,192, 76,109,192
130 DATA 76,137,192, 76,134,192, 76,
      82,193, 76, 57,193
140 DATA 76,117,193, 76, 97,193, 32,
      60,192,173, 17,208
150 DATA 141,112,193,173, 24,208,141,
      113,193,169, 59,141
160 DATA 17,208,169, 24,141, 24,208,
      162, 16, 76, 87,192
170 DATA 160, 0,162, 32,132,253,134,
      254,152,234,145,253
180 DATA 200,208,251,230,254,202,208,
      246, 96, 32,253,174
190 DATA 32,158,183,160, 0,169, 4,
      132,253,133,254,138
200 DATA 162, 4,145,253,200,208,251,
      230,254,202,208,246
210 DATA 96,160, 0,169, 32,132,253,
      133,254,162, 32,177

```

220 DATA 253, 73,255,145,253,200,208,
 247,230,254,202,208
 230 DATA 242, 96,169,128, 44,169, 0,
 133,151, 32,253,174
 240 DATA 32,235,183,224,200,176,238,
 165, 21,201, 1,144
 250 DATA 8,208,230,165, 20,201, 64,
 176,224,138, 74, 74
 260 DATA 74, 10,168,185,255,192,141,
 115,193,185, 0,193
 270 DATA 141,116,193,138, 41, 7, 24,
 109,115,193,141,115
 280 DATA 193,165, 20, 41,248,141,114,
 193, 24,169, 0,109
 290 DATA 115,193,133,253,169, 32,109,
 116,193,133,254, 24
 300 DATA 165,253,109,114,193,133,253,
 165,254,101, 21,133
 310 DATA 254,165, 20, 41, 7, 73, 7,
 170,189, 49,193,160
 320 DATA 0, 36,151, 16, 5, 73,255,
 49,253, 44, 17,253
 330 DATA 145,253, 96, 0, 0, 64, 1,
 128, 2,192, 3, 0
 340 DATA 5, 64, 6,128, 7,192, 8,
 0, 10, 64, 11,128
 350 DATA 12,192, 13, 0, 15, 64, 16,
 128, 17,192, 18, 0
 360 DATA 20, 64, 21,128, 22,192, 23,
 0, 25, 64, 26,128
 370 DATA 27,192, 28, 0, 30, 1, 2,
 4, 8, 16, 32, 64
 380 DATA 128, 32,253,174, 32,212,225,
 162, 0,160, 64,169
 390 DATA 0,133,253,169, 32,133,254,
 169,253,133,185, 76
 400 DATA 216,255, 32,253,174, 32,212,
 225,169, 1,133,185
 410 DATA 169, 0, 76,213,255,173,112,
 193,141, 17,208,173
 420 DATA 113,193,141, 24,208, 76, 68,
 229, 0, 0, 0, 0
 430 DATA 0, 32,253,174, 32,158,183,
 32,201,255,169, 0
 440 DATA 160, 32,133,253,132,254,162,
 25,160, 7, 44,160
 450 DATA 5,185,214,193, 32,210,255,
 136, 16,247,169, 40

```

460 DATA 133, 21,169,128,133,151,169,
      0,133, 20,160, 7
470 DATA 177,253, 37,151,240, 7,165,
      20, 25,222,193,133
480 DATA 20,136,208,240,165, 20, 32,
      210,255, 70,151,144
490 DATA 225,165,253,105, 7,133,253,
      144, 2,230,254,198
500 DATA 21,208,207,202,208,189,169,
      13, 32,210,255, 76
510 DATA 204,255, 1, 64, 6, 42, 27,
      13, 49, 27,128, 64
520 DATA 32, 16, 8, 4, 2, 1
530 IF S <> 60459 THEN PRINT "FOUT IN DATA !!" : END
540 PRINT "OK !"

```

Toelichting:

De volgende opmerkingen betreffen het gebruik van het hulpprogramma. De oproep van afzonderlijke functies gebeurt door SYS-instructies, waarin voor een deel parameters besloten liggen. Aan het begin van het programma worden de adressen van de routines doorgegeven aan variabelen die bij een latere oproep worden gebruikt. Daarna kunnen voor zover dat wenselijk is parameters door komma's van elkaar worden gescheiden. In het voorbeeld werden de volgende namen gebruikt:

X - horizontale coördinaten van een punt in de grafiek, 0-139
Y - verticale coördinaten, 0-199

De coördinaat 0,0 ligt in de linker bovenhoek.

PF - kleur van een punt in de grafiek, 0-15
HF - kleur van de achtergrond, 0-15
LF - logisch filenummer van de printer, 1-255

```

100 IN = 124096: REM SYS IN - grafiek inschakelen
110 CL = IN+3 : REM SYS CL - grafiek wissen
120 CO = IN+6 : REM SYS CO,PK*16+AK - kleur instellen
120 RV = IN+9 : REM SYS RV - grafiek inverteren
130 SE = IN+12: REM SYS SE,X,Y - grafiekpunt instellen
140 RS = IN+15: REM SYS RS,X,Y - grafiekpunt wissen
150 GL = IN+18: REM SYS GL,"NAAM",8 - grafiek laden
160 GS = IN+21: REM SYS GS,"NAAM",8 - grafiek opslaan
170 HC = IN+24: REM SYS HC,LF - hardcopy op printer
180 OF = IN+27: REM SYS OF - grafiek uitschakelen
200 SYS IN : REM grafiek in
210 PK = 1 : REM kleur van punt = wit

```



```

220 AK kleur van achtergrond = zwart
230 SYS CO,16*PK+AK : REM kleur instellen
240 REM tekenen van de X-as
250 FOR X=0 TO 319 : SYS SE,X,100 : NEXT
260 REM tekenen van de Y-as
270 FOR Y=0 TO 199 : SYS SE,160,Y : NEXT
280 REM tekenen van de sinus-kromme
290 PI = 3.14159265: X=0
300 FOR I=-PI TO PI STEP 2*PI/319
310 SYS SE,X,100+99*SIN(I)
320 X=X+1 : NEXT
330 REM grafiek opslaan op diskette
340 SYS GS,"sinuscurve",8
350 REM printer openen, epson-mode
360 OPEN 1,4,1
370 REM hardcopy op printer uitdraaien
380 SYS HC,1
390 REM printerfile weer sluiten en grafiek uitschakelen
400 CLOSE 1 : SYS OF : END

```

3.7 Sprites - het toverwoord van de Commodore 64

Sprites zijn kleine grafieken die onafhankelijk van elkaar kunnen worden gecontroleerd. Omdat de Commodore 64 ze op een ander manier regelt dan de normale grafiek, hoeft niet steeds naar de grafische bladzijde te worden omgeschakeld.

Op het scherm kunnen ten hoogste 8 sprites (0-7) worden afgebeeld. U hoeft alleen een geheugenbereik te kiezen, de sprite een nummer te geven en hem in of uit te schakelen. Door de X-Y-coördinaten aan te geven kan de sprite over het scherm bewegen. Beeldscherm-RAM of kleur-RAM worden automatisch gewijzigd door de 6569 video display chip. Sprites kunnen verticaal en horizontaal worden vergroot. Een botsing tussen sprites kunt u vaststellen en er vervolgens op reageren. U kunt een achtergrond kiezen en ook de prioriteit bepalen - sprites kunnen voor of achter de achtergrond verschijnen. Hierdoor is een ruimtelijke afbeelding van de sprites mogelijk. Met de Commodore 64 kunnen dus driedimensionale grafieken worden gemaakt. Deze overlapping van sprites maakt effecten mogelijk, waarvoor vroeger veel programmeerwerk nodig was.

Voor het gebruik van sprites moet u met het binaire rekenen en de registers van de VIC, de grafische processor, vertrouwd zijn (paragrafen 1.1 en 3.1.1). Op de volgende bladzijden gaan we ervan uit dat u die kennis paraat heeft. Bij het maken van sprites hebben de 46 registers de belangrijkste functie (paragraaf 3.1.1). Beweging, kleur en alle andere bijzonderheden worden aldus geregeld. Ieder register bestaat uit 8 bits die men naar behoeven kan zetten of wissen. Ook de positie van de punten is belangrijk. Iedere sprite heeft 24x21 punten, die door programmering van de grafiek kunnen worden gezet. Een POKE-instructie kan ten hoogste 8 punten zetten of wissen; sprites bestaan horizontaal uit 24 punten. Daarom moet een andere weg worden bewandeld. Een sprite is uit 3 series samengesteld en elke serie uit een matrix van 3x21 punten. Bij de programmering moeten de 3 series van een regel altijd naast elkaar staan.

	1	2	3
REGEL 1			
REGEL 2			
REGEL 3			
.			
REGEL 21			

Iedere regel bestaat dus uit 3 serie van 8 bits. Er zijn twee soorten sprites:

1. Eenkleurige sprites

Ieder bit correspondeert met een punt die de kleur uit zijn register heeft of die van de achtergrond. De punt wordt in het laatste geval wel gezet, maar verschijnt niet op het scherm.

2. MultiColor sprites

Twee bits corresponderen met een punt. Ze informeren over het register, waaruit de kleur van de sprite komt. Een multicolor sprite kan op die manier, afgezien van de kleur van de achtergrond, uit maximaal drie verschillende kleuren bestaan.

3.7.1 Programmering van de sprites

De programmering van sprites verloopt net zo als de programmering in machinetaal. Eerst bepaalt u de positie van de punten en vervolgens wijzigt u dit binaire patroon in decimale getallen, de waarden voor de POKE-instructies. Anders dan bij de programmering van de grafiek hoeven de punten niet steeds opnieuw te worden berekend.. Daarom kunt u ze in DATA-regels vastleggen. Daarna leest u een FOR ... NEXT-lus de waarden in en slaat ze ergens in het geheugen op. Denk eraan dat punten die niet op het scherm verschijnen de waarde 0 krijgen. DATA-regels zien er zo uit:

```
1000 DATA 000, 000, 000
1010 DATA 000, 000, 000
1020 DATA 000, 000, 000
1030 DATA 000, 000, 000
1040 DATA 000, 000, 000
1050 DATA 000, 000, 000
1060 DATA 000, 000, 000
1070 DATA 003, 255, 255
1080 DATA 000, 002, 000
1090 DATA 192, 170, 128
1100 DATA 194, 150, 080
1110 DATA 234, 150, 080
1120 DATA 194, 170, 168
1130 DATA 192, 170, 168
1140 DATA 000, 032, 128
1150 DATA 000, 170, 160
1160 DATA 000, 000, 000
1170 DATA 000, 000, 000
1180 DATA 000, 000, 000
1190 DATA 000, 000, 000
1200 DATA 000, 000, 000
```

De 3x21-structuur van de programmering van de sprites is duidelijk te zien. Ieder getal tussen 1 en 255 komt binnen een sprite met een puntpatroon overeen. Elke figuur van een punt tot een vol blok van 21x24 kan dus worden geprogrammeerd.

Sprites worden hoofdzakelijk voor spelletjes en trucages gebruikt. In tegenstelling tot een grafiek blijven de vormen van de figuren altijd gelijk. Hoogstens kunnen ze groter worden. Grafieken hebben geen vaste vorm - denk maar aan het plotten van een functie of een taartdiagram in de vorm van een schijf die als een taart is aangesneden. Zo kunnen sprites net zo goed voor utilitaire doeleinden worden gebruikt. Stelt u zich eens de

presentatie van een produkt voor met de grafiek, de kleur en het geluid van de Commodore 64.

Terug naar het laatste voorbeeld. Om te zien wat er eigenlijk in de DATA-regels stond moeten de decimale getallen weer in het binaire stelsel worden omgezet.

REGEL	1	00000000	00000000	00000000
REGEL	2	00000000	00000000	00000000
REGEL	3	00000000	00000000	00000000
REGEL	4	00000000	00000000	00000000
REGEL	5	00000000	00000000	00000000
REGEL	6	00000000	00000000	00000000
REGEL	7	00000000	00000000	00000000
REGEL	8	00000001	11111111	11111111
REGEL	9	00000000	00000010	00000000
REGEL	10	11000000	10101010	10000000
REGEL	11	11000010	10010110	01010000
REGEL	12	11101010	10010110	01010000
REGEL	13	11000010	10101010	10101000
REGEL	14	11000000	10101010	10101000
REGEL	15	00000000	00100000	10000000
REGEL	16	00000000	10101010	10100000
REGEL	17	00000000	00000000	00000000
REGEL	18	00000000	00000000	00000000
REGEL	19	00000000	00000000	00000000
REGEL	20	00000000	00000000	00000000
REGEL	21	00000000	00000000	00000000

Misschien kunt u het niet zo duidelijk zien, maar hier staat het zijaanzicht van een helicopter die naar links vliegt.

Nu de sprite is gemaakt kan hij naar believen worden veranderd. Ten eerste moet hij worden ingeschakeld. In register 21 staat zijn kengetal.

sprite:	7	6	5	4	3	2	1	0	
bit	:	b7	b6	b5	b4	b3	b2	b1	b0

Door POKE 21,1 wordt sprite 0 ingeschakeld, POKE 21,3 schakelt de sprites 0 en 1 in en POKE 21,255 zou ze allemaal inschakelen.

Vervolgens moet het adres van de sprite worden gedefinieerd. Daarvoor zijn de adressen 2040 en 2047 beschikbaar. Het beginbereik van de sprite wordt hierin opgeslagen.

Een sprite bestaat uit 63 bytes. Door het blokgetal in het adres van de sprite in te lezen wordt de geheugenruimte gedefinieerd. Als voorbeeld nemen we blok 13. 13*64 bytes (lengte van een sprite

+ 1, voor het startadres van de volgende sprite) levert startadres 832 op.

register- pointer		adres
----------------------	--	-------

11	* 64 =	704
13	* 64 =	832
14	* 64 =	896
15	* 64 =	960

Met deze blokken kunt u slechts 4 sprites adresseren. Om ook andere te kunnen aanspreken moet u eerst de BASIC Start verschuiven, zodat er genoeg ruimte voor de nieuwe sprites ontstaat. Dat gaat zo:

```
POKE 44,10      (BASIC Start aan $0A00
POKE 10*256,0    (eerste BASIC-byte = 0)
NEW              (terugzetten van pointer)
```

De volgende tabel geeft de adressen voor de registerpointers van de sprites.

adres :	2040	2041	2042	2043	2044	2045	2046	2047
sprite:	0	1	2	3	4	5	6	7

Als u op adres 2040 de waarde 13 schrijft, betekent dat dat de sprite vanaf geheugenplaats 832 te vinden is. Als u dezelfde waarde op adres 2041 schrijft, hebben de sprites 1 en 2 dezelfde vorm. Alle andere informatie kan echter verschillend zijn, wat het programmeerwerk tot een minimum beperkt. Bij de programmering hoeven dus geen andere adressen te worden veranderd, het nummer van de sprite is voldoende. Door een FOR ... NEXT-lus worden alle waarden van de sprites in het geheugen opgeslagen vanaf plaats 832.

```
FOR I=0 TO 62:      REM 63 bytes van een sprite
  READ X:           REM lezen van bytes
  POKE 832+I,X:     REM bytes in het blok schrijven
NEXT I:             REM eind van de lus
```

Daarna kunt u alle informatie over de sprite met een of twee POKES veranderen.

Een sprite heeft twee registers voor de positie op het scherm. Het eerste register voor sprite 0 is register 0. Hier staat de plaats van de sprite op de X-as (horizontaal). Hetzelfde, maar dan

voor de Y-as staat in het volgende register. In de volgende voorbeelden wordt de variabele V (voor videocontroller) gebruikt. De waarde daarvan is 53248. Dit is het startadres van de videocontroller. De eerste regel van ieder programma, waarin grafiek of sprites voorkomen, moet daarom zijn:

```
V=53248: REM start van videocontroller
```

Door

```
POKE V+0, KOLOM: REM sprite 0-X
POKE V+1, REGEL:  REM sprite 0-Y
```

verschijnt de helicopter op het midden van het beeldscherm.

Voor een vloeiende beweging moet u de registers met 1 veranderen. Dat gebeurt met een FOR ... NEXT-lus:

```
FOR I=159 TO 100 STEP -1
    POKE V+0,I
NEXT I
```

Deze routine verschuift de helicopter een heel stuk naar links, maar hij blijft op dezelfde horizontale as. De beweging van de sprite kan zichtbaar worden gemaakt door voor NEXT I nog een FOR ... NEXT-lus in te voegen:

```
FOR II=1 TO 100
NEXT II
```

De sprite schuift nu veel langzamer op; de beweging is zichtbaar. Er zijn 320 horizontale posities, maar de maximale waarde bij een POKE in het X-register kan slechts 255 zijn. Om de sprite naar de rechter beeldschermrand te brengen gebruikt u een ander register. In register 16 bevindt zich een bijzondere bit voor iedere sprite. Hier wordt aangegeven of een X-coördinaat die boven de 255 ligt moet worden gebruikt. Om dit te bewerkstelligen zet u voordat u adresseert de corresponderende bit op 1. Voor de helicopter betekent dat:

```
POKE V+16,1
```

Bij de adressering wordt nu steeds 255 opgeteld. Een POKE V+0,1 zou dus de keuze van positie 256 tot gevolg hebben. Om weer op punten kleiner dan 256 terecht te komen moet deze bit worden teruggezet:

```
POKE V+16,0
```

Een sprite kan dus op alle mogelijke manieren en onafhankelijk van de anderen over het beeldscherm wandelen.

De registers 39-46 zijn bestemd voor de kleuren van de sprites. Voor iedere sprite staat er een register ter beschikking.

```
register: 39 40 41 42 43 44 45 46
sprite  : 0  1  2  3  4  5  6  7
```

kleuren:

```
0  zwart
1  wit
2  rood
3  cyaan
4  violet
5  groen
6  blauw
7  geel
8  oranje
9  bruin
10 lichtrood
11 grijs 1
12 grijs 2
13 lichtgroen
14 lichtblauw
```

Door POKE V+39,14 wordt onze sprite dus lichtblauw.

De sprites kunnen in horizontale of verticale richting worden vergroot. Daarvoor staan weer twee registers ter beschikking, een voor vergroting langs de X-as en de ander voor vergroting langs de Y-as. Deze registers nemen alle sprites voor hun rekening. Daarom moeten ze gecodeerd worden. Een spritecontrolregister is daarom zo opgebouwd:

```
byte  : 7 6 5 4 3 2 1 0
sprite: 7 6 5 4 3 2 1 0
```

Om de eerste sprite (sprite 0) in de X- en de Y-richting groter te laten worden, moet u deze POKE-instructies geven:

```
POKE V+23,1: REM vergroot sprite 0 in Y-richting
POKE V+29,1: idem in X-richting
```

Om de beide sprites 0 en 1 in de Y-richting groter te laten worden, moet u POKE V+23,3 gebruiken enzovoort. Een sprite kan in beide richtingen twee keer zo groot worden gemaakt. Dat betekent dat bijvoorbeeld bij het maken van loopschrift een teken vier keer

kan worden vergroot.

Een sprite kan voor of achter de achtergrond verschijnen. Probeer eens

POKE V+27,1

De helicopter verschijnt achter de achtergrond. Om dat vast te stellen gaat u met de cursor naar de regel, waar de sprite staat. Als u een paar karakters laat verschijnen, ziet u dat ze de sprite bedekken. U kunt het beste zelf met het kleurcontrast tussen sprite en tekens experimenteren.

POKE V+27,0

haalt de sprite weer naar voren. De tekst ligt er nu achter. Het register dat de achtergrondsprite prioriteit verleent, ziet er als volgt uit:

```
byte   : b7 b6 b5 b4 b3 b2 b1 b0
prior. : 7  6  5  4  3  2  1  0
```

U kunt de prioriteit van een sprite dus naar wens wijzigen. Wanneer de corresponderende bit niet is gezet (=0), dan staat de sprite dus voor de achtergrond. Bij een ingestelde bit staat de sprite achter de achtergrond. Door de sprites verschillend in te stellen kunnen ze elkaar overlappen, terwijl de voor- en achtergrond duidelijk herkenbaar blijft. Een driedimensionaal beeld is zo makkelijk te verkrijgen.

In register 30 wordt een botsing tussen sprites opgetekend. Het staat op 0 totdat twee of meer sprites op elkaar zijn gebotst. Daarna blijft het register ingesteld totdat het door een POKE-instructie is teruggezet. De waarde van het register geeft aan om welke sprites het gaat. Als na

PRINT PEEK (V+30) of KO=PEEK(V+30)

de waarde 3 verschijnt, weet u, dat de sprites 1 en 2 op elkaar zijn gebotst. De bitstructuur van het register is verder net zo als dat van het andere. Door

POKE V+30,0

zet u het register weer op 0. Als dat niet gebeurt, worden de beide sprites als op elkaar gebotst beschouwd, ook al is dat niet het geval.

Register 31 maakt het mogelijk op een botsing van sprites met de achtergrond te reageren. U gaat net zo te werk als bij register 30. Het enige verschil is het resultaat van de PEEK-instructie: er wordt alleen duidelijk welke sprite(s) op een teken zijn gebotst. Aard en positie van het teken blijven onbekend. Om daar achter te komen moet u de andere registers en geheugenplaatsen nazoeken. Ter verduidelijking nog een keer de schrijfwijze van de instructie:

PRINT PEEK (V+31) of KO=PEEK (V+31)

Na een botsing moet register 31 natuurlijk weer worden teruggezet. Met behulp van de registers 17 en 22 kan het beeldscherm in vier richtingen ten hoogste 8 stappen worden verschoven. Register 17 zorgt voor verschuiving langs de Y-as en 22 voor verschuiving langs de X-as. Twee dingen zijn in dit verband belangrijk.

1. Pas wanneer bit 3 van het register dat wordt gebruikt is ingesteld, kan de verschuiving plaatsvinden.
2. Als de inhoud van de registers moet worden veranderd, mag er niet zo maar een nieuwe waarde in deze registers worden geschreven. Er kunnen dan ook andere bits worden veranderd. De POKE-instructie moet samen met de oude inhoud gecodeerd worden. In het volgende voorbeeld wordt bit 3 van register 22 gezet:

POKE V+22,PEEK(V+22) OR 8

Nu kan de inhoud van het beeldscherm 8 stappen verschuiven.

Een sprite kan ten hoogste uit drie kleuren bestaan, maar heeft dan een kleinere oplossing, omdat twee bits als een punt worden beschouwd. In plaats van een 8x8 matrix is er dan een 4x8 matrix. De twee bits informeren over de kleur en geven dus ook aan of een punt gezet is. Twee bits kunnen vier inlichtingen doorgeven: 00, 01, 10 en 11.

- 00 - de punt heeft de kleur van de achtergrond en is dus onzichtbaar.
- 01 - de kleur wordt uit register 37 gehaald
- 10 - de kleur wordt uit het spritekleurregister gehaald (registers 39-46): vaststaande kleuren
- 11 - de kleur wordt uit register 38 gehaald

Afgezien van de kleur van de achtergrond kan een sprite dus een eigen kleur hebben en twee kleuren die voor alle sprites gelden. De helicopter ziet er wat merkwaardig uit, omdat hij als MC-sprite is ontwikkeld. Dergelijke sprites bestaan uit minder punten, dan

een monochrome sprite. Daarom laat het beeld in de normale spritemodus enigszins te wensen over. Hieronder geven we het programma voor een helicopter die uit meer kleuren bestaat.

```
10 REM spritevoorbeeld helicopter
20 V=53248: REM begin van videocontroller
30 POKE V+32,15: POKE V+33,14: REM kleuren van achtergrond
40 PRINT "<CTRL>-7": REM druk CTRL en 7 tegelijk in
50 POKE V+21,3: REM sprite 0 en 1 openen
60 POKE V+28,3: REM sprite 0 en 1 zijn multicolor
70 POKE V+39,6: REM kleur van sprite 0: blauw
80 POKE V+40,2: REM kleur van sprite 1: rood
90 POKE V+37,14: REM multicolor kleur 1: lichtblauw
100 POKE V+38,0: REM multicolor kleur 2: zwart
110 POKE 2040,13: REM sprite 0 uit bereik 832-895
120 POKE 2041,13: REM sprite 1 uit bereik 832-895
130 FOR I=0 TO 62: REM lus voor inlezen van data
140 : READ X: REM lezen van puntencombinaties
150 : POKE 832+I,X: REM opslaan van puntencombinaties
160 NEXT I: REM eind van lus
170 POKE V+0,24: POKE V+1,50: REM positie van sprite 0
180 POKE V+2,60: POKE V+3,50: REM positie van sprite 1
190 END
1000 DATA 000, 000, 000
1010 DATA 000, 000, 000
1020 DATA 000, 000, 000
1030 DATA 000, 000, 000
1040 DATA 000, 000, 000
1050 DATA 000, 000, 000
1060 DATA 000, 000, 000
1070 DATA 003, 255, 255
1080 DATA 000, 002, 000
1090 DATA 192, 170, 128
1100 DATA 194, 150, 080
1110 DATA 234, 150, 080
1120 DATA 194, 170, 168
1130 DATA 192, 170, 168
1140 DATA 000, 032, 128
1150 DATA 000, 170, 160
1160 DATA 000, 000, 000
1170 DATA 000, 000, 000
1180 DATA 000, 000, 000
1190 DATA 000, 000, 000
1200 DATA 000, 000, 000
```

In de appendix van dit boek staat een raster dat u bij het ontwerpen van sprites kunt gebruiken. Eerst maakt u een grove schets en besluit hoeveel kleuren u gaat gebruiken. Vervolgens tekent u op het raster het uiteindelijke ontwerp. Daarbij

gebruikt u dus ofwel alle punten van de 8x8 matrix, ofwel per bladzijde drie kleuren. Bij MC-sprites corresponderen 2 punten op het papier met 1 punt op het beeldscherm. Tegelijkertijd moeten de registers worden uitgekozen. U kunt het beste niet de punten inkleuren, maar de bitcombinaties in de vakjes schrijven. Als dat klaar is, kunnen de bits worden geteld. Bij het raster staan al decimale getallen: u hoeft alleen maar die getallen op te tellen die met een ingestelde bit corresponderen. Op iedere regel staan daarna drie decimale getallen die u net als in de dataregels hierboven naast elkaar moet schrijven. Bij monochrome sprites is iedere punt op het papier een punt op het scherm; alle vakjes kunnen dus worden ingekleurd. Daarna maakt u van de getallen weer dataregels.

4 STURING VAN IN- EN UITVOER - CIA 6526

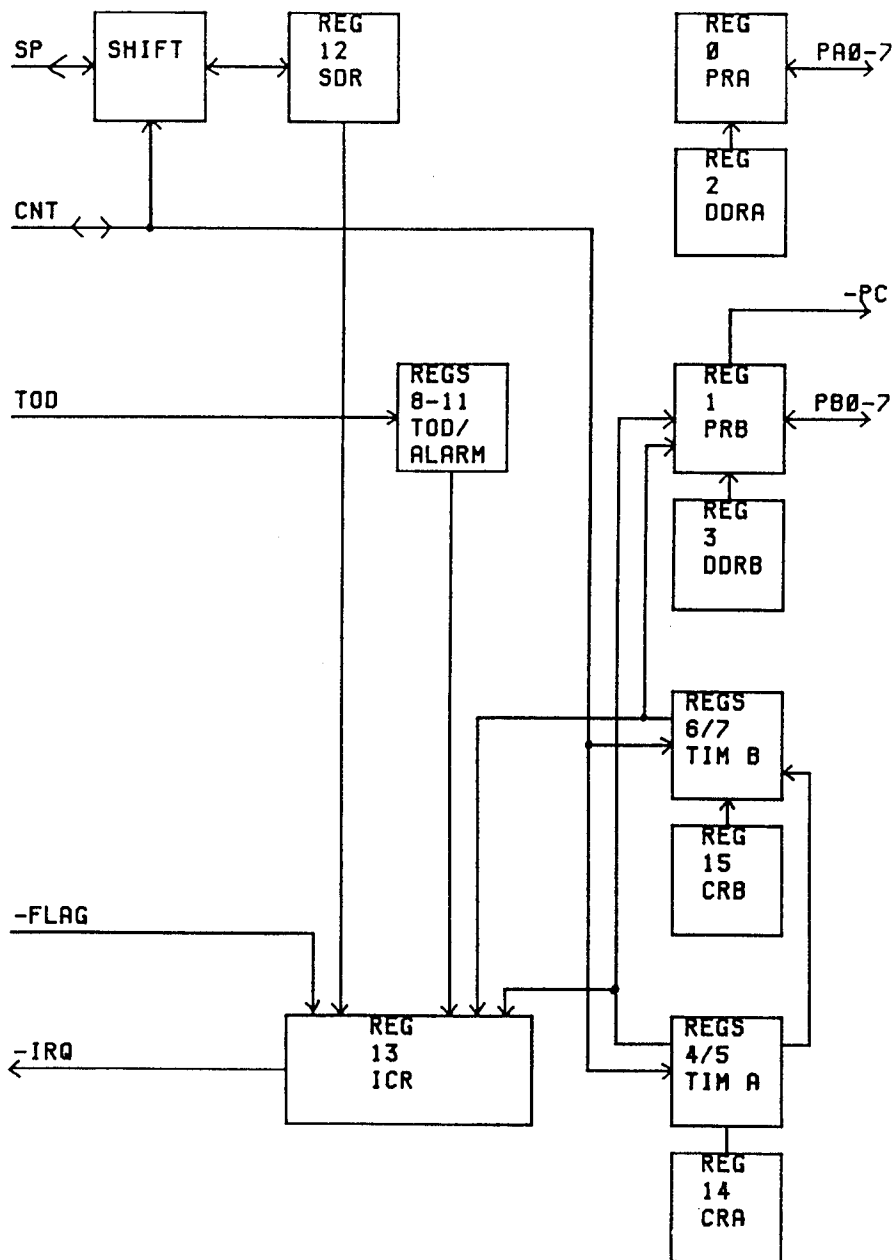
4.1 Algemene opmerkingen over de 6526

De Complex Interface Adapter (CIA) 6526 is een nieuwe perifere bouwsteen uit de 65xx familie. Hij bestaat uit:

- * 16 afzonderlijk programmeerbare in- en uitvoerlijnen
- * 8- of 16bits handshake zowel bij in- als uitvoer
- * 2 onafhankelijke, cascadeerbare 16bits intervaltimers
- * 24uurs klok (AM/PM) met programmeerbare alarmtijd
- * 8bits schuifregister voor de seriële in- en uitvoer

Het blokschema staat op de volgende bladzijde. In hoofdstuk 4.6 vindt u informatie over de speciale functie van de CIA's in de Commodore 64. De pinconfiguratie is als volgt:

- 1 massa
- 2 - 9 I/O-poort A; 8 bits bidirectioneel
- 10-17 I/O-poort B; 8 bits bidirectioneel. De bits 6+7 kunnen zo geprogrammeerd worden, dat ze de underflow van van de timers aangeven.
- 18 PC(port control); alleen uitgang; geeft aan of er aan poort B of aan beide poorten data ter beschikking staan
- 19 TOD(time of day); alleen ingang 50/60 Hz.; triggert de real time clock
- 20 +5V voedingsspanning
- 21 IRQ (interrupt request)
- 22 R/W(read/write); alleen uitgang; 0=overname van databus; 1=uitvoer op databus
- 23 CS(chip select); alleen ingang; 0=databus geldig; 1=databus hoogohmig (tri-state)
- 24 FLAG; alleen ingang; betekenis als PC
- 25 O2(systeemklok 2); alleen ingang; alle acties op de databus vinden alleen plaats, wanneer O2=1
- 26-33 DB7-DB0(databus); bidirectioneel; aansluiting op de processor
- 34 RES(reset); alleen ingang; 0=terugzetten van CIA in beginstand
- 35-38 RS3-RS0(register select); alleen ingang; dient voor keuze van een van de 16 registers van de CIA; alleen geldig wanneer CS=0
- 39 SP(serial port); bidirectioneel; in-en uitgang van het schuifregister
- 40 CNT(count); bidirectioneel; in- en uitgang van de schuifregisterklok of triggeringang voor de intervaltimers



4.2 Registerbeschrijving van de CIA

- REG 0 PRA (port register A)
gebruik: READ/WRITE
bits 0-7 correspondeert met de pins PA0-7
- REG 1 PRB (port register B)
gebruik: READ/WRITE
bits 0-7 komt met de pins PBO-7 overeen
- REG 2 DDRA (datarichtingsregister A)
gebruik: READ/WRITE
bits 0-7 bepalen datarichting van de corresponderende databits van poort A. 0=ingang, 1=uitgang.
- REG 3 DDRB (datarichtingsregister B)
gebruik: READ/WRITE
bits 0-7 bepalen datarichting van de corresponderende databits van poort B. 0=ingang, 2=uitgang.
- REG 4 TA LO (timer A lobyte (=LSB))
gebruik: READ
bits 0-7 geeft de momentele toestand aan van het LSB van timer A
gebruik: WRITE
bits 0-7 in dit register wordt het lobyte geladen van de waarde, waarbij de timer begint met aftellen naar nul
- REG 5 TA HI (timer A hibyte)
gebruik: READ
bits 0-7 geven de momentele waarde aan van het hibyte van timer A.
gebruik: WRITE
bits 0-7 in dit register wordt het MSB geladen van de waarde, waarbij de timer begint met aftellen naar nul.
- REG 6 TB LO (timer B lobyte)
(zie REG 4)
- REG 7 TB HI (timer B hibyte)
(zie REG 5)
- REG 8 TOD IOTHS (klok 1/10 seconden)
gebruik: READ
bits 0-3 tiende seconden van de realtimeclock in BCD
BCD -formaat

- bits 4-7 altijd 0
 - gebruik: WRITE en CRB bit 7=0
 - bits 0-3 tiende seconden in BCD-formaat
 - bits 4-7 moeten 0 zijn
 - gebruik: WRITE en CRB bit 7=1
 - bits 0-3 voorkeuze van tiende seconden van de alarmtijd in BCD-formaat
 - bits 4-7 moeten 0 zijn
- REG 9 TOD SEC (klok seconden)
- gebruik: READ
 - bits 0-3 seconden in BCD-formaat
 - bits 4-6 10-seconden interval in BCD-formaat
 - bit 7 altijd 0
 - Zie REG 8 voor verdere gebruiksmogelijkheden.
- REG 10 TOD MIN (klok minuten)
- gebruik: READ
 - bits 0-3 minuten in BCD-formaat
 - bits 4-6 10-minuten interval in BCD-formaat
 - bit 7 altijd 0
 - Zie REG 8 voor verdere gebruiksmogelijkheden
- REG 11 TOD HR (klok uren)
- gebruik: READ
 - bits 0-3 uren in BCD-formaat
 - bit 4 10-uren interval
 - bits 5-6 altijd 0
 - bit 7 0=voor 12 uur 's middags (AM), na 12 uur 's middags (PM)
 - Zie REG 8 voor verdere gebruiksmogelijkheden
- REG 12 SDR (serial data register)
- gebruik: READ/WRITE
 - bits 0-7 de data uit dit register worden bit voor bit via pin SP naar buiten ofwel in het register geschoven
- REG 13 ICR (interrupt control register)
- gebruik: READ (INT DATA)
 - bit 0 1=underflow timer A
 - bit 1 1=underflow timer B
 - bit 2 1=kloktijd en gekozen alarmtijd gelijk
 - bit 3 1=SDR vol/leeg (afhankelijk van het soort gebruik)
 - bit 4 1=signaal aan pin FLAG
 - bits 5-6 altijd 0
 - bit 7 minstens een bit van INT MASK en INT DATA corresponderen met elkaar
 - OPGELET: bij het lezen van dit register worden alle

bits gewist
gebruik: WRITE (INT MASK)
betekenis van bits als boven, behalve bit 7

- bit 7 1=iedere ingestelde bit zet de corresponderende maskerbit. De andere bits blijven onaangetast.
0=iedere ingestelde bit wist de corresponderende maskerbit. De andere bits blijven onaangetast.

REG 14 CRA (control register A)

gebruik: READ/WRITE

- bit 0 1=timer A start, 0=stop
bit 1 1=underflow van timer A wordt gesignaleerd aan pin PB6
bit 2 1=elke underflow van timer A zet PB6 in de andere stand
0=elke underflow van timer A produceert aan PB6 een hipuls met de pulslengte van een systeemklok
bit 3 1=timer A telt slechts een keer van de beginwaarde terug naar nul
0=timer telt voortdurend terug naar nul
bit 4 1=direkt een nieuwe beginwaarde laden in timer A. Deze bit fungeert als strobe. Het moet iedere keer, dat er direkt wordt geladen, opnieuw worden ingesteld
bit 5 bepaalt oorsprong van de timertrigger
1=timer telt de stijgende CNT-flanken,
0=timer telt pulsen van systeemklok
bit 6 1=SP is uitgang, 0=SP is ingang
bit 7 1= realtimeclocktrigger is 50 Hz,
0=trigger is 60 Hz.

REG 15 CRB (control register B)

gebruik: READ/WRITE

- bits 0-4 zie REG 14, met dien verstande, dat ze op timer B en pin PB7 betrekking hebben
bits 5-6 bepalen oorsprong van timertrigger
00=timer telt klokpulsen, 01 timer telt stijgende CNT-flanken
10=timer B telt underflows van timer A
11=timer B telt underflows van timer A als CNT=1
bit 7 1=alarm instellen
0=kloktijd instellen

4.3 I/O-ports

De poorten en B bestaan elk uit een 8bits dataregister (PR) en een 8bits datarichtingsregister (DDR). Wanneer een bit in het DDR is ingesteld, dient de corresponderende bit in het PR als uitgang. Als een bit in het DDR=0 staat, functioneert de corresponderende bit in het PR als ingang. Tijdens het lezen geeft het PR de momentele toestand van de ingangs- en uitgangspins aan (PA0-7, PBO-7). PB6 en PB7 kunnen de uitgangsfuncties van de timers overnemen. De overdracht van gegevens tussen CIA en de buitenwereld, die op PA/PB is aangesloten, wordt geregeld door PC en FLAG.

Voorafgegaan door een lees- of schrijffactie op PRB wordt PC=0 gedurende een klokpuls. Dit signaal kan op die manier aangeven of er data voor PB beschikbaar zijn, dan wel of PB data aanneemt. FLAG is een ingang die negatief door de flankpuls getriggerd wordt. Hij kan bijvoorbeeld met PC van een andere ingang worden verbonden. Een dalende flank aan FLAG zet ook de FLAG-interruptbit. De seriële datapoort SDR is een synchroon 8bits schuifregister. CRA bit 6 bepaalt de modus van in- of uitvoer. In de invoermodus worden de databits op SP in het schuifregister overgenomen, tijdens een stijgende flank van een signaalpuls op CNT. Na 8 CNT-pulsen wordt de inhoud van het schuifregister naar CNT gebracht en de SP-bit in ICR ingesteld. In de uitvoermodus fungeert timer A als baudgenerator. De data uit SDR worden met de halve underflowfrequentie van timer A naar SP geschoven. De theoretisch hoogste baud is 1/4 van de klokpuls. Wanneer timer A loopt en in de continue modus staat (CRA bit 0=1 en bit 0=3), begint de overdracht, nadat de gegevens in het SDR zijn geschreven. De klokpuls die van timer A is afgeleid, verschijnt op CNT. De gegevens uit SDR worden in het schuifregister geladen en dan bij iedere dalende flank op CNT uit SP geschoven. Na 8 CNT-pulsen wordt de SP-interrupt geproduceerd. Als SDR daarvoor met nieuwe gegevens wordt geladen dan worden die automatisch in het schuifregister geladen en eruit geschoven. Dan verschijnt er geen interrupt. Als er data uit SDR worden geschoven, dan staat de hibernate (MSB) voorop. De invoer moet hetzelfde format hebben.

4.4 Timer

Beide intervaltimers bestaan uit een 16bits teller (read only) en een 16bits tussengeheugen (write only). Gegevens die in de timer worden geschreven, komen terecht in het tussengeheugen; leesdata geven de momentele stand van de teller aan. Beide timers kunnen onafhankelijk of tezamen worden gebruikt. Door de verschillende gebruiksmogelijkheden kunnen lange tijdsvertragingen, variabele pulslengtes en impulsketens worden geproduceerd. Als de CNT-ingang wordt gebruikt, kunnen de timers externe impulsen tellen of frequenties meten. Elke timer heeft een stuurregister (CRA/CRB) dat de volgende functies mogelijk maakt:

START/STOP (bit 0)

Deze bit kan de timer elk moment laten beginnen of stoppen.

PB ON/OFF (bit 1)

Hiermee wordt de timeronderflow naar PB geleid (PB6 voor timer A, PB7 voor B). Deze functie heeft voorrang voor de datarichting die in DDRB is bepaald.

TOGGLE/PULSE

Deze bit bepaalt de soort underflowpuls die aan PB verschijnt. Of PB wordt bij elke underflow in een andere stand gezet of er wordt een positieve puls met de lengte van een klokpuls geproduceerd.

ONE SHOT/CONTINUE MODUS(bit 3)

De timer telt terug naar nul, beginnend bij de waarde van het tussengeheugen. Daarna zet hij de IRC-bit, laadt de teller opnieuw met de waarde van het tussengeheugen en stopt dan. In de continue modus wordt dit proces steeds herhaald.

FORCE LOAD (bit 4)

De timer kan altijd worden geladen, ongeacht of deze loopt of niet.

INPUT MODE (bit 5 CRA, bit 5-6 CRB)

Keuze van de klokpuls, waardoor de timer terug kan tellen. Timer A kan met de puls van de systeemklok gevoed of met een puls die op CNT wordt gegeven. Timer B kan bovendien nog met

de underflowpulsen van timer A worden gevoed.
Naar keuze hangt dit af van CNT=1.

4.5 Real-time clock

De realtimeclock (TOD) heeft een verdeling van 24 uur (AM/PM) met een resolutie-interval van 1/10 seconde. Ze bestaat uit vier registers: 1/10 seconde, seconden, minuten, uren. De AM/PM-bit is het MSB (hibyte) van het urenregister. Elk register heeft het BCD-format, zodat de gelezen waarden zonder veel rekenen kunnen worden gebruikt. Als klokpuls dient een signaal van 50/60 Hz. aan pin TOD (programmeerbaar, CRA-bit 7).

Bovendien is er nog een alarmregister, waarmee men op elk moment een interrupt kan plegen. Dit register gebruikt hetzelfde adres als het TOD-register. Daarom wordt een actie op dit register met CRB bit 7 geregeld. In het alarmregister kan alleen geschreven worden. Onafhankelijk van CRB bit 7 geeft iedere leesactie de stand van het TOD-register aan. Om de tijd goed te kunnen instellen en aflezen moet een bepaalde volgorde worden gehandhaafd: als er in het urenregister wordt geschreven, stopt de klok vanzelf. Pas wanneer een schrijfactie op het 1/10 secondenregister is uitgevoerd, loopt de klok verder. Op die manier begint de klok op de gewenste tijd.

Tijdens het uitlezen van de tijd kan een overdracht in een register plaatsvinden dat al gelezen is. Daarom wordt de tijd in een tussengeheugen opgeslagen. Het tussengeheugen wordt pas vrijgegeven, nadat de 1/10 seconden gelezen zijn. Een enkel register kan natuurlijk ook tussentijds gelezen worden. Als dit het urenregister is, moet daarna het 1/10 secondenregister worden gelezen om het tussengeheugen weer vrij te maken.

4.5.1 De juiste tijd met een trucje

Afhankelijk van het systeem laat de nauwkeurigheid van klok TI\$ die door het bedrijfssysteem wordt gevoed, op de lange duur te

wensen over. U moet met een maximale fout van een half uur per dag rekening houden. Voor wie de precieze tijd wil weten is er in de CIA's de real-time clock die door de frequentie van de netspanning wordt gevoed. Daardoor is ze ook op de lange duur zeer nauwkeurig.

De volgende twee BASIC-programma's maken het gebruik van de real-time clock gemakkelijker. Het eerste stelt de klok in, het tweede leest hem uit. Bij het eerste wordt de waarde voor 1/10 seconde altijd op nul gezet.

```

10 C=56328: REM basisadres van klok in CIA 1
20 REM C=56584 voor klok in CIA 2
30 POKE C+7,PEEK(C+7)AND127
35 POKE C+6,PEEK(C+6)OR128
40 INPUT"TIJD IN FORMAAT HHMMSS INVOEREN";A$
50 IF LEN(A$)<>6 THEN 40
60 H=VAL(LEFT$(A$,2))
70 M=VAL(MID$(A$,3,2))
80 S=VAL(RIGHT$(A$,2))
90 IF H>23 THEN 40
100 IF H>11 THEN H=H+68
110 POKE C+3,16*INT(H/10)+H-INT(H/10)*10
120 IF M>59 THEN 40
130 POKE C+2,16*INT(M/10)+H-INT(M/10)*10
140 IF S>59 THEN 40
150 POKE C+1,16*INT(S/10)+S-INT(S/10)*10
160 POKE C,0

```

Het volgende programma maakt het uitlezen van de tijd mogelijk.

```

10 C=56328: REM basisadres van klok in CIA 1
20 PRINT "(shft/clr)":REM C=56584 voor klok in CIA 2
30 H=PEEK(C+3):M=PEEK(C+2):S=PEEK(C+1):T=PEEK(C)
40 FL=1
50 IF H>32 THEN H=H-128:FL=0
60 H=INT(H/16)*10+H-INT(H/16)*16:ON FL GOTO 80
65 IF H=12 THEN 85
70 H=H+12
80 IF H=12 THEN H=0
85 M=INT(M/16)*10+M-INT(M/16)*16
90 S=INT(S/16)*10+S-INT(S/16)*16
100 T$=STR$(T)
110 H$=STR$(H):IF LEN(H$)=2 THEN H$=" 0"+RIGHT$(H$,1)
120 M$=STR$(M):IF LEN(M$)=2 THEN M$=" 0"+RIGHT$(M$,1)
130 H$=STR$(H):IF LEN(H$)=2 THEN H$=" 0"+RIGHT$(H$,1)
140 PRINT "(home)"
150 PRINT"RIGHT$(H$,2)": "RIGHT$(M$,2)": "RIGHT$(S$,2)": "0";

```

```
160 PRINT"RIGHT$(T$,1)"
170 GOTO 30
```

Na het printen van STOP/RESTORE moet u de tijd opnieuw instellen, omdat het bedrijfssysteem alle registers op de uitgangswaarde zet. Dat geldt ook voor het bit dat 50 of 60Hz kiest.

4.6 De CIA's van de Commodore 64

Als u de CIA's voor eigen doeleinden wilt gebruiken, vergeet dan niet dat ze bepaalde functies hebben. Dat geldt speciaal voor het gebruik van interrupts, waardoor het bedrijfssysteem bepaalde routines doorloopt. Verander het masker in de ICR liever nooit.

In de Commodore 64 vervullen de CIA's de volgende functies.

CIA 1 basisadres \$DC00(56320)

REG 0 (PRA)

bits 0-7 Bij normaal gebruik worden hier de rijen van het toetsenbord gekozen. Enkele bits zijn met controlport 1 aan de buitenkant van de computer verbonden. Daar worden joysticks of paddles aangesloten:

bits 0-4 joystick 0, volgorde: boven, onder, links, rechts, toets

bits 6-7 keuze paddlestel A/B. Slechts een van beide bits mag 1 zijn.

REG 1 (PRB)

bits 0-7 Als een toets is ingedrukt, worden hier de kolommen van de toetsenbordmatrix teruggemeld.

bits 0-4 Zie REG 0, maar dan voor controlport 2 (joystick 1).

REG 13 (ICR)

bit 4 invoergegevens van cassettepoort

CIA 2 basisadres \$DD00(56576)

REG 0 (PRA)

bits 0-1 VA 14-15 (meest significante adresbits van video-RAM)

bit 2 TXD (alleen wanneer aangesloten op een RS232-cartridge, anders vrij)

bit 3 ATN (uitgang seriële bus)
 bit 4 CLOCK (idem)
 bit 5 DATA (idem)
 bit 6 CLOCK (idem)
 bit 7 DATA (idem)

REG 1 (PRB)

bits 0-7 gewoonlijk vrij. Als een RS232-cartridge wordt aangesloten, krijgen ze de volgende betekenis:

bit 0 RXD (receive data)
 bit 1 RTS (request to send)
 bit 2 DTR (data terminal ready)
 bit 3 RI (ring indicator)
 bit 4 DCD (data carrier detect)
 bit 6 CTS (clear to send)
 bit 7 DSR (data set ready)

REG 13 (ICR)

bit 4 RXD (alleen bij gebruik van RS232, anders vrij)

4.7 Het gebruik van joysticks

Bij normaal gebruik worden de CIA-bits gebruikt om het toetsenbord uit te lezen. Om met joysticks te kunnen werken moet die functie worden uitgeschakeld. Dat kan alleen met een programma dat niet via het toetsenbord loopt: er mag dus geen INPUT of iets dergelijks in zitten. Het volgende programma voldoet aan die eis.

```

10 POKE56322,224
20 J=PEEK(56320)
30 IF(JAND1)=OTHEN?"boven"
40 IF(JAND2)=OTHEN?"beneden"
50 IF(JAND3)=OTHEN?"links"
60 IF(JAND4)=OTHEN?"rechts"
70 IF(JAND16)=OTHEN?"knop"
80 GOTO20
  
```

Dit programma verwacht de joystick aan controlport 2. Als u een joystick op poort 1 gebruikt, dan moet het adres in regel 20 worden veranderd in 56321. Alleen STOP/RESTORE kan het programma

onderbreken. In een toepassingsprogramma is de laatste regel:

100 POKE56322,255

Deze instructie maakt het toetsenbord weer vrij. Deze regel kan natuurlijk ook voor elke actie met het toetsenbord worden gezet, wanneer uw programma toetsenbord en joysticks nodig heeft. Voor u een joystick opvraagt, moet er wel een instructie als in regel 10 staan.

4.8 De IEC-bus

4.8.1 Wat geschiedenis

In 1974 besloten verschillende fabrikanten van meetapparatuur de aansluitingen te standaardiseren volgens een bestaand systeem van de Amerikaanse firma Hewlett Packard: de General Purpose Interface Bus (GPIB). Van microcomputers was in die tijd nog geen sprake. Het nieuwe bussysteem was gericht op het gebruik van meetapparatuur, x-t-schrijvers en dergelijke. De internationale benaming voor de bus is IEC-625 (DIN: IEC-625 1). In de Verenigde Staten is ook de naam IEEE-488 gangbaar. Voor de computerwereld werd deze bus pas relevant, toen men met de computergestuurde interpretatie van meetreeksen begon.

4.8.2 De grote IEC-bus

Kennis van de IEC-bus van de grote CBM-computers draagt bij tot een goed begrip van de veel complexere stand van zaken bij de seriële IEC-bus van de Commodore 64. Bovendien is de grote IEC-bus verkrijgbaar als uitbreiding van de Commodore 64. De bus van een computer is de verbinding tussen verschillende onderdelen waarlangs gegevens worden overgedragen. De bus bestaat uit een reeks verbindingslijnen. Het gegevensverkeer op de bus moet geregeld worden. Bij de IEEE-bus wordt dit gedaan door de buscontroller. Deze moet weten welk apparaat de talker is en welk apparaat de listenersfunctie heeft. Er kan per keer maar 1 talker operationeel zijn, maar er kunnen vele listeners op de bus worden aangesloten. Elk apparaat dient over een eigen IEEE-interface te beschikken. Daarmee regelt de buscontroller het openen en sluiten

van verschillende voor sturing en gegevensoverdracht per apparaat. Voor het in bedrijf nemen van de apparaten dienen deze al als talker en/of als listener te zijn ingesteld. Een printer is doorgaans alleen operationeel als listener terwijl een meetapparaat doorgaans alleen als talker functioneert. Een robot werkt in de regel als talker en/of listener.

4.8.2.1 Definities

Om u in de volgende paragrafen een overvloed aan tekst te besparen, sommen we de meest gebruikte benamingen op en geven er verklaringen voor.

4.8.2.1.1 Gebruik van de connector en namen van signalen

Meestal wordt voor de IEC-bus een cinch-stekkerverbinding in de vorm van een trapezium met 24 contacten gebruikt. Deze verbinding heeft een stekker- en een buskant, zodat verschillende op elkaar kunnen worden gestoken. Heel soms treft men een 25-polige verbinding van het type D-subminiatur aan. Daar zullen we ons echter niet mee bezighouden. Nu volgt de pinconfiguratie:

- 1 - 4 DIO1-4 (Datalijnen In Out) Deze worden door de talker met gegevens gevuld. De listener ontvangt . DIO1 is het lowbyte.
- 5 EOI (End Or Identify) Deze lijn heeft twee functies. Ten eerste zendt de talker dit signaal uit bij de overdracht van het laatste databyte. Ten tweede gebruikt de controller het in verbinding met ATN om een apparaat dat met SRQ bediend moet worden, zijn adres te laten zenden.
- 6 DAV (Data Valid) Met dit signaal geeft de talker aan dat er een geldige byte op de datalijnen zit.
- 7 NRFD (Not Ready For Data) Een listener laat de talker weten dat hij niet nog meer gegevens op kan nemen, omdat hij bijvoorbeeld nog met de verwerking van oude data bezig is.

- 8 NDAC (No Data ACcepted) Een listener geeft aan dat hij het byte dat op de bus ligt, nog niet heeft overgenomen. Een overname wordt bevestigd door het wegvallen van dit signaal.
- 9 IFC (InterFace Clear) De controller kan hiermee alle aangesloten apparatuur in de uitgangstoestand brengen.
- 10 SRQ (Service ReQuest) Als een apparaat na een opgave te hebben volbracht nieuwe gegevens nodig heeft, af wil geven of een andere actie verlangt, dan kan het dit met dit signaal aan de controller mededelen. Die leidt vervolgens een identify-cyclus in (met EOI en ATN) om vast te stellen om welk apparaat het gaat. Bij Commodore wordt deze functie niet gebruikt.
- 11 ATN (ATteNtion) Als de controller een instructie door wil geven, activeert hij deze lijn. Alle apparaten die op de bus zijn aangesloten krijgen deze instructie, omdat van tevoren niet vaststaat welk apparaat bedoeld is. Dat blijkt pas bij de overdracht van het adres. De andere apparaten kunnen dan hun parate toestand opgeven.
- 12 SHIELD is de afscherming van de kabel van de bus.
- 13-16 DIO5-DIO8 (Data In Out) is de meest significante helft van de databus.
- 17 REN (Remote ENable) Deze lijn is bij Commodore altijd actief. Voor de aangesloten apparatuur betekent het dat de bus werkt.
- 18-24 GND (GrouND = massa) Pin 24 is nog eens speciaal als massa van de logische signalen aangeduid. De signalen op de buslijnen hebben dus op pin 24 betrekking.

4.8.2.1.2 Adressen van de apparatuur

Omdat alle apparaten die op de IEC-bus zijn aangesloten, gelijke rechten hebben, is het nodig een apparaat te selecteren voor een bepaalde actie. Daarom heeft elk apparaat een adres, waarin niet alleen het nummer van het apparaat is aangegeven, maar ook alle actie die wordt verlangd: zenden, ontvangen of activiteiten

beëindigen. Het adresseringsbyte wordt daartoe in tweeën gedeeld. In de minst significante helft staat het adres (daarom een bereik van 0-15) en in de meest significante de actie (een bereik van 16-240). De volgende acties zijn mogelijk:

- 32 Het apparaat wordt als listener geadresseerd en ontvangt dus gegevens.
Dit commando is bijvoorbeeld het gevolg van een PRINT#-instructie in BASIC.
- 64 Het apparaat moet talker zijn en zendt dus gegevens.
Aan dit commando kan een GET# of INPUT# ten grondslag liggen.
- 48 De listen-modus wordt beëindigd. Het minst significante halve bit is altijd =15.
- 80 De talk-modus wordt beëindigd. Ook hier is het adresgedeelte =15.

Het hele adresseringsbyte voor bijvoorbeeld een printer met als nummer 4 is $32 + 4 = 36$ (\$24).

4.8.2.1.3 Secundaire adressen

Op een floppy kunnen meer dan een bestanden worden geopend. Behalve verschillende logisch filenummers moet u ook andere datakanalen (secundaire adressen) in de OPEN-instructie aangeven:

```
10 OPEN 1,8,6,"BESTAND1"  
20 OPEN 2,8,7,"BESTAND2"
```

Een secundair adres start dus functies op, die specifiek zijn voor een apparaat. Bij een floppy houdt dit adres de verschillende bestanden uit elkaar. Daarvoor wordt het bij elke actie direct naar het adres van het apparaat verzonden. Het secundaire adres heeft dus geen algemene stuurfunctie op de IEC-bus, maar per apparaat een verschillende betekenis. Ook hier vindt een tweedeling plaats, waarbij het minst significante deel het eigenlijke secundaire adres is. Dit wordt dan in de OPEN-instructie aangegeven (bereik 0-15). Het meest significante deel informeert over de constellatie waarin dit adres een rol speelt:

96 = PRINT, INPUT of GET
224 = CLOSE
240 = OPEN

De secundaire adressen 0 en 1 vormen een apart geval. De floppy interpreteert 0 als LOAD-instructie en 1 als SAVE. Deze waarden mogen daarom nooit in een OPEN-instructie staan. Bij Commodore-printers kan via het secundaire adres onder andere de karakterset worden uitgekozen.

4.8.2.1.4 De systeemvariabele ST

De numerieke variabele ST informeert over de afloop van een actie op de IEC-bus. De programmeur moet zo mogelijk na elke PRINT, INPUT enzovoort deze variabele opvragen. Bij storingen op de bus kunnen er anders gegevens verloren gaan.

ST kan de volgende waarden aannemen:

- 1 kan na OPEN of PRINT optreden en betekent dat een byte na overgave niet binnen 64 msec. door het opheffen van NDAC bevestigd werd.
- 2 treedt eventueel na INPUT of GET op. Nadat het apparaat als talker is geadresseerd, werd het eerste byte niet binnen 64 msec. ter beschikking gesteld.
- 64 Het laatste byte werd in verbinding met het signaal EOI overgedragen. Bij de floppy betekent dat gewoonlijk einde van het bestand.
- 128 een poging tot adressering had geen reactie op de bus tot gevolg. In zo'n geval eindigt een BASIC-programma met de foutmelding DEVICE NOT PRESENT.

Soms is er een combinatie van de genoemde waarden. Het is daarom handig de relevante getallen bij het opvragen te maskeren. Wilt u bijvoorbeeld het eind van het bestand opvragen, dan doet u dat met

```
100 GET#1,A$:IF (ST AND 64) THEN ...
```

In dit verband kan namelijk heel goed de waarde 66 optreden.

4.8.2.2 De adressering

De BASIC-regels

```
10 OPEN 1,4  
20 PRINT#1,"A"
```

worden gebruikt bij de volgende illustraties van acties op de IEC-bus. Het gaat om het printen van een A op een printer met adres 4. Regel 20 fungeert als trigger voor de acties. Van het OPEN merkt de printer nog niets. Anders dan bij de floppy is daarmee geen naam verbonden van een bestand dat moet worden doorgegeven. De getallen die aan de volgende alinea's voorafgaan, corresponderen met de getallen die in de timing-diagrammen op de X-as staan.

1De bus is nog in de rusttoestand en de stuurlijnen zijn geen van alle actief, dus op highniveau. Stuurlijnen kunnen indifferent zijn; bij Commodore zijn ze gewoonlijk high.

2De lijn ATN maakt alle apparaten duidelijk, dat nu een instructie volgt. NDAC van alle apparaten wordt naar low getrokken om de controller te vertellen dat de bus werkt. De computer toetst deze toestand niet apart, maar kijkt of de lijnen NRFD en NDAC zich in een verschillende toestand bevinden. Als dat niet het geval is, verschijnt de foutmelding DEVICE NOT PRESENT.

3Het adres van het apparaat wordt op de datalijnen gezet: in dit geval 36 ($32 + 4$). De printer met nummer 4 wordt als listener geadresseerd.

4De geldigheid van de datalijnen wordt door DAV aangegeven. NRDF wordt low, omdat de apparaten eerst de aangeboden gegevens moeten verwerken en niet nog meer data kunnen opnemen.

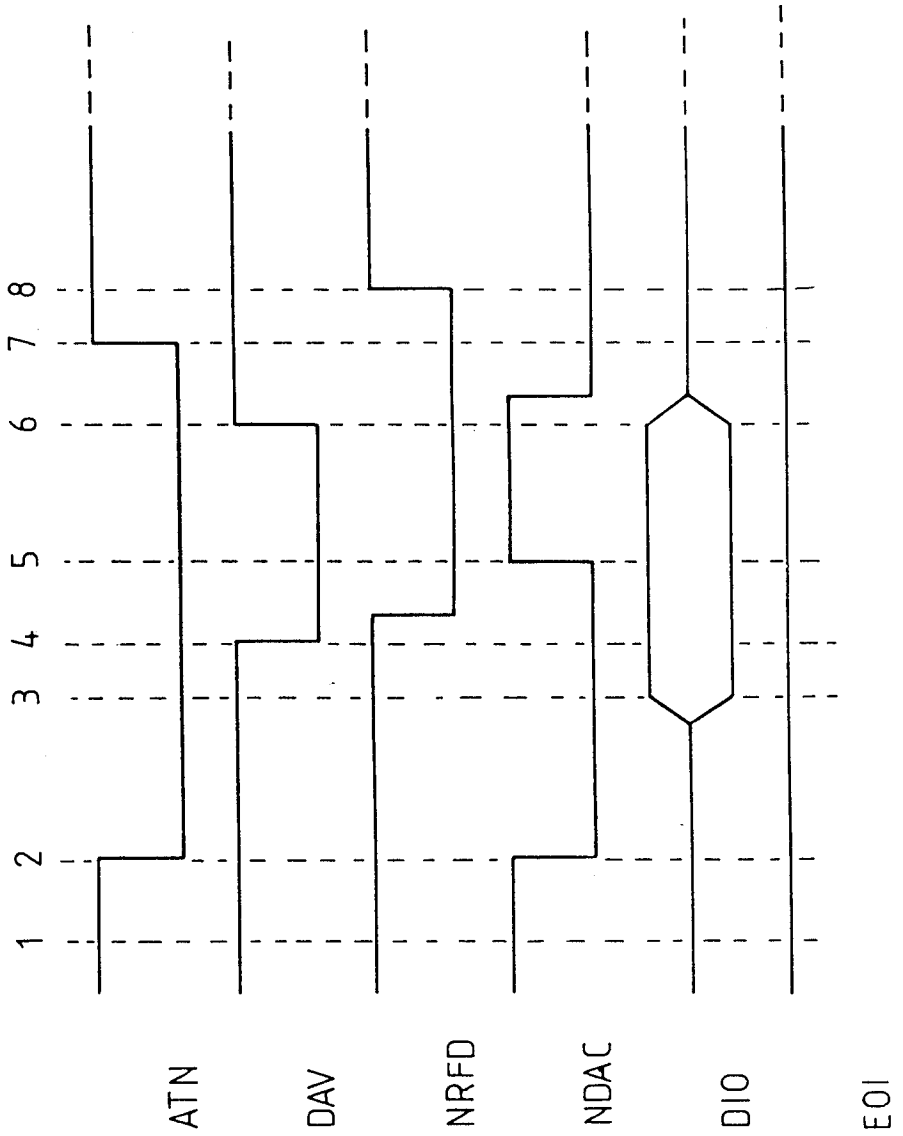
5Binnen 64millisec. verwacht de computer een bevestiging van het apparaat (NDAC=high). Anders wordt ST=1 gezet en gaat de actie niet door.

6Op grond van de bevestiging haalt de computer DAV en het adres van de datalijnen af.

7Door de lijn ATN high te maken geeft de computer aan, dat de adresseringscyclus beëindigd is. Als er nog een secundair adres had moeten worden overgedragen, dan was ATN low gebleven en was dit adres met een DAV op de datalijnen gezet.

8Nadat de printer het adres heeft verwerkt, geeft hij door NRFD aan dat hij andere gegevens kan ontvangen. Aan de volgende acties nemen de andere aangesloten apparaten geen deel.

Adressering



4.8.2.3 Gegevensoverdracht

9De A wordt op de datalijnen gezet en

10zijn geldigheid wordt door DAV aangegeven.

11Na met NRFD te hebben aangegeven dat hij geen andere gegevens kan gebruiken, bevestigt de printer door NDAC.

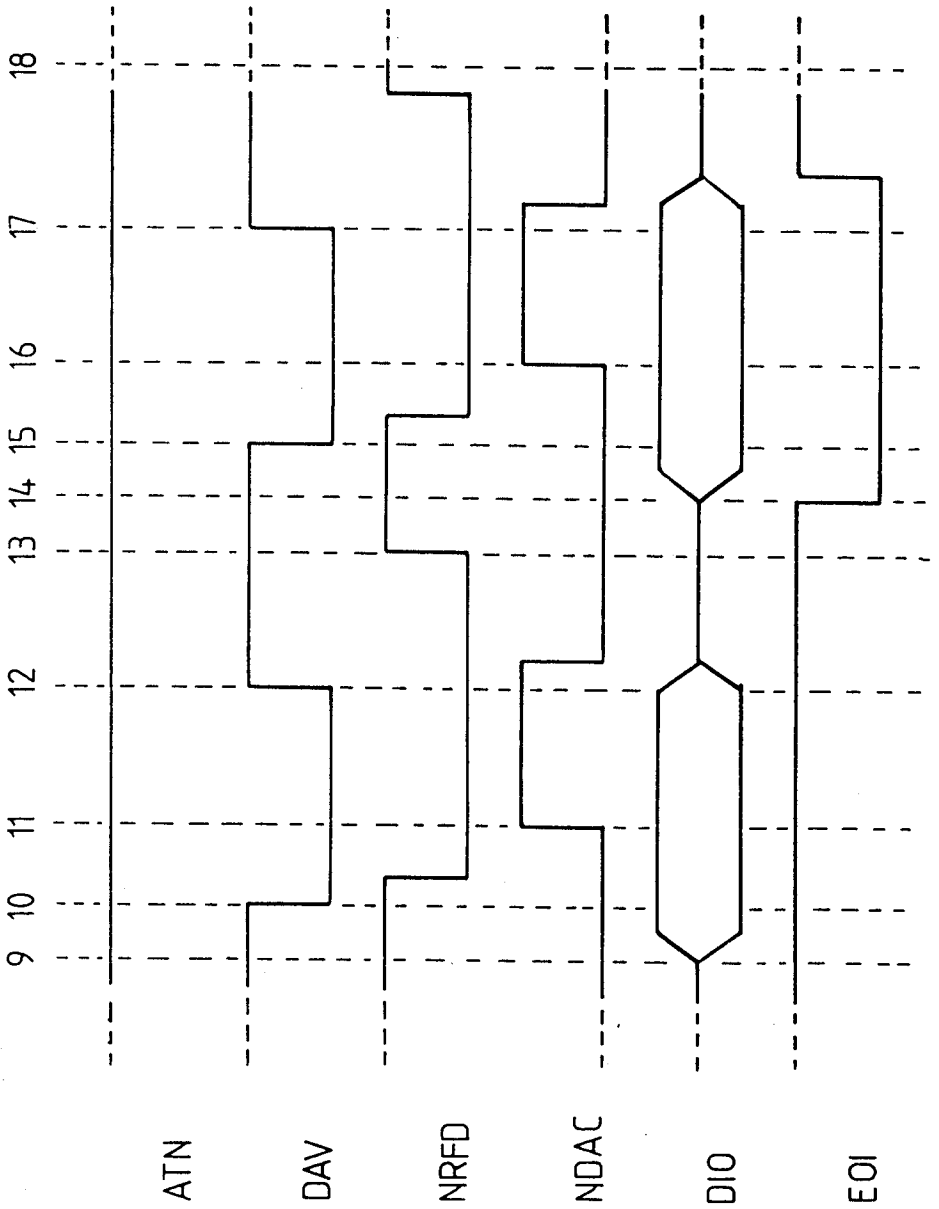
12DAV wordt weer high en de gegevens worden van de datalijnen gehaald.

13De printer heeft de A verwerkt en geeft door NRFD aan dat hij voor andere acties gereed is.

14-15In regel 20 van het BASIC-programma stond alleen PRINT#1,"A", maar het bedrijfssysteem voegt automatisch een chr\$(13) toe (= einde van de regels). Dat staat nu op de datalijnen. Omdat dit de laatste tekens zijn die moeten worden overgedragen, wordt tijdens deze cyclus ook de lijn EOI actief.

16-18De rest verloopt als voorheen en de bus gaat in de rusttoestand.

Gegevens-overdracht



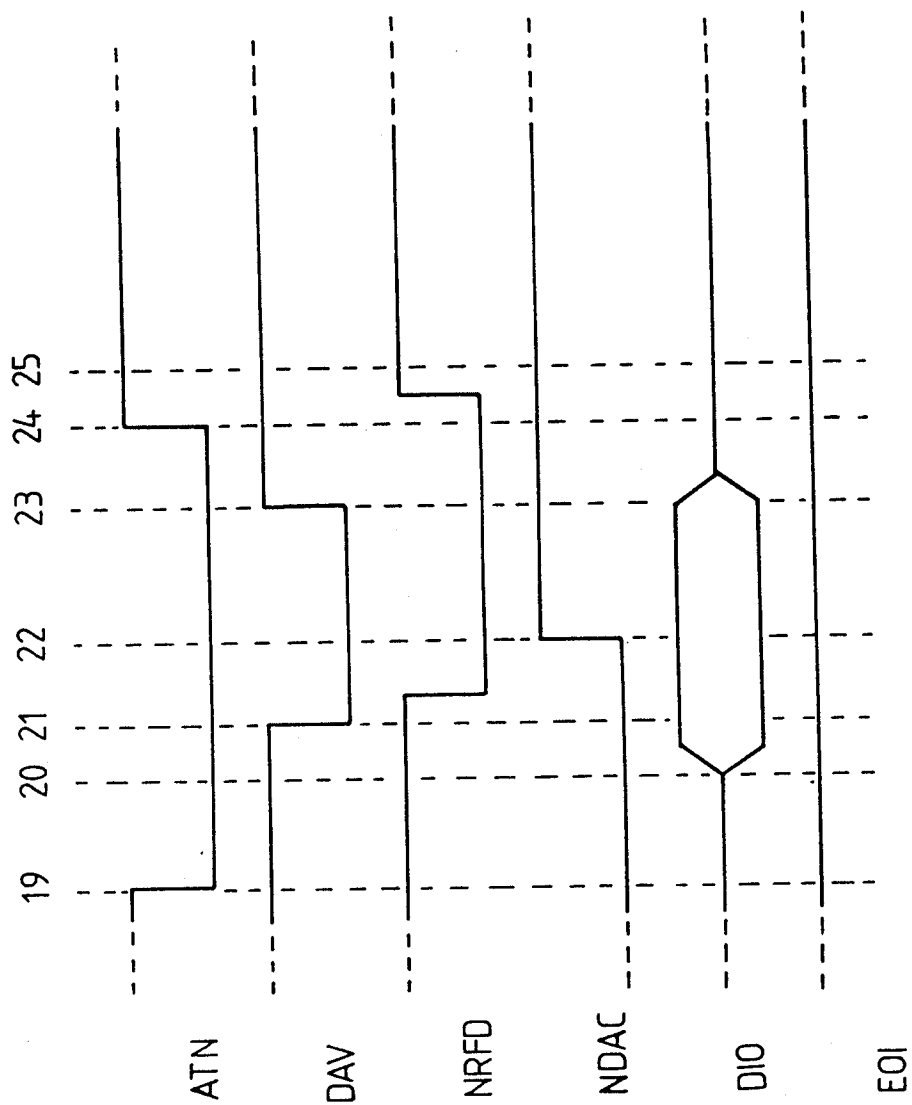
4.8.2.4 adressering beëindigen

19ATN informeert de apparaten dat er weer een instructie volgt.

20-21Die wordt als byte met de waarde 63 ($48 + 15$) op de datalijnen gezet. Net als in paragraaf 4.8.2.1.2 staat, is 48 de beëindiging van de listen-modus waarbij het minst significante halfbyte altijd =15.

22-25De volgende acties zijn dezelfde als onder 5-8 en de bus gaat in de rusttoestand.

Ontkoppelen



4.8.3 De seriële IEC-bus van de Commodore 64

De IEC-bus van de Commodore 64 lijkt aan de buitenkant helemaal niet op het systeem dat we net beschreven. Dit verschil in vorm neemt niet weg, dat de logica ervan voor het grootste deel met de afspraken van 1974 overeenkomt. Het is moeilijk aan te geven waarom Commodore van de standaard IEC-bus is afgeweken. Het kan zijn dat men wegens plaatsgebrek een kleinere aansluiting heeft gekozen. Of misschien wilde men bewust een bijzondere en goedkope reeks randapparatuur aanbieden, omdat bijvoorbeeld een floppy van het type CBM 8050 voor hobbydoeleinden te hoog gegrepen is.

4.8.3.1 Basiskennis

De seriële IEC-bus heeft maar vijf lijnen:

1. SRQ betekenis als in paragraaf 4.8.2.1.1
2. GND
3. ATN betekenis als in paragraaf 4.8.2.1.1
4. CLK (clock) Omdat de gegevens niet meer per byte (8 bits in een keer) maar bit-serieel worden doorgegeven, geeft de talker bij elke bit een klokpuls op de lijn CLK. De puls kenmerkt de geldigheid van de datalijn.
5. DATA is de enige lijn, waarover een byte serieel kan worden geschoven met het minst significante bit voorop.

Het hierboven beschreven proces met DAV, NRFD, NDAC enzovoort is verborgen in de seriële datastroom van de lijnen CLK en DATA. Aan de tijd tussen de sprongen kan van de lijnen wordt afgelezen of het om NDAC, EOI of iets anders gaat. De volgende paragrafen verduidelijken alleen het principe. Voor het gebruik van eigen randapparatuur zijn ze niet zonder meer geschikt. U moet daarvoor meetapparatuur hebben, waarmee u de processen op de seriële bus zichtbaar kunt maken. Op die manier kunt u dan uw conclusies trekken. Hieronder hanteren we het voorbeeld dat we in de paragrafen 4.8.2.2 - 4.8.2.4 hebben behandeld.

4.8.3.2 Adressering

1Net als in paragraaf 4.8.2.2 wordt ook hier ATN low ten teken dat een instructie volgt.

2Als reactie hierop moet het apparaat binnen 1millisec. de datalijn naar low brengen. De computer geeft vervolgens door CLK=high aan dat een byte klaar staat.

3De lijn DATA neemt de functie NRFD over. Zolang ze low is, kan het apparaat geen gegevens opnemen. Als het zover is wordt DATA=high.

3-4Het adres van het apparaat komt over de lijn DATA: een bit met iedere highpuls op de lijn CLK. CLK is ongeveer met DAV te vergelijken.

5Binnen 1 millisec. moet de printer de lijn DATA naar low brengen. Dit correspondeert met NDAC.

DATA heeft nu al drie functies. In de terminologie van de parallelle IEC-bus zijn dat DIO, NRFD en NDAC.

Bij de grote IEC-bus worden de datalijnen uitsluitend geregeld door de talker van dat moment. Hier wisselen talker en listener elkaar af.

4.8.3.3 Overdracht van gegevens

6De overdracht begint als de talker CLK high maakt. Dit betekent dat er een byte gereed staat.

7Als de listener klaar is voor ontvangst, maakt hij DATA high (functie NRFD).

8-9Bij elke klokpuls gaat er een bit over de lijn DATA.

9De listener bevestigt de ontvangst van de complete byte (de A uit regel 20) door DATA=low. Ook hier is sprake van de dubbele functie NDAC en NRFD.

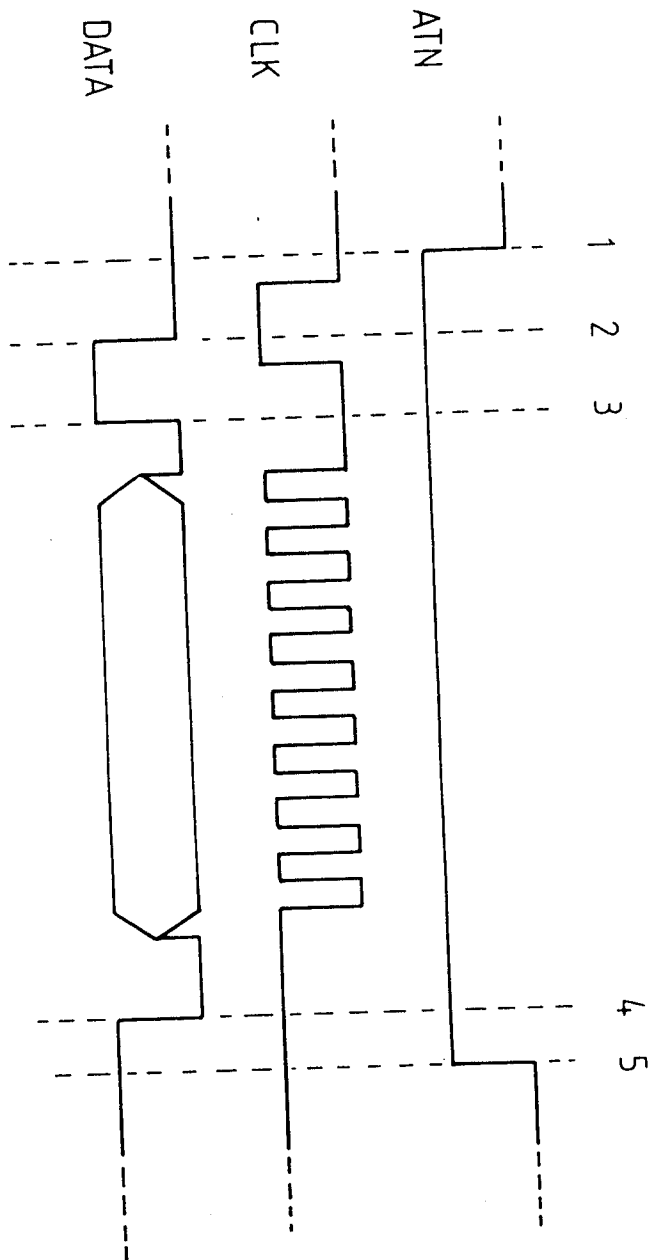
11-12Op dezelfde manier volgt nu het afsluitende chr\$(13). Daarna komt het signaal EOI, de vierde functie van de lijn DATA.

13-14Nadat de listener de lijn DATA high had gemaakt (NRFD), moet de talker eigenlijk binnen 0,2millisec. door CLK=low de overdracht van gegevens inleiden. Maar nu blijft CLK langer high. Dit betekent dat de byte die werd overgedragen het laatste was. Ook die situatie moet de listener met DATA=low bevestigen.

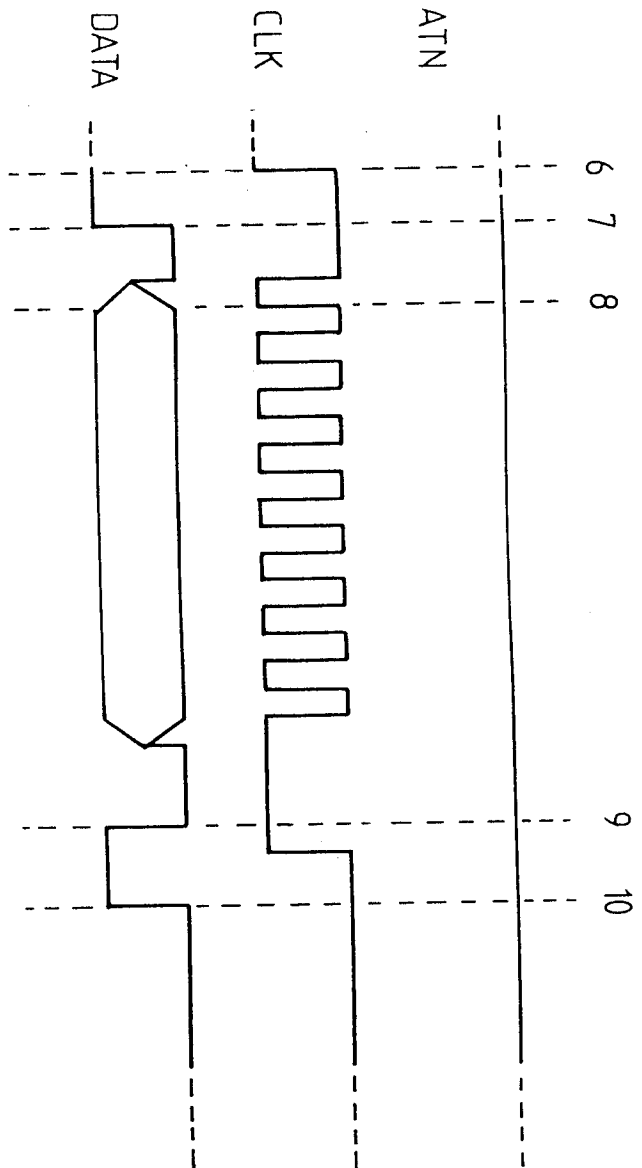
15Wanneer DATA weer high wordt (NRFD), geeft de talker een lege byte door (chr\$(0) vanaf punt 16). Vervolgens gaat de bus in rusttoestand. (

Op dezelfde manier als bij punt 1-5 beschreven werd, waarbij de byte de waarde 63 heeft, wordt de adressering ongedaan gemaakt. Afhankelijk van het systeem is een seriële IEC-bus aanmerkelijk langzamer dan een parallelle. De korte wachttijden 1millisec. tegenover 64millisec. ondervangen dit euvel maar ten dele.

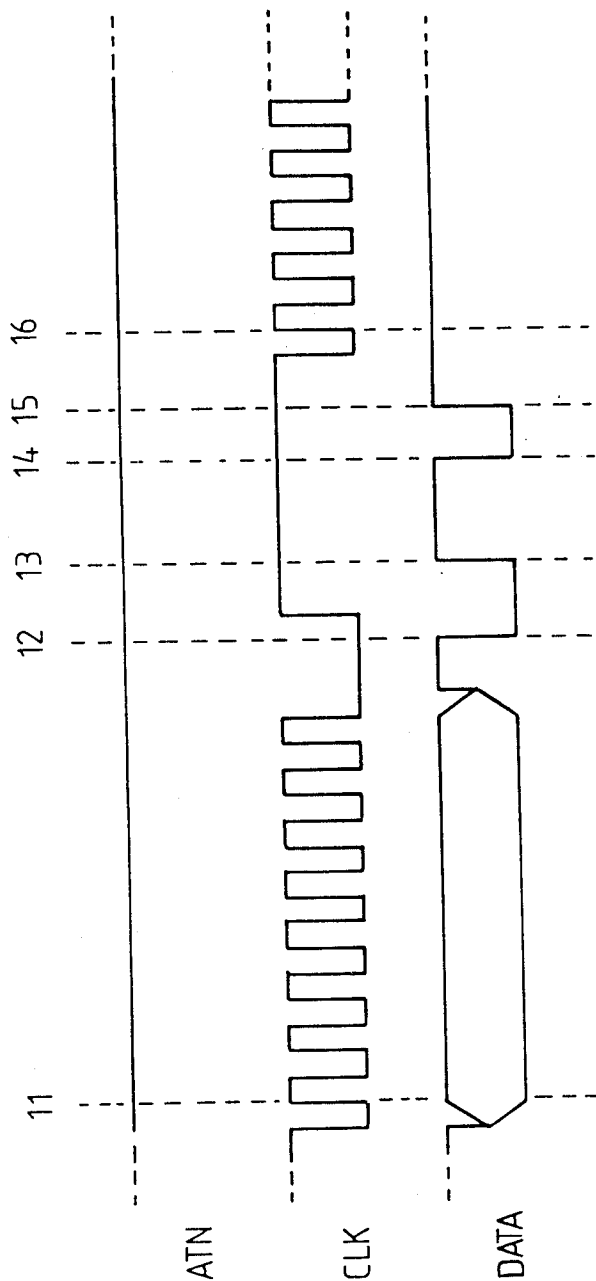
Adressering



Gegevensoverdracht



Laatste gegevensbyte (EOI)



5. BASIC VANUIT EEN ANDER PERSPECTIEF

5.1 De werking van de BASIC-interpreter

De Commodore 64 heeft op ROM een BASIC-interpreter die ontwikkeld is uit de interpreter en die indertijd door MICROSOFT (MICROSOFT BASIC) voor de Commodore PET 2001 is geconstrueerd. Het nummer van deze versie van BASIC is 2.0. Qua functies komt ze overeen met de 3000-serie van Commodore. De werking van de interpreter gaat als volgt: als u een regel invoert, kijkt de interpreter of er BASIC-instructies in voorkomen. Is dat het geval dan verandert hij het woord in een byte: dit is de interpretercode of token. Vervolgens zet hij de regel met behulp van het nummer op de juiste plaats in het programma. Als het programma wordt uitgevoerd, herkent hij een instructie aan de interpretercode. De meest significante bit (bit 7) van deze lbytes code is altijd ingesteld; de waarde is dus groter dan 127. Uit een tabel haalt hij daarna het adres van de routine die de instructie als subroutine uitvoert. De volgende instructie wordt vervolgens met een lus gehaald en uitgevoerd. De adressen van de instructies staan in hoofdstuk 7, de routine zelf wordt in hoofdstuk 8 uitvoerig beschreven. Voordat u eigen functies en routines gaat gebruiken, moet u weten hoe de interpreter de gegevens opslaat.

BASIC onderscheidt drie typen gegevens: reële getallen, hele getallen en strings. Reële getallen liggen tussen +/- 1E-39 en +/- 1E-38. Integere getallen zijn de hele getallen tussen -32768 en 32768. Strings zijn reeksen tekens met een lengte tot 255 karakters. De invoer van variabelen neemt 7 bytes in beslag. De eerste twee zijn de naam (ASCII-code) en de volgende vijf geven de waarde aan. Bij reële getallen is het eerste byte de exponent en zijn de volgende vier de mantissa (halflogaritmische afbeelding). Bij integere getallen zijn maar twee bytes in gebruik: de high- en lowbyte van de binaire constellatie die uit 16 bits bestaat. Bij strings geeft de eerste byte de lengte aan (0-255), de volgende twee geven het adres. Om de typen variabelen te kunnen onderscheiden wordt bij gehele getallen de meest significante bit van de eerste en tweede letter van de naam gezet. Bij strings is de meest significante bit van het tweede teken ingesteld. De variabelentabel in het RAM begint direkt na het BASIC-programma. Een wijzer in de zeropage geeft het adres aan (\$2D/\$2E, 45/46).

Als u de volgende waarden geeft:

A = 10.3
B% = -23
C\$ = "Commodore 64"

ziet de inhoud van het geheugen er op het beeldscherm als volgt uit:

002D	03 08 variabelentabel begint bij \$0803
0803	41 00 84 24 CC CC CD variabele A, reëel getal 10.3
080A	C2 80 FF E9 00 00 00 variabele B%, binaire waarde -23
0810	43 80 0C F4 9F 00 00 variabele C\$, lengte 12, adres \$9FF4

5.2 Van invoer tot verwerking

Op de hieronder beschreven manier kunnen variabelen of uitdrukkingen vanuit BASIC aan machinetaalprogramma's worden doorgegeven. De BASIC-interpreter heeft een routine die een willekeurige uitdrukking haalt en interpreteert. Er wordt met reële getallen gerekend. Integere getallen worden eerst in reële omgezet. Deze routine heet FRMEVL (formule-interpretatie), heeft \$AD9E als adres en interpreteert numerieke en stringparameters. Ter onderscheiding wordt een typeflag gezet (adres \$0D, 13 - \$00 is numeriek, \$FF is string). Numerieke getallen worden in de eerste accu voor reële getallen (FAC) opgeslagen. Voor rekenkundige operaties staat een tweede accu voor reële getallen ter beschikking (ARG), (vanaf adres \$69 (105 in het decimale stelsel). Nadat zo'n routine is opgeroepen, staat het resultaat altijd in de FAC. Ook bij het oproepen van functies wordt het argument aan de FAC overgedragen en het resultaat daar weer opgehaald. Na de interpretatie van een string wijst de wijzer op \$64/\$65 (100/101) naar de stringdescriptor waarin lengte en adres van de string zijn opgeslagen. Als routine \$B475 wordt opgeroepen, wordt de lengte van de string uit de accu en het adres uit het X-(lowbyte) en Y-register (highbyte) gehaald. Hieronder enkele nuttige routines van de BASIC-interpreter. Ze vervullen rekenkundige functies.

adres functie

```
=====
$B853  min          FAC=ARG-FAC
$B86A  plus         FAC=ARG+FAC
$BA28  vermenigvuldigen FAC=ARG*FAC
$BB12  delen        FAC=ARG/FAC
$BF7B  machtsverheffen FAC=ARG^FAC
```

De adressen van de andere routines en het gebruik van de zeropage door de interpreter en het bedrijfssysteem staan in hoofdstuk 7.

5.3 Doe meer met BASIC

5.3.1 Het uitbreiden van BASIC

Enkele machinetaalroutines zijn eleganter in BASIC te formuleren dan via de USR- en SYS-functies. De plaats in de interpreter die een BASIC-instructie haalt en uitvoert, heeft de volgende ROM-listing:

A7E1 6C 08 03 JMP (\$0308); wijst gewoonlijk op \$A7E4

A7E4 20 73 00 JSR \$0073 ; volgende teken uit

 ; BASIC-tekst halen

A7E7 20 A7 ED JSR \$A7ED ; instructie uitvoeren

A7EA 4C AE A7 JMP \$A7AE ; terug naar interpreterlus

Hier kan worden ingegrepen. Op adres \$0308/\$0309 staat een wijzer die op een eigen routine kan worden gezet om de code te herkennen. Het is gangbaar om eigen uitbreidingen door een speciaal teken (bijvoorbeeld een uitroepteken) vooraf te laten gaan.

100 !PRINT

roept een aangepaste printerroutine op. Het kenmerk van de routine is het uitroepteken. Als dat wordt aangetroffen, kan op de eigen routine worden overgegaan. Zo niet, dan wordt de routine

van de BASIC-interpreter opgeroepen. Dat deel van het programma kan er als volgt uitzien.

```

DECODE    JSR $0073    ; CHRGET, volgend teken
          CMP #"! "    ; met speciaal teken vergelijken
          BEQ FOUND    ; aftakken naar eigen routine
          JSR $0079    ; CHRGOT, opnieuw flags zetten
          JMP $A7E7    ; instructie van interpreter volgen
FOUND     JSR $0073    ; CHRGET, volgend teken halen
          JSR COMMAND  ; eigen instructie uitvoeren
          JMP $A7AE    ; terug naar interpreterlus

```

De wijzer op \$0308/\$0309 moet bij het starten van de eigen BASIC-instructies op het adres DECODE van het vorige voorbeeld worden gezet. Moeten er meer dan een instructie worden uitgevoerd, dan kunt u ter onderscheiding daarvan een routine inbouwen, die ze selecteert.

5.3.2 HARDCOPY - RENEW - PRINT USING

Hieronder enkele voorbeelden voor eigen routines in BASIC.

Voorbeeld 1: hardcopyfunctie

Met de hardcopyfunctie kan de inhoud van het beeldscherm op de printer (apparaat nummer 4) worden afgedrukt. Deze kan direkt met SYS 9*4096 worden opgeroepen:

```

; HARDCOPY FUNCTIE
9000 A9 04          LDA #4      ;
9002 85 BA          STA FA      ; nummer van printer
9004 A9 7E          LDA #126    ;
9006 85 B8          STA LF      ; logisch filennummer
9008 A9 00          LDA #0      ; adres low van beeldscherm
900A A0 04          LDY #4      ; adres high van beeldscherm
900C 85 71          STA TEMP    ; als wijzer kenmerken
900E 84 72          STY TEMP+1;
9010 85 B7          STA FNLEN   ; geen filenamen
9012 85 B9          STA SA      ; secundair adres op nul
9014 20 C0 FF       JSR OPEN    ; printerfile openen
9017 A6 B8          LDX LF      ; logisch filennummer van printer
9019 20 C9 FF       JSR CKOUT   ; printer bestemd voor uitvoer
901C A2 19          LDX #25     ; aantal regels van beeldscherm

```

```

901E A9 OD      LOOP      LDA #13 ; nieuwe regel
9020 20 D2 FF      JSR BSOUT ; aan printer
9023 20 E1 FF      JSR STOP ; stoptoets opvragen
9026 F0 2E      BEQ EXIT ; is er gedrukt, dan beëindigen
9028 A0 00      LDY #0 ;
902A B1 71      LOOP2     LDA (TEMP),Y ; teken van beeldscherm halen
902C 85 67      STA STORE ;
902E 29 3F      AND #$3F ;
9030 06 67      ASL STORE ;
9032 24 67      BIT STORE ; beeldschermcode
9034 10 02      BPL *+4 ; in ASCII-code wijzigen
9036 09 80      ORA #$80 ;
9038 70 02      BVS *+4 ;
903A 09 40      ORA #$40 ;
903C 20 D2 FF      JSR BSOUT ; en naar de printer vragen
903F C8      INY ;
9040 C0 28      CPY #40 ; regel afgelopen?
9042 D0 E6      BNE LOOP2 ;
9044 98      TYA ;
9045 18      CLC ; zo ja, wijzer op volgende
9046 65 71      ADC TEMP ; regel zetten
9048 85 71      STA TEMP ;
904A 90 02      BBC *+4 ;
904C E6 72      INC TEMP+1;
904E CA      DEX ; zijn alle regels al uitgevoerd?
904F D0 CD      BNE LOOP ;
9051 A9 OD      LDA #13 ;
9053 20 D2 FF      JSR BSOUT ; nieuwe regel
9055 20 CC FF      JSR CLRCH ; uitvoer weer op beeldscherm
9059 A9 7E      LDA #126 ;
905B 4C C3 FF      JMP CLOSE ; printbestand sluiten - klaar

```

Nu het equivalent in BASIC:

```

100 POKE 56,9*16 : CLR : FOR I=36864 TO 36957
110 READ X : POKE I,X : S=S+X
120 DATA 169, 4,133,186,169,126,133,184,169, 0,160, 4
130 DATA 133,113,132,114,133,183,133,185, 32,192,255,166
140 DATA 184, 32,201,255,162, 25,169, 13, 32,210,255, 32
150 DATA 225,255,240, 46,160, 0,177,113,133,103, 41, 63
160 DATA 6,103, 36,103, 16, 2, 9,128,112, 2, 9, 64
170 DATA 32,210,255,200,192, 40,208,230,152, 24,101,113
180 DATA 133,113,144, 2,230,114,202,208,205,169, 13, 32
190 DATA 210,255, 32,204,255,169,126, 76,195,255
200 IF S<>12023 THEN PRINT "FOUT IN DATA!": END
210 PRINT "OK"

```

Voorbeeld 2: RENEW

Dit programma is nuttig als u toevallig door NEW een programma heeft gewist. Het eind van het gewiste programma wordt opgespoord en als er geen nieuwe regels worden ingevoerd of variabelen gebruikt, wordt de BASIC-wijzer weer op de oude waarde gezet. Het beginadres is $(12 \cdot 4096 + 15 \cdot 256) = 52992$.

					; RENEW FUNCTIE
					; haalt gewist programma terug
CF00 A5 2B		LDA PRGSTRT			; start van BASIC-programma
CF02 A4 2C		LDY PRGSTRT+1			
CF04 85 22		STA TEMP			; als wijzer opslaan
CF06 84 23		STY TEMP+1			
CF08 A0 03		LDY #3			
CF0A C8	NULL	INY			
CF0B B1 22		LDA (TEMP),Y			; zoekt einde van eerste regel
CF0D D0 FB		BNE NULL			; nulbyte
CF0F C8		INY			
CF10 98		TYA			
CF11 18		CLC			
CF12 65 22		ADC TEMP			; offset optellen
CF14 A0 00		LDY #0			
CF16 91 2B		STA (PRGSTRT),Y			; als wijzer op volgende
CF18 A5 23		LDA TEMP+1			
CF1A 69 00		ADC #0			; regel opslaan
CF1C C8		INY			
CF1D 91 2B		STA (PRGSTRT),Y			
CF1F 88		DEY			; bevat nu nul
CF20 A2 03	TO	LDX #3			
CF22 E6 22	TDREIO	INC TEMP			
CF24 D0 02		BNE *+4			; eind programma: direkt
CF26 E6 23		INC TEMP+1			; drie nulbytes zoeken
CF28 B1 22		LDA (TEMP),Y			
CF2A D0 F4		BNE TO			
CF2C CA		DEX			
CF2D D0 F3		BNE TDREIO			
CF2F A5 22		LDA TEMP			
CF31 69 02		ADC #2			
CF33 85 2D		STA PRGEND			
CF35 A5 23		LDA TEMP+1			; wijzer op eind van
CF37 69 00		ADC #0			; programma zetten
CF39 85 2E		STA PRGEND+1			
CF3B 4C 63 A6		JMP CLR			; CLR en ready

Nu weer het equivalent in BASIC. Voordat dit programma op per ongeluk gewiste BASIC-programma's kan worden losgelaten moet het natuurlijk eerst geladen en gestart worden.

```

100 FOR I=52992 TO 53053
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 165, 43,164, 44,133, 34,132, 35,160, 3,200,177
130 DATA 34,208,251,200,152, 24,101, 32,160, 0,145, 43
140 DATA 165, 53,105, 0,200,145, 43,136,162, 3,230, 34
150 DATA 208, 2,230, 35,177, 34,208,244,202,208,243,165
160 DATA 34,105, 2,133, 45,165, 35,105, 0,133, 46, 76
170 DATA 99,166
180 IF S<>7000 THEN PRINT "FOUT IN DATA !" : END
190 PRINT "OK"

```

Een hele handige routine die de interpreter van de Commodore 64 niet heeft is de geformateerde uitvoer van getallen (PRINT USING). Deze routine functioneert als volgt:

Eerst wordt een numeriek getal uit de accu voor integere getallen gehaald. Met de routine van de BASIC-interpreter wordt dat getal in een uit getallen bestaande ASCII-string veranderd die vanaf adres \$100 wordt opgeslagen. Op dit punt grijpt de routine in en voert de formatering uit. Daarna wordt de geformateerde string uitgevoerd. Een oproep via de SYS-functie heeft het volgende formaat:

SYS (AD) X

AD is het startadres van de routine en X de variabele die moet worden uitgedrukt of een positief numeriek getal. De parameters worden met POKE's ingesteld:

POKE 51612, X	0=heel getal, 1=getal in decimalen
POKE 51613, L	0-10, totale lengte-1
POKE 51614, N	aantal cijfers na de komma
POKE 51615, ASC(" ")	teken dat plaats voor getal opvult
POKE 51549, ASC(" ")	teken dat aan getal voorafgaat

De vooraf gekozen waarden zijn: decimaal getal, lengte 10 (9+1), 2 cijfers na de komma, spatie als teken dat aan getal voorafgaat, en als teken dat plaats voor getal opvult.

SYS (51456) 100 resulteert in

100.00.

Door

POKE 51614,3 : POKE 51615,ASC("*") : POKE 51549,ASC("\$")

ziet de uitvoer er zo uit:

**\$100.00

Als u de uitvoer wilt printen moet u de CMD-modus gebruiken:

OPEN 1,4 : CMD 1
SYS (51456) X
PRINT#1 : CLOSE1

Voorbeeld 3: PRINT USING

```
                ; PRINT USING
C900 20 8A AD      JSR FRMNUM      ; numerieke expressie halen
C903 20 DD BD      JSR ASCII      ; in ASCII omzetten
C906 20 0D C9      JSR USING
C909 20 1E AB      JSR OUT        ; string uitvoeren
C90C 60           RTS
C90D A9 45        USING LDA #"E"
C90F 20 8E C9      JSR CHECK      ; test op exponentweergave
C912 B0 59        BCS SETPTR
C914 AD 9C C9      LDA DECINT     ; flag voor decimaal of geheel
C917 F0 59        BEQ INTEGER    ; getal
C919 AD 02 01      LDA $102
C91C D0 0B        BNE L1
C91E AC 9D C9      LDY LENGHT    ; totale lengte - 1
C921 A9 30        LDA #"O"
C923 99 02 01      L2 STA $102,Y  ; buffer met nullen vullen
C926 88           DEY
C927 D0 FA        BNE L2
C929 A9 2E        L1 LDA #". "
C92B 20 8E C9      JSR CHECK
C92E A8           TAY
C92F 90 02        BCC *+4
C931 A0 30        LDY #"O"
C933 A9 00        LDA #0
C935 20 8E C9      JSR CHECK
C938 98           TYA
C939 9D 00 01      STA $100,X
C93C A9 2E        LDA #". "
C93E 20 8E C9      JSR CHECK
C941 AC 9E C9      LDY DECLN     ; aantal decimale posities
C944 E8           L3 INX
C945 88           DEY
C946 D0 FC        BNE L3
```


C948 EC 9D C9	L8	CPX LENGHT	
C94B B0 20		BCS SETPTR	
C94D AC 9D C9		LDY LENGHT	
C950 A9 00		LDA #0	
C952 99 01 01		STA \$101,Y	
C955 BD 00 01	L6	LDA \$100,X	
C958 C9 20		CMP #" "	; eerste teken
C95A D0 02		BNE L5	
C95C A9 20		LDA #" "	
C95E 99 00 01	L5	STA \$100,Y	
C961 CA		DEX	
C962 10 06		BPL L4	
C964 AD 9F C9		LDA FILLER	
C967 88		DEY	
C968 10 F4		BPL L5	
C96A 88	L4	DEY	
C96B 10 E8		BPL L6	
C96D A9 00	SETPTR	LDA #0	; pointer op buffer zetten
C96F A0 01		LDY #1	
C971 60		RTS	
C972 A9 00	INTEGER	LDA #0	
C974 20 8E C9		JSR CHECK	
C977 90 F4		BCC SETPTR	
C979 8A		TXA	
C97A A8		TAY	
C97B AD 02 01		LDA \$102	
C97E F0 09		BEQ L7	
C980 A9 2E		LDA #". "	
C982 20 8E C9		JSR CHECK	
C985 90 02		BCC L7	
C987 8A		TXA	
C988 A8		TAY	
C989 98	L7	TYA	
C98A AA		TAX	
C98B CA		DEX	
C98C 10 BA		BPL L8	
C98E A2 00	CHECK	LDX #0	
C990 DD 00 01	L9	CMP \$100,X	
C993 F0 06		BEQ L10	
C995 E8		INX	
C996 E0 0C		CPX #12	
C998 D0 F6		BNE L9	
C99A 18		CLC	
C99B 60	L10	RTS	
C99C 01	DECINT	.BYT 1	; decimaal
C99D 09	LENGHT	.BYT 9	; lengte 9
C99E 02	DECLEN	.BYT 2	; aantal decimaalposities
C99F 20	FILLER	.BYT " "	; vultekens
C9A0	LEADING	= L5 - 1	; eerste teken

Het equivalent in BASIC:

```
100 FOR I=51456 TO 51615
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 32,138,173, 32,221,189, 32, 13,201, 32, 30,171
130 DATA 96,169, 69, 32,142,201,176, 89,173,156,201,240
140 DATA 89,173, 2, 1,208, 11,172,157,201,169, 48,153
150 DATA 2, 1,136,208,250,169, 46, 32,142,201,168,144
160 DATA 2,160, 48,169, 0, 32,142,201,152,157, 0, 1
170 DATA 169, 46, 32,142,201,172,158,201,232,136,208,252
180 DATA 236,157,201,176, 32,172,157,201,169, 0,153, 1
190 DATA 1,189, 0, 1,201, 32,208, 2,169, 32,153, 0
200 DATA 1,202, 16, 6,173,159,201,136, 16,244,136, 16
210 DATA 232,169, 0,160, 1, 96,169, 0, 32,142,201,144
220 DATA 244,138,168,173, 2, 1,240, 9,169, 46, 32,142
230 DATA 201,144, 2,138,168,152,170,202, 16,182,162, 0
240 DATA 221, 0, 1,240, 6,232,224, 12,208,246, 24, 96
250 DATA 0, 9, 2, 32
260 IF S<>18657 THEN PRINT "FOUT IN DATA !" : END
270 PRINT "OK"
```

5.3.3 Mathematische routines naar eigen ontwerp

Wie vaak wiskundige routines nodig heeft die niet tot de standaarduitrusting van de interpreter behoren, kan het beste zelf een programma in machinetaal schrijven. Als een functie slechts 1 argument heeft, kan men de USR-functie gebruiken. Net als alle andere functie-oproepen van de interpreter (bijvoorbeeld de SIN-functie) kan ze in instructies voor de berekening van variabelen of in een PRINT-instructie staan. Om de interpreter te laten weten waar de eigen routine staat, wordt hem in twee geheugenplaatsen het startadres van de routine meegegeeld. Dat kan in BASIC door POKE-instructies. Bij de oproep van de USR-functie wordt de waarde van het argument in de eerste accu voor reële getallen (FAC) geladen. De functie kan dan worden berekend. Als het resultaat in de FAC staat en RTS (RETURN van het programma in machinetaal) wordt uitgevoerd, wordt de waarde weer aan BASIC overgedragen. De volgende voorbeelden laten zien hoe u eigen functies schrijft.

5.3.4 SQR-, SUM- en PROD-functie

Eerst een routine voor het trekken van tweedemachtswortels. De BASIC-interpreter heeft weliswaar al zo'n routine, maar deze nieuwe is sneller en nauwkeuriger. Hij maakt namelijk gebruik van herhalingen en verheft niet zoals de interpreter eerst tot de tweede macht, wat het oproepen van LOG en EXP vereist. Als beginwaarde nemen we het argument en halveren de exponent. Dat levert al een goede schatting van de wortel op. De herhalingen komen tot stand volgens het principe $x(n+1)=(x(n)+a/x(n))/2$. Hierbij is het a argument, $x(n)$ de oude waarde en $x(N+1)$ de nieuwe geschatte waarde. Na vier herhalingen verandert het resultaat niet meer.

C800	20	2B	BC		JSR SIGN	; voorteken controleren
C803	FO	34			BEQ EIND	; waarde=0: klaar
C805	10	03			BPL OK	; positief: in orde
C807	4C	48	B2		JMP ILL	; negatief: illegal quantity
C80A	20	C7	BB	OK	JSR FACA4	; FAC naar accu#4 overbrengen
C80D	A5	61			LDA EXP	
C80F	38				SEC	
C810	E9	81			SBC #\$81	; exponent normaliseren
C812	08				PHP	
C813	4A				LSR A	; exponent halveren
C814	18				CLC	
C815	69	01			ADC #1	
C817	28				PLP	
C818	90	02			BCC S1	
C81A	69	7F			ADC #\$7F	; exponent weer invoeren
C81C	85	61		S1	STA EXP	
C81E	A9	04			LDA #4	; 4 herhalingen
C820	85	67			STA COUNT	
C822	20	CA	BB	ITER	JSR FACA3	; FAC naar accu#3
C825	A9	5C			LDA #\$5C	
C827	A0	00			LDY #\$00	; wijzer op accu#4
C829	20	0F	BB		JSR DIV	; delen door FAC
C82C	A9	57			LDA #\$57	
C82E	A0	00			LDY #\$00	; wijzer op accu#3
C830	20	67	D8		JSR PLUS	; bij FAC optellen
C833	C6	61			DEC EXP	; FAC/2 (exponent min 1)
C835	C6	67			DEC COUNT	; teller verlagen
C837	D0	E9			BNE ITER	; nog een herhaling
C839	60			EIND	RTS	; klaar

Voordat de nieuwe USR-functie kan worden opgeroepen, moet de interpreter weten waar de functie begint. Daarom poken we de

lowbyte van het adres naar \$311 (785) en de highbyte naar \$312 (786):

POKE 785,0: POKE 786, 12*16+8

Deze POKEs maken deel uit van het laadprogramma in BASIC.

```
100 FOR I=51200 TO 51257
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 32, 43,188,240, 52, 16, 3, 76, 72,178, 32,199
130 DATA 187,165, 97, 56,233,129, 8, 74, 24,105, 1, 40
140 DATA 144, 2,105,127,133, 97,169, 4,133,103, 32,202
150 DATA 187,169, 92,160, 0, 32, 15,187,169, 87,160, 0
160 DATA 32,103,184,198, 97,198,103,208,233, 96
170 IF S<>6211 THEN PRINT "FOUT IN DATA!" : END
180 POKE 785,0 : POKE 786,200 : PRINT "OK"
```

Door ? USR(A) wordt de routine opgeroepen. Het uitvoeren hiervan duurt ongeveer 12 millisec. Het SQR-programma van de interpreter neemt ongeveer 52 msec in beslag. Een iets moeilijker voorbeeld is het optellen van een getallen. Dat is onontbeerlijk bij statistische operaties als de berekening van gemiddelden. We nemen aan dat de gegevens in een array (gedimensioneerde variabele) ter beschikking staan.

```
10 DIM A(1000)
... berekening of inlezen van gegevens
100 S=0
110 FOR I=0 TO 1000: S=S+A(I): NEXT
120 PRINT S
```

Onze USR-functie moet de BASIC-regels 100 en 110 vervangen door

```
100 S=USR(A)
```

Parameter A is de naam van de array. Later zullen we zien dat door twee machinetaalinstructies te vervangen ook het produkt van de arrayelementen kan worden berekend.

033C 20 8D AD	JSR NUMTEST	; variabele numeriek?
033F A6 2F	LDX ARRTAB	;
0341 A5 30	LDA ARRTAB+1	; wijzer op begin van
0343 86 5F	S3 STX TEMP	; arraytabel
0345 85 60	STA TEMP+1	; lopende wijzer
0347 C5 32	CMP ARREND+1	;
0349 D0 04	BNE S1	;
034B E4 31	CPX ARREND	; eind van arraytabel?
034D F0 1D	BEQ NOTFOUND	;
034F A0 00	SL LDY #0	;

0351	B1	5F		LDA (TEMP),Y	; eerste letter van naam
0353	C8			INY	;
0354	C5	45		CMP VARNAM	; met gezochte naam vergelijken
0356	D0	06		BNE S2	; nee: volgende array testen
0358	A5	46		LDA VARNAM+1	; tweede letter
035A	D1	5F		CMP (TEMP),Y	; vergelijken
035C	F0	17		BEQ FOUND	; gevonden
035E	C8		S2	INY	;
035F	B1	5F		LDA (TEMP),Y	;
0361	18			CLC	;
0362	65	5F		ADC TEMP	; offset voor volgende array
0364	AA			TAX	; optellen
0365	C8			INY	;
0366	B1	5F		LDA (TEMP),Y	;
0368	65	60		ADC TEMP+1	;
036A	90	D7		BCC S3	;
036C	A2	E2	NOTFOUND	LDX #<TAB	;
036E	86	22		STX \$22	; wijzer op foutmelding
0370	A9	03		LDA #>TAB	;
0372	4C	45 A4		JMP ERROUT	; foutmelding uitlezen
0375	C8		FOUND	INY	;
0376	B1	5F		LDA (TEMP),Y	;
0378	18			CLC	;
0379	65	5F		ADC TEMP	;
037B	85	24		STA STORE	;
037D	C8			INY	;
037E	B1	5F		LDA (TEMP),Y	;
0380	65	60		ADC TEMP+1	;
0382	85	25		STA STORE+1	;
0384	C8			INY	;
0385	B1	5F		LDA (TEMP),Y	; aantal indices
0387	20	96 B1		JSR SETARR	; wijzer op eerste arrayelement
038A	85	5F		STA TEMP	;
038C	84	60		STY TEMP+1	; wijzer naar TEMP
038E	24	0E		BIT INTFLG	; flag voor integere getallen
0390	30	1F		BMI INTEGER	; testen
0392	20	A2 BB		JSR MEMAC1	; element in FAC
0395	18			CLC	;
0396	90	04		BCC LOOP	; sprong in lus
0398	20	67 B8	S5	JSR MEMPLUS	; variabele plus FAC
039B	18			CLC	;
039C	A5	5F	LOOP	LDA TEMP	;
039E	69	05		ADC #5	; wijzer op volgende element
03A0	85	5F		STA TEMP	;
03A2	90	02		BCC S4	;
03A4	E6	60		INC TEMP+1	;
03A6	A4	60	S4	LDY TEMP+1	;
03A8	C5	24		CMP STORE	; einde van array?
03AA	90	EC		BCC S5	;

```

03AC C4 25          CPY STORE+1      ;
03AE 90 E8          BCC S5            ;
03B0 60             READY RTS          ; ja, klaar
03B1 20 D5 03      INTEGER JSR INTAKK   ; integere variabele naar FAC
03B4 20 0C BC          S6 JSR A1TOA2    ; FAC naar ARG
03B7 18             CLC               ;
03B8 A5 5F          LDA TEMP          ;
03BA 69 02          ADC #2            ; wijzer op volgende
03BC 85 5F          STA TEMP          ; arrayelement
03BE 90 02          BCC S7            ;
03C0 E6 60          INC TEMP+1        ;
03C2 C5 24          S7 CMP STORE      ; einde van arraybereik?
03C4 90 06          BCC S8            ;
03C6 A5 60          LDA TEMP+1        ;
03C8 C5 25          CMP STORE+1       ;
03CA B0 E4          BCS READY         ;
03CC 20 D5 03      S8 JSR INTAKK       ; integere variabele naar FAC
03CF 20 6F B8      JSR AKPLUS         ; FAC+ARG
03D2 4C B4 03      JMP S6             ;
03D5 A0 00      INTAKK LDY #0         ;
03D7 B1 5F          LDA (TEMP),Y      ;
03D9 AA            TAX               ;
03DA C8            INY               ;
03DB B1 5F          LDA (TEMP),Y      ;
03DD A8            TAY               ;
03DE 8A            TXA               ;
03DF 4C 91 B3      JMP INTFLOAT      ; naar vlottende komma
03E2 41 52 52      TAB .ASC "array not found"
03E5 41 59 20 4E 4F 54 20 46 50 55 4E
03F0 C4            .BYTE "D"+$80

```

100 FOR I = 828 TO 1008

```

110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 32,141,173,166, 47,165, 48,134, 95,133, 96,197
130 DATA 50,208, 4,228, 49,240, 29,160, 0,177, 95,200
140 DATA 197, 69,208, 6,165, 70,209, 95,240, 23,200,177
150 DATA 95, 24,101, 95,170,200,177, 95,101, 96,144,215
160 DATA 162,226,134, 34,169, 3, 76, 69,164,200,177, 95
170 DATA 24,101, 95,133, 36,200,177, 95,101, 96,133, 37
180 DATA 200,177, 95, 32,150,177,133, 95,132, 96, 36, 14
190 DATA 48, 31, 32,162,187, 24,144, 4, 32,103,184, 24
200 DATA 165, 95,105, 5,133, 95,144, 2,230, 96,164, 96
210 DATA 197, 36,144,236,196, 37,144,232, 96, 32,213, 3
220 DATA 32, 12,188, 24,165, 95,105, 2,133, 95,144, 2
230 DATA 230, 96,197, 36,144, 6,165, 96,197, 37,176,228
240 DATA 32,213, 3, 32,111,184, 76,180, 3,160, 0,177
250 DATA 95,170,200,177, 95,168,138, 76,145,179, 65, 82
260 DATA 82, 65, 89, 32, 78, 79, 84, 32, 70, 79, 85, 78
270 DATA 196

```

```

280 IF S <> 20399 THEN PRINT "FOUT IN DATA" : STOP
290 POKE 785,3*16+12: POKE 786,3: PRINT "OK"

```

Het programma kan arrays met reële en arrays met integere getallen verwerken. Als een array niet wordt gevonden leidt dat tot een array not found error. De logica voor het berekenen van het produkt van de array-elementen is dezelfde. De produktfunctie wordt verkregen door de oproepen voor optellen te vervangen door de routine voor vermenigvuldigen. Vanaf adres \$0398 moet dan 20 28 BA staan en vanaf adres \$03CF 20 2B BA. Vanuit BASIC kan dat door POKE 921,40: POKE 922,186: POKE 976,43. Om de routine die deze keer in de bandbuffer ligt, te kunnen gebruiken, moet weer het beginadres worden gePOKEt (in het BASIC laadprogramma is dat al gebeurd):

```
POKE 785,3*16+12: POKE 786,3
```

Bereken ter vergelijking eens de som met een BASIC-lus en daarna met de nieuwe routine: het verschil in tijd is verbluffend.

5.3.5 Verandering van formaten van gegevens

Als meer dan een parameter moeten worden overgedragen, is de USSR-functie niet meer geschikt. We moeten onze toevlucht nemen tot een variant van de SYS-instructie. Gewoonlijk voert de SYS-instructie vanaf dit adres alleen het machinetaalprogramma uit en geeft geen andere parameters door. De routine FRMEVL leest een volgende parameter en geeft hem door aan de accu voor integere getallen. Door komma's of haakjes van elkaar gescheiden kunnen op die manier zoveel parameters als nodig worden overgedragen. Behalve de routine die formules interpreteert is er nog een aantal andere inspringpunten en subroutines die parameters interpreteren, die tussen haakjes staan of die controleren op een volgende komma. Ook het type variabele (string of numeriek) kan worden herkend. Bij numerieke variabelen is bovendien nog een controle van het bereik mogelijk. De belangrijkste routines staan hieronder. Voor verder bijzonderheden wordt verwezen naar de ROM-listing of paragraaf 6.4.

adres	functie
-------	---------

AD8A	argument interpreteren en controleren of het numeriek is
AD8D	numeriek?
AD8F	string?
AD9E	interpretatie van willekeurig type argument
AEF1	interpretatie van argument tussen haakjes
AEF7	haakje gesloten?
AEFA	haakje open?
AEFD	komma?
B79E	haalt byte (0-255) uit X-register
0073	haalt volgend teken uit BASIC-tekst

De overschrijding van het bereik is een ILLEGAL QUANTITY ERROR, een verkeerd type levert TYPE MISMATCH op. De verandering van verschillende formaten is met de volgende routines mogelijk:

adres	beschrijving
B1BF	verandert FAC in integer
B395	verandert 16-bits integer getal in A/X in reëel getal
B3A2	verandert byte in Y in reëel getal
BC9B	verandert FAC in 16-bits getal
BCF3	verandert cijferstring in reëel getal
BDDD	verandert FAC in cijferstring

Nu een voorbeeld van een SYS-oproep met overdracht van parameters. Als u vanuit BASIC ergens op het beeldscherm een uitvoer wilt hebben, dan moet u na de HOME-toets te hebben ingedrukt met de cursorbesturing het aantal plaatsen rechts en naar beneden regelen. Dit is omslachtig en neemt veel geheugenruimte in beslag. Het gaat eenvoudiger met een zelfgeschreven routine in machinetaal die de volgende syntax heeft:

SYS PR, kolom, regel, lijst

PR is het beginadres van de routine: kolom en regel bepalen de positie van de cursor, waar net als bij een normale PRINT-instructie de variabelen of andere elementen van de lijst worden uitgelezen.

C000 20 FD AE	JSR CECOM ; controleert of er een komma staat
C003 20 9E B7	JSR GETBYT; haalt waarde van kolom naar X
C006 8A	TXA
C007 48	PHA ; kolomnummer kenmerken
C008 20 FD AE	JSR GETCOM; komma?

COOB 20 9E B7	JSR GETBYT; haalt waarde van regel
COOE 68	PLA
COOF A8	TAY ; waarde van kolom naar Y
CO10 18	CLC
CO11 20 F0 FF	JSR CURSOR; zet cursor op goede plaats
CO14 20 FD AE	JSR CKCOM ; komma?
CO17 4C A4 AA	JMP PRINT ; verder met PRINT-instructie

```

100 FOR I=49152 TO 49177
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 32,253,174, 32,158,183,138, 72, 32,253,174, 32
130 DATA 158,183,104,168, 24, 32,240,255, 32,253,174, 76
140 DATA 164,170
150 IF S<>3566 THEN PRINT "FOUT IN DATA!" : END
160 PRINT "OK"

```

Als u aan het begin van het programma het beginadres \$C000 van de routine aan de variabele PR toewijst, laat de volgende instructie de tekst "voorbeeld" vanaf de vierentwintigste kolom van de twintigste regel op uw beeldscherm verschijnen:

```

10 PR=12*4096
100 SYS PR,24,20, "voorbeeld".

```

6. PROGRAMMEREN IN MACHINETAAL OP DE COMMODORE 64

6.1 De monitor

Om makkelijk in machinetaal te kunnen programmeren heeft u een monitor nodig. Dat is geen beeldscherm, maar een hulpprogramma waarmee geheugenplaatsen en registers kunnen worden veranderd en programma's in machinetaal uitgevoerd, geladen, opgeslagen en gedisassembleerd. Bovendien kunt u met de monitor programma's in machinetaal makkelijk wijzigen en corrigeren. Na de monitor vanaf cassette of diskette te hebben ingelezen, kan hij worden gestart met de normale instructie voor het oproepen van programma's in machinetaal:

```
SYS xxxxx
```

Voor de monitor is dat SYS 12*4096. Dit is het beginadres van de monitor als decimaal getal (hexadecimaal: C000). Daarna meldt de monitor zich met PROMPT. Nu kan hij monitor-instructies ontvangen. In ons geval zien die er zo uit:

```
C*
  PC  IRQ  SR AC XR YR SP
>; E145 EA13 31 32 AC 34 F8
>
```

Ter verduidelijking bekijken we de registers van de 6502 (6510).

programmateller PC

In dit register staat de volgende geheugenplaats die moet worden verwerkt. In dit voorbeeld is dat adres E145.

interruptvector IRQ

Dit register stuurt een mogelijke onderbreking en vertakking van het programma.

statusregister SR

Hierin bevinden zich de toestanden van de flags:

b7 = negatieve flag (negatief resultaat)
b6 = interne overflow (rekenkundige overflow)

b5 = altijd gezet
 b4 = afbrekingsflag (afbreking van programma)
 b3 = decimale flag (wisselt van hexadecimaal naar BCD
 (binary coded decimal))
 b2 = interrupt schakelaar (stuurt interrupt)
 b1 = nulflag (resultaat=0)
 b0 = overdrachtsflag (rekenkundige overdracht)

In de meeste handboeken wordt dat als volgt aangegeven:

b7	b6	b5	b4	b3	b2	b1	b0
N	V	1	B	D	I	Z	C

accumulator AC

In dit register worden alle vergelijkings- en rekenoperaties uitgevoerd.

X-register XR

Dit indexregister dient voor de opname van gegevens tijdens het verloop van het programma

Y-register YR

Zie X-register.

stapelwijzer SP

Dit register wijst op het stapelgeheugen, waarin de adressen van de terugsprong in subroutines zijn opgeslagen.

De inhoud van deze registers kan voortdurend worden gecontroleerd en veranderd. Bedenk, dat alleen hexadecimale getallen kunnen worden ingevoerd. Om de inhoud van de registers te veranderen zet u de cursor boven de waarde en toetst daarna de nieuwe waarde in. Een druk op RETURN effectueert de operatie. Op dezelfde manier kunt u de inhoud van het geheugen behandelen. Dat is de eigenlijke programmering in machinetaal. De volgende instructie wijst een bepaald geheugenbereik aan:

M xxxx yyyy

M staat voor MEMORY, xxxx is het beginadres en yyyy het einde van het geheugenbereik. Alle adressen moeten als hexadecimale getallen van vier cijfers worden ingelezen. Daarna wordt de

inhoud aangewezen. Als het geheugenbereik niet volledig op het scherm past, schuift het regel voor regel naar boven totdat het eindadres bereikt is. Een druk op STOP breekt dit proces af en u bent weer terug bij de instructieregel. De inhoud van het geheugen kan op dezelfde manier worden veranderd als bij de registers: cursor instellen, nieuwe waarde(n) intoetsen, RETURN.

Een bepaald geheugenbereik zou er op het scherm als volgt uit kunnen zien:

```
>M C000 C010

>: C000  A9 10 8D 16 03 A9 C0 8D

>: C008  17 03 A9 43 85 97 D0 16

>: C010  A9 42 85 97 D8 4A 68 8D
```

enzovoort

Door programma's in machinetaal te disassembleren kan het geheugen ook worden afgebeeld. Het voordeel daarvan is dat de hexadecimale getallen begrijpelijke instructiereksen worden, MNEMONICS genaamd.

Maar er is ook een nadeel aan verbonden. Als er in het machinetaalprogramma een tekst staat, probeert de disassembler die instructie als OPERATIECODE te interpreteren en hem in mnemonics te veranderen. Zo'n tekst is geen instructie in machinetaal en gewoonlijk zullen er vraagtekens verschijnen, omdat de opcode niet bestaat. Maar het kan ook gebeuren dat er schijnbaar normale mnemonics ontstaan. Een ervaren programmeur zal snel kunnen vaststellen of het om tekst, variabelen of instructies in machinetaal gaat. Nadat een programma in machinetaal is ingelezen of veranderd, is de uitvoering de volgende stap. Dat gebeurt door

G xxxx

G staat voor GO TO en betekent een sprong naar adres xxxx, het begin of een inspringadres van een programma in machinetaal. Komt het programma tijdens de uitvoering een BRK-(BREAK)-instructie tegen (= de BASIC-instructie STOP), dan wordt de uitvoering afgebroken en meldt de monitor zich weer met B*. Door deze instructie in machinetaalprogramma's in te voegen kunnen ze snel en eenvoudig worden getest. Als dat is gebeurd kan de BRK-instructie tegen de oorspronkelijke worden ingewisseld. Met de monitorinstructie

S "NAME",xx,yyyy,zzzz

kunnen programma's in machinetaal op cassette of diskette worden opgeslagen. S betekent SAVE (opslaan). Een naam voor het programma mag u zelf kiezen, zolang de aanhalingstekens maar niet ontbreken. xx is het adres van cassette (01) of diskette (08), yyyy het begin- en zzzz het eindadres van het programma. Een druk op RETURN effectueert de opslag.

Met de monitorinstructie

L "NAME",xx

worden programma's in machinetaal geladen. L betekent LOAD (laden). Met sommige monitors is het mogelijk programma's op verschillende geheugenplaatsen te laden. Zo kunt u een programma, dat bij adres 0800 begint naar 2800 schuiven. Maar let op! Het kan zijn dat het verschoven programma niet kan wandelen. Dan moeten eerst alle absolute adressen worden veranderd.

Om het programmeren in de monitor af te sluiten is er de X-instructie. X betekent EXIT (uitgang). Op deze manier komt men weer terug in BASIC. Gelukkig blijft daarbij het machinetaalprogramma en de BASIC-programmatuur altijd intact. Let er wel op dat het programma in machinetaal niet op het adres van het BASIC-programma wordt opgeslagen. Dat zou dan over het BASIC-programma heen worden geschreven.

Als u een machinetaalprogramma heeft geladen, geef dan nog voor het starten de instructie NEW. Daardoor worden alle wijzers in de normale toestand gezet. Als dat niet gebeurt kan er bij de invoer van het BASIC-programma al na de eerste regel een foutmelding verschijnen. Ook zonder monitor kan er in machinetaal worden geprogrammeerd. In elk geval kunnen de programma's worden ingelezen en gestart. Maar bedenk wel dat iedere geheugenplaats vanuit BASIC moet worden veranderd. Eerst zou er dus een BASIC-programma moeten worden geschreven, waarmee de operatiecodes kunnen worden ingevoerd. Dat is niet eenvoudig, omdat ze in decimale vorm moeten worden ingelezen: een POKE-instructie verlangt immers decimale getallen. Eventueel zou een routine die hexadecimale in decimale getallen verandert, uitkomst kunnen brengen. Maar bij het uitlezen van de bytes hebben de getallen van de PEEK-uitvoer een decimale vorm. Dan staat u dus voor het tegenovergestelde probleem. Had u aan het begin de beschikking over hexadecimale getallen, terwijl decimale getallen nodig waren. Nu krijgt u decimale getallen, terwijl u alleen iets aan hexadecimale getallen heeft (bijvoorbeeld om ze in een opcode-tabel na te slaan). De aanschaf van een monitor is dus aan te bevelen.

6.2 Nuttige adressen van het bedrijfssysteem van de Commodore 64

Wie zelf programma's in machinetaal schrijft kan door gebruik te maken van ROM-routines veel werk besparen. De routines van de perifere bouwstenen zijn daarvoor uitermate geschikt. De belangrijkste routines van de Commodore 64 staan als sprongtabel op de eigenlijke routines aan het eind van de ROM samengevat. Ze zijn bij nieuwe CBM-computers alleen maar uitgebreid. De routines die voor een grote CBM-computer zijn geschreven, kunt u zonder meer op uw C64 gebruiken zolang het maar om kernal-routines gaat. Op drie extra adressen na is de sprongtabel van de Commodore 64 dezelfde als die van de VIC 20. Daarvan kunt u dus gebruik maken om programma's van de VIC 20 voor de Commodore 64 geschikt te maken. Hieronder een aantal routines voor de Commodore 64.

adres	functie
\$FF90	zet flag voor uitvoer van systeemmeldingen
\$FF93	stuurt secundair adres na een LISTEN-instructie op de IEC-bus
\$FF96	stuurt secundair adres na een TALK-instructie op de IEC-bus
\$FF99	haalt als carry flag is gezet, het hoogste RAM-adres naar X en Y; bij gewiste carry flag wordt het adres van X en Y gezet
\$FF9C	zie boven, maar dan voor begin van RAM
\$FF9F	vraagt toetsenbord op
\$FFA2	zet time-out flag voor IEC-bus
\$FFA5	haalt byte van IEC-bus naar accu
\$FFA8	geeft byte uit accu aan IEC-bus
\$FFAB	zendt UNTALK-instructie op IEC-bus
\$FFAE	zendt UNLISTEN-instructie op IEC-bus
\$FFB1	zendt LISTEN-instructie op IEC-bus

\$FFB4	zendt TALK-instructie naar IEC-bus
\$FFB7	haalt statuswoord naar accu
\$FFBA	zet fileparameters, accu moet logisch filenummer bevatten, X=nummer van apparaat, Y=secundair adres
\$FFBD	zet parameter van filenaam, accu moet lengte van naam bevatten, X en Y bevatten adres van filenaam
\$FFC0	OPEN-instructie, opent logisch bestand
\$FFC3	CLOSE-instructie, sluit logisch bestand
\$FFC6	CHKIN zet volgende invoer op logisch bestand, dat aan X wordt overgedragen. Logisch bestand moet eerst met OPEN-routine worden geopend
\$FFC9	CKOUT zet volgende uitvoer op logisch bestand dat aan X wordt overgedragen. Logisch bestand moet eerst met OPEN-routine worden geopend
\$FFCC	CLRCH zet in- en uitvoer weer in standaardmodus (toetsenbord en beeldscherm)
\$FFCF	BASIN invoer, haalt teken naar accu
\$FFD2	BSOUT uitvoer, geeft teken aan accu
\$FFD5	LOAD, laadt programma in geheugen
\$FFD8	SAVE, slaat programma op
\$FFDB	stelt lopende tijd opnieuw in
\$FFDE	haalt lopende tijd
\$FFE1	vraagt STOP-toets op
\$FFE4	GET, haalt teken naar accu
\$FFE7	CLALL, zet alle in- en uitvoerkanalen terug; de bestanden worden niet gesloten
\$FFEA	verhoogt tijd met 1/60 seconde
\$FFED	SCREEN haalt regels en kolommen van het beeldscherm

\$FFFO als carry flag gewist is, wordt cursor op positie X/Y gezet. Als carry flag is gezet wordt cursorpositie gehaald

\$FFF3 haalt beginadres van I/O-bouwsteen

Er zijn ook routines waarmee het beeldscherm kan worden bediend, bijvoorbeeld de positionering van de cursor (zie boven). De volgende routines zijn het belangrijkste:

adres	functie
\$E518	terugzetten van beeldscherm en opvraagfunctie van toetsenbord
\$E544	CLR, wist beeldscherm
\$E566	HOME, brengt cursor in linker bovenhoek van beeldscherm
\$E56C	berekent positie van cursor
\$E5A0	laadt videocontroller met standaardwaarden
\$E5B4	haalt teken uit buffer van toetsenbord
\$E5CA	wacht op inlezen via toetsenbord
\$E8EA	beeldscherm rollen (scroll), schuift beeldscherm een regel naar boven
\$E9FF	wist regel van beeldscherm
\$EA1C	zet in kleur een teken op beeldscherm (beeldschermcode in accu, kleur in X)

6.3 In- en uitvoer in machinetaalprogramma's

Voor het schrijven van eigen programma's in machinetaal is het vooral voor in- en uitvoer van gegevens handig om op routines van het bedrijfssysteem terug te vallen. Ze staan in een sprongtabel aan het einde van de ROM (zie laatste bladzijde van de ROM-listings).

6.3.1 In- en uitvoer van afzonderlijke bytes

De belangrijkste routines zijn

BSOUT \$FFD2 uitvoer van een byte
en
BASIN \$FFCF invoer van een byte

De byte die uit- of ingelezen moet worden, wordt aan de accu doorgegeven. De accu is het universele register van de processor, waarin alle operaties plaatsvinden.

Voorbeeld: uitvoer van een tekst op het beeldscherm.

```
UITVOER  LDX  #0
L1        LDA  TEKST,X    ; tekst halen
          JSR  BSOUT      ; en uitvoeren
          INX
          CPX  #12        ; reeds alle tekens?
          BNE  L1
          RTS
TEKST     .ASC "voorbeeldtekst"
```

De invoer gebeurt net zo. Als er bijvoorbeeld een tekst via het toetsenbord ingelezen en opgeslagen moet worden, verschijnt de cursor en worden de tekens overgenomen totdat u op RETURN drukt.

```
INVOER   LDX  #0
L1        LDA  BASIN      ; een teken halen
          STA  TEXT,X     ; en opslaan
          INX
          CMP  #13        ; = RETURN?
          BNE  L1        ; nee, dan volgende halen
          RTS
TEKST     =      . . .    ; plaats voor het opslaan van tekst
```

Deze routines laten een teken op het beeldscherm verschijnen of halen een teken van het toetsenbord. Bij de uitvoer op het beeldscherm kan natuurlijk de besturing van het scherm worden gebruikt. Een onderdeel daarvan zijn de codes die de cursor sturen of het beeldscherm wissen. De code moet in de accu worden geladen en samen met de uitleesroutine doorgegeven.

Voorbeeld: beeldscherm wissen

```
LDA  #147    ; code beeldscherm wissen
JSR  BSOUT   ; uitvoeren
```

Speciaal voor uitvoer op het beeldscherm zijn er enkele nuttige routines om de programmering te vereenvoudigen. De routine waarmee het scherm wordt gewist, kan direkt worden opgeroepen:

JSR CLRSCR

Er is ook een routine die de cursor home stuurt:

JSR HOME

Het is handig dat de cursor direkt naar een bepaalde plaats op het scherm kan worden gestuurd:

```
LDX REGEL ; cursorregel 0-24
LDY KOLOM ; cursorkolom 0-39
CLC      ; carry clear, direkt cursor instellen
JSR CURSOR; cursor instellen
LDA #"A" ; uit te voeren teken
JSR BSOUT ; op beeldscherm
```

De subroutine CURSOR heeft twee functies. Als de carryflag is gewist, wordt de cursor op de regel en in de kolom gezet die in het X- en Y-register staan. Wordt de cursor opgeroepen als de carryflag is gezet, dan wordt de momentele cursorpositie gehaald en in het X- en Y-register opgeslagen. Bij deze routines horen de volgende adressen:

```
CLRSCR $E544
HOME   $E566
CURSOR $FFFO
```

6.3.2 In- en uitvoer via de randapparatuur

Om te beginnen behandelen we in het kort het principe van in- en uitvoer. De randapparatuur krijgt een nummer van 0-15, waarmee het door het bedrijfsysteem wordt aangesproken.

nummer	apparaat
0	toetsenbord
1	datasette
2	RS-232 interface
3	beeldscherm
4-15	apparatuur op IEC-bus (printer, floppy)

Bij zo'n primair adres (=adres van het apparaat) komt eventueel een secundair adres, dat de filenaam aangeeft en bepaalt hoe het apparaat werkt. Om niet elke keer alle parameters van een apparaat te moeten aangeven, wordt het logisch filenummer ingevoerd. Met OPEN krijgt elk filenummer een primair adres, een secundair adres en een filenaam. Daarna kan het apparaat met het logisch filenummer worden aangesproken. Voor de eerste in- of uitvoer moet het bestand worden geopend. Dat kan met de BASIC-instructie OPEN, maar ook in machinetaal. Eerst worden de fileparameters ingesteld. Het logisch filenummer moet in \$D8 (184) staan, het nummer van het apparaat in \$BA (186), het secundaire adres in \$B9 (185), de lengte van de filenaam in \$B7 (183) (0 wanneer er geen filenaam is) en het adres van de filenaam in \$BB/\$BC (187/188). Vervolgens roept men de subroutine OPEN \$FFCO op. Als de uitvoer naar het geopende bestand moet, roept u de volgende routine op:

```
LDX LF ; logisch filenummer
JSR CKOUT; $FFC9, uitvoer naar bestand
```

Met de routine BSOUT (zie boven) gaat de uitvoer niet naar het beeldscherm, maar naar het apparaat met het genoemde logisch filenummer. De routine CLRCH zorgt ervoor dat de uitvoer weer op het scherm verschijnt:

```
JSR CLRCH; $FFCC, uitvoer op scherm
```

Het invoeren van gegevens uit een bestand (bijvoorbeeld band of floppy) gaat als volgt:

```
LDX LF ; logisch filenummer
JSR CHKIN; $FFC6
```

Met BASIN (zie boven) worden gegevens uit het geopende bestand gehaald. Het subprogramma INVOER haalt gegevens van bijvoorbeeld de floppy tot het subprogramma INCLRCH weer op de standaardinvoer (toetsenbord) overschakelt. Het bestand wordt pas gesloten als CLOSE is opgeroepen:

```
LDA LF ; logisch filenummer
JSR CLOSE; $FFC3, bestand sluiten
```

6.3.3 De techniek van het opslaan van gegevens - LOAD en SAVE

Om gegevens of programma's op te slaan kunt u cassettes of diskettes gebruiken.

Cassette

Met een cassette recorder kunnen gegevens alleen sequentieel worden opgeslagen en in dezelfde volgorde worden gelezen. Ze kunnen niet direkt worden aangesproken. Ook kunnen ze niet gedeeltelijk worden veranderd. Het bestand moet in zijn geheel worden gelezen, dan kan er worden veranderd en daarna moet het weer op de band worden gezet. Omdat het opslaan en lezen van programma's een bepaalde volgorde vereist, is de cassette recorder een goedkope manier om die taak te vervullen. Een nadeel is echter de geringe snelheid. Omdat de gegevens serieel (bit voor bit) worden doorgegeven, is dat een principieel probleem. Om drop outs te corrigeren worden alle gegevens twee maal doorgegeven. Voordat de gegevens op band worden vastgelegd start de computer deze automatisch. Eerst wordt er een toon op de band gezet die de synchronisatie regelt en daarna volgen de gegevens in tweevoud. Om deze tijdverslindende procedure tot het uiterste te beperken worden de gegevens in een buffer opgeslagen, voordat ze op band worden gezet. Deze bandbuffer (adres 828-1019 (\$33C-\$3FB)) telt 192 tekens. Bij het inlezen worden de data uit de buffer gehaald. De verschillende soorten gegevens die op de band komen, moeten kunnen worden herkend. Daarom gaat aan elke opslag een header vooraf. De header wordt in de buffer geschreven en op de band gezet en later weer van de band naar de buffer geladen. Een header ziet er als volgt uit:

```
byte 1    ; specificceert het type header
byte 2    ; beginadres, LSB
byte 3    ; beginadres, MSB
byte 4    ; eindadres, LSB
byte 5    ; eindadres, MSB
byte 6-21 ; filenaam
```

Het eerste byte (type header) heeft de volgende betekenis:

- 1 - BASIC-programma; wordt geladen vanaf het begin van BASIC
- 2 - datablok; de bytes 2-192 bevatten de gegevens
- 3 - programma in machinetaal; wordt absoluut geladen
- 4 - dataheader; kondigt datafile aan
- 5 - end of tape block, einde van de band

Als het om een programmaheader van het type 1 of 3 gaat, staan in de volgende 4 bytes het start- en eindadres van het programma en de naam. Op de band staat in een blok het programma (twee keer). Bij een header van het type 3 wordt het programma geladen vanaf het adres dat in de programmaheader staat. Bij type 1 wordt het startadres gelezen, maar er wordt niets mee gedaan. Het programma wordt geladen vanaf het begin van BASIC (gewoonlijk \$800). Door bij het laden het secundaire adres 1 op te geven wordt absoluut geladen, dat wil zeggen op het gelezen startadres. Voor programma's in machinetaal is dat altijd nodig, omdat ze alleen in het geheugenbereik lopen, waarvoor ze zijn geschreven. Bij een header van het type 2 gaat het om gegevens, die met PRINT# op band zijn gezet. Bij elke INPUT# of GET# worden gegevens uit de buffer gelezen. Als die leeg is, wordt de uitvoering van het programma onderbroken en het volgende datablok van de band in de buffer ingelezen. Het secundaire adres geeft in de eerste plaats aan of een programma als BASIC-programma of als programma in machinetaal wordt ingelezen. Opslaan zonder secundair adres (of 0) betekent een programma in BASIC, headertype 1. Het secundaire adres 1 of een ander oneven getal produceert een programma in machinetaal. Het secundaire adres 2 of 3 schrijft na het programma nog een End Of Tape blok. Overzichtelijk opgesomd is er het volgende aan de hand:

LADEN

LOAD "NAAM",1	laadt BASIC-programma verschuifbaar; een gespecificeerd machinetaalprogramma (type 3) wordt absoluut geladen
LOAD "NAAM", 1,1	laadt elk programma absoluut

OPSLAAN

SAVE "NAAM",1	slaat op als BASIC-programma
SAVE "NAAM",1,1	slaat op als machinetaalprogramma
SAVE "NAAM",1,2	slaat op als BASIC-programma met EOT-blok
SAVE "NAAM",1,3	slaat op als machinetaalprogramma met EOT-blok

Bij het openen van een bandbestand heeft het secundaire adres de volgende betekenis:

OPEN 1,1,0, "NAAM"	opent bestand om te lezen
OPEN 1,1,1, "NAAM"	opent bestand om te schrijven
OPEN 1,1,2, "NAAM"	opent bestand om te schrijven met extra EOT-blok

Als bij het laden van het programma of het openen van een bestand een EOT-blok wordt aangetroffen, resulteert dat in de foutmelding "FILE NOT FOUND ERROR". De CLOSE-instructie sluit het bestand. Als dit was geopend om te schrijven, wordt daardoor in de bandbuffer het einde gemarkeerd (nulbyte) en de buffer op band gezet. Zo wordt duidelijk dat het noodzakelijk is het bandbestand met CLOSE te sluiten, nadat erin is geschreven. Doet u dat niet, dan komen de laatste gegevens niet op de band en gaan verloren. Het secundaire adres 2 zorgt ervoor, dat er nog een EOT-blok op de band komt. Ook de overdracht van gegevens is bij bandgebruik aan beperkingen onderhevig. Gewoonlijk komen ze in ASCII-formaat op de band (letters, cijfers en bijzondere tekens). Als dat met PRINT#1, CHR\$(I) gebeurt, kunnen niet alle tekens worden weergegeven. In het bijzonder CHR\$(0) wordt weggefilterd, omdat het bedrijfsstelsel het als eindmarkering gebruikt.

Diskette

Bij de overdracht van gegevens op diskette bestaan al deze problemen niet. Een diskette is verdeeld in sporen en sectoren die afzonderlijk direkt kunnen worden aangesproken. Een programma wordt opgeslagen nadat de low- en highbyte van het adres zijn doorgegeven. Er wordt geen verschil gemaakt tussen programma's in BASIC en machinetaal. Dat gebeurt alleen bij het laden.

```
LOAD "NAAM",8      laadt een BASIC-programma waarbij het
                   programma-adres wordt genegeerd
LOAD "NAAM",8,1     laadt machinetaalprogramma; wordt geladen
                   vanaf het opgeslagen beginadres
```

Het beginadres van een programma op diskette komt u te weten door:

```
10 OPEN 1,8,0,"NAAM"      : REM programmaple openen om te lezen
20 GET#1, A$, B$          : REM beginadres halen
30 IF A$=" " THEN A$=CHR$(0)
40 IF B$=" " THEN B$=CHR$(0)
50 PRINT ASC(A$)+256*ASC(B$): REM decimaal adres
60 CLOSE 1
```

Het beginadres van het programma wordt in decimalen gegeven. De lege strings in regel 30 en 40 zijn nodig, omdat de GET-instructie geen nulbytes accepteert. Op de volgende manier kunt u een programma in machinetaal op diskette zetten:

```
10 OPEN 1,8,1,"NAAM"      :REM programmaple openen
20 PRINT#1,CHR$(AD-INT(AD/256)*256);:REM lowbyte van beginadres
30 PRINT#1, CHR$(AD/256);  :REM highbyte van beginadres
40 FOR I=0 TO N-1         :REM n bytes opslaan
```

```

50 PRINT#1, CHR$(PEEK(AD+I));      :NEXT
60 CLOSE 1                          :REM programmafile sluiten

```

De variabele AD bevat het beginadres, N is de lengte van het programma in bytes. Programma's of geheugenbereiken kunnen vanuit de machinetaal worden opgeslagen met twee routines van het bedrijfsstelsel:

```

LOAD $FFD5
SAVE $FFD8

```

Bij het gebruik van de LOAD-routine wordt de accu met 0 geladen, het teken voor LOAD. Als de LOAD-routine wordt opgeroepen terwijl de waarde 1 in de accu zit, wordt VERIFY uitgevoerd. Het secundaire adres bepaalt waarheen geladen wordt. Is het secundaire adres niet 0, dan wordt op het adres geladen dat in het programma staat. Bij 0 wordt op het adres geladen dat in het X-(low) en Y-register (high) wordt gezet. Verder moeten de adressen van de apparaten en de filenaam opgeslagen zijn. Ook daarvoor zijn er ROM-routines. Voorbeeld: laden van een programma in machinetaal op het oorspronkelijke adres vanaf diskette:

```

LDX #8      ; apparaatnummer van de floppy
LDY #1      ; secundair adres
JSR $FFBA   ; fileparameters instellen
LDA #5      ; lengte filenaam
LDX #<NAME  ; adres filenaam
LDY #>NAME  ; highbyte van adres
JSR $FFBD   ; parameters filenaam instellen
LDA #0      ; LOAD-flag
JSR LOAD    ; programma laden
STX ADR     ; lowbyte eindadres
STY ADR+1   ; highbyte eindadres

```

Uit dit voorbeeld blijkt dat de LOAD-routine het eindadres van het geladen programma in het X- en het Y-register zet. In het volgende voorbeeld wordt een programma vanaf de band op adres \$6000 geladen. Het opgeslagen adres heeft geen invloed.

```

LDX #2      ; primaire nummer van recorder
LDY #0      ; secundair adres
JSR $FFBA   ; fileparameter geven
LDA #5      ; lengte van filenaam
LDX #<NAME  ; adres van filenaam
LDY #>NAME  ; MSB van adres
JSR $FFBD   ; parameter voor filenaam geven
LDA #0      ; LOAD-flag
LDX #$00    ; LOAD-adres low
LDY #$60    ; LOAD-adres high

```

```

JSR  LOAD      ; programma laden
STX  ADR       ; eindadres low
STY  ADR+1     ; eindadres high

```

Met de monitor voor de Commodore 64 wordt een programma altijd absoluut geladen en (op cassette) opgeslagen. Het laden van een machinetaalprogramma vanuit een BASIC-programma met LOAD levert een paar problemen op. Door het secundaire adres 1 op te geven kan er absoluut worden geladen, maar er is nog een probleem. Na elke LOAD wordt het eindadres van het laden altijd het eindadres van het BASIC-programma en begint de uitvoering van het programma weer van voren af aan. Voor het naladen (overlay) van BASIC-programma's is dat heel zinvol, maar bij het laden van programma's in machinetaal en in een ander geheugenbereik komen er problemen. In zo'n geval is een programmaatje in machinetaal de uitkomst. Vanuit het BASIC-programma kan het met SYS worden opgeroepen. Ook is een overdracht van parameters mogelijk, bijvoorbeeld filenaam en primair adres.

Het opslaan van programma's of geheugenbereiken met de SAVE-routine gebeurt op dezelfde manier. De routine moet op de hoogte zijn van het begin- en eindadres, de adressen van de apparatuur en lengte en adres van de filenaam. Voor diskettegebruik is dat noodzakelijk. Het beginadres moet op de zeropage op twee opeenvolgende adressen staan (low- en highbyte). In de accu wordt een wijzer op dit adres gezet. Het eindadres staat in het X- (LSB) en Y-register (MSB). Parameters voor apparatuur en filenaam worden op dezelfde manier gegeven als bij de LOAD-routine. Als voorbeeld zetten we het geheugenbereik \$C000-\$CFFF onder de naam "PROGRAMMA" op diskette. Het beginadres van het programma staat op \$FB/\$FC.

```

LDX  #8          ; primaire nummer van floppy
JSR  $FFBA       ; fileparameter geven
LDA  #8          ; lengte van filenaam
LDX  #<NAME      ; adres van filenaam
LDY  #>NAME      ; MSB van adres
JSR  $FFBD       ; parameter voor filenaam geven
LDA  #$FB        ; wijzer op beginadres
LDX  #$00        ; eindadres + 1 low
LDY  #$D0        ; eindadres + 1 high
JSR  SAVE        ; programma opslaan
NAME .ASC "PROGRAMMA" ; filenaam

```

Uit dit voorbeeld blijkt, dat de inhoud van het eindadres niet meer wordt opgeslagen. Daarom moet het eindadres plus 1 worden opgegeven.

Tot slot zetten we een BASIC-programma zonder filenaam met EOT-markering op band.

```
LDX #1      ; primair adres van recorder
LDY #2      ; secundair adres voor end of tape
JSR $FFBA   ; fileparameter geven
LDA #0      ; geen filenaam
JSR $FFBD   ; parameter voor filenaam geven
LDA #$2B    ; wijzer op begin BASIC-programma
LDX $2D     ; eindadres van programma low
LDY $2E     ; eindadres high
JSR SAVE    ; programma opslaan
```

6.3.4 De programmering van de RS-232 interface

RS-232 is een Amerikaanse benaming voor een standaard seriële interface. De Europese benaming V24 wordt ook veel gebruikt. Bij seriële overdracht worden niet als bij een parallelle interface 8 bits tegelijkertijd op verschillende lijnen tegelijk gezet, maar bit voor bit op een lijn voor ontvangen of voor zenden. De seriële interface heeft daardoor minder lijnen nodig, zodat gegevens bijvoorbeeld per telefoon kunnen worden verzonden. Een nadeel is de geringe snelheid, bij een parallelle interface heeft u daar geen last van. In het bedrijfssysteem van de Commodore 64 is de complete software verwerkt voor de bediening van de seriële RS-232 interface. De aansluiting zelf is verkrijgbaar als RS-232-cartridge die op de userport aangesloten wordt en voor de benodigde spanning zorgt (+/- 12 V). U kunt hiermee met andere apparatuur die is voorzien van een seriële aansluiting communiceren. Het bedrijfssysteem heeft de aan RS-232 interface het primaire adres 2 toegewezen. Als een logische file wordt geopend met het primaire adres 2, legt het bedrijfssysteem voor de in- en uitvoer twee buffers aan van 256 bytes elk. Gewoonlijk liggen ze aan het einde van het BASIC-RAM. Bij gebruik van de RS-232 interface in een BASIC-programma moet eerst de OPEN-instructie worden gegeven, omdat daardoor alle variabelen worden gewist. Ook wordt niet nagegaan of er nog genoeg geheugenruimte is. Bij CLOSE-instructies wordt deze buffer weer vrijgegeven. Ook in dat geval wordt in BASIC een CLR-instructie uitgevoerd. Het bestand moet dus pas aan het eind van het programma worden gesloten. Er kan maar 1 datakanaal tegelijk voor RS-232 open staan. Bij het sluiten van een datakanaal wordt een lopende overdracht afgebroken en worden de bufferwijzers teruggezet. De instructie

SYS 61604

wacht totdat de hele inhoud van de buffer is doorgegeven. Deze instructie moet aan de CLOSE-instructie voorafgaan. Als u in machinetaal programmeert kunt u

JSR \$FOA4

gebruiken. Een controleregister en een instructieregister bevatten de parameters voor de overdracht van gegevens. Deze registers worden doorgegeven als de eerste twee tekens van de filenaam. Het controleregister bevat de snelheid van de transmissie in bauds (een baud is een bit per seconde) en het aantal data- en stopbits dat wordt doorgegeven. Stopbits worden na elk doorgegeven datawoord (5-8 bits) verzonden.

Het instructieregister bevat aanwijzingen over de manier waarop de overdracht plaatsvindt, de pariteitscontrole en de manier van handshake. De onderste 4 bits van het controleregister bepalen de transmissiesnelheid volgens deze tabel:

bit	3	2	1	0	decimaal	aantal bauds
0	0	0	0	0	0	door gebruiker te definiëren
0	0	0	0	1	1	50
0	0	0	1	0	2	75
0	0	1	1	1	3	110
0	1	0	0	0	4	134.5
0	1	0	1	1	5	150
0	1	1	0	0	6	300
0	1	1	1	1	7	600
1	0	0	0	0	8	1200
1	0	0	1	1	9	1800
1	0	0	0	0	8	1200
1	0	0	1	1	9	1800
1	0	1	0	0	10	2400

de volgende baudrates zijn niet geïmplementeerd:

1	0	1	1	11	3600
1	1	0	0	12	4800
1	1	0	1	13	7200
1	1	1	0	14	9600
1	1	1	1	15	19200

U kunt dus transmissiesnelheden programmeren tussen de 50 en 2400 baud. De bits 5 en 6 bepalen het aantal databits:

bit 6 5 decimaal 7 bits

1	0	64	6 bits
1	1	96	5 bits

Bit 7 bepaalt het aantal stopbits:

bit 7 decimaal aantal stopbit

0	0	1 stopbit
1	128	2 stopbits

Het instructieregister is als volgt georganiseerd:

bit 0 decimaal handshake

0	0	3-draads handshake
1	1	X-draads handshake

bit 4 decimaal transmissiewijze

0	0	fullduplex
1	16	halfduplex

bit 7 6 5 decimaal pariteitscontrole

X X 0	0	niet; ook geen achtste databit
0 0 1	32	oneven pariteit
0 1 1	96	even pariteit
1 0 1	160	niet; achtste databit altijd 1
1 1 1	224	niet; achtste databit altijd 0

Als u 3-draads handshake heeft gekozen, worden de stuurlijnen CTS (clear to send) en DSR (data set ready) bij het zenden en de ontvangst overgeslagen. De computer stuurt de gegevens bijvoorbeeld naar een printer ongeacht of deze ze al verwerkt heeft. Wilt u dat het aangesloten apparaat de overdracht kan stoppen, gebruikt u de X-draads handshake. De genoemde stuurlijnen moeten in dat geval worden aangesloten en de printer moet ze kunnen bedienen. Als er een uitwisseling van gegevens plaatsvindt tussen twee computers, is de 3-draads handshake vaak voldoende (zie ook de aanwijzingen hieronder). U moet dan een RS-232 datakanaal openen met de parameters:

transmissiesnelheid 2400 baud
7 bits gegevens in ASCII-formaat
2 stopbits
geen pariteitscontrole
achtste databit altijd 0

fullduplex
3-draads handshake

De OPEN-instructie ziet er zo uit:

OPEN 1,2,0, CHR\$(10+0+128)+CHR\$(0+0+224)

Programmering van eigen transmissiesnelheden

In standaardgevallen wordt de snelheid van de overdracht gerealiseerd door de programmering van de timer in de CIA's. Als u daarvan afwijkt, kunt u de bovengrens van 2400 baud niet overschrijden. Dat is de hoogste snelheid die met de software voor de overdracht via de RS-232 kan worden bereikt. Na een bepaalde tijd die afhankelijk is van de transmissiesnelheid, geven de timers gedurende de ontvangst van de bits een niet maskeerbare interrupt (NMI). Als u eigen transmissiesnelheden gebruikt, kunt u het derde en vierde teken van de filenaam gebruiken om de waarden van de timers bij de OPEN-instructie door te geven. Deze waarden kunnen met behulp van de snelheid van de overdracht worden uitgerekend:

$T = 492662 / \text{BAUD} - 101$

De low- en highbyte van deze waarde worden het derde en vierde teken van de filenaam. In plaats van de code voor het aantal bauds zetten we een nul in het controleregister. Dan weet het bedrijfssysteem dat er eigen transmissiesnelheden worden gebruikt. In het volgende voorbeeld staan dezelfde parameters als hierboven. De snelheid van de overdracht is 1000 baud.

100 BAUD=1000

110 T=492662/BAUD-101

120 H=INT(T/256): TL=T-TH*256

130 OPEN 1,2,0, CHR\$(128)+CHR\$(224)+CHR\$(TL)+CHR\$(TH)

De eigen transmissiesnelheden liggen tussen ongeveer 8 en 2400 baud. Het statusregister voor het verkeer via de RS-232 interface heeft een andere betekenis dan bij de normale overdracht van gegevens. Dit register kan in BASIC door de variabele TL worden opgevraagd, maar bij een leesactie wordt het gewist. Is de statuswaarde vaker nodig, dan moet zij eerst aan een andere variabele worden toegewezen. ST geeft alleen de status van RS-232 aan, als ook het laatste verkeer via de interface loopt. Vanuit een programma in machinetaal kan men de status lezen zonder dat er wordt gewist. Hieronder staat de betekenis van de bits van de

RS-232-status. Een ingestelde bit wil zeggen, dat aan de voorwaarde is voldaan.

bit beschrijving

- 0 pariteitsfout
- 1 kaderfout
- 2 ontvangstbuffer vol
- 3 ontvangstbuffer leeg
- 4 geen CTS (clear to send) signaal
- 5 onbenut
- 6 geen DSR (data set ready) signaal
- 7 break signaal ontvangen

Wordt de RS-232 interface in machinetaal geprogrammeerd, dan kunt u de buffers voor in- en uitvoer in een willekeurig geheugenbereik kiezen. De wijzers op de buffers worden bij de OPEN-instructie ingesteld, maar kunnen daarna in een ander geheugenbereik worden gezet. De wijzers staan op de zeropage. \$F7/\$F8 wijst op de invoerbuffer en \$F9/\$FA op die voor de uitvoer. De programmering van in- en uitvoer via de RS-232 interface gaat net zo als bij andere apparaten met het primaire adres 2 (zie paragraaf 6.3.2).

Hieronder volgen enkele adressen voor de in- en uitvoer via de RS-232-interface.

\$0293	659	controlewoord
\$0294	660	instructiewoord
\$0295/\$0296	661/662	timerwaarde voor transmissiesnelheid
\$0297	663	RS-232 statuswoord
\$0298	664	aantal databits; wordt bij OPEN berekend
\$0299/\$029A	665/666	timerwaarde voor transmissiesnelheid bij het zenden
\$029B	667	schrijfwijzer van ontvangstbuffer
\$029C	668	leeswijzer van ontvangstbuffer
\$029D	669	leeswijzer van zendbuffer
\$029E	670	schrijfwijzer van zendbuffer

Op de zeropage zijn verder nog een paar adressen nodig als werkgeheugen tijdens de overdracht van gegevens.

De vorm van de aansluiting voor de RS-232 interface.

De standaardplug voor de RS-232 interface is een 25-polige Cannonplug, die op de volgende manier is opgebouwd en aangesloten:

1	2	3	4	5	6	7	8	9	10	11	12	13
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
14	15	16	17	18	19	20	21	22	23	24	25	

pinconfiguratie van de 25-polige Cannonplug

pin	afk.	richting	betekenis
-----	------	----------	-----------

1	GND		protective ground
2	TXD	OUT	transmitted data
3	RXD	IN	received data
4	RTS	OUT	request to send
5	CTS	IN	clear to send
6	DSR	IN	data set ready
7	GND		signal ground
8	DCD	IN	data carrier detected
20	DTR	OUT	data terminal ready
22	RI	IN	ring indicator

Om twee computers met elkaar te kunnen verbinden moet de aansluiting er zo uit zien:

computer 1	pin	pin	computer 2
TD	2 --->	3	RD
RD	3 <---	2	TD
RTS	4 --->	5	CTS
CTS	5 <---	4	RTS
DSR	6 <---	20	DTR
DTR	20 --->	6	DSR
GND	7 <-->	7	GND

De pijlen geven de richting van de overdracht aan. Bij deze aansluiting kunt u de X-draads handshake kiezen. Voor de 3-draads is het voldoende dat pin 2, 3 en 7 verbonden zijn als hierboven is beschreven. Als u de transmissieparameters van bijvoorbeeld een printer niet kent, verbind dan de pinnen 2, 3 en 7 als hierboven en sluit op elke plug de pinnen 4, 5, 8 en 6, 20 aan. Op die manier kunnen de meeste apparaten worden gebruikt. Als de gegevens verminkt of verkeerd aankomen, klopt de transmissiesnelheid of de pariteit waarschijnlijk niet. Door de stuurlijnen kort te sluiten werkt u als met een 3-draads

handshake. Kunnen de gegevens niet snel genoeg worden verzonden of ontvangen, gebruik dan eventueel machinetaal.

6.3.5 De seriële IEC-bus van de Commodore 64

Zoals bekend, wordt op de IEC-bus randapparatuur aangesloten en eventueel meer apparaten tegelijk. De IEC-bus van de Commodore 64 is de seriële pendant van de parallelle IEC-bus (IEEE 488) van de grote CBM computers. Omdat de gegevens bit voor bit en niet per byte worden doorgegeven, is de transmissiesnelheid lager. Als er meer apparaten op de bus worden aangesloten, hebben ze verschillende nummers de primaire adressen. Voor de IEC-bus zijn de nummers 4-15 gereserveerd. Wordt een apparaat aangesproken, dan zet de buscontroller (in dit geval altijd de computer) een ATN-sigitaal (attention) op een stuurlijn. Vervolgens zendt de computer het adres van het apparaat, waarna ATN weer wordt ingetrokken. Aan het apparaat wordt meegedeeld of het gegevens moet ontvangen of zenden. Men spreekt in dat verband van listener en talker en van LISTEN- en TALK-instructies. Daarna kan een secundair adres bepalen hoe het apparaat wordt gebruikt, met andere woorden hoe de gegevens worden verwerkt. Om de overdracht goed te laten verlopen kan een byte pas worden doorgegeven als de ontvanger het vorige heeft gekregen en verwerkt. Daarvoor dient de handshake. De zender deelt mee wanneer hij gegevens op de bus heeft gezet. Als de ontvanger klaar is voor de volgende gegevens, geeft hij dat door aan de zender. De overdracht kan verder gaan en heeft dus altijd de maximale snelheid die beide apparaten toestaan. Is de overdracht beëindigd, dan wordt het apparaat gedeadresseerd met UNTALK of UNLISTEN. Dat hangt ervan af of het heeft gezonden of ontvangen. De bus is nu vrij voor de volgende operatie.

Voor de bediening van de IEC-bus in machinetaal staan er in het bedrijfssysteem subroutines ter beschikking. In de sprongtabel in het bovenste ROM-bereik staan ze opgesomd (zie de laatste bladzijde van de ROM-listings). Als voorbeeld nemen we het inlezen van de foutmelding van de floppy disk. In BASIC ziet dat er zo uit:

```

10 OPEN 15,8,15: REM openen van foutkanaal
20 INPUT#15,A$,B$,C$,D$: REM foutmelding halen
30 PRINT A$;"",B$;"",C$;"",D$: REM en uitlezen
40 CLOSE 15: REM kanaal sluiten

```

In BASIC is dit vanwege de INPUT-instructie alleen mogelijk in de programmamodus. Nu volgt de tegenhanger in machinetaal:

```

C000 A9 08      LDA #8          ; primaire adres van floppy
C002 85 BA      STA FA
C004 20 B4 FF    JSR TALK       ; talk zenden
C007 A9 6F      LDA #15+$60    ; secundaire adres 5 plus $60
C009 85 B9      STA SA
C00B 20 96 FF    JSR SECTALK    ; secundaire adres voor talk
C00E 20 A5 FF L  JSR IECIN      ; teken van floppy halen
C011 20 D2 FF    JSR PRINT      ; op beeldscherm zetten
C014 C9 0D      CMP #13        ; carriage return?
C016 D0 F6      BNE L          ; nee, andere tekens
C018 20 AB FF    JSR UNTALK     ; untalk zenden
C01A 60         RTS            ; klaar

```

Nu een laadprogramma in BASIC:

```

100 FOR I=49152 TO 49179
110 READ X: POKE I,X: S=S+X: NEXT
120 DATA 169, 8,133,186, 32,180,255,169,111,133,185, 32
130 DATA 150,255, 32,165,255, 32,210,255,201, 13,208,246
140 DATA 32,171,255, 96
150 IF S <> 4169 THEN PRINT "FOUT IN DATA!": END
160 PRINT "OK!"

```

Vanuit BASIC wordt het programma met SYS 12*4096 opgeroepen.

Met een iets langer machinetaalprogramma kunt u gemakkelijk de inhoudsopgave van de diskette uitlezen. De directory hoeft dan niet met LOAD "\$",8 als BASIC-programma te worden geladen. Het BASIC-programma gaat daarbij in het geheugen verloren en dat nadeel heeft u met dit programma niet.

```

C000 A9 24      LDA #$24       ; dollarteken als filenaam
C002 85 FB      STA $FB        ; opslaan
C004 A9 FB      LDA #$FB       ; adres van filenaam
C006 85 BB      STA $BB
C008 A9 00      LDA #$00       ; highbyte
C00A 85 BC      STA $BC
C00C A9 01      LDA #$01       ; lengte van filenaam
C00E 85 B7      STA $B7
C010 A9 09      LDA #$08       ; apparaatnummer 8

```


C012	85 BA	STA \$BA	
C014	A9 60	LDA #\$60	; secundair adres voor LOAD
C016	85 B9	STA \$B9	
C018	20 D5 F3	JSR \$F3D5	; file met naam openen
C01B	A5 BA	LDA \$BA	
C01D	20 B4 FF	JSR \$FFB4	; talk zenden
C020	A5 B9	LDA \$B9	
C022	20 96 FF	JSR \$FF96	; secundair adres zenden
C025	A9 00	LDA #\$00	
C027	85 90	STA \$90	; status wissen
C029	A0 03	LDY #\$03	; eerste drie bytes overslaan
C02B	84 FB	STY \$FB	; als teller merken
C02D	20 A5 FF	JSR \$FFA5	; byte van floppy halen
C030	85 FC	STA \$FC	; en merken
C032	A4 90	LDY \$90	; status testen
C034	D0 2F	BNE \$9065	
C036	20 A5 FF	JSR \$FFA5	; byte van floppy halen
C039	A4 90	LDY \$90	; status testen
C03B	D0 28	BNE \$9065	
C03D	A4 FB	LDY \$FB	; teller halen
C03F	88	DEY	; en verlagen
C040	D0 E9	BNE \$902B	
C042	A6 FC	LDX \$FC	; byte terughalen
C044	20 CD BD	JSR \$BDCD	; 16bits getal uitvoeren
C047	A9 20	LDA #\$20	; aantal bezette bloks
C049	20 D2 FF	JSR \$FFD2	; spatie uitvoeren
C04C	20 A5 FF	JSR \$FFA5	; volgende byte halen
C04F	A6 90	LDX \$90	; status testen
C051	D0 12	BNE \$9065	
C053	AA	TAX	; byte testen
C054	F0 06	BEQ \$905C	; nul? dan regeleinde
C056	20 D2 FF	JSR \$FFD2	; anders uitvoeren
C059	4C 4C C0	JMP \$C04C	; en volgende teken halen
C05C	A9 0D	LDA #\$0D	; carriage return
C05E	20 D2 FF	JSR \$FFD2	; uitvoeren
C061	A0 02	LDY #\$02	; twee bytes voor linkadres
C063	D0 C6	BNE \$902B	; doorgaan
C065	20 42 F6	JSR \$F642	; bestand sluiten
C068	60	RTS	

Nu volgt weer het laadprogramma

```

100 FOR I = 49152 TO 49256
110 READ X : POKE I,X : S=S+X : NEXT
120 DATA 169, 36,133,251,169,251,133,187,169, 0,133,188
130 DATA 169, 1,133,183,169, 9,133,186,169, 96,133,185
140 DATA 32,213,243,165,186, 32,180,255,165,185, 32,150
150 DATA 255,169, 0,133,144,160, 3,132,251, 32,165,255
160 DATA 133,252,164,144,208, 47, 32,165,255,164,144,208

```

```

170 DATA 40,164,251,136,208,233,166,252, 32,205,189,169
180 DATA 32, 32,210,255, 32,165,255,166,144,208, 18,170
190 DATA 240, 6, 32,210,255, 76, 76,192,169, 13, 32,210
200 DATA 255,160, 2,208,198, 32, 66,246, 96
210 IF S <> 15344 THEN PRINT "FOUT IN DATA !!" : END
220 PRINT "OK !"

```

De oproep vanuit BASIC gebeurt weer met SYS 12*4096. De inhoudsopgave van de diskette verschijnt dan op het beeldscherm zonder dat een opgeslagen BASIC-programma verloren gaat. In plaats van de seriële IEC-bus kan het vooral vanwege de snelheid lonend zijn een IEC-bus module te gebruiken die op de memory expansion poort wordt aangesloten. Alle randapparatuur van de grote Commodore computers kan dan worden gebruikt, zoals de grote dubbele diskdrives. Bij de programmering van de parallelle IEC-bus veranderen alleen de adressen van de routines, bijvoorbeeld de in- en uitvoer van een byte op de IEC-bus of het zenden van TALK of LISTEN. Programmeert u de in- en uitvoer op de IEC-bus via logische bestanden met BASIN en BASOUT (zie paragraaf 6.3), dan hoeven er geen veranderingen te worden aangebracht.

7.1 De zeropage en andere belangrijke geheugenlocaties

adressen		gebruik
hex.	dec.	
00	0	datarichtingsregister voor processorpoort
01	1	processorpoort
02	2	onbenut
03-04	3- 4	vector voor verandering van reëel getal naar geheel getal
05-06	5- 6	vector voor verandering van geheel getal naar reëel getal
07	7	zoekteken
08	8	apostrof-flag
09	9	geheugen voor kolom bij TAB-instructie
0A	10	flag van interpreter: LOAD=0, VERIFY=1
0B	11	wijzer in invoerbuffers, aantal dimensies
0C	12	flag voor DIM
0D	13	typeflag \$00=numeriek, \$FF=string
0E	14	flag voor integer=\$80, reëel=\$00
0F	15	apostrof-flag bij LIST
10	16	flag voor FN
11	17	flag voor INPUT \$00, GET \$40, READ \$98
12	18	teken voorafgaand aan ATN
13	19	actief I/O-apparaat
14-15	20- 21	integer adres, bijvoorbeeld regelnummer
16	22	wijzer op stringstack
17-18	23- 24	wijzer op laatst gebruikte string
19-21	25- 33	stringstack
22-25	34- 37	wijzer voor verschillende doeleinden
2B-2C	43- 44	wijzer op begin van BASIC-programma
2D-2E	45- 46	wijzer op begin van variabelen
2F-30	47- 48	wijzer op begin van array
31-32	49- 50	wijzer op eind van array
33-34	51- 52	wijzer op begin van string
35-36	53- 54	hulpwijzer voor strings
37-38	55- 56	wijzer op eind van BASIC-RAM
39-3A	57- 58	actueel regelnummer van BASIC
3B-3C	59- 60	vorig regelnummer van BASIC
3D-3E	61- 62	wijzer op volgende BASIC-instructie voor CONT
3F-40	63- 64	momenteel regelnummer voor DATA
41-42	65- 66	wijzer op volgende DATA-element
43-44	67- 68	wijzer op herkomst van invoer

45-46	69- 70	naam van variabele
47-48	71- 72	adres van variabele
49-4A	73- 74	wijzer op waarde van variabele
4B-4C	75- 76	tussengeheugen voor programmawijzer
4D	77	masker voor vergelijkingsoperaties
4E-4F	78- 79	wijzer voor FN
50-53	80- 83	stringdescriptor
54	84	constante \$4C voor functies
55-56	85- 86	sprongvector voor functies
57-5B	87- 91	register voor rekenoperaties, accu#3
5C-60	92- 96	register voor rekenoperaties, accu#4
61-65	97-101	accu#1 voor reële getallen, FAC
66	102	teken voorafgaand aan FAC
67	103	teller voor polynoominterpretatie
68	104	rondingsbyte voor FAC
69-6D	105-109	accu#2 voor reële getallen, ARG
6E	110	teken voorafgaand aan ARG
6F	111	vergelijkt tekens voor begin van FAC en ARG
70	112	rondingsbyte voor FAC
71-72	113-114	wijzer voor polynoominterpretatie
73-8A	115-138	CHRGET-routine, haalt teken uit BASIC-tekst
7A-7B	122-123	programmawijzer
8B-8F	139-143	laatste RND-waarde
90	144	statuswoord ST
91	145	flag voor stoptoets
92	146	tijdconstante voor band
93	147	flag voor LOAD \$00 of VERIFY \$01
94	148	flag bij IEC-uitvoer
95	149	uitvoerbuffer voor IEC-bus
96	150	flag voor EOT van band ontvangen
97	151	tussengeheugen van register
98	152	aantal open files
99	153	actief invoerapparaat
9A	154	actief uitvoerapparaat
9B	155	pariteit voor band
9C	156	flag voor byte ontvangen
9D	157	flag voor direktmodus \$80, programma \$00
9E	158	bandpass 1 controletotaal
9F	159	bandpass 2 foutcorrectie
A0-A2	160-162	tijd
A3	163	bitteller voor seriële uitvoer
A4	164	teller voor band
A5	165	teller voor band schrijven
A6	166	wijzer in bandbuffer
A7-AB	167-171	werkgeheugen voor in/uitvoer via band
AC-AD	172-173	wijzer voor bandbuffer en scrolling
AE-AF	174-175	wijzer op eind van programma bij LOAD/SAVE
B0-B1	176-177	tijdconstanten voor bandtiming
B2-B3	178-179	wijzer op bandbuffer

B4	180	bitteller voorband
B5	181	volgend bit voor RS-232
B6	182	buffer voor uit te lezen byte
B7	183	lengte van filenaam
B8	184	logisch filennummer
B9	185	secundair adres
BA	186	primaair adres
BB-BC	187-188	wijzer op filenaam
BD	189	werkgeheugen voor seriële in/uitvoer
BE	190	pass-teller voor band
BF	191	buffer voor seriële uitvoer
C0	192	flag voor cassettemotor
C1-C2	193-194	beginadres voor in/uitvoer van scherm
C3-C4	195-196	eindadres voor in/uitvoer van scherm
C5	197	nummer van gebruikte toets, 64=geen toets
C6	198	aantal ingedrukte toetsen
C7	199	flag voor RVS-modus
C8	200	regeleinde voor uitvoer
C9	201	cursorregel voor invoer
CA	202	cursorkolom voor invoer
CB	203	ingedrukte toets, 64=geen toets
CC	204	flag voor cursor: 0=cursor aan, 1=cursor uit
CD	205	teller om cursor te laten knipperen
CE	206	teken onder de cursor
CF	207	flag voor cursor: 1=aan-fase, 0=uit-fase
D0	208	flag voor invoer van toetsenbord of scherm
D1-D2	209-210	wijzer op begin van actuele beeldschermregel
D3	211	cursorkolom
D4	212	flag voor apostrofmodus
D5	213	lengte van beeldschermregel
D6	214	cursorregel
D7	215	verschillende doeleinden
D8	216	aantal inserts
D9-F2	217-242	MSB van begin van beeldschermregels
F3-F4	243-244	wijzer in kleur-RAM
F5-F6	245-246	wijzer op decodeertabel van toetsenbord
F7-F8	247-248	wijzer op RS-232 invoerbuffer
F9-FA	249-250	wijzer op RS-232 uitvoerbuffer

00FF-010A 255-256 buffer voor veranderen van reël getal
naar ASCII

0100-013E 256-318 geheugen voor correctie bij invoer vanaf band
0100-01FF 256-511 processor stack
0200-0258 512-600 BASIC invoerbuffer
0259-0262 601-610 tabel van logisch filenummers
0263-026C 611-630 tabel van primaire adressen
0277-0280 631-640 buffer van toetsenbord
0281-0282 641-642 begin van BASIC-RAM
0283-0284 643-644 eind van BASIC-RAM

0285	645	time-out flag voor seriële IEC-bus
0286	646	actuele kleur
0287	647	kleur onder de cursor
0288	648	highbyte video-RAM
0289	649	lengte van toetsenbordbuffer
028A	650	flag voor repeatfunctie van alle toetsen
028B	651	teller voor repeatsnelheid
028C	652	teller voor repeatvertraging
028D	653	flag voor SHIFT, C= en CTRL (bit 0,1 en 2)
028E	654	SHIFTflag
028F-0290	655-656	wijzer voor decodering van toetsenbord
0291	657	flag voor shift/Commodore geblokkeerd
0292	658	flag voor scrolling
0293	659	RS-232 besturingswoord
0294	660	RS-232 instructiewoord
0295-0286	661-662	bit timing
0297	663	RS-232 status
0298	664	aantal databits voor RS-232
0299-029A	665-666	RS-232 transmissiesnelheid
029B	667	wijzer op byte, ontvangen van RS-232
029C	668	wijzer op input van RS-232
029D	669	wijzer op byte dat via RS-232 wordt doorgegeven
029E	670	wijzer op uitvoer op RS-232
029F-02A0	671-672	geheugen voor IRQ tijdens bandgebruik
02A1	673	CIA 2 NMI-flag
02A2	674	CIA 1 timer A
02A3	675	CIA 1 interruptflag
02A4	676	CIA 1 flag voor timer A
02A5	677	beeldschermregel
02A6	678	flag voor PAL- (1) of NTSC-versie (0)
02A7-02BF	679-703	niet gebruikt
02C0-02FE	704-766	sprite 11
02FF	767	niet gebruikt
0300-0301	768,769	\$E38B vector voor BASIC warmstart
0302-0303	770-771	\$A483 vector voor invoer van regel
0304-0305	772-773	\$A57C vector voor verandering in interpretercode
0306-0307	774-775	\$A71A vector voor verandering in begrijpelijke tekst (LIST)
0308-0309	776-777	\$A7E4 vector voor halen van BASIC-instructieadressen
030A-030B	778-779	\$AE86 vector voor interpretatie
030C	780	accu voor SYS-instructie
030D	781	X-REG voor SYS-instructie
030E	782	Y-REG voor SYS-instructie
030F	783	status-REG voor SYS-instructie
0310	784	\$4C JMP-instructie voor USR-instructie
0311-0312	785-786	\$B248 USR-vector

0313	787	niet gebruikt
0314-0315	788-789	\$EA31 IRQ-vector
0316-0317	790-791	\$FE66 BRK-vector
0318-0319	792-793	\$FE47 NMI-vector
031A-031B	794-795	\$F34A OPEN-vector
031C-031D	796-797	\$F291 CLOSE-vector
031E-031F	798-799	\$F20E CHKIN-vector
0320-0321	800-801	\$F250 CKOUT-vector
0322-0323	802-803	\$F333 CLRCH-vector
0324-0325	804-805	\$F157 INPUT-vector
0326-0327	806-807	\$F1CA OUTPUT-vector
0328-0329	808-809	\$F6ED STOP-vector
032A-032B	810-811	\$F13E GET-vector
032C-032D	812-813	\$F32F GLALL-vector
032E-032F	814-815	\$FE66 warmstartvector
0330-0331	816-817	\$F4A5 LOAD-vector
0332-0333	818-819	\$F5ED SAVE-vector
0334-033B	820-827	niet gebruikt
033C-03FB	828-1019	bandbuffer
0340-037E	832-894	sprite 13
037F	895	niet gebruikt
0380-03BE	896-985	sprite 14
03BF	986	niet gebruikt
03C0-03FE	960-1022	sprite 15

7.2 De adressen van de BASIC-routines

In de Commodore 64 zit dezelfde BASIC-interpreter als in de VIC 20. Alleen de adressen verschillen. Bij de adressen \$A000-\$BFFF van de Commodore 64 telt u \$2000 om die van de VIC 20 te krijgen. \$A860 is voor de VIC 20 dus \$C860. Van de adressen \$A000-\$E37A van de Commodore 64 trekt u 3 af: \$E30E komt overeen met \$E30B van de VIC 20.

adres omschrijving

A000	startvector
A002	NMI-vector
A004	Commodore BASIC
A00C	adressen van de BASIC-instructies min 1
A052	adressen van de BASIC-functies
A080	hiërarchiecodes en adressen van de BASIC-operatoren
A09E	lijst van BASIC-instructiewoorden

A19E BASIC-foutmeldingen
 A328 adressen van de foutmeldingen
 A364 meldingen van de BASIC-interpreter
 A38A stack-zoekroutine voor FOR-NEXT en GOSUB
 A3B8 blokverschuivingsroutine
 A3FB controleert op plaats in stapel
 A408 maakt plaats in stapel
 A435 out of memory uitlezen
 A437 foutmelding uitlezen
 A469 break-insprong
 A474 ready-insprong
 A480 invoer-wachtlus
 A49C wissen en invoegen van programmaregels
 A533 BASIC-programmaregels opnieuw koppelen
 A560 haalt teken in invoerbuffer
 A571 string too long uitlezen
 A579 veranderen van regel in interpretercode
 A613 beginadres van BASIC-regel zoeken
 A642 BASIC-instructie NEW
 A65E BASIC-instructie CLR
 A68E programmawijzer op begin van BASIC zetten
 A69C BASIC-instructie LIST
 A717 interpretercode in instructiewoord veranderen
 A742 BASIC-instructie FOR
 A7AE interpreterlus, voert BASIC-instructies uit
 A7ED voert een BASIC-instructie uit
 A81D BASIC-instructie RESTORE
 A82C breekt programma af bij ingedrukte stop-toets
 A82F BASIC-instructie STOP
 A831 BASIC-instructie END
 A857 BASIC-instructie CONT
 A871 BASIC-instructie RUN
 A883 BASIC-instructie GOSUB
 A8A0 BASIC-instructie GOTO
 A8D2 BASIC-instructie RETURN
 A8F8 BASIC-instructie DATA
 A906 zoekt volgende instructie
 A909 zoekt volgende regel
 A928 BASIC-instructie IF
 A93B BASIC-instructie REM
 A94B BASIC-instructie ON
 A96B zoekt adres van BASIC-regel
 A9A5 BASIC-instructie LET
 AA80 BASIC-instructie PRINT#
 AA86 BASIC-instructie CMD
 AAA0 BASIC-instructie PRINT
 AB1E string uitlezen
 AB3E spatie/cursor right uitlezen
 AB4D behandeling van fouten bij invoer

AB7B	BASIC-instructie GET
ABA5	BASIC-instructie NPUT#
ABBF	BASIC-instructie INPUT
AC06	BASIC-instructie READ
ACFC	?extra ignored en ?redo from start
AD1D	BAS C-instructie NEXT
AD8A	FRMNUM haalt gegeven; controle: numeriek?
AD8D	controle: numeriek?
AD8F	controle: string?
AD99	typ mismatch uitlezen
AD9E	FRMEVL haalt en interpreteert willekeurig gegeven
AE83	rekenkundig gegeven halen
AEA8	constante: reëel getal pi
AED4	BASIC-instructie NOT
AEF1	haalt gegeven tussen haakjes
AEF7	controle: haakje gesloten?
AEFA	controle: haakje open?
AEFD	controle: komma?
AEFF	controle: teken in accu?
AF08	syntax error uitlezen
AF28	haalt variabele
AFE6	BASIC-instructie OR
AFE9	BASIC-instructie AND
B016	vergelijkingsoperaties
B081	BASIC-instructie DIM
B113	controle: letter?
B194	berekent wijzer als eerste element van array
B1A5	constante: reëel getal -32768
B1AA	FAC in integer veranderen
B245	bad subscript uitlezen
B248	illegal quantity uitlezen
B34C	berekent omvang van array
B37D	BASIC-functie FRE
B39E	BASIC-functie POS
B3A6	controle: direkt-modus?
B3AB	illegal direkt uitlezen
B3AE	undef'd function uitlezen
B3B3	BASIC-instructie DEF
B3E1	FN-syntax controleren
B3F4	BASIC-functie FN
B465	BASIC-functie STR\$
B475	stringbeheer: wijzer op string berekenen
B487	string samenstellen
B526	garbage collection, ongebruikte strings verwijderen
B63D	koppeling van strings: +
B6A3	FRESTR: stringbeheer
B6EC	BASIC-functie CHR\$
B700	BASIC-functie LEFT\$
B72C	BASIC-functie RIGHT\$
B737	BASIC-functie MID\$
B77C	BASIC-functie LEN

B782 stringparameters halen
 B78B BASIC-functie ASC
 B79B haalt byte (0-255)
 B7AD BASIC-functie VAL
 B7EB haalt adres (0-65535) en bytewaarde (0-255)
 B7F7 FAC in adresformaat veranderen (bereik 0-65535)
 B80D BASIC-functie PEEK
 B824 BASIC-instructie POKE
 B82D BASIC-instructie WAIT
 B849 $FAC = FAC + 0.5$
 B850 aftrekken: $FAC = constante(A/Y) - FAC$
 B853 aftrekken: $FAC = ARG - FAC$
 B867 optellen: $FAC = constante(A/Y) + FAC$
 B86A optellen: $FAC = ARG + FAC$
 B97E overflow uitlezen
 B9BC constanten: reële getallen voor LOG
 B9EA BASIC-functie LOG
 BA28 vermenigvuldigen: $FAC = constante(A/Y) * FAC$
 BA2B vermenigvuldigen: $FAC = ARG * FAC$
 BA8C $ARG = constante(A/Y)$
 BAE2 $FAC = FAC * 10$
 BAF9 constante: reël getal 10
 BAFE $FAC = FAC / 10$
 BBOF delen: $FAC = constante(A/Y) / FAC$
 BB12 delen: $FAC = ARG / FAC$
 BB8A division by zero uitlezen
 BBA2 $FAC = constante(A/Y)$
 BBC4 $accu\#4 = FAC$
 BBCA $accu\#3 = FAC$
 BBDO variabele = FAC
 BBFC $FAC = ARG$
 BCOC $ARG = FAC$
 BC1B FAC ronden
 BC2B teken halen, dat aan FAC voorafgaat
 BC39 BASIC-functie SGN
 BC58 BASIC-functie ABS
 BC5B constante (A/Y) met FAC vergelijken
 BC9B FAC in integer veranderen
 BCCC BASIC-functie INT
 BCF3 ASCII in reël veranderen
 BDB3 reële constanten voor reël naar ASCII
 BDC2 regelnummer uitlezen bij foutmelding
 BD CD positief integer getal uitlezen (0-65535)
 BDDD FAC in ASCII-formaat veranderen
 BF11 constante: reël getal 0.5
 BF16 binaire getallen om FAC in ASCII te veranderen
 BF71 BASIC-functie SQR
 BF78 machtsverheffen: $FAC = constante(A/Y)^{FAC}$
 BF7B machtsverheffen: $FAC = ARG^{FAC}$

BFBF	constanten: reële getallen voor EXP
BFED	BASIC-functie EXP
E043	polynoomberekening
E059	polynoomberekening
E08D	constanten: reële getallen voor RND
E097	BASIC-functie RND
E107	break uitlezen
E10C	BSOUT: teken uitlezen
E112	BSIN: teken ontvangen
E118	CKOUT: uitvoerapparaat bepalen
E11E	CHKIN: invoerapparaat bepalen
E124	GETIN: teken halen
E12A	BASIC-instructie SYS
E156	BASIC-instructie SAVE
E165	BASIC-instructie VERIFY
E168	BASIC-instructie LOAD
E1BE	BASIC-instructie OPEN
E1C7	BASIC-instructie CLOSE
E1D4	parameters voor LOAD en SAVE halen
E219	parameter voor OPEN halen
E264	BASIC-functie COS
E26B	BASIC-functie SIN
E2B4	BASIC-functie TAN
E2EO	constanten: reële getallen voor SIN en COS
E30E	BASHC-functie ATN
E33E	constanten: reële getallen voor ATN
E37B	BASIC-NMI-sprong
E394	BASIC koude start
E3A2	kopie van de CHRGET-routine
E3BA	beginwaarde voor RND-functie
E3BF	RAM voor BASIC starten
E447	tabel van BASIC-vectoren
E453	BASIC-vectoren laden

7.3 Tabel ter vergelijking van de VIC 20 en de Commodore 64

C 64 VIC 20 omschrijving

E45F	E429	meldingen van bedrijfssysteem
E4E0	-	wacht op Commodore-toets
E4EC	-	constanten voor timing van RS-232
E500	E500	haalt BASIC-adressen van CIA/VIA

E505	E505	haalt beeldschermformaat: regels/kolommen
E50A	E50A	cursor instellen/cursorpositie halen
E518	E518	beeldschermreset
E544	E55F	scherm wissen
E566	E581	cursor home
E5A0	E5BB	videocontroller starten
E5B4	E5CF	teken halen uit buffer van toetsenbord
E5CA	E5E5	wachtlus voor invoer van toetsenbord
E632	E64F	teken van beeldscherm halen
E684	E6B8	controle: apostrof?
E6B6	E6EA	MSB voor begin van regels berekenen
E8DA	E921	tabel van kleurcodes
E8EA	E975	scherm scrollen
E9C8	EA56	regel naar boven schuiven
E9FF	EA8D	beeldschermregel wissen
EA1C	EAAl	teken en kleur op scherm zetten
EA24	EAB2	wijzer op kleur-RAM berekenen
EA31	EABF	interruptroutine
EA87	EB1E	opvragen van toetsenbord
EB48	EBDC	controle van toetsen: SHIFT? CTRL? Commodore?
EB79	EC46	wijzer op decodeertabellen van toetsenbord
EB81	EC5E	decodeertabellen
EC44	ED21	controle: stuurteken?
EC78	ED69	decodeertabellen
ECB9	EDE4	constanten voor videocontroller
ECE7	EDF3	load (cr) run (cr)
ECFO	EDFE	tabel: LSB's van eerste beeldschermposities
ED09	EE14	TALK zenden
ED0C	EE17	LISTEN zenden
ED40	EEE4	byte op IEC-bus uitvoeren
EDB9	EECO	secundaire adres voor LISTEN zenden
EDC7	EECE	secundaire adres voor TALK zenden
EDEF	EEF6	UNTALK zenden
EDFE	EFO4	UNLISTEN zenden
EE13	EF19	byte van IEC-bus halen
EEB3	EF96	een milliseconde vertraging
EEBB	EFA3	RS-232 uitvoer
EF4A	F027	aantal databits van RS-232 berekenen
F014	FOED	uitvoer in buffer van RS-232
F086	F14F	GET van RS-232
FOA4	F160	timer instellen voor IEC-timeout
FOBD	F174	foutmeldingen van bedrijfsstelsel
F12B	F1EO	meldingen uitlezen
F157	F2OE	BASIN: teken halen
F1CA	F27A	BSOUT: teken uitlezen
F2OE	F2C7	CHKIN: uitvoerapparaat bepalen
F250	F309	CKOUT: invoerapparaat bepalen
F291	F34A	CLOSE
F30F	F3CF	logische filenummer zoeken

F31F	F3DF	fileparameters geven
F32F	F3EF	CLALL sluit alle I/O-kanalen
F333	F3F3	CLRCH sluit I/O-kanaal
F34A	F40A	OPEN
F49E	F542	LOAD
F5AF	F647	searching for filename uitlezen
F5D2	F66A	loading/verifying uitlezen
F5DD	F675	SAVE
F68F	F728	saving filenaam uitlezen
F69B	F734	UDTIM: lopende tijd verhogen
F6DD	F760	tijd halen
F6E4	F767	tijd instellen
F6ED	F770	stoptoets opvragen
F6FB	F77E	foutmeldingen van bedrijfssysteem uitlezen
F72C	F7AF	programmaheader van band lezen
F76A	F7E7	header op band zetten
F7D0	F84D	beginadres uit bandbuffer halen
F7D7	F854	begin- en eindadres van badbuffer instellen
F7EA	F867	bandheader naar naam zoeken
F80D	F88A	bandbufferwijzer verhogen
F817	F894	wacht op bandtoets voor lezen
F82E	F8AB	vraagt bandtoets op
F838	F8B7	wacht op bandtoets voor schrijven
F841	F8C0	blok van band lezen
F84A	F8C9	programma van band laden
F864	F8EA	bandbuffer op band zetten
F86B	F8EB	blok/programma op band zetten
F8BE	F92F	I/O-afsluiting afwachten
F8E1	F94B	controle: stoptoets?
F92C	F98E	interruptroutine voor band lezen
FB97	F8DB	bitteller instellen voor seriële uitvoer
FBA6	FBEA	bit op band zetten
FBCD	FC0B	interruptroutine voor band schrijven
FCB8	FCF6	IRQ-vector zetten
FCCA	FD08	cassettomotor uitschakelen
FCD1	FD11	controle: eindadres bereikt?
FCDB	FD1B	adreswijzer verhogen
FCE2	FD22	RESET
FD02	FD3F	controle: ROM in \$8000/\$A000?
FD10	FD4D	identificatie van ROM-moduul
FD15	FD52	hardware en I/O-vectoren instellen/halen
FD30	FD6D	tabel van hardware en I/O-vectoren
FD50	FD8D	werkgeheugen starten
FD9B	FD6D	tabel van IRQ-vectoren
FDF9	FE49	parameters voor filenamen geven
FE00	FE50	parameter voor actuele file geven
FE07	FE57	status halen
FE18	FE66	flag zetten voor meldingen bedrijfssysteem
FE1C	FE6A	status instellen

FE21	FEF6	timeout-flag voor IEC-bus zetten
FE25	FE73	RAM-bovengrens instellen/halen
FE34	FE82	RAM-ondergrens instellen/halen
FE43	FEA9	NMI-routine
FEC2	FF5C	constanten voor aantal bauds van RS-232
FF48	FF72	interrupthandler
FF81	FF8A	sprongtabel van bedrijfssysteemroutines

7.4 Tabel ter vergelijking van de CBM 8000 en de Commodore 64

De onderstaande tabel vergemakkelijkt de overdracht naar de Commodore 64 van programma's in machinetaal, die voor computers uit de 8000-serie zijn geschreven. Het gaat om een vergelijking van de BASIC-interpretadressen van beide machines.

8000	64	omschrijving
B000	A00C	adressen van de BASIC-instructies min 1
B066	A052	adressen van de BASIC-functies
B094	A080	hiërarchiecodes en adressen van BASIC-operatoren
B0B2	A09E	lijst van BASIC-instructies
B20D	A19E	BASIC-foutmeldingen
B322	A38A	stack-zoekroutine voor FOR-NEXT en GOSUB
B350	A3B8	blokverschuivingsroutine
B393	A3FB	controle: plaats in stack?
B3CD	A435	out of memory uitlezen
B3CF	A437	foutmelding uitlezen
B3FF	A474	ready-modus
B406	A480	invoerwachtlus
B4B6	A533	BASIC-programmaregels opnieuw koppelen
B4E2	A560	haalt regel naar invoerbuffer
B4FB	A579	regel in interpretercode veranderen
B5A3	A613	zoekt BASIC-regel
B5D2	A642	BASIC-instructie NEW
B5EE	A65E	BASIC-instructie CLR
B622	A68E	programmawijzer op begin van BASIC zetten
B630	A69C	BASIC-instructie LIST
B6B5	A717	interpretercode veranderen in instructiewoord
B6DE	A742	BASIC-instructie FOR
B74A	A7AE	interpreterlus: voert BASIC-instructies uit
B785	A7ED	voert BASIC-instructie uit
B7B7	A81D	BASIC-instructie RESTORE

B7C6	A82C	STOP en END
B7EE	A857	BASIC-instructie CONT
B808	A871	BASIC-instructie RUN
B813	A883	BASIC-instructie GOSUB
B830	A8A0	BASIC-instructie GOTO
B85D	A8D2	BASIC-instructie RETURN
B883	A8F8	BASIC-instructie DATA
B891	A906	zoekt volgende instructie
B894	A909	zoekt volgende regel
B8B3	A92B	BASIC-instructie IF
B8C6	A93B	BASIC-instructie REM
B8D6	A94B	BASIC-instructie ON
B8F6	A96B	zoekt adres van BASIC-regel
B930	A9A5	BASIC-instructie LET
BA88	AA80	BASIC-instructie PRINT#
BA8E	AA86	BASIC-instructie CMD
BAA8	AAA0	BASIC-instructie PRINT
BB1D	AB1E	string uitlezen
BB4C	AB4D	behandeling van fouten bij invoer
BB7A	AB7B	BASIC-instructie GET
BBA4	ABA5	BASIC-instructie INPUT#
BBBE	ABBF	BASIC-instructie READ
BCF7	ACFC	?extra ignored en ?redo from start
BD19	AD1D	BASIC-instructie NEXT
BD84	AD8A	FRMNUM haalt gegeven; controle: numeriek?
BD87	AD8D	controle: numeriek?
BD89	AD8F	controle: string?
BD93	AD99	type mismatch uitlezen
BD98	AD9E	FRMEVL haalt en interpreteert willekeurig gegeven
BDD1	AE83	volgend element van gegeven halen
BEA0	AEA8	reël getal pi
BEE9	AEF1	haalt rekenkundig gegeven tussen haakjes
BEEF	AEF7	controle: haakje gesloten?
BEF2	AEFA	controle: haakje open?
BEF5	AEFD	controle: komma?
BEF7	AEFF	controle: teken in accu?
BF00	AF08	syntax error uitlezen
BFOC	AF28	haalt variabele
C086	AFE6	BASIC-operator OR
C089	AFE9	BASIC-operator AND
COB6	B016	vergelijkingsoperaties
C121	B081	BASIC-instructie DIM
C1B6	B113	controle: letter?
C2D9	B1A5	reël getal -32768
C2DD	B1AA	FAC in integer veranderen
C370	B245	bad subscript uitlezen
C373	B248	illegal quantity uitlezen
C477	B34C	berekent omvang van array
C4A8	B37D	BASIC-functie FRE

C4C9	B39E	BASIC-functie POS
C4CF	B3A6	controle: direkt-modus?
C4DC	B3B3	BASIC-instructie DEF
C50A	B3E1	FN-syntax controleren
C51D	B3F4	BASIC-functie FN
C58E	B465	BASIC-functie STR\$
C59E	B475	stringbeheer: wijzer op string berekenen
C5B0	B487	string samenstellen
C66A	B526	garbage collection
C74F	B63D	koppeling van strings
C7B5	B6A3	FRESTR: stringbeheer
C822	B6EC	BASIC-functie CHR\$
C836	B700	BASIC-functie LEFT\$
C862	B72C	BASIC-functie RIGHT\$
C86D	B737	BASIC-functie MID\$
C8B2	B77C	BASIC-functie LEN
C8B8	B782	stringparameter halen
C8C1	B78B	BASIC-functie ASC
C8D1	B78B	haalt byte (0-255)
C8E3	B7AD	BASIC-functie VAL
C921	B7EB	haalt adres en waarde van byte
C92D	B7F7	FAC in adresformaat veranderen (bereik 0-65535)
C943	B80D	BASIC-instructie PEEK
C95A	B824	BASIC-instructie POKE
C963	B82D	BASIC-instructie WAIT
C97F	B849	FAC=FAC+0.5
C986	B850	afrekken: FAC=constante(A/Y)-FAC
C989	B853	afrekken: FAC=ARG-FAC
C99D	B867	optellen: FAC=constante(A/Y)+FAC
C9A0	B86A	optellen: FAC=ARG+FAC
CAB4	B97E	overflow uitlezen
CAF2	B9BC	constanten: reële getallen voor LOG
CB20	B9EA	BASIC-functie LOG
CB5E	BA28	vermenigvuldigen: FAC=constante(A/Y)*FAC
CB61	BA2B	vermenigvuldigen: FAC=ARG*FAC
CBC2	BA8C	ARG=constante(A/Y)
CC18	BAE2	FAC=FAC*10
CC2F	BAF9	reëel getal 10
CC34	BAFE	FAC=FAC/10
CC45	BBOF	delen: FAC=constante(A/Y)/FAC
CC4A	BB12	delen: FAC=ARG/FAC
CC00	BB8A	division by zero uitlezen
CCD8	BBA2	FAC=constante(A/Y)
CCFD	BBC4	accu#4=FAC
CD03	BBCA	accu#3=FAC
CD0A	BBD0	variabele=FAC
CD32	BBFC	FAC=ARG
CD42	BC0C	ARG=FAC
CD51	BC1B	FAC ronden

CD61	BC2B	teken halen, dat aan FAC voorafgaat
CD6F	BC39	BASIC-functie SGN
CD8E	BC58	BASIC-functie ABS
CD91	BC5B	constante (A/Y) met FAC vergelijken
CDD1	BC9B	FAC in integer veranderen
CE02	BCCC	BASIC-functie INT
CE29	BCF3	ASCII in FAC veranderen
CEE9	BDB3	reële constanten voor reël naar ASCII
CF78	BDC2	regelnummer bij foutmelding uitlezen
CF93	BDDD	FAC veranderen in ASCII-formaat
DOC2	BF11	reël getal 0.5
DOC7	BF16	binair getallen om FAC in ASCII te veranderen
D108	BF71	BASIC-functie SQR
D112	BF78	machtsverheffen: $FAC = constante(A/Y)^{FAC}$
D115	BF7B	machtsverheffen: $FAC = ARG^{FAC}$
D156	BFBF	reële constanten voor EXP
D184	BFED	BASIC-functie EXP
D1D7	EO43	polynoomberekening
D1ED	EO59	polynoomberekening
D221	EO8D	reële constanten voor RND
D229	EO97	BASIC-functie RND
D282	E264	BASIC-functie COS
D289	E26B	BASIC-functie SIN
D2D2	E2B4	BASIC-functie TAN
D2FE	E2EO	reële constanten voor SIN en COS
D32C	E30E	BASIC-functie ATN
D35C	E33E	reële constanten voor ATN
D399	E3A2	kopie van CHRGET-routine

7.5 Aanpassing van programma's voor de VIC 20

De Commodore 64 heeft meer mogelijkheden dan de VIC 20: HI-RES grafiek in kleur, sprites, zijn capaciteiten als synthesizer. Dat neemt niet weg, dat de software voor de VIC 20 heel goed te gebruiken is. Maar de modules en cassettes kunnen niet eenvoudigweg worden overgenomen.

1) Programma's in machinetaal

De aanpassing van deze programmatuur is niet moeilijk, maar wel gecompliceerder dan bij BASIC-programma's. U heeft er in ieder geval de tabel uit paragraaf 7.1 bij nodig.

Komt u op een sprongadres dat op een adres in de ROM wijst, dan zoekt u dat adres op in de tabel van de Commodore 64. In het programma wisselt u het in tegen het adres van de VIC 20.

Het verschil tussen de adressen kunt u ook zelf berekenen. Bij de VIC 20 is \$CB7B het beginadres van de BASIC-routine GET, \$2000 hoger dan bij de Commodore 64 (\$AB7B). Vanaf \$E000 berekent u de VIC 20-adressen door die van de Commodore 64 met 3 te verlagen.

2) BASIC-programma's

De Commodore 64 heeft een ander beeldschermformaat. Het formaat in de programma's voor de VIC 20 moet worden herzien of er moet na 22 regels een RETURN worden gegeven. Anders verdwijnen er tekens in dezelfde regel.

De POKE-instructies leveren problemen op. Er zijn twee soorten:

1) POKE-instructies in het beeldschermgeheugen

Alle adressen moeten worden veranderd. Vergelijk de tabellen van de tekengeneratoren en de beeldschermpagina.

2) Waarden, die de ROM van de VIC 20 betreffen.

Hiervoor geldt hetzelfde als bij de verandering van programma's in machinetaal.

7.6 Aanpassing van programma's voor de grote CBM-computers

De meeste grote CBM-computers hebben hetzelfde beeldschermformaat als de Commodore 64. Bovendien kan hun BASIC-versie volledig worden overgenomen. Alleen de POKE-instructies moeten worden veranderd. Daarvoor moet u de adressen kennen van de grote Commodore computers.

Literatuur over de PET/CBM-machines vindt u in de literatuurlijst.

Globaal gezien verschillen BASIC en machinetaal van deze computers niet van hun tegenhangers in de Commodore 64. De 6510 is volledig compatibel met de 6502. Alleen de sprongadressen naar de routines in het bedrijfsysteem zijn anders, evenals de adressen van het

beeldschermgeheugen en de vectoren van de zeropage. Alle programma's van de PET- en CBM-computers en van de VIC 20 kunnen dus zonder veel moeite op de Commodore 64 worden gebruikt. Bovendien kunt u deze programma's uitbreiden met kleur, klank en grafiek van de Commodore 64. Zulke mogelijkheden heeft geen enkele PET- en CBM-machine. Het RAM van 64K is als werkgeheugen groot genoeg om de kwaliteit van de programma's aanzienlijk te vergroten.

A propos: cassettes voor de VIC 20 kunt u alleen inlezen nadat ze op een grote CBM-machine zijn geladen en vervolgens op cassette zijn gezet.

De Commodore 64 kan een grote CBM simuleren. Het volgende programma realiseert deze emulatie:

```
10 POKE 56576,1: REM CBM-beeldscherm naar 32768
20 POKE 53272,4
30 POKE 648,128: REM BASIC ontvangt beeldschermadres
40 POKE 1024,0: POKE 56,128: REM BASIC begin/eind
50 PRINT CHR$(147)
60 NEW
```

Deze emulatie zet de video-RAM op de nieuwe adressen van de BASIC-wijzers.

8 ROM-LISTING VAN DE COMMODORE 64

8.1 Gebruik van de ROM-listing

De assemblerlisting van het bedrijfssysteem en de BASIC-interpreter van de Commodore 64 nemen meer dan 100 bladzijden in beslag. U kunt er voor een eigen assemblerprogrammering veel ideeën uit opdoen. Het grootste nut heeft de ROM-listing wanneer u de routines van bedrijfssysteem en BASIC-interpreter als subroutines in uw eigen programma's verwerkt. Dat bespaart veel programmeerwerk.

Aan het begin van de BASIC-interpreter staan alle adressen van de BASIC-instructies in een sprongtabel naar de routines. Bij GOTO hoort het adres \$A8A0. Daar staat de instructie. De routines voor rekenoperaties met reële getallen (zo worden de normale variabelen opgeslagen) beginnen bij adres \$B800 (zie hoofdstuk 5). Het bedrijfssysteem begint bij \$E400. Eerst komt de beeldschermeditor, daarna de bediening van het toetsenbord en de interruptroutine. Dan volgen de in- en uitvoerroutines voor de seriële IEC-bus, de RS-232 interface en de bediening van de cassetterecorder. De belangrijkste routines staan weer in een sprongtabel aan het eind van de ROM, vanaf adres \$FF81. In hoofdstuk 7 zijn de sprongpunten op een rijtje gezet. Het voorbeeld van de blokverschuivingsroutine van de BASIC-interpreter illustreert hoe zulke routines worden gebruikt.

Bij het programmeren in machinetaal moet vaak een geheugenbereik worden verschoven, bijvoorbeeld een grafiek of tabel. De verschuiving hoeft niet te worden geprogrammeerd en getest, men kan een bestaande routine gebruiken. Na het inlezen van de parameters kan de subroutine worden opgeroepen. Het sprongadres is \$A3BF.

Om \$24BA-\$29C0 zo te verschuiven, dat ze bij \$3000 eindigen, leest u de begin- en eindadressen van het oude en nieuwe blok in:

```
LDA #$BA ;
LDY #$24 ; wijzer op begin van oude blok
STA $5F ;
STY $60 ;
LDA #$C1 ;
LDY #$29 ; wijzer op eind van oude blok + 1
STA $5A ;
STY $5B ;
```

```

LDA #$01 ;
LDY #$30 ; wijzer op eind van nieuwe blok + 1
STA $58 ;
STY $59 ;
JSR $A3BF; oproep van verschuivingsroutine

```

Deze routine verschuift geheugenbereiken naar hogere adressen. De BASIC-interpreter heeft de routine nodig om het array-bereik naar boven te schuiven, wanneer een nieuwe variabele moet worden ingevoerd.

De ROM-listing verduidelijkt de werking van BASIC-interpreter en bedrijfssysteem. Een juist gebruik van deze routines bespaart niet alleen tijd, maar ook geheugenruimte.

8.2 Alfabetische lijst van ROM-routines

Met de volgende 62 trefwoorden en hun verwijzingen heeft u het adres van een ROM-routine zo gevonden. Op een vetgedrukt trefwoord volgt (meestal) een opsomming van lemmata die relevante informatie voor het hoofdlemma bevatten. Achter de verwijzingen staan tussen haakjes de trefwoorden vermeld die afwijken van hoofdlemma of verwijzing. Een hoofdlemma wordt (meestal) afgesloten met een lijst van adressen met bijbehorende ROM-routines. De hoofdlemmata zijn zo algemeen mogelijk gehouden.

ACCU

zie : FAC

ADRES

zie ook: regel (adresformaat)
array (array-element)
band (beginadres)
BASIC (sprongadres)
BASIC-functie
BASIC-instructie
byte (GETADR)
CIA (basisadres)
controle (eindadres)
FAC (GETADR)

fout (foutmelding)
IEC-bus (secundair adres)
NMI (NMI-sprong)
programma (beginadres)

\$FCDB adreswijzer verhogen

ARG

zie ook: FAC

\$BA8C ARG=constante(A/Y)

\$BBFC ARG naar FAC

ARRAY

zie ook: routine

\$B30E adres van array-element berekenen

\$B2E9 array-element zoeken

\$B261 array-variabele invoegen

\$B194 wijzer op eerste array-element berekenen

ASCII

zie ook: FAC

reël

\$E691 ASCII-code in beeldschermcode veranderen

\$BCF3 ASCII in reël veranderen

BAND

zie ook: interrupt (interruptroutine)
programma

\$F8E2 band voor leesactie voorbereiden

\$F864 bandbuffer op band zetten

\$F80D bandbufferwijzer verhogen

\$F7EA bandheader naar naam zoeken

\$F7D0 beginadres van bandbuffer halen

\$FBA6 bit op band zetten

\$F814 block van band lezen

\$F199 byte van band halen

\$F76A header op band zetten

\$FCCA motor van recorder uit
\$F82E inlezen van ingedrukte bandtoets
\$F817 wachten op bandtoets
\$F838 wachten op bandtoets voor schrijfactie

BASIC

zie ook: fout (BASIC-foutmelding)
 programma (BASIC-start)
 RAM

\$A717 BASIC-code in begrijpelijke tekst veranderen
\$E394 BASIC koude start
\$E453 BASIC-vectoren laden
\$E447 tabel van BASIC-vectoren
\$E37B sprongadres van BASIC NMI

BASIC-FUNCTIE

\$A052 adressen van BASIC-functies
\$BC58 ABS
\$B78B ASC
\$E30E ATN
\$E33E ATN-constanten
\$B6EC CHR\$
\$E264 COS
\$E2E0 COS-constanten
\$BFED EXP
\$BFBF EXP-constanten
\$B3F4 FN
\$B3E1 FN-syntax controleren
\$B37D FRE
\$B6A3 FRESTR: stringbeheer
\$BCCC INT
\$B700 LEFT\$
\$B77C LEN
\$B9EA LOG
\$B98C LOG-constanten
\$B737 MID\$
\$B80D PEEK
\$B824 POKE
\$B39E POS
\$B72C RIGHT\$
\$E097 RND
\$E3BA RND-beginwaarde
\$BC39 SGN

\$E26B SIN
\$E2E0 SIN-constanten
\$BF71 SQR
\$B465 STR\$
\$E2B4 TAN
\$B7AD VAL

BASIC-INSTRUCTIE

\$A00C adressen van BASIC-instructies (min 1)
\$A09E BASIC-instructies
\$A7ED BASIC-instructie uitvoeren
\$E1C7 CLOSE
\$A65E CLR
\$AA86 CMD
\$A857 CONT
\$A8F8 DATA
\$B3B3 DEF
\$B081 DIM
\$A831 END
\$A742 FOR
\$AB7B GET
\$A883 GOSUB
\$A8A0 GOTO
\$A928 IF
\$ABBF INPUT
\$ABA5 INPUT#
\$A9A5 LET
\$A69C LIST
\$E168 LOAD
\$A642 NEW
\$AD1D NEXT
\$A94B ON
\$E1B1 OPEN
\$B824 POKE
\$AAA0 PRINT
\$AA80 PRINT#
\$AC06 READ
\$A93B REM
\$A81D RESTORE
\$A8D2 RETURN
\$A871 RUN
\$E156 SAVE
\$A82F STOP
\$E12A SYS
\$E165 VERIFY
\$B82D WAIT

BASIC-OPERATOR

\$AFE9 AND
\$A080 hiërarchiecodes van BASIC-operatoren
\$AED4 NOT
\$AFE6 OR

BASIC-ROUTINE

\$E112 BASIN
\$E10C BSOUT
\$E11E CHKIN
\$E118 CKOUT
\$E124 GETIN

BEDRIJFSSYSTEEM

zie ook: fout (foutmelding)
 controle

\$FE18 flag zetten voor meldingen van bedrijfssysteem
\$E45F, \$FOBD meldingen van bedrijfssysteem
\$F12B uitvoer van meldingen van bedrijfssysteem

BEELDSCHERM

zie ook: ASCII (beeldschermcode)

\$E4DA achtergrondkleur instellen
\$E505 beeldschermformaat halen
\$E9FF beeldschermregel wissen
\$E518 beeldscherm reset
\$E8EA beeldscherm scrollen
\$E544 beeldscherm wissen
\$E50A cursor instellen/halen
\$E556 cursor home
\$E56C cursorpositie berekenen
\$ECFO tabel: LSB van begin beeldschermregels
\$EA1C teken en kleur op beeldscherm zetten
\$E716 teken op beeldscherm uitvoeren
\$E632 teken van beeldscherm halen
\$E5A0 videocontroller starten

BIT

zie ook: band
byte
FAC
IEC-bus (bitteller)
RS-232 (databit)

\$BA59 vermenigvuldiging van bits

BUFFER

zie : band (bandbuffer/bandbufferwijzer)
RS-232 (RS-232 buffer)
toets

BYTE

zie ook: band
beeldscherm (LSB)
FAC (GETBYT)
IEC-bus
regel (MSB)
RS-232

\$B79B GETBYT: bytewaarde naar X halen

\$B7EB GETADR, GETBYT: waarde van 16 en 8 bits halen

CIA

\$E500 basisadres van CIA halen

CODE

zie : ASCII (ASCII-code/beeldschermcode)
BASIC (BASIC-code)
BASIC-operator (hiërarchiecode)
interpreter (interpretercode)
toets (decodeertabel)
kleur (kleurcode)

CONSTANTE

zie ook: ARG
BASIC-functie
FAC
regel
tijd (timerconstante)

\$AEAD constante: pi

CONTROLE

zie ook: FAC (vergelijken)

\$E684 apostrof?
\$FD02 auto-start ROM?
\$B3A6 direkte-modus?
\$FCD1 eindadres bereikt?
\$AEFD gelijk aan actueel teken?
\$AEFA haakje open?
\$AEF7 haakje gesloten?
\$AA1D heeft teken cijfer?
\$AEFD komma?
\$B113 letter?
\$AD8D numeriek?
\$A3FB plaats in geheugen?
\$EB48 SHIFT, CTRL, C=?
\$EC44 stuurteken?
\$A82C, \$F8D0 stopptoets?
\$AD8F string?
\$B016 vergelijken
\$AF14 variabele van bedrijfssysteem?

CURSOR

zie : beeldscherm

ELEMENT

zie ook: array (array-element)

\$AE83 volgend element van term halen

FAC

zie ook: ARG

\$BC5B constant(A/Y) met FAC vergelijken
\$B86A $FAC = ARG + FAC$
\$B853 $FAC = ARG - FAC$
\$BA2B $FAC = ARG * FAC$
\$BB12 $FAC = ARG / FAC$
\$BF7B $FAC = ARG^FAC$
\$BBA2 $FAC = constante(A/Y)$
\$B867 $FAC = constante(A/Y) + FAC$
\$B850 $FAC = constante(A/Y) - FAC$
\$BA28 $FAC = constante(A/Y) * FAC$
\$BBOF $FAC = constante(A/Y) / FAC$
\$BF78 $FAC = ARG^{constante(A/Y)}$
\$B849 $FAC = FAC + 0.5$
\$BAE2 $FAC = FAC * 10$
\$BAFE $FAC = FAC / 10$
\$BBCA FAC naar accu3#
\$BBC7 FAC naar accu4#
\$BCOC FAC naar ARG
\$BBDO FAC naar variabele
\$BDDD FAC in ASCII veranderen en naar \$100
\$B7F7 GETADR: FAC veranderen in positief 16-bits getal
\$B947 mantisse van FAC inverteren
\$BC2B voorteken van FAC halen

FILE

\$F31F fileparameters geven
\$F30F logisch filenummer zoeken
\$FE00 parameter voor actieve file geven
\$FDF9 parameter voor filenaam geven

FLAG

zie : bedrijfssysteem
IEC-bus (time-out flag)

FRMEVL

\$AD9E FRMEVL: willekeurig term interpreteren

FRMNUM

\$AD8A term halen; numeriek?

FOUT

\$A328 adressen van foutmeldingen
\$A19E BASIC-foutmeldingen
\$AB4D foutenbehandeling bij invoer
\$F6FB foutmelding van bedrijfssysteem
\$A437 uitvoer van foutmelding
\$BDC2 uitvoer van regelnummer bij foutmelding

GARBAGE COLLECTION

\$B526 garbage collection

GEHEUGEN

zie ook: controle (ROM)

\$FE25 bovengrens van RAM instellen/halen
\$FD10 identificatie van ROM-moduul
\$A408 maakt plaats in geheugen
\$FE34 ondergrens van RAM instellen/halen
\$E3BF RAM om BASIC te starten
\$FD50 start werkgeheugen (RAM)
\$EA24 wijzer op kleur-RAM berekenen

HARDWARE

\$FD15 hardware en I/O-vectoren instellen/halen
\$FD30 tabel van hardware en I/O-vectoren

HEADER

zie : band (bandheader/header)
programma (programmaheader)

IEC-BUS

\$FB97 bitteller instellen voor seriële uitvoer
\$ED40, \$EDDD byte op seriële bus uitvoeren
\$EE13 byte van seriële bus halen
\$EDOC LISTEN zenden
\$EDB9 secundair adres naar LISTEN
\$EDC7 secundair adres naar TALK
\$ED09 TALK zenden
\$FE21 time-out flag voor IEC-bus zetten
\$EDFE UNLISTEN zenden
\$EDEF UNTALK zenden

INSTRUCTIE

\$A906 volgende instructie zoeken

INTEGER

zie ook: reël

\$BDCD uitvoer van positief integer getal in A/X
\$A9C4 waardetoewijzing INTEGER

INTERPRETER

\$A7AE interpreterlus
\$A364 meldingen van interpreter
\$A579 verandering van regel in interpretercode

INTERRUPT

\$EA31 interruptroutine
\$F92C interruptroutine voor band lezen
\$FBCD, \$FC6A interruptroutine voor band schrijven
\$FF48 IRQ-sprong
\$FCB8 IRQ-vector instellen
\$FD98 IRQ-vectoren

INVOER

zie : fout (foutenbehandeling)
lus (wachtlus)
regel
hardware (I/O-vector)

KERNAL

\$FF81 KERNAL sprongtabel

KLEUR

zie ook: beeldscherm (achtergrondkleur)
geheugen (kleur-RAM)

\$E8DA tabel van kleurcodes

LUS

zie ook: interpreter (interpreterlus)

\$A480 wachtlus voor invoer

\$E5CA wachtlus voor invoer via toetsenbord

NMI

zie ook: BASIC (BASIC NMI)
RS-232

\$FE43 NMI-sprong

PARAMETER

zie : file (fileparameter)
routine
string (stringparameter)

POLYNOM

\$E043 polynoomberekening 1

\$E059 polynoomberekening 2

PROGRAMMA

\$A613 beginadres van programmaregel berekenen
\$F72C programmaheader van band lezen
\$A4ED programmaregel invoegen
\$A4A9 programmaregel wissen
\$A533 programmaregels opnieuw koppelen
\$A49C programmaregels wissen en invoegen
\$F84A programma van band laden
\$A68E programmawijzer op BASIC-start

REEL

zie ook: ASCII

\$BF16, \$BDB3 constanten voor re^uel naar ASCII
\$B1A5 constante: re^uel getal -32768
\$BF11 constante: re^uel getal 0.5
\$BAF9 constante: re^uel getal 10
\$B1B2, \$BC9B re^uel veranderen in integer
\$A9D6 waardetoewijzing REEL

REGEL

zie ook: beeldscherm (beeldschermregel)
fout (regelnummer)
interpreter
programma (programmaregel)

\$E965 invoegen van voortzettingsregel
\$A560 invoer van regel
\$E6B6 MSB voor begin van regels opnieuw berekenen
\$E9C8 regel naar boven schuiven
\$A96B regelnummer halen en in adresformaat veranderen
\$A909 volgende regel zoeken

REGISTER

\$B983 register naar rechts

ROUTINE

zie ook: BASIC-routine
IEC-bus (LISTEN, TALK, UNLISTEN, UNTALK)
interrupt (interruptroutine)
RS-232 (CHKIN-, GET-, NMI-routine)

\$F157 BASIN
\$F1CA BSOUT
\$A3B8 blockverschuivingsroutine
\$E3A2 CHRGET (kopie)
\$F13E GETIN
\$F20F CHKIN
\$F250 CKOUT
\$F32F CLALL
\$F291 CLOSE
\$F333 CLRCH
\$B34C hulproutine voor berekening van array
\$F49E LOAD
\$F34A OPEN
\$E219 parameter voor OPEN halen
\$E1D4 parameters voor LOAD en SAVE halen
\$FCE2 resetroutine
\$F5DD SAVE
\$A38E stack-zoekroutine

RS-232

zie ook: tijd

\$F014 uitvoer in RS-232 buffer
\$EEBB, \$F208 uitvoer van RS-232
\$F1B8 byte van RS-232 halen
\$EF4A databits voor RS-232 berekenen
\$FED6 NMI-routine voor RS-232
\$F04D CHKIN-routine voor RS-232
\$F086 GET-routine voor RS-232
\$FED6 NMI-routine voor RS-232

SPATIE

\$AB3B uitvoer van een spatie

STACK

zie : routine (stack-zoekroutine)
string

STATUS

\$FE07 status halen

STRING

zie ook: BASIC-functie (FRESTR: stringbeheer)
controle

\$B4F4 plaats reserveren voor string
\$B67A string naar gereserveerd bereik
\$B487 string halen; wijzer naar A/Y
\$B782 stringparameter halen
\$B761 stringparameter van stack halen
\$B475 stringwijzer berekenen
\$AB1E uitvoer van string
\$B63D verbinding van strings
\$B02E vergelijken van strings
\$AA2C waardetoewijzing aan normale string
\$A9D9 waardetoewijzing string

TERM

\$AEF1 term tussen haakjes halen

TIJD

zie ook: IEC-bus (time-out flag)

\$BF3A constanten om TI in TI\$ te veranderen
\$AF84, \$F6DD tijd halen
\$F6E4 tijd instellen
\$F69B tijd verhogen
\$FEC2 timerconstanten voor RS-232 baud rate, NTSC-versie
\$E4EC timerconstanten voor RS-232 baud rate, PAL-versie

TOETS

zie ook: band (bandtoets)
 controle (C=/CTRL/SHIFT/stoptoets)
 invoer (toetsenbord)

\$EB81 decodeertabel 1 voor toetsenbord
\$EBC2 decodeertabel 2 voor toetsenbord
\$EC03 decodeertabel 3 voor toetsenbord
\$EC78 decodeertabel 4 voor toetsenbord
\$F6ED stoptoets inlezen
\$E5B4 teken halen uit buffer van toetsenbord
\$EA87 toetsenbord uitlezen
\$E4E0 wachten op C=-toets
\$EB79 wijzer op decodeertabellen voor toetsenbord

UITVOER

zie : fout (foutmelding)
 hardware (I/O-vector)
 IEC-bus (seriële uitvoer)
 RS-232
 spatie
 string
 vraagteken

VARIABELE

zie ook: array (array-variabele)
 controle
 FAC

\$B1D1 gedimensioneerde variabele halen
\$AF28, \$B08B variabele halen
\$B11D variabele invoegen

VECTOR

zie : BASIC (BASIC-vector)
 hardware (I/O-vector)
 interrupt (IRQ-vector)

VRAAGTEKEN

\$AB45 uitvoer van vraagteken

WIJZER

zie : adres (adreswijzer)
array
band (bandbufferwijzer)
geheugen
programma (programmawijzer)
string (stringwijzer)
toets

8.3 Commodore 64 ROM-listing

```

programma (programmawijzer)
string (stringwijzer)
toets

```

```
A000 94 E3 ; STARTVECTOR $E394
```

```
A002 7B E3 ; NMI-VECTOR $E37B
```

```
A004 43 42 4D 42 41 53 49 43; CBMBASIC
```

***** ADRESSEN VAN BASIC-INSTRUCTIES

	INTERPRETERCODE	ADRES	INSTRUCTIE
A00C	30 A8	\$80	\$A831 END
A00E	41 A7	\$81	\$A742 FOR
A010	1D AD	\$82	\$AD1E NEXT
A012	F7 A8	\$83	\$A8F8 DATA
A014	A4 AB	\$84	\$ABA5 INPUT#
A016	BE AB	\$85	\$ABBF INPUT
A018	80 B0	\$86	\$B081 DIM
A01A	05 AC	\$87	\$AC06 READ
A01C	A4 A9	\$88	\$A9A5 LET
A01E	9F A8	\$89	\$A8A0 GOTO
A020	70 A8	\$8A	\$A871 RUN
A022	27 A9	\$8B	\$A928 IF
A024	1C A8	\$8C	\$A81D RESTORE
A026	82 A8	\$8D	\$A883 GOSUB
A028	D1 A8	\$8E	\$A8D2 RETURN
A02A	3A A9	\$8F	\$A93B REM
A02C	2E A8	\$90	\$A82F STOP
A02E	4A A9	\$91	\$A94B ON
A030	2C B8	\$92	\$B82D WAIT
A032	67 E1	\$93	\$E168 LOAD
A034	55 E1	\$94	\$E156 SAVE
A036	64 E1	\$95	\$E165 VERIFY
A038	B2 B3	\$96	\$B3B3 DEF
A03A	23 B8	\$97	\$B824 POKE
A03C	7F AA	\$98	\$AA80 PRINT#
A03E	9F AA	\$99	\$AAA0 PRINT
A040	56 A8	\$9A	\$A857 CONT
A042	9B A6	\$9B	\$A69C LIST
A044	5D A6	\$9C	\$A65E CLR
A046	85 AA	\$9D	\$AA86 CMD

A048	29	E1	\$9E	\$E12A	SYS
A04A	BD	E1	\$9F	\$E1BE	OPEN
A04C	C6	E1	\$A0	\$E1C7	CLOSE
A04E	7A	AB	\$A1	\$AB7B	GET
A050	41	A6	\$A2	\$A642	NEW

***** ADRESSEN VAN BASIC-FUNCTIES

A052	39	BC	\$B4	\$BC39	SGN
A054	CC	BC	\$B5	\$BCCC	INT
A056	58	BC	\$B6	\$BC58	ABS
A058	10	03	\$B7	\$0310	USR
A05A	7D	B3	\$B8	\$B37D	FRE
A05C	9E	B3	\$B9	\$B39E	POS
A05E	71	BF	\$BA	\$BF71	SQR
A060	97	E0	\$BB	\$E097	RND
A062	EA	B9	\$BC	\$B9EA	LOG
A064	ED	BF	\$BD	\$BFED	EXP
A066	64	E2	\$BE	\$E264	COS
A068	6B	E2	\$BF	\$E26B	SIN
A06A	B4	E2	\$C0	\$E2B4	TAN
A06C	0E	E3	\$C1	\$E30E	ATN
A06E	0D	B8	\$C2	\$B80D	PEEK
A070	7C	B7	\$C3	\$B77C	LEN
A072	65	B4	\$C4	\$B465	STR\$
A074	AD	B7	\$C5	\$B7AD	VAL
A076	8B	B7	\$C6	\$B78B	ASC
A078	EC	B6	\$C7	\$B6EC	CHR\$
A07A	00	B7	\$C8	\$B700	LEFT\$
A07C	2C	B7	\$C9	\$B72C	RIGHT\$
A07E	37	B7	\$CA	\$B737	MID\$

***** HIERARCHIECODES EN ADRESSEN (-1)
; VAN OPERATOREN

A080	79	69	B8	\$79,	\$B86A	OPTELLEN
A083	79	52	B8	\$79,	\$B853	AFTREKKEN
A086	7B	2A	BA	\$7B,	\$BA2B	VERMENIGVULDIGEN
A089	7B	11	BB	\$7B,	\$BB12	DELEN
A08C	7F	7A	BF	\$7F,	\$BF7B	MACHTSVERHEFFEN
A08F	50	E8	AF	\$50,	\$AFE9	AND
A092	46	E5	AF	\$46,	\$AFE6	OR
A095	7D	B3	BF	\$7D,	\$BFB4	VOORTEKENVERANDERING
A098	5A	D3	AE	\$5A,	\$AED4	NOT
A09B	64	15	BO	\$64,	\$B016	VERGELIJK

***** BASIC-INSTRUCTIES

A09E	45	4E		END	
AOA0	C4	46	4F D2 4E 45 58 D4	FOR	NEXT
AOA8	44	41	54 C1 49 4E 50 55	DATA	INPUT#
AOB0	54	A3	49 4E 50 55 D4 44	INPUT	DIM
AOB8	49	CD	52 45 41 C4 4C 45	READ	LET
AOC0	D4	47	4F 54 CF 52 55 CE	GOTO	RUN
AOC8	49	C6	52 45 53 54 4F 52	IF	RESTORE

A0D0	C5 47 4F 53 55 C2 52 45	GOSUB	RETURN	
A0D8	54 55 52 CE 52 45 CD 53	REM	STOP	
A0E0	54 4F D0 4F CE 57 41 49	ON	WAIT	
A0E8	D4 4C 4F 41 C4 53 41 56	LOAD	SAVE	
A0F0	C5 56 45 52 49 46 D9 44	VERIFY	DEF	
A0F8	45 C6 50 4F 4B C5 50 52	POKE	PRINT#	
A100	49 4E 54 A3 50 52 49 4E	PRINT		
A108	D4 43 4F 4E D4 4C 49 53	CONT	LIST	
A110	D4 43 4C D2 43 4D C4 53	CLR	CMD	SYS
A118	59 D3 4F 50 45 CE 43 4C	OPEN	CLOSE	
A120	4F 53 C5 47 45 D4 4E 45	GET	NEW	
A128	D7 54 41 42 A8 54 CF 46	TAB(	TO	
A130	CE 53 50 43 A8 54 48 45	SPC(	THEN	
A138	CE 4E 4F D4 53 54 45 D0	NOT	STOP	
A140	AB AD AA AF DE 41 4E C4	+--*/^	AND	
A148	4F D2 BE BD BC 53 47 CE	OR <=>	SGN	
A150	49 4E D4 41 42 D3 55 53	INT	ABS	USR
A158	D2 46 52 C5 50 4F D3 53	FRE	POS	SQR
A160	51 D2 52 4E C4 4C 4F C7	RND	LOG	
A168	45 58 D0 43 4F D3 53 49	EXP	COS	SIN
A170	CE 54 41 CE 41 54 CE 50	TAN	ATN	PEEK
A178	45 45 CB 4C 45 CE 53 54	LEN	STR\$	
A180	52 A4 56 41 CC 41 53 C3	VAL	ASC	
A188	43 48 52 A4 4C 45 46 54	CHR\$	LEFT\$	
A190	A4 52 49 47 48 54 A4 4D	RIGHT\$	MID\$	
A198	49 44 A4 47 CF 00	GO		
***** BASIC-FOUTMELDING				
A19E	54 4F	1	TOO MANY FILES	
A1A0	4F 20 4D 41 4E 59 20 46			
A1A8	49 4C 45 D3 46 49 4C 45	2	FILE OPEN	
A1B0	20 4F 50 45 CE 46 49 4C	3	FILE NOT OPEN	
A1B8	45 20 4E 4F 54 20 4F 50			
A1C0	45 CE 46 49 4C 45 20 4E	4	FILE NOT FOUND	
A1C8	4F 54 20 46 4F 55 4E C4	5	DEVICE NOT PRESENT	
A1D0	44 45 56 49 43 45 20 4E			
A1D8	4F 54 20 50 52 45 53 45			
A1E0	4E D4 4E 4F 54 20 49 4E	6	NOT INPUT FILE	
A1E8	50 55 54 20 46 49 4C C5			
A1F0	4E 4F 54 20 4F 55 54 50	7	NOT OUTPUT FILE	
A1F8	55 54 20 46 49 4C C5 4D			
A200	49 53 53 49 4E 47 20 46	8	MISSING FILENAME	
A208	49 4C 45 20 4E 41 4D C5			
A210	49 4C 4C 45 47 41 4C 20	9	ILLEGAL DEVICE NUMBER	
A218	44 45 56 49 43 45 20 4E			
A220	55 4D 42 45 D2 4E 45 58	10	NEXT WITHOUT FOR	
A228	54 20 57 49 54 48 4F 55			
A230	54 20 46 4F D2 53 59 4E	11	SYNTAX	
A238	54 41 D8 52 45 54 55 52	12	RETURN WITHOUT GOSUB	
A240	4E 20 57 49 54 48 4F 55			

A248	54 20 47 4F 53 55 C2 4F	13 OUT OF DATA
A250	55 54 20 4F 46 20 44 41	
A258	54 C1 49 4C 4C 45 47 41	14 ILLEGAL QUANTITY
A260	4C 20 51 55 41 4E 54 49	
A268	54 D9 4F 56 45 52 46 4C	15 OVERFLOW
A270	4F D7 4F 55 54 20 4F 46	16 OUT OF MEMORY
A278	20 4D 45 4D 4F 52 D9 55	17 UNDEF'D STATEMENT
A280	4E 44 45 46 27 44 20 53	
A288	54 41 54 45 4D 45 4E D4	
A290	42 41 44 20 53 55 42 53	18 BAD SUBSCRIPT
A298	43 52 49 50 D4 52 45 44	19 REDIM'D ARRAY
A2A0	49 4D 27 44 20 41 52 52	
A2A8	41 D9 44 49 56 49 53 49	20 DIVISION BY ZERO
A2B0	4F 4E 20 42 59 20 5A 45	
A2B8	52 CF 49 4C 4C 45 47 41	21 ILLEGAL DIRECT
A2C0	4C 20 44 49 52 45 43 D4	
A2C8	54 59 50 45 20 4D 49 53	22 TYPE MISMATCH
A2D0	4D 41 54 43 C8 53 54 52	23 STRING TOO LONG
A2D8	49 4E 47 20 54 4F 4F 20	
A2E0	4C 4F 4E C7 46 49 4C 45	24 FILE DATA
A2E8	20 44 41 54 C1 46 4F 52	25 FORMULA TOO COMPLEX
A2F0	4D 55 4C 41 20 54 4F 4F	
A2F8	20 43 4F 4D 50 4C 45 D8	
A300	43 41 4E 27 54 20 43 4F	26 CAN'T CONTINUE
A308	4E 54 49 4E 55 C5 55 4E	27 UNDEF'D FUNCTION
A310	44 45 46 27 44 20 46 55	
A318	4E 43 54 49 4F CE 56 45	28 VERIFY
A320	52 49 46 D9 4C 4F 41 C4	29 LOAD

***** ADRESSEN VAN FOUTMELDINGEN

A328 9E A1 AC A1 B5 A1 C2 A1
A330 D0 A1 E2 A1 F0 A1 FF A1
A338 10 A2 25 A2 35 A2 3B A2
A340 4F A2 5A A2 6A A2 72 A2
A348 7F A2 90 A2 9D A2 AA A2
A350 BA A2 C8 A2 D5 A2 E4 A2
A358 ED A2 00 A3 OE A3 1E A3
A360 24 A3 83 A3

***** MELDINGEN VAN DE INTERPRETER

A364	OD 4F 4B OD	OK
A368	00 20 20 45 52 52 4F 52	ERROR
A370	00 20 49 4E 20 00 OD OA	IN
A378	52 45 41 44 59 2E OD OA	READY.
A380	00 OD OA 42 52 45 41 4B	BREAK
A388	00 A0	


```

***** STAPELZOEKROUTINE
;      VOOR FOR-NEXT EN GOSUB
A38A  BA      TSX
A38B  E8      INX
A38C  E8      INX
A38D  E8      INX
A38E  E8      INX
A38F  BD 01 01 LDA $0101,X
A392  C9 81    CMP #$81      ; FOR-CODE
A394  D0 21    BNE $A3B7
A396  A5 4A    LDA $4A
A398  D0 0A    BNE $A3A4
A39A  BD 02 01 LDA $0102,X
A39D  85 49    STA $49
A39F  BD 03 01 LDA $0103,X
A3A2  85 4A    STA $4A
A3A4  DD 03 01 CMP $0103,X
A3A7  D0 07    BNE $A3B0
A3A9  A5 49    LDA $49
A3AB  DD 02 01 CMP $0102,X
A3AE  F0 07    BEQ $A3B7
A3B0  8A      TXA
A3B1  18      CLC
A3B2  69 12    ADC #$12      ; STAPELWIJZER MET 18 VERHOGEN
A3B4  AA      TAX
A3B5  D0 D8    BNE $A38F      ; VOLGENDE LUS TOETSEN
A3B7  60      RTS
***** BLOK-VERSCHUIF-ROUTINE
A3B8  20 08 A4 JSR $A408      ; TOETST OP PLAATS IN GEHEUGEN
A3BB  85 31    STA $31
A3BD  84 32    STY $32
A3BF  38      SEC      ; INSPRINGPUNT
A3C0  A5 5A    LDA $5A      ; $5F/$60 OUD BLOKBEGIN
A3C2  E5 5F    SBC $5F      ; $5A/$5B OUD BLOKEINDE+1
A3C4  85 22    STA $22      ; $58/$59 NIEUW BLOKEINDE+1
A3C6  A8      TAY
A3C7  A5 5B    LDA $5B
A3C9  E5 60    SBC $60
A3CB  AA      TAX
A3CC  E8      INX
A3CD  98      TYA
A3CE  F0 23    BEQ $A3F3
A3D0  A5 5A    LDA $5A
A3D2  38      SEC
A3D3  E5 22    SBC $22
A3D5  85 5A    STA $5A
A3D7  B0 03    BCS $A3DC
A3D9  C6 5B    DEC $5B
A3DB  38      SEC

```

```

A3DC  A5 58    LDA $58
A3DE  E5 22    SBC $22
A3E0  85 58    STA $58
A3E2  B0 08    BCS $A3EC
A3E4  C6 59    DEC $59
A3E6  90 04    BCC $A3EC
A3E8  B1 5A    LDA ($5A),Y
A3EA  91 58    STA ($58),Y
A3EC  88        DEY
A3ED  D0 F9    BNE $A3E8
A3EF  B1 5A    LDA ($5A),Y
A3F1  91 58    STA ($58),Y
A3F3  C6 5B    DEC $5B
A3F5  C6 59    DEC $59
A3F7  CA        DEX
A3F8  D0 F2    BNE $A3EC
A3FA  60        RTS

```

***** PLAATS IN STACK?

```

A3FB  0A        ASL          ; ACCU MOET HET HALVE GETAL
A3FC  69 3E    ADC #$3E      ; AAN BENODIGDE PLAATS BEVATTEN
A3FE  B0 35    BCS $A435     ; GEEFT 'OUT OF MEMORY'
A400  85 22    STA $22
A402  BA        TSX
A403  E4 22    CPX $22
A405  90 2E    BCC $A435     ; GEEFT 'OUT OF MEMORY'
A407  60        RTS

```

***** MAAKT PLAATS IN GEHEUGEN
VOOR INVVOEGEN VAN REGELS
EN VARIABLEN

```

;
;
A408  C4 34    CPY $34
A40A  90 28    BCC $A434
A40C  D0 04    BNE $A412     ; A/Y=ADRES, TOTDAT PLAATS
A40E  C5 33    CMP $33       ; GEBRUIKT WORDT
A410  90 22    BCC $A434
A412  48        PHA
A413  A2 09    LDX #$09
A415  98        TYA
A416  48        PHA
A417  B5 57    LDA $57,X     ; REGISTER VOOR REKENOPERATIES
A419  CA        DEX          ; SAVEN
A41A  10 FA    BPL $A416
A41C  20 26 B5 JSR $B526     ; GARBAGE COLLECTION
A41F  A2 F7    LDX #$F7
A421  68        PLA
A422  95 61    STA $61,X     ; REGISTER TERUGHALEN
A424  E8        INX
A425  30 FA    BMI $A421
A427  68        PLA
A428  A8        TAY

```

```

A429 68      PLA
A42A C4 34    CPY $34
A42C 90 06    BCC $A434      ; OK, KLAAR
A42E D0 05    BNE $A435      ; GEEN PLAATS: OUT OF MEMORY
A430 C5 33    CMP $33
A432 B0 01    BCS $A435
A434 60      RTS
A435 A2 10    LDX #$10      ; FOUTNUMMER: OUT OF MEMORY
***** FOUTSPRONG
A437 6C 00 03 JMP ($0300)    ; JMP $E38B, BASIC WARME START
***** FOUTMELDING UITVOEREN
A43A 8A      TXA      ; FOUTMELDING IN X-REGISTER, 1-30
A43B 0A      ASL
A43C AA      TAX
A43D BD 26 A3 LDA $A326,X
A440 85 22    STA $22
A442 BD 27 A3 LDA $A327,X    ; ADRES VAN FOUTMELDING HALEN
A445 85 23    STA $23
A447 20 CC FF JSR $FFCC      ; CLRCH: ACTIEVE I/O-KANALEN
;                               TERUGZETTEN
A44A A9 00    LDA #$00
A44C 85 13    STA $13      ; I/O-FLAG TERUGZETTEN
A44E 20 D7 AA JSR $AAD7      ; CR EN LF UITVOEREN
A451 20 45 AB JSR $AB45      ; ? UITVOEREN
A454 A0 00    LDY #$00
A456 B1 22    LDA ($22),Y    ; TEKST VAN DE FOUTMELDING
A458 48      PHA
A459 29 7F    AND #$7F
A45B 20 47 AB JSR $AB47      ; FOUTMELDING UITVOEREN
A45E C8      INY
A45F 68      PLA
A460 10 F4    BPL $A456
A462 20 7A A6 JSR $A67A      ; BASIC-WIJZER INITIALISEREN,
;                               CONT BLOKKEREN
A465 A9 69    LDA #$69
A467 A0 A3    LDY #$A3      ; POINTER A/Y OP ERROR
A469 20 1E AB JSR $AB1E      ; STRING UITVOEREN
A46C A4 3A    LDY $3A      ; PROGRAMMA MODUS
A46E C8      INY
A46F F0 03    BEQ $A474      ; NEE
A471 20 C2 BD JSR $BDC2      ; IN REGELNUMMER UITVOEREN
A474 A9 76    LDA #$76
A476 A0 A3    LDY #$A3      ; POINTER OP READY
A478 20 1E AB JSR $AB1E      ; STRING UITVOEREN
A47B A9 80    LDA #$80
A47D 20 90 FF JSR $FF90      ; FLAG VOOR DIREKTE MODUS ZETTEN
***** WACHTLUS BIJ INVOER
A480 6C 02 03 JMP ($0302)    ; JMP $A483
A483 20 60 A5 JSR $A560      ; BASIC-REGEL IN INVOERBUFFER

```

```

A486 86 7A STX $7A ; HALEN
A488 84 7B STY $7B ; CHRGET: WIJZER OP INVOERBUFFER
A48A 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
A48D AA TAX
A48E FO FO BEQ $A480 ; BUFFER LEEG, BLIJF WACHTEN
A490 A2 FF LDX #$FF
A492 86 3A STX $3A ; KENMERK VOOR DIREKTE MODUS
A494 90 06 BCC $A49C ; CIJFER, BASIC-REGEL INVVOEGEN
A496 20 79 A5 JSR $A579 ; BASIC-REGEL IN INTERPRETERCODE
; VERANDERN
A499 4C E1 A7 JMP $A7E1 ; INSTRUCTIE UITVOERN
***** WISSEN EN INVVOERN VAN
; PROGRAMMAREGELS
A49C 20 6B A9 JSR $A96B ; REGELNUMMER EN ADRESFORMAAT
; VERANDERN IN $14/$15
A49F 20 79 A5 JSR $A579 ; BASIC-REGEL IN INTERPRETERCODE
; VERANDERN
A4A2 84 0B STY $0B ; WIJZER IN INVVOERBUFFER
A4A4 20 13 A6 JSR $A613 ; ADRES VAN BASIC-REGEL BEREKENEN
A4A7 90 44 BCC $A4ED ; NIET AANWEZIG, DAN WISSEN
***** PROGRAMMAREGEL WISSEN
A4A9 A0 01 LDY #$01
A4AB B1 5F LDA ($5F),Y
A4AD 85 23 STA $23
A4AF A5 2D LDA $2D
A4B1 85 22 STA $22
A4B3 A5 60 LDA $60
A4B5 85 25 STA $25
A4B7 A5 5F LDA $5F
A4B9 88 DEY
A4BA F1 5F SBC ($5F),Y
A4BC 18 CLC
A4BD 65 2D ADC $2D
A4BF 85 2D STA $2D
A4C1 85 24 STA $24
A4C3 A5 2E LDA $2E
A4C5 69 FF ADC #$FF
A4C7 85 2E STA $2E
A4C9 E5 60 SBC $60
A4CB AA TAX
A4CC 38 SEC
A4CD A5 5F LDA $5F
A4CF E5 2D SBC $2D
A4D1 A8 TAY
A4D2 B0 03 BCS $A4D7
A4D4 E8 INX
A4D5 C6 25 DEC $25
A4D7 18 CLC
A4D8 65 22 ADC $22

```

```

A4DA 90 03      BCC $A4DF
A4DC C6 23      DEC $23
A4DE 18         CLC
A4DF B1 22      LDA ($22),Y ; VERSCHUIFLUS
A4E1 91 24      STA ($24),Y
A4E3 C8         INY
A4E4 D0 F9      BNE $A4DF
A4E6 E6 23      INC $23
A4E8 E6 25      INC $25
A4EA CA         DEX
A4EB D0 F2      BNE $A4DF
***** PROGRAMMAREGEL INVOEGEN
A4ED 20 59 A6 JSR $A659      ; CLR-INSTRUCTIE
A4FO 20 33 A5 JSR $A533      ; PROGRAMMAREGELS WEER KOPPELEN
A4F3 AD 00 02 LDA $0200      ; TEKEN IN BUFFER
A4F6 F0 88      BEQ $A480    ; NEE, DAN NAAR WACHTLUS BIJ INVOER
A4F8 18         CLC
A4F9 A5 2D      LDA $2D
A4FB 85 5A      STA $5A
A4FD 65 0B      ADC $0B

A4FF 85 58      STA $58
A501 A4 2E      LDY $2E
A503 84 5B      STY $5B
A505 90 01      BCC $A508
A507 C8         INY
A508 84 59      STY $59
A50A 20 B8 A3 JSR $A3B8      ; BASIC-REGELS VERSCHUIVEN
A50D A5 14      LDA $14
A50F A4 15      LDY $15
A511 8D FE 01 STA $01FE
A514 8C FF 01 STY $01FF
A517 A5 31      LDA $31
A519 A4 32      LDY $32
A51B 85 2D      STA $2D
A51D 84 2E      STY $2E
A51F A4 0B      LDY $0B
A521 88         DEY
A522 B9 FC 01 LDA $01FC,Y    ; TEKEN UIT INVOERBUFFER
A525 91 5F      STA ($5F),Y ; IN PROGRAMMA KOPIEREN
A527 88         DEY
A528 10 F8      BPL $A522
A52A 20 59 A6 JSR $A659      ; CLR-INSTRUCTIE
A52D 20 33 A5 JSR $A533      ; PROGRAMMAREGELS OPNIEUW KOPPELEN
A530 4C 80 A4 JMP $A480      ; TOT INVOERWACHTLUS
***** BASIC-PROGRAMMAREGELS
; OPNIEUW KOPPELEN

```

```

A533 A5 2B LDA $2B
A535 A4 2C LDY $2C
A537 85 22 STA $22
A539 84 23 STY $23
A53B 18 CLC
A53C A0 01 LDY #$01
A53E B1 22 LDA ($22),Y
A540 F0 1D BEQ $A55F
A542 A0 04 LDY #$04
A544 C8 INY
A545 B1 22 LDA ($22),Y
A547 D0 FB BNE $A544
A549 C8 INY
A54A 98 TYA
A54B 65 22 ADC $22
A54D AA TAX
A54E A0 00 LDY #$00
A550 91 22 STA ($22),Y
A552 A5 23 LDA $23
A554 69 00 ADC #$00
A556 C8 INY
A557 91 22 STA ($22),Y
A559 86 22 STX $22
A55B 85 23 STA $23
A55D 90 DD BCC $A53C
A55F 60 RTS
***** INVOER VAN EEN REGEL
A560 A2 00 LDX #$00
A562 20 12 E1 JSR $E112 ; TEKEN HALEN
A565 C9 0D CMP #$0D ; RETURN-TOETS
A567 F0 0D BEQ $A576 ; JA, INVOER STOPPEN
A569 9D 00 02 STA $0200,X ; TEKEN IN INVOERBUFFER
; OPSLAAN
A56C E8 INX
A56D E0 59 CPX #$59 ; 89STE TEKEN?
A56F 90 F1 BCC $A562 ; NEE, MEER TEKENS HALEN
A571 A2 17 LDX #$17 ; NUMMER VAN STRING TOO LONG
A573 4C 37 A4 JMP $A437 ; FOUTMELDING UITVOEREN
A576 4C CA AA JMP $AACA ; BUFFER MET $0 AFSLUITEN,
; CR UITVOEREN
***** VERANDERING VAN EEN REGEL IN
; INTERPRETER-CODE
A579 6C 04 03 JMP ($0304) ; JMP $A57C
A57C A6 7A LDX $7A
A57E A0 04 LDY #$04
A580 84 0F STY $0F ; FLAG VOOR AANHALINGSTEKEN
A582 BD 00 02 LDA $0200,X ; TEKEN UIT BUFFER HALEN
A585 10 07 BPL $A58E ; GEEN BASIC-CODE?
A587 C9 FF CMP #$FF ; CODE VOOR PI?

```

A589	FO 3E	BEQ \$A5C9	
A58B	E8	INX	
A58C	DO F4	BNE \$A582	
A58E	C9 20	CMP #\$20	; SPATIE
A590	FO 37	BEQ \$A5C9	
A592	85 08	STA \$08	
A594	C9 22	CMP #\$22	; AANHALINGSTEKEN
A596	FO 56	BEQ \$A5EE	
A598	24 0F	BIT \$0F	
A59A	70 2D	BVS \$A5C9	
A59C	C9 3F	CMP #\$3F	; VRAAGTEKEN
A59E	DO 04	BNE \$A5A4	
A5A0	A9 99	LDA #\$99	; DOOR PRINT-CODE VERVANGEN
A5A2	DO 25	BNE \$A5C9	
A5A4	C9 30	CMP #\$30	; 0
A5A6	90 04	BCC \$A5AC	; KLEINER?
A5A8	C9 3C	CMP #\$3C	
A5AA	90 1D	BCC \$A5C9	
A5AC	84 71	STY \$71	
A5AE	A0 00	LDY #\$00	
A5B0	84 0B	STY \$0B	
A5B2	88	DEY	
A5B3	86 7A	STX \$7A	
A5B5	CA	DEX	
A5B6	C8	INY	
A5B7	E8	INX	
A5B8	BD 00 02	LDA \$0200,X	; TEKEN IN BUFFER
A5BB	38	SEC	
A5BC	F9 9E A0	SBC \$A09E,Y	; VERGELIJK MET INSTRUCTIES
			IN TABEL
A5BF	FO F5	BEQ \$A5B6	
A5C1	C9 80	CMP #\$80	
A5C3	DO 30	BNE \$A5F5	
A5C5	05 0B	ORA \$0B	; GEVONDEN, BASIC-CODE =
			TELLER+\$80
A5C7	A4 71	LDY \$71	
A5C9	E8	INX	
A5CA	C8	INY	
A5CB	99 FB 01	STA \$01FB,Y	; BASIC-CODE OPSLAAN
A5CE	B9 FB 01	LDA \$01FB,Y	; EN STATUSREGISTER INSTELLEN
A5D1	FO 36	BEQ \$A609	; EINDE, DAN KLAAR
A5D3	38	SEC	
A5D4	E9 3A	SBC #\$3A	; PUNKKOMMA
A5D6	FO 04	BEQ \$A5DC	; VERBINDINGSSTREEPJE?
A5D8	C9 49	CMP #\$49	; DATA-CODE?
A5DA	DO 02	BNE \$A5DE	
A5DC	85 0F	STA \$0F	
A5DE	38	SEC	
A5DF	E9 55	SBC #\$55	; REM-CODE?

```

A5E1 DO 9F BNE $A582
A5E3 85 08 STA $08
A5E5 BD 00 02 LDA $0200,X
A5E8 FO DF BEQ $A5C9
A5EA C5 08 CMP $08
A5EC FO DB BEQ $A5C9
A5EE C8 INY
A5EF 99 FB 01 STA $01FB,Y
A5F2 E8 INX
A5F3 DO FO BNE $A5E5
A5F5 A6 7A LDX $7A
A5F7 E6 0B INC $0B
A5F9 C8 INY
A5FA B9 9D AO LDA $A09D,Y
A5FD 10 FA BPL $A5F9 ; VERGELIJK MET TABEL
A5FF B9 9E AO LDA $A09E,Y
A602 DO B4 BNE $A5B8
A604 BD 00 02 LDA $0200,X
A607 10 BE BPL $A5C7
A609 99 FD 01 STA $01FD,Y
A60C C6 7B DEC $7B
A60E A9 FF LDA #$FF ; WIJZER OP INVOERBUFFER
A610 85 7A STA $7A
A612 60 RTS
***** BEGINADRES VAN PROGRAMMA-
; REGEL BEREKENEN
A613 A5 2B LDA $2B
A615 A6 2C LDX $2C ; WIJZER OP BEGIN PROGRAMMA
A617 A0 01 LDY #$01
A619 85 5F STA $5F
A61B 86 60 STX $60
A61D B1 5F LDA ($5F),Y ; LINK-ADRES HIGH
A61F FO 1F BEQ $A640 ; O, DAN EINDE, NIET GEVONDEN
A621 C8 INY
A622 C8 INY
A623 A5 15 LDA $15 ; GEZOCHTE REGELNUMMER HIGH
A625 D1 5F CMP ($5F),Y ; MET ACTUEEL NUMMER VERGELIJKEN
A627 90 18 BCC $A641 ; KLEINER, DAN NIET GEVONDEN
A629 FO 03 BEQ $A62E ; GELIJK, DAN NUMMER LOW TESTEN
A62B 88 DEY
A62C DO 09 BNE $A637 ; ABSOLUTE SPRONG
A62E A5 14 LDA $14
A630 88 DEY
A631 D1 5F CMP ($5F),Y ; REGELNUMMER LOW VERGELIJKEN
A633 90 0C BCC $A641 ; KLEINER
A635 FO 0A BEQ $A641 ; OF GELIJK C=1
A637 88 DEY
A638 B1 5F LDA ($5F),Y
A63A AA TAX

```



```

A63B 88      DEY      ; ADRES VAN VOLGENDE
;             PROGRAMMAREGEL
A63C B1 5F    LDA ($5F),Y
A63E B0 D7    BCS $A617 ; VERDER ZOEKEN
A640 18      CLC
A641 60      RTS
***** BASIC-INSTRUCTIE NEW
A642 D0 FD    BNE $A641
A644 A9 00    LDA #$00
A646 A8      TAY
A647 91 2B    STA ($2B),Y
A649 C8      INY      ; 2 x $00 BIJ BEGIN VAN PROGRAMMA
A64A 91 2B    STA ($2B),Y
A64C A5 2B    LDA $2B
A64E 18      CLC
A64F 69 02    ADC #$02
A651 85 2D    STA $2D  ; BEGIN VAN VARIABELEN=BEGIN
;             VAN PROGRAMMA+2
A653 A5 2C    LDA $2C
A655 69 00    ADC #$00
A657 85 2E    STA $2E
A659 20 8E A6 JSR $A68E ; CHRGET-WIJZER OP BEGIN VAN
;             PROGRAMMA
A65C A9 00    LDA #$00
***** BASIC-INSTRUCTIE CLR
A65E D0 2D    BNE $A68D
A660 20 E7 FF JSR $FFE7 ; CLALL: I/O-KANALEN TERUGZETTEN
A663 A5 37    LDA $37
A665 A4 38    LDY $38
A667 85 33    STA $33  ; BEGIN VAN STRINGS OP EINDE VAN
;             BASIC-RAM
A669 84 34    STY $34
A66B A5 2D    LDA $2D
A66D A4 2E    LDY $2E
A66F 85 2F    STA $2F  ; EINDE VAN VARIABELEN=BEGIN
;             VAN VARIABELEN
A671 84 30    STY $30
A673 85 31    STA $31
A675 84 32    STY $32
A677 20 1D A8 JSR $A81D ; RESTORE-INSTRUCTIE
A67A A2 19    LDX #$19
A67C 86 16    STX $16  ; STRINGDESCRIPTOR-INDEX
;             TERUGZETTEN
A67E 68      PLA
A67F A8      TAY
A680 68      PLA
A681 A2 FA    LDX #$FA
A683 9A      TXS      ; STACKPOINTER INITIALISEREN
A684 48      PHA

```

```

A685 98      TYA
A686 48      PHA
A687 A9 00   LDA #$00
A689 85 3E   STA $3E      ; CONT BLOKKEREN
A68B 85 10   STA $10
A68D 60      RTS
***** PROGRAMMAWIJZER OP
; BASIC-START
A68E 18      CLC
A68F A5 2B   LDA $2B
A691 69 FF   ADC #$FF      ; BASIC-START
A693 85 7A   STA $7A
A695 A5 2C   LDA $2C
A697 69 FF   ADC #$FF      ; MIN 1
A699 85 7B   STA $7B
A69B 60      RTS
***** BASIC-INSTRUCTIE LIST
A69C 90 06   BCC $A6A4      ; CIJFER? (REGELNUMMER)
A69E F0 04   BEQ $A6A4      ; ALLEEN LIST?
A6A0 C9 AB   CMP #$AB      ; CODE VOOR VERBINDINGSSTREEPJE
A6A2 D0 E9   BNE $A68D      ; ANDERE CODE, SYNTAX ERROR
A6A4 20 6B A9 JSR $A96B      ; REGELNUMMER HALEN
A6A7 20 13 A6 JSR $A613      ; STARTADRES PROGRAMMAREGEL
; BEREKENEN
A6AA 20 79 00 JSR $0079      ; CHRGET: LAATSTE TEKEN HALEN
A6AD F0 0C   BEQ $A6BB      ; GEEN REGELNUMMER
A6AF C9 AB   CMP #$AB      ; CODE VOOR VERBINDINGSSTREEPJE
A6B1 D0 8E   BNE $A641      ; NEE, SYNTAX ERROR
A6B3 20 73 00 JSR $0073      ; CHRGET: VOLGEND TEKEN HALEN
A6B6 20 6B A9 JSR $A96B      ; REGELNUMMER HALEN
A6B9 D0 86   BNE $A641
A6BB 68      PLA
A6BC 68      PLA
A6BD A5 14   LDA $14
A6BF 05 15   ORA $15      ; TWEDE REGELNUMMER=0?
A6C1 D0 06   BNE $A6C9      ; NEE
A6C3 A9 FF   LDA #$FF
A6C5 85 14   STA $14
A6C7 85 15   STA $15      ; LIST TOT EINDE
A6C9 A0 01   LDY #$01
A6CB 84 0F   STY $0F
A6CD B1 5F   LDA ($5F),Y   ; LINKADRES HIGH
A6CF F0 43   BEQ $A714      ; JA, KLAAR
A6D1 20 2C A8 JSR $A82C      ; STOP-TOETS?
A6D4 20 D7 AA JSR $AAD7      ; CR UITVOEREN, NIEUWE REGEL
A6D7 C8      INY
A6D8 B1 5F   LDA ($5F),Y
A6DA AA      TAX
A6DB C8      INY      ; REGELNUMMER HALEN

```

A6DC	B1 5F	LDA (\$5F),Y	
A6DE	C5 15	CMP \$15	
A6E0	D0 04	BNE \$A6E6	; VERGELIJK MET EINDNUMMER
A6E2	E4 14	CPX \$14	
A6E4	F0 02	BEQ \$A6E8	
A6E6	B0 2C	BCS \$A714	; GROTER, DAN KLAAR
A6E8	84 49	STY \$49	
A6EA	20 CD BD	JSR \$BDCD	; REGELNUMMER UITVOEREN
A6ED	A9 20	LDA #\$20	; SPATIE
A6EF	A4 49	LDY \$49	
A6F1	29 7F	AND #\$7F	
A6F3	20 47 AB	JSR \$AB47	; TEKEN UITVOEREN
A6F6	C9 22	CMP #\$22	; AANHALINGSTEKEN?
A6F8	D0 06	BNE \$A700	
A6FA	A5 0F	LDA \$0F	
A6FC	49 FF	EOR #\$FF	; FLAG VOOR AANHALINGSTEKEN
			OMDRAAIEN
A6FE	85 0F	STA \$0F	
A700	C8	INY	; GEEN REGELEINDE NA 255 TEKENS,
A701	F0 11	BEQ \$A714	; DAN OPHOUDEN
A703	B1 5F	LDA (\$5F),Y	; TEKEN HALEN
A705	D0 10	BNE \$A717	; GEEN REGELEINDE, DAN LISTEN
A707	A8	TAY	
A708	B1 5F	LDA (\$5F),Y	
A70A	AA	TAX	; BEGINADRES VOLGENDE REGEL
A70B	C8	INY	
A70C	B1 5F	LDA (\$5F),Y	
A70E	86 5F	STX \$5F	
A710	85 60	STA \$60	; ALS WIJZER MARKEREN
A712	D0 B5	BNE \$A6C9	; VERDER GAAN
A714	4C 86 E3	JMP \$E386	; NAAR BASIC WARME START
*****			BASIC-CODE IN INSTRUCTIES
			VERANDEREN
A717	6C 06 03	JMP (\$0306)	; JMP \$A71A
A71A	10 D7	BPL \$A6F3	; GEEN INTERPRETERCODE,
			DAN ZO UITVOEREN
A71C	C9 FF	CMP #\$FF	; CODE VOOR PI
A71E	F0 D3	BEQ \$A6F3	; ZO UITVOEREN
A720	24 0F	BIT \$0F	; AANHALINGSTEKENMODUS?
A722	30 CF	BMI \$A6F3	; DAN TEKEN ZO UITVOEREN
A724	38	SEC	
A725	E9 7F	SBC #\$7F	; OFFSET AFTREKKEN
A727	AA	TAX	; CODE NAAR X
A728	84 49	STY \$49	; TEKENWIJZER MARKEREN
A72A	A0 FF	LDY #\$FF	
A72C	CA	DEX	; EERSTE INSTRUCTIE?
A72D	F0 08	BEQ \$A737	
A72F	C8	INY	
A730	B9 9E A0	LDA \$A09E,Y	; OFFSET VOOR X _e INSTRUCTIE

;			VINDEN
A733	10 FA	BPL \$A72F	
A735	30 F5	BMI \$A72C	; BIT 7 INGESTELD, VOLGENDE
;			INSTRUCTIE
A737	C8	INY	
A738	B9 9E A0	LDA \$A09E,Y	; INSTRUCTIE UIT TABEL
;			HALEN
A73B	30 B2	BMI \$A6EF	; LAATSTE LETTER, DAN KLAAR
A73D	20 47 AB	JSR \$AB47	; TEKEN UITVOEREN
A740	DO F5	BNE \$A737	; VOLGENDE LETTER UITVOEREN
*****			BASIC-INSTRUCTIE FOR
A742	A9 80	LDA #\$80	; INTEGER BLOKKEREN
A744	85 10	STA \$10	
A746	20 A5 A9	JSR \$A9A5	; LET-INSTRUCTIE, ZET
;			FOR-VARIABLE
A749	20 8A A3	JSR \$A38A	; ZOEKT OPEN FOR-NEXT-LUS MET
;			ZELFDE VARIABLEN
A74C	DO 05	BNE \$A753	; NIET GEVONDEN
A74E	8A	TXA	
A74F	69 0F	ADC #\$0F	; STAPELPOINTER VERHOGEN
A751	AA	TAX	
A752	9A	TXS	
A753	68	PLA	; SPRONGADRES VAN STACK HALEN
A754	68	PLA	
A755	A9 09	LDA #\$09	
A757	20 FB A3	JSR \$A3FB	; GENOEG PLAATS IN STACK?
A75A	20 06 A9	JSR \$A906	; ZOEKT VOLGEND BASIC-STATEMENT
A75D	18	CLC	
A75E	98	TYA	; PROGRAMMAWIJZER OP VOLGENDE
;			INSTRUCTIE
A75F	65 7A	ADC \$7A	
A761	48	PHA	; EN OP STACK LEGGEN
A762	A5 7B	LDA \$7B	
A764	69 00	ADC #\$00	
A766	48	PHA	
A767	A5 3A	LDA \$3A	
A769	48	PHA	; ACTUEEL REGELNUMMER OP STACK
A76A	A5 39	LDA \$39	
A76C	48	PHA	
A76D	A9 A4	LDA #\$A4	; TO-CODE
A76F	20 FF AE	JSR \$AEFF	; CODE?
A772	20 8D AD	JSR \$AD8D	; NUMERIEKE VARIABLE?
A775	20 8A AD	JSR \$AD8A	; HAALT NUMERIEKE TERM NAAR FAC
A778	A5 66	LDA \$66	
A77A	09 7F	ORA #\$7F	
A77C	25 62	AND \$62	
A77E	85 62	STA \$62	
A780	A9 8B	LDA #\$8B	
A782	A0 A7	LDY \$A7	; SPRONGADRES ONTHOUDEN

```

A784 85 22 STA $22
A786 84 23 STY $23
A788 4C 43 AE JMP $AE43 ; EINDWAARDE VAN LUS OP STACK
A78B A9 BC LDA #$BC
A78D A0 B9 LDY #$B9 ; WIJZER OP CONSTATE 1
A78F 20 A2 BB JSR $BBA2 ; ALS DEFAULT STEPWAARDE IN FAC
A792 20 79 00 JSR $0079 ; CHRGET: LAATSTE TEKEN HALEN
A795 C9 A9 CMP #$A9 ; STEP-CODE
A797 D0 06 BNE $A79F ; GEEN STEP-WAARDE
A799 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
A79C 20 8A AD JSR $AD8A ; NUMERIEKE TERM NAAR FAC
A79F 20 2B BC JSR $BC2B ; HAALT VOORTEKEN
A7A2 20 38 AE JSR $AE38 ; LEGT VOORTEKEN EN
; STEPWAARDE OP STACK
A7A5 A5 4A LDA $4A
A7A7 48 PHA ; VARIABELENAAM
A7A8 A5 49 LDA $49
A7AA 48 PHA
A7AB A9 81 LDA #$81 ; EN FOR-CODE OP STACK
A7AD 48 PHA
***** INTERPRETERLUS
A7AE 20 2C A8 JSR $A82C ; TEST OP STOPTOETS
A7B1 A5 7A LDA $7A
A7B3 A4 7B LDY $7B ; PROGRAMMAWIJZER
A7B5 C0 02 CPY #$02 ; DIREKTE MODUS
A7B7 EA NOP
A7B8 F0 04 BEQ $A7BE ; JA
A7BA 85 3D STA $3D ; ALS WIJZER VOOR CONT MARKEREN
A7BC 84 3E STY $3E
A7BE A0 00 LDY #$00
A7C0 B1 7A LDA ($7A),Y ; ACTUEEL TEKEN
A7C2 D0 43 BNE $A807 ; GEEN REGELEINDE
A7C4 A0 02 LDY #$02
A7C6 B1 7A LDA ($7A),Y ; EINDE PROGRAMMA
A7C8 18 CLC ; FLAG VOOR END ZETTEN
A7C9 D0 03 BNE $A7CE
A7CB 4C 4B A8 JMP $A84B ; JA, DAN END UITVOEREN
A7CE C8 INY
A7CF B1 7A LDA ($7A),Y
A7D1 85 39 STA $39
A7D3 C8 INY ; ACTUEEL REGELNUMMER ONTHOUDEN
A7D4 B1 7A LDA ($7A),Y
A7D6 85 3A STA $3A
A7D8 98 TYA
A7D9 65 7A ADC $7A ; PROGRAMMAWIJZER OP
; PROGRAMMAREGEL ZETTEN
A7DB 85 7A STA $7A
A7DD 90 02 BCC $A7E1
A7DF E6 7B INC $7B

```

```

A7E1  6C 08 03 JMP ($0308) ; JMP $A7E4 BASIC-INSTRUCTIE
;
A7E4  20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
A7E7  20 ED A7 JSR $A7ED ; STATEMENT UITVOEREN
A7EA  4C AE A7 JMP $A7AE ; TERUG NAAR INTERPRETERLUS
***** BASIC-STATEMENT UITVOEREN
A7ED  FO 3C BEQ $A82B ; EINDE REGEL, DAN KLAAR
A7EF  E9 80 SBC #$80 ; TOKEN?
A7F1  90 11 BCC $A804 ; NEE, DAN NAAR LET-INSTRUCTIE
A7F3  C9 23 CMP #$23
A7F5  B0 17 BCS $A80E ; FUNCTIE-TOKEN OF GO TO
A7F7  0A ASL ; BASIC-INSTRUCTIE, CODE * 2
A7F8  A8 TAY
A7F9  B9 OD AO LDA $A00D,Y
A7FC  48 PHA ; INSTRUCTIE-ADRES UIT TABEL
A7FD  B9 OC AO LDA $A00C,Y
A800  48 PHA ; ALS SPRONGADRES OP STACK
A801  4C 73 00 JMP $0073 ; VOLGEND TEKEN EN INSTRUCTIE
;
A804  4C A5 A9 JMP $A9A5 ; NAAR LET-INSTRUCTIE
*****
A807  C9 3A CMP #$3A ; DUBBELE PUNT
A809  FO D6 BEQ $A7E1
A80B  4C 08 AF JMP $AF08 ; SYNTAX ERROR
***** GOTO-CODE?
080E  C9 4B CMP #$4B ; GO (MIN $80)
0810  DO F9 BNE $A80B
A812  20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
A815  A9 A4 LDA #$A4 ; TO
A817  20 FF AE JSR $AEFF ; CODE?
A81A  4C A0 A8 JMP $A8A0 ; NAAR GOTO-INSTRUCTIE
***** BASIC-INSTRUCTIE RESTORE
A81D  38 SEC
A81E  A5 2B LDA $2B
A820  E9 01 SBC #$01 ; PROGRAMMASTART-1
A822  A4 2C LDY $2C
A824  B0 01 BCS $A827
A826  88 DEY
A827  85 41 STA $41
A829  84 42 STY $42 ; =DATAWIJZER
A82B  60 RTS
***** STOPTOETS?
A82C  20 E1 FF JSR $FFE1 ; STOPTOETS INLEZEN
***** BASIC-INSTRUCTIE STOP
A82F  B0 01 BCS $A832 ; C=1 FLAG VOOR STOP
***** BASIC-INSTRUCTIE END
A831  18 CLC ; C=0 FLAG VOOR END
A832  DO 3C BNE $A870
A834  A5 7A LDA $7A

```

```

A836 A4 7B LDY $7B ; PROGRAMMAWIJZER
A838 A6 3A LDX $3A ; DIREKTE MODUS?
A83A E8 INX
A83B F0 0C BEQ $A849 ; JA
A83D 85 3D STA $3D
A83F 84 3E STY $3E ; WIJZER VOOR CONT INSTELLEN
A841 A5 39 LDA $39
A843 A4 3A LDY $3A ; NUMMER VAN ACTUELE REGEL
A845 85 3B STA $3B
A847 84 3C STY $3C ; ALS REGELNUMMER VOOR CONT
; MARKEREN
A849 68 PLA
A84A 68 PLA ; SPRONGADRES VAN STACK HALEN
A84B A9 81 LDA #$81
A84D A0 A3 LDY #$A3 ; WIJZER OP BREAK
A84F 90 03 BCC $A854 ; END FLAG?
A851 4C 69 A4 JMP $A469 ; NEE, 'BREAK IN XXX' UITVOEREN
A854 4C 86 E3 JMP $E386 ; NAAR BASIC WARME START
***** BASIC-INSTRUCTIE CONT
A857 D0 17 BNE $A870
A859 A2 1A LDX #$1A ; FOUTNUMMER VAN CAN'T CONTINUE
A85B A4 3E LDY $3E ; CONT GEBLOKKEERD?
A85D D0 03 BNE $A862 ; NEE
A85F 4C 37 A4 JMP $A437 ; FOUTMELDING UITVOEREN
A862 A5 3D LDA $3D
A864 85 7A STA $7A ; PROGRAMMAWIJZER
A866 84 7B STY $7B
A868 A5 3B LDA $3B ; EN
A86A A4 3C LDY $3C
A86C 85 39 STA $39 ; REGELNUMMER INSTELLEN
A86E 84 3A STY $3A
A870 60 RTS
***** BASIC-INSTRUCTIE RUN
A871 08 PHP
A872 A9 00 LDA #$00
A874 20 90 FF JSR $FF90 ; FLAG VOOR PROGRAMMAMODUS
; ZETTEN
A877 28 PLP
A878 D0 03 BNE $A87D ; ANDERE TEKENS (REGELNUMMER)?
A87A 4C 59 A6 JMP $A659 ; PROGRAMMAWIJZER OP BEGIN
; VAN PROGRAMMA, CLR
A87D 20 60 A6 JSR $A660 ; CLR-INSTRUCTIE
A880 4C 97 A8 JMP $A897 ; GOTO-INSTRUCTIE
***** BASIC-INSTRUCTIE GOSUB
A883 A9 03 LDA #$03
A885 20 FB A3 JSR $A3FB ; GENOEG PLAATS IN STACK?
A888 A5 7B LDA $7B
A88A 48 PHA ; PROGRAMMAWIJZER
A88B A5 7A LDA $7A

```

```

A88D 48      PHA
A88E A5 3A    LDA $3A
A890 48      PHA      ; EN REGELNUMMER MARKEREN
A891 A5 39    LDA $39
A893 48      PHA
A894 A9 8D    LDA #$8D      ; GOSUB-CODE OP STACK
A896 48      PHA
A897 20 79 00 JSR $0079      ; CHR GOT: LAATSTE TEKEN HALEN
A89A 20 A0 A8 JSR $A8A0      ; GOTO-INSTRUCTIE
A89D 4C AE A7 JMP $A7AE      ; NAAR INTERPRETER-LUS
***** BASIC-INSTRUCTIE GOTO
A8A0 20 6B A9 JSR $A96B      ; REGELNUMMER NAAR $14/$15
A8A3 20 09 A9 JSR $A909      ; VOLGEND REGELBEGIN ZOEKEN
A8A6 38      SEC
A8A7 A5 39    LDA $39
A8A9 E5 14    SBC $14      ; REGELNUMMER < ACTUELE REGEL?
A8AB A5 3A    LDA $3A
A8AD E5 15    SBC $15
A8AF B0 0B    BCS $A8BC      ; NEE
A8B1 98      TYA
A8B2 38      SEC
A8B3 65 7A    ADC $7A
A8B5 A6 7B    LDX $7B      ; ZOEKT VANAF ACTUELE REGEL
A8B7 90 07    BCC $A8C0
A8B9 E8      INX
A8BA B0 04    BCS $A8C0
A8BC A5 2B    LDA $2B      ; ZOEKT VANAF PROGRAMMASTART
A8BE A6 2C    LDX $2C
A8C0 20 17 A6 JSR $A617      ; ZOEKT PROGRAMMAREGEL
A8C3 90 1E    BCC $A8E3      ; NIET GEVONDEN,
; UNDEF'D STATEMENT
A8C5 A5 5F    LDA $5F
A8C7 E9 01    SBC #$01
A8C9 85 7A    STA $7A      ; PROGRAMMAWIJZER OP NIEUWE
; REGEL
A8CB A5 60    LDA $60
A8CD E9 00    SBC #$00
A8CF 85 7B    STA $7B
A8D1 60      RTS
***** BASIC-INSTRUCTIE RETURN
A8D2 D0 FD    BNE $A8D1
A8D4 A9 FF    LDA #$FF
A8D6 85 4A    STA $4A
A8D8 20 8A A3 JSR $A38A      ; VOLGENDE ZIN MET GOSUB-GEGEVEN
; IN STACK ZOEKEN
A8DB 9A      TXS
A8DC C9 8D    CMP #$8D      ; GOSUB-CODE
A8DE F0 0B    BEQ $A8EB      ; GEVONDEN?
A8EO A2 0C    LDX #$0C      ; NUMMER VOOR RETURN WITHOUT

```



```

;
A8E2 2C      .BYTE $2C
A8E3 A2 11    LDX #$02      ; NUMMER VAN UNDEF'D STATEMENT
A8E5 4C 37 A4 JMP $A437     ; FOUTMELDING UITVOEREN
A8E8 4C 08 AF JMP $AF08     ; SYNTAX ERROR UITVOEREN
A8EB 68      PLA
A8EC 68      PLA
A8ED 85 39    STA $39
A8EF 68      PLA      ; REGELNUMMER VAN STACK HALEN
A8F0 85 3A    STA $3A
A8F2 68      PLA
A8F3 85 7A    STA $7A
A8F5 68      PLA      ; PROGRAMMAWIJZER VAN STACK
;                      HALEN
A8F6 85 7B    STA $7B
A8F8 20 06 A9 JSR $A906
A8FB 98      TYA
A8FC 18      CLC
A8FD 65 7A    ADC $7A
A8FF 85 7A    STA $7A

A901 90 02    BCC $A905     ; BIJ PROGRAMMAWIJZER OPTELLEN
A903 E6 7B    INC $7B
A905 60      RTS
***** OFFSET VOLGEND SCHEIDINGS-
;                      TEKEN VINDEN
A906 A2 3A    LDX #$3A      ; DUBBELE PUNT
A908 2C      .BYTE $2C
A909 A2 00    LDX #$00      ; EIND VAN REGEL
A90B 86 07    STX $07
A90D A0 00    LDY #$00      ; Y BEVAT OFFSET
A90F 84 08    STY $08
A911 A5 08    LDA $08
A913 A6 07    LDX $07      ; GEZOCHT TEKEN
A915 85 07    STA $07
A917 86 08    STX $08
A919 B1 7A    LDA ($7A),Y   ; TEKEN HALEN
A91B F0 E8    BEQ $A905     ; EIND VAN REGEL, DAN KLAAR
A91D C5 08    CMP $08
A91F F0 E4    BEQ $A905
A921 C8      INY      ; WIJZER VERHOGEN
A922 C9 22    CMP #$22     ; AANHALINGSTEKEN
A924 D0 F3    BNE $A919
A926 F0 E9    BEQ $A911
***** BASIC-INSTRUCTIE IF
A928 20 9E AD JSR $AD9E     ; FRMEVL: TERM BEREKENEN
A92B 20 79 00 JSR $0079     ; CHRGT: LAATSTE TEKEN
A92E C9 89    CMP #$89     ; GOTO-CODE

```

```

A930 FO 05      BEQ $A937      ; JA
A932 A9 A7      LDA #$A7       ; THEN-CODE
A934 20 FF AE    JSR $AEFF      ; CODE?
A937 A5 61      LDA $61        ; RESULTAAT VAN IF-TERM
A939 DO 05      BNE $A940      ; TERM JUIST?
***** BASIC-INSTRUCTIE REM
A93B 20 09 A9    JSR $A909      ; NEE, BEGIN VAN VOLGENDE
;                                REGEL ZOEKEN
A93E FO BB      BEQ $A8FB      ; PROGRAMMAWIJZER OP VOLGENDE
;                                REGEL
A940 20 79 00    JSR $0079      ; CHRGET: LAATSTE TEKEN
A943 B0 03      BCS $A948      ; GEEN TEKEN?
A945 4C A0 A8    JMP $A8A0      ; NAAR GOTO-INSTRUCTIE
A948 4C ED A7    JMP $A7ED      ; VOLGENDE INSTRUCTIE DECODE-
;                                REN EN UITVOEREN
***** BASIC-INSTRUCTIE ON
A94B 20 9E B7    JSR $B79E      ; BYTE-WAARDE (0-255) HALEN
A94E 48          PHA           ; CODE ONTHOUDEN
A94F C9 8D      CMP #$8D       ; GOSUB-CODE
A951 FO 04      BEQ $A957      ; JA
A953 C9 89      CMP #$89       ; GOTO-CODE
A955 DO 91      BNE $A8E8      ; NEE, DAN SYNTAX ERROR
A957 C6 65      DEC $65        ; TELLER VERLAGEN
A959 DO 04      BNE $A95F      ; NOG NIET NUL?
A95B 68          PLA           ; JA, CODE TERUGHALEN
A95C 4C EF A7    JMP $A7EF      ; EN INSTRUCTIE UITVOEREN
A95F 20 73 00    JSR $0073      ; CHRGET: VOLGEND TEKEN
A962 20 6B A9    JSR $A96B      ; REGELNUMMER HALEN
A965 C9 2C      CMP #$2C       ; KOMMA
A967 FO EE      BEQ $A957      ; JA, DAN VERDER
A969 68          PLA           ; GEEN SPRONG, CODE TERUGHA-
;                                LEN, KLAAR
A96A 60          RTS
***** REGELNUMMER NAAR $14/$15
A96B A2 00      LDX #$00
A96D 86 14      STX $14
A96F 86 15      STX $15      ; VOORINSTELLING REGELNUM-
;                                MER =0
A971 B0 F7      BCS $A96A      ; GEEN CIJFER, DAN KLAAR
A973 E9 2F      SBC #$2F      ; 0, DAN -1 (=HEXWAARDE)
A975 85 07      STA $07      ; ONTHOUDEN
A977 A5 15      LDA $15
A979 85 22      STA $22
A97B C9 19      CMP #$19      ; GETAL AL >= 6400?
A97D B0 D4      BCS $A953      ; DAN SYNTAX ERROR
A97F A5 14      LDA $14
A981 0A          ASL A
A982 26 22      ROL $22      ; GETAL*10 (=2*2+GETAL*2)
A984 0A          ASL A

```

```

A985 26 22 ROL $22
A987 65 14 ADC $14
A989 85 14 STA $14
A98B A5 22 LDA $22
A98D 65 15 ADC $15
A98F 85 15 STA $15
A991 06 14 ASL $14
A993 26 15 ROL $15
A995 A5 14 LDA $14
A997 65 07 ADC $07 ; EN CIJFER ONDER TIEN OPTELLEN
A999 85 14 STA $14
A99B 90 02 BCC $A99F
A99D E6 15 INC $15
A99F 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
A9A2 4C 71 A9 JMP $A971 ; EN VERDER GAAN
***** BASIC-INSTRUCTIE LET
A9A5 20 8B B0 JSR $B08B ; ZOEKT VARIABLE
A9A8 85 49 STA $49
A9AA 84 4A STY $4A ; VARIABLENADRES ONTHOUDEN
A9AC A9 B2 LDA #$B2 ; ==CODE
A9AE 20 FF AE JSR $AEFF ; CODE?
A9B1 A5 0E LDA $0E ; FLAG (INTEGERE GETALLEN)
A9B3 48 PHA
A9B4 A5 0D LDA $0D ; EN TYPE-FLAG (STRING/NUME-
; RIEK) ONTHOUDEN
A9B6 48 PHA
A9B7 20 9E AD JSR $AD9E ; FRMEVL: TERM HALEN
A9BA 68 PLA
A9BB 2A ROL A ; TYPE-FLAG TERUGHALEN
A9BC 20 90 AD JSR $AD90 ; JUIST TYPE?
A9BF D0 18 BNE $A9D9
A9C1 68 PLA
A9C2 10 12 BPL $A9D6 ; REEEL?
***** WAARDETOEWIJZING INTEGER
A9C4 20 1B BC JSR $BC1B ; FAC AFRONDEN
A9C7 20 BF B1 JSR $B1BF ; EN IN INTEGER VERANDEREN
A9CA A0 00 LDY #$00
A9CC A5 64 LDA $64
A9CE 91 49 STA ($49),Y ; WAARDE IN VARIABLE BRENGEN
A9D0 C8 INY
A9D1 A5 65 LDA $65
A9D3 91 49 STA ($49),Y
A9D5 60 RTS
***** WAARDETOEWIJZING REEEL
A9D6 4C D0 BB JMP $BBD0 ; FAC NAAR VARIABLE BRENGEN
***** WAARDETOEWIJZING STRING
A9D9 68 PLA
A9DA A4 4A LDY $4A ; ADRES VAN VARIABLE HIGH
A9DC C0 BF CPY #$BF ; IS VARIABLE TI$?

```

```

A9DE DO 4C      BNE $AA2C      ; NEE
A9EO 20 A6 B6 JSR $B6A6      ; FRESTR
A9E3 C9 06      CMP #$06      ; STRINGLENGTE =6
A9E5 DO 3D      BNE $AA24      ; NEE, ILLEGAL QUANTITY
A9E7 A0 00      LDY #$00
A9E9 84 61      STY $61
A9EB 84 66      STY $66
A9ED 84 71      STY $71
A9EF 20 1D AA JSR $AA1D      ; HEEFT VOLGEND TEKEN CIJFER?
A9F2 20 E2 BA JSR $BAE2      ; FAC=FAC*10
A9F5 E6 71      INC $71      ; PLAATSTELLER VERHOGEN
A9F7 A4 71      LDY $71
A9F9 20 1D AA JSR $AA1D      ; HEEFT VOLGEND TEKEN CIJFER?
A9FC 20 OC BC JSR $BCOC      ; FAC NAAR ARG KOPIEREN
A9FF AA        TAX
AA00 FO 05      BEQ $AA07      ; FAC=0?
AA02 E8        INX
AA03 8A        TXA
AA04 20 ED BA JSR $BAED      ; FAC=FAC+ARG
AA07 A4 71      LDY $71      ; PLAATSTELLER
AA09 C8        INY          ; VERHOGEN
AA0A CO 06      CPY #$06      ; AL 6 PLAATSEN?
AA0C DO DF      BNE $A9ED
AA0E 20 E2 BA JSR $BAE2      : FAC=FAC*10
AA11 20 9B BC JSR $BC9B      ; RECHTER KANTLIJN VAN FAC
;                          RECHT MAKEN
AA14 A6 64      LDX $64
AA16 A4 63      LDY $63      ; INGEVOERDE TIJD
AA18 A5 65      LDA $65
AA1A 4C DB FF JMP $FFDB      ; TIJD INSTELLEN
***** HEEFT TEKEN CIJFER?
AA1D B1 22      LDA ($22),Y   ; HEEFT TEKEN
AA1F 20 80 00 JSR $0080      ; CIJFER?
AA22 90 03      BCC $AA27
AA24 4C 48 B2 JMP $B248      ; ILLEGAL QUANTITY
AA27 E9 2F      SBC #$2F      ; ASCII IN HEX VERANDEREN
AA29 4C 7E BD JMP $BD7E      ; AAN FAC EN ARG DOORGEVEN
***** WAARDETOEWIJZING AAN NORMALE STRING
AA2C A0 02      LDY #$02
AA2E B1 64      LDA ($64),Y   ; STRINGADRES HIGH
AA30 C5 34      CMP $34      ; MET BEGINADRES VAN STRING
;                          VERGELIJKEN
AA32 90 17      BCC $AA4B      ; KLEINER, STRING STAAT IN
;                          PROGRAMMA
AA34 DO 07      BNE $AA3D
AA36 88        DEY
AA37 B1 64      LDA ($64),Y   ; STRINGADRES LOW
AA39 C5 33      CMP $33      ; VERGELIJKEN
AA3B 90 OE      BCC $AA4B      ; STRING IN PROGRAMMA

```

AA3D	A4 65	LDY \$65	
AA3F	C4 2E	CPY \$2E	
AA41	90 08	BCC \$AA4B	
AA43	D0 0D	BNE \$AA52	
AA45	A5 64	LDA \$64	
AA47	C5 2D	CMP \$2D	
AA49	B0 07	BCS \$AA52	
AA4B	A5 64	LDA \$64	
AA4D	A4 65	LDY \$65	
AA4F	4C 68 AA	JMP \$AA68	
AA52	A0 00	LDY #\$00	
AA54	B1 64	LDA (\$64),Y	; LENGTE VAN STRING
AA56	20 75 B4	JSR \$B475	; CONTROLEERT PLAATS IN GEHEU- GEN, ZET STRINGWIJZER
AA59	A5 50	LDA \$50	
AA5B	A4 51	LDY \$51	
AA5D	85 6F	STA \$6F	
AA5F	84 70	STY \$70	
AA61	20 7A B6	JSR \$B67A	; STRING AAN STRINGBEREIK DOORGEVEN
AA64	A9 61	LDA #\$61	
AA66	A0 00	LDY #\$00	
AA68	85 50	STA \$50	
AA6A	84 51	STY \$51	
AA6C	20 DB B6	JSR \$B6DB	; DESCRIPTOR UIT STRINGSTACK WISSEN
AA6F	A0 00	LDY #\$00	
AA71	B1 50	LDA (\$50),Y	; LENGTE
AA73	91 49	STA (\$49),Y	
AA75	C8	INY	
AA76	B1 50	LDA (\$50),Y	; ADRES LOW
AA78	91 49	STA (\$49),Y	
AA7A	C8	INY	
AA7B	B1 50	LDA (\$50),Y	; EN ADRES HIGH
AA7D	91 49	STA (\$49),Y	; IN VARIABELE BRENGEN
AA7F	60	RTS	
***** BASIC-INSTRUCTIE PRINT#			
AA80	20 86 AA	JSR \$AA86	; CMD-INSTRUCTIE
AA83	4C B5 AB	JMP \$ABB5	; HAALT BYTE
AA86	20 9E B7	JSR \$B79E	
AA89	FO 05	BEQ \$AA90	
AA8B	A9 2C	LDA #\$2C	; KOMMA
AA8D	20 FF AE	JSR \$AEFF	; KOMMA?
AA90	08	PHP	
AA91	86 13	STX \$13	; NUMMER VAN UITVOERAPPARAAT ONTHOUDEN
AA93	20 18 E1	JSR \$E118	; CKOUT: UITVOERAPPARAAT INSTELLEN
AA96	28	PLP	
AA97	4C A0 AA	JMP \$AAAA	; NAAR PRINT-INSTRUCTIE

```

AA9A 20 21 AB JSR $AB21 ; STRING PRINTEN
AA9D 20 79 00 JSR $0079 ; CHRGET: LAATSTE TEKEN
***** BASIC-INSTRUCTIE PRINT
AAA0 FO 35 BEQ $AAD7
AAA2 FO 43 BEQ $AAE7
AAA4 C9 A3 CMP #$A3 ; 'TAB(' -CODE
AAA6 FO 50 BEQ $AAF8
AAA8 C9 A6 CMP #$A6 ; 'SPC(' -CODE
AAAA 18 CLC
AAAB FO 4B BEQ $AAF8
AAAD C9 2C CMP #$2C ; KOMMA
AAAF FO 37 BEQ $AAE8
AAB1 C9 3B CMP #$3B ; PUNKKOMMA
AAB3 FO 5E BEQ $AB13
AAB5 20 9E AD JSR $AD9E ; FRMEVL: TERM HALEN
AAB8 24 0D BIT $0D ; TYPE-FLAG
AABA 30 DE BMI $AA9A ; STRING?
AABC 20 DD BD JSR $BDDD ; FAC VERANDEREN IN ASCII
AABF 20 87 B4 JSR $B487 ; STRINGPARAMETER HALEN
AAC2 20 21 AB JSR $AB21 ; STRING PRINTEN
AAC5 20 3B AB JSR $AB3B ; CURSOR RECHTS/SPATIE
; UITVOEREN
AAC8 D0 D3 BNE $AA9D ; VERDER GAAN
AACA A9 00 LDA #$00 ; INVOERBUFFER
AACC 9D 00 02 STA $0200,X ; MET $0 AFSLUITEN
AACF A2 FF LDX #$FF
AAD1 A0 01 LDY #$01 ; WIJZER OP INVOERBUFFER
AAD3 A5 13 LDA $13 ; NUMMER VAN UITVOERAPPARAAT
AAD5 D0 10 BNE $AAE7
AAD7 A9 0D LDA #$0D ; CR: CARRIAGE RETURN
AAD9 20 47 AB JSR $AB47 ; UITVOEREN
AADC 24 13 BIT $13 ; LOGISCHE FILENUMMER
AADE 10 05 BPL $AAE5 ; < 128?
AAEO A9 0A LDA #$0A ; LF: LINE FEED
AAE2 20 47 AB JSR $AB47 ; UITVOEREN
AAE5 49 FF EOR #$FF
AAE7 60 RTS
AAE8 38 SEC ; TABULATOR VOOR TIENTALLEN
; MET KOMMA
AAE9 20 FO FF JSR $FFFO ; CURSORPOSITIE HALEN
AAEC 98 TYA
AAED 38 SEC
AAEE E9 0A SBC #$0A ; MIN 10
AAFO B0 FC BCS $AAEE ; NIET NEGATIEF?
AAF2 49 FF EOR #$FF ; INVERTEREN
AAF4 69 01 ADC #$01 ; PLUS 1
AAF6 D0 16 BNE $AB0E
***** TAB( (C=1) EN SPC( (C=0)

```

```

AAF8 08      PHP      ; FLAG ONTHOUDEN
AAF9 38      SEC
AAFA 20 FO FF JSR $FFFO ; CURSORPOSITIE HALEN
AAFD 84 09    STY $09   ; EN ONTHOUDEN
AAFF 20 9B B7 JSR $B79B ; BYTE-WAARDE HALEN
AB02 C9 29    CMP #$29  ; HAAKJE GESLOTEN?
AB04 D0 59    BNE $AB5F ; NEE, SYNTAX ERROR
AB06 28      PLP
AB07 90 06    BCC $ABOF ; NAAR SPC(
AB09 8A      TXA      ; TAB-WAARDE IN ACCU
AB0A E5 09    SBC $09   ; MET CURSORPOSITIE VERGELIJKEN
ABOC 90 05    BCC $AB13 ; WAARDE < CURSORPOSITIE, DAN
; KLAAR
ABOE AA      TAX
ABOF E8      INX
AB10 CA      DEX
AB11 D0 06    BNE $AB19
AB13 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN
AB16 4C A2 AA JMP $AAA2 ; EN VERDER GAAN
AB19 20 3B AB JSR $AB3B ; CURSOR NAAR RECHTS/SPATIE
; UITVOEREN
AB1C D0 F2    BNE $AB10 ; NAAR BEGIN VAN LUS
***** STRING UITVOEREN
AB1E 20 87 B4 JSR $B487 ; STRINGPARAMETER HALEN
AB21 20 A6 B6 JSR $B6A6 ; FRESTR
AB24 AA      TAX      ; STRINGLENGTE
AB25 A0 00    LDY #$00
AB27 E8      INX
AB28 CA      DEX
AB29 FO BC    BEQ $AAE7 ; STRING KLAAR?
AB2B B1 22    LDA ($22),Y ; TEKEN VAN STRING
AB2D 20 47 AB JSR $AB47 ; UITVOEREN
AB30 C8      INY
AB31 C9 0D    CMP #$0D   ; CR: CARRIAGE RETURN
AB33 D0 F3    BNE $AB28 ; NEE, VERDER
AB35 20 E5 AA JSR $AAE5 ; FOUT! LF-UITVOER JSR $AADC?
AB38 4C 28 AB JMP $AB28 ; EN VERDER GAAN
***** CURSOR NAAR RECHTS/SPATIE
; UITVOEREN
AB3B A5 13    LDA $13    ; UITVOER IN FILE?
AB3D FO 03    BEQ $AB42 ; NEE, BEELDSCHERM, DAN
; CURSOR NAAR RECHTS
AB3F A9 20    LDA #$20   ; SPATIE
AB41 2C 1D    .BYTE $2C
AB42 A9 1D    LDA #$1D   ; CURSOR NAAR RECHTS
AB44 2C      .BYTE $2C
AB3F A9 3F    LDA #$3F   ; VRAAGTEKEN
AB47 20 0C E1 JSR $E10C ; UITVOEREN
AB4A 29 FF    AND $FF    ; FLAGS ZETTEN

```

```

AB4C 60      RTS
***** FOUTENBEHANDELING BIJ INVOER
AB4D A5 11    LDA $11      ; FLAG VOOR INPUT/GET/READ
AB4F FO 11    BEQ $AB62    ; INPUT
AB51 30 04    BMI $AB57    ; READ
AB53 A0 FF    LDY #$FF
AB55 DO 04    BNE $AB5B    ; GET
***** FOUT BIJ READ
AB57 A5 3F    LDA $3F      ; NUMMER VAN DATAREGEL
AB59 A4 40    LDY $40
***** FOUT BIJ GET
AB5B 85 39    STA $39
AB5D 84 3A    STY $3A      ; =REGELNUMMER VAN FOUT
AB5F 4C 08 AF JMP $AF08    ; SYNTAX ERROR
***** FOUT BIJ INPUT
AB62 A5 13    LDA $13      ; NUMMER VAN INVOERAPPARAAT
AB64 FO 05    BEQ $AB6B    ; TOETSENBORD?
AB66 A2 18    LDX #$18     ; NUMMER VOOR FILE DATA
AB68 4C 37 A4 JMP $A437    ; FOUTMELDING UITVOEREN
AB6B A9 0C    LDA #$0C
AB6D A0 AD    LDY #$AD     ; WIJZER OP REDO FROM START
AB6F 20 1E AB JSR $AB1E    ; STRING UITVOEREN
AB72 A5 3D    LDA $3D
AB74 A4 3E    LDY $3E      ; PROGRAMMAWIJZER TERUG
AB76 85 7A    STA $7A
AB78 84 7B    STY $7B      ; OP INPUT INSTRUCTIE
AB7A 60      RTS
***** BASIC-INSTRUCTIE GET
AB7B 20 A6 B3 JSR $B3A6    ; DIRECTE MODUS?
AB7E C9 23    CMP #$23     ; #
AB80 DO 10    BNE $AB92    ; NEE
AB82 20 73 00 JSR $0073    ; CHRGET: VOLGEND TEKEN HALEN
AB85 20 9E B7 JSR $B79E    ; BYTEWAARDE HALEN
AB88 A9 2C    LDA #$2C     ; KOMMA
AB8A 20 FF AE JSR $AEFF    ; CODE?
AB8D 86 13    STX $13
AB8F 20 1E E1 JSR $E11E    ; CHKIN
AB92 A2 01    LDX #$01
AB94 A0 02    LDY #$02     ; WIJZER OP EINDE BUFFER ($201)
AB96 A9 00    LDA #$00     ; BUFFER MET $0 AFSLUITEN
AB98 8D 01 02 STA $0201
AB9B A9 40    LDA #$40     ; GET-FLAG
AB9D 20 0F AC JSR $ACOF    ; WAARDETOEWIJZING AAN VARIABLE
ABAO A6 13    LDX $13      ; INVOERAPPARAAT
ABA2 DO 13    BNE $ABB7    ; GEEN TOETSENBORD, DAN CLRCH
ABA4 60      RTS
***** BASIC-INSTRUCTIE INPUT#
ABA5 20 9E B7 JSR $B79E    ; HAALT BYTEWAARDE
ABA8 A9 2C    LDA #$2C     ; KOMMA

```



```

ABAA 20 FF AE JSR $AEFF ; KOMMA?
ABAD 86 13 STX $13 ; INVOERAPPARAAT
ABAF 20 1E E1 JSR $E11E ; CHKIN
ABB2 20 CE AB JSR $ABCE ; INPUT ZONDER DIALOGSTRING
ABB5 A5 13 LDA $13
ABB7 20 CC FF JSR $FFCC ; CLRCH ZET INVOERAPPARAAT
; TERUG
ABBA A2 00 LDX #$00
ABBC 86 13 STX $13 ; INVOER WEER VIA TOETSENBOARD
ABBE 60 RTS
***** BASIC-INSTRUCTIE INPUT
ABBF C9 22 CMP #$22 ; AANHALINGSTEKEN
ABC1 D0 0B BNE $ABCE ; NEE
ABC3 20 BD AE JSR $ABED ; DIALOGSTRING HALEN
ABC6 A9 3B LDA #$3B ; PUNKKOMMA
ABC8 20 FF AE JSR $AEFF ; CODE?
ABCB 20 21 AB JSR $AB21 ; STRING UITVOEREN
ABCE 20 A6 B3 JSR $B3A6 ; DIREKTE MODUS?
ABD1 A9 2C LDA #$2C ; KOMMA
ABD3 8D FF 01 STA #01FF ; AAN BEGIN VAN BUFFER
ABD6 20 F9 AB JSR $ABF9 ; VRAAGTEKEN UITVOEREN
ABD9 A5 13 LDA $13 ; NUMMER VAN INVOERAPPARAAT
ABDB F0 0D BEQ $ABEA ; TOETSENBOARD?
ABDD 20 B7 FF JSR $FFB7 ; STATUS HALEN
ABE0 29 02 AND #$02
ABE2 F0 06 BEQ $ABEA ; TIME-OUT?
ABE4 20 B5 AB JSR $ABB5 ; JA, CLRCH, TOETSENBOARD WEER
; ACTIVEREN
ABE7 4C F8 A8 JMP $A8F8 ; VOLGENDE INSTRUCTIE UITVOEREN
ABEA AD 00 02 LDA $0200 ; EERSTE TEKEN HALEN
ABED D0 1E BNE $ACOD ; EINDE?
ABEF A5 13 LDA $13 ; JA, INVOERAPPARAAT
ABF1 D0 E3 BNE $ABD6 ; NIET TOETSENBOARD?
ABF3 20 06 A9 JSR $A906 ; OFFSET VAN VOLGENDE
; INSTRUCTIE ZOEKEN
ABF6 4C FB A8 JMP $A8FB ; PROGRAMMAWIJZER OP VOLGENDE
; INSTRUCTIE
ABF9 A5 13 LDA $13 ; INVOERAPPARAAT
ABFB D0 06 BNE $AC03 ; NIET TOETSENBOARD?
ABFD 20 45 AB JSR $AB45 ; VRAAGTEKEN UITVOEREN
AC00 20 3B AB JSR $AB3B ; SPATIE UITVOEREN
AC03 4C 60 A5 JMP $A560 ; INVOERREGEL HALEN
***** BASIC-INSTRUCTIE READ
AC06 A6 41 LDX $41
AC08 A4 42 LDY $42 ; DATAWIJZER HALEN
AC0A A9 98 LDA #$98 ; READ-FLAG
AC0C 2C .BYTE $2C
AC0D A9 00 LDA #$00
AC0F 85 11 STA $11 ; ZETTEN

```

AC11	86	43	STX	\$43	
AC13	84	44	STY	\$44	; INPUTWIJZER OP INVOER
AC15	20	8B	BO JSR	\$B08B	; ZOEKT VARIABELE
AC18	85	49	STA	\$49	
AC1A	84	4A	STY	\$4A	; VARIABELENADRES OPSLAAN
AC1C	A5	7A	LDA	\$7A	
AC1E	A4	7B	LDY	\$7B	
AC20	85	4B	STA	\$4B	; PROGRAMMAWIJZER TUSSENTIJD
					OPSLAAN IN \$4B/\$4C
AC22	84	4C	STY	\$4C	
AC24	A6	43	LDX	\$43	
AC26	A4	44	LDY	\$44	; INPUT-WIJZER
AC28	86	7A	STX	\$7A	; =PROGRAMMA-WIJZER
AC2A	84	7B	STY	\$7B	
AC2C	20	79	00 JSR	\$0079	; CHRGET: LAATSTE TEKEN HALEN
AC2F	D0	20	BNE	\$AC51	
AC31	24	11	BIT	\$11	; FLAG VOOR INVOER
AC33	50	0C	BVC	\$AC41	; GEEN GET?
AC35	20	24	E1 JSR	\$E124	; GETIN
AC38	8D	00	02 STA	\$0200	; WIJZER IN BUFFER SCHRIJVEN
AC3B	A2	FF	LDX	#\$FF	
AC3D	A0	01	LDY	#\$01	
AC3F	D0	0C	BNE	\$AC4D	
AC41	30	75	BMI	\$ACB8	
AC43	A5	13	LDA	\$13	; INVOERAPPARAAT
AC45	D0	03	BNE	\$AC4A	; NIET TOETSENBOARD?
AC47	20	45	AB JSR	\$AB45	; VRAAGTEKEN UITVOEREN
AC4A	20	F9	AB JSR	\$ABF9	; TWEEDE VRAAGTEKEN UITVOEREN
AC4D	86	7A	STX	\$7A	; PROGRAMMAWIJZER INSTELLEN
AC4F	84	7B	STY	\$7B	
AC51	20	73	00 JSR	\$0073	; CHRGET: VOLGEND TEKEN HALEN
AC54	24	0D	BIT	\$0D	; TYPE-FLAG
AC56	10	31	BPL	\$AC89	
AC58	24	11	BIT	\$11	; FLAG VOOR INVOER
AC5A	50	09	BVC	\$AC65	; GEEN GET?
AC5C	E8		INX		
AC5D	86	7A	STX	\$7A	
AC5F	A9	00	LDA	#\$00	
AC61	85	07	STA	\$07	
AC63	F0	0C	BEQ	\$AC71	
AC65	85	07	STA	\$07	
AC67	C9	22	CMP	#\$22	; AANHALINGSTEKEN
AC69	F0	07	BEO	\$AC72	
AC6B	A9	3A	LDA	#\$3A	; DUBBELE PUNT
AC6D	85	07	STA	\$07	
AC6F	A9	2C	LDA	#\$2C	; KOMMA
AC71	18		CLC		
AC72	85	08	STA	\$08	
AC74	A5	7A	LDA	\$7A	

AC76	A4	7B	LDY	\$7B	
AC78	69	00	ADC	#\$00	
AC7A	90	01	BCC	\$AC7D	
AC7C	C8		INY		
AC7D	20	8D	B4	JSR	\$B48D ; STRING OVERNEMEN
AC80	20	E2	B7	JSR	\$B7E2 ; PROGRAMMAWIJZER ACHTER STRING ZETTEN
AC83	20	DA	A9	JSR	\$A9DA ; STRING AAN VARIABLE TOEWIJZEN
AC86	4C	91	AC	JMP	\$AC91 ; VERDER GAAN
AC89	20	F3	BC	JSR	\$BCF3 ; CIJFERSTRING IN FAC HALEN
AC8C	A5	0E		LDA	\$0E ; FLAG VOOR INTEGER/REEEL
AC8E	20	C2	A9	JSR	\$A9C2 ; FAC AAN NUMERIEKE VARIABLE TOEWIJZEN
AC91	20	79	00	JSR	\$0079 ; CHRGOT: LAATSTE TEKEN HALEN
AC94	F0	07		BEQ	\$AC9D ; EINDE?
AC96	C9	2C		CMP	#\$2C ; KOMMA
AC98	F0	03		BEQ	\$AC9D
AC9A	4C	4D	AB	JMP	\$AB4D ; VOOR FOUTENBEHANDELING
AC9D	A5	7A		LDA	\$7A
AC9F	A4	7B		LDY	\$7B ; PROGRAMMAWIJZER
ACA1	85	43		STA	\$43
ACA3	84	44		STY	\$44 ; =DATAWIJZER
ACA5	A5	4B		LDA	\$4B
ACA7	A4	4C		LDY	\$4C ; PROGRAMMAWIJZER
ACA9	85	7A		STA	\$7A ; WEER TERUGHALEN
ACAB	84	7B		STY	\$7B
ACAD	20	79	00	JSR	\$0079 ; CHRGOT: LAATSTE TEKEN HALEN
ACB0	F0	2D		BEQ	\$ACDF
ACB2	20	FD	AE	JSR	\$AEFD ; CKCOM: KOMMA?
ACB5	4C	15	AC	JMP	\$AC15 ; VERDER
ACB8	20	06	A9	JSR	\$A906 ; VOLGENDE INSTRUCTIE ZOEKEN
ACBB	C8			INY	
ACBC	AA			TAX	
ACBD	D0	12		BNE	\$ACD1 ; EINDE VAN REGEL?
ACBF	A2	0D		LDX	#\$0D ; NEE
ACC1	C8			INY	
ACC2	B1	7A		LDA	(\$7A),Y
ACC4	F0	6C		BEQ	\$AD32 ; EIND VAN PROGRAMMA?
OUT OF DATA X=0					
ACC6	C8			INY	
ACC7	B1	7A		LDA	(\$7A),Y
ACC9	85	3F		STA	\$3F
ACCB	C8			INY	
ACCC	B1	7A		LDA	(\$7A),Y
ACCE	C8			INY	
ACCF	85	40		STA	\$40
ACD1	20	FB	A8	JSR	\$A8FB ; PROGRAMMAWIJZER OP VOLGENDE
					INSTRUCTIE
ACD4	20	79	00	JSR	\$0079 ; CHRGOT: LAATSTE TEKEN HALEN
ACD7	AA			TAX	
ACD8	E0	83		CPX	#\$83 ; DATA

```

ACDA DO DC      BNE $ACB8      ; NEE, VERDER ZOEKEN
ACDC 4C 51 AC    JMP $AC51      ; DATA LEZEN
ACDF A5 43      LDA $43
ACE1 A4 44      LDY $44        ; INPUTWIJZER
ACE3 A6 11      LDX $11        ; VLAG VOOR INVOER
ACE5 10 03      BPL $ACEA      ; GEEN DATA?
ACE7 4C 27 A8    JMP $A827      ; DATAWIJZER INSTELLEN
ACEA A0 00      LDY #$00
ACEC B1 43      LDA ($43),Y
ACEE F0 0B      BEQ $ACFB
ACFO A5 13      LDA $13
ACF2 D0 07      BNE $ACFB
ACF4 A9 FC      LDA #$FC
ACF6 A0 AC      LDY #$AC      ; WIJZER OP ?EXTRA IGNORED
ACF8 4C 1E AB    JMP $AB1E      ; STRING UITVOEREN
ACFB 60          RTS
ACFC 3F 45 58 54 52 41 20 49; ?EXTRA IGNORED

```

```

AD04 47 4E 4F 52 45 44 0D 00
AD0C 3F 52 45 44 4F 20 46 52; ?REDO FROM START
AD14 4F 20 53 54 41 52 54 0D
AD1C 00

```

```

***** BASIC-INSTRUCTIE NEXT
AD1D DO 04      BNE $AD24      ; VOLGT VARIABLENAAM?
AD20 A0 00      LDY #$00
AD22 F0 03      BEQ $AD27
AD24 20 8B B0    JSR $B08B      ; ZOEKT VARIABLE
AD27 85 49      STA $49
AD29 84 4A      STY $4A        ; ADRES VAN VARIABLE
AD2B 20 8A A3    JSR $A38A      ; ZOEKT FOR-NEXT-LUS IN STACK
AD2E F0 05      BEQ $AD35      ; GEVONDEN
AD30 A2 0A      LDX #$0A      ; NUMMER VOOR NEXT WITHOUT FOR
AD32 4C 37 A4    JMP $A437      ; FOUTMELDING UITVOEREN
AD35 9A          TXS
AD36 8A          TXA
AD37 18          CLC
AD38 69 04      ADC #$04
AD3A 48          PHA
AD3B 69 06      ADC #$06
AD3D 85 24      STA $24
AD3F 68          PLA
AD40 A0 01      LDY #$01
AD42 20 A2 BB    JSR $BBA2      ; VARIABLE VAN STACK NAAR FAC
AD45 BA          TSX
AD46 BD 09 01    LDA $0109,X
AD49 85 66      STA $66
AD4B A5 49      LDA $49        ; ADRES VAN VARIABLE
AD4D A4 4A      LDY $4A

```

```

AD4F 20 67 B8 JSR $B867 ; TELT STEPWAARDE BIJ FAC OP
AD52 20 DO BB JSR $BBD0 ; FAC NAAR VARIABLELE BRENGEN
AD55 A0 01 LDY #$01
AD57 20 5D BC JSR $BC5D ; FAC VERGELIJKEN MET
; EINDWAARDE VAN LUS
AD5A BA TSX
AD5B 38 SEC
AD5C FD 09 01 SBC $0109,X
AD5F FO 17 BEQ $AD78
AD61 BD OF 01 LDA $010F,X
AD64 85 39 STA $39
AD66 BD 10 01 LDA $0110,X ; REGELNUMMER HALEN
AD69 85 3A STA $3A
AD6B BD 12 01 LDA $0112,X
AD6E 85 7A STA $7A
AD70 BD 11 01 LDA $0111,X ; PROGRAMMAWIJZER HALEN
AD73 85 7B STA $7B
AD75 4C AE A7 JMP $A7AE ; NAAR INTERPRETERLUS
AD78 8A TXA
AD79 69 11 ADC #$11
AD7B AA TAX
AD7C 9A TXS
AD7D 20 79 00 JSR $0079 ; CHRGET: LAATSTE TEKEN HALEN
AD80 C9 2C CMP #$2C ; KOMMA
AD82 D0 F1 BNE $AD75 ; NEE, DAN KLAAR
AD84 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
AD87 20 24 AD JSR $AD24 ; VOLGENDE NEXT-VARIABLELE
***** FRMNUM: TERM HALEN,
; NUMERIEK?
AD8A 20 9E AD JSR $AD9E ; FRMEVL: TERM HALEN
***** NUMERIEK?
AD8D 18 CLC
AD8E 24 .BYTE $24
***** STRING?
AD8F 38 SEC
AD90 24 OD BIT $OD ; TYPE-FLAG ZETTEN
AD92 30 03 BMI $AD97
AD94 B0 03 BCS $AD99
AD96 60 RTS
AD97 B0 FD BCS $AD96
AD99 A2 16 LDX #$16 ; NUMMER VOOR TYPE MISMATCH
AD9B 4C 37 A4 JMP $A437 ; FOUTMELDING UITVOEREN
***** FRMEVL: INTERPRETEREN VAN
; WILLEKEURIGE TERM
AD9E A6 7A LDX $7A
ADAO DO 02 BNE $ADA4 ; PROGRAMMAWIJZER MIN 1
ADA2 C6 7B DEC $7B
ADA4 C6 7A DEC $7A
ADA6 A2 00 LDX #$00

```

ADA8	24		.BYTE \$24	
ADA9	48		PHA	
ADAA	8A		TXA	
ADAB	48		PHA	
ADAC	A9	01	LDA #\$01	; TWEE BYTES
ADAE	20	FB A3	JSR \$A3FB	; GENOEG PLAATS IN STACK?
ADB1	20	83 AE	JSR \$AE83	; VOLGEND ELEMENT HALEN
ADB4	A9	00	LDA #\$00	
ADB6	85	4D	STA \$4D	; MASKER VOOR
				VERGELIJKINGSOPERATOR
ADB8	20	79 00	JSR \$0079	; CHRGET: LAATSTE TEKEN HALEN
ADBB	38		SEC	
ADBC	E9	B1	SBC #\$B1	
ADBE	90	17	BCC \$ADD7	
ADCO	C9	03	CMP #\$03	
ADC2	B0	13	BCS \$ADD7	
ADC4	C9	01	CMP #\$01	
ADC6	2A		ROL A	; MASKER VOOR <, =, >
ADC7	49	01	EOR #\$01	
ADC9	45	4D	EOR \$4D	
ADCB	C5	4D	CMP \$4D	
ADCD	90	61	BCC \$AE30	
ADCF	85	4D	STA \$4D	
ADD1	20	73 00	JSR \$0073	; CHRGET: VOLGEND TEKEN HALEN
ADD4	4C	BB AD	JMP \$ADBB	; TERUG
ADD7	A6	4D	LDX \$4D	
ADD9	D0	2C	BNE \$AE07	
ADDB	B0	7B	BCS \$AE58	
ADDD	69	07	ADC #\$07	
ADDF	90	77	BCC \$AE58	
ADE1	65	0D	ADC \$0D	
ADE3	D0	03	BNE \$ADE8	; NUMERIEK?
ADE5	4C	3D B6	JMP \$B63D	; VERBINDING VAN STRINGS
ADE8	69	FF	ADC #\$FF	; CODE-\$AA
ADEA	85	22	STA \$22	
ADEC	0A		ASL A	
ADED	65	22	ADC \$22	; MAAL 3
ADEF	A8		TAY	
ADFO	68		PLA	
ADF1	D9	80 A0	CMP \$A080,Y	; MET HIERACHIEFLAG VERGELIJKEN
ADF4	B0	67	BCS \$AE5D	
ADF6	20	8D AD	JSR \$AD8D	; NUMERIEK?
ADF9	48		PHA	
ADFA	20	20 AE	JSR \$AE20	; OPERATORADRES EN OPERANDEN
OP STACK				
ADFD	68		PLA	
ADFE	A4	4B	LDY \$4B	

```

.A
AE00 10 17    BPL $AE19
AE02 AA       TAX
AE03 FO 56    BEQ $AE5B
AE05 DO 5F    BNE $AE66
AE07 46 OD    LSR $OD      ; STRINGFLAG WISSEN
AE09 8A       TXA
AE0A 2A       ROL A
AE0B A6 7A    LDX $7A
AE0D DO 02    BNE $AE11    ; PROGRAMMAWIJZER 1 TERUG
AE0F C6 7B    DEC $7B
AE11 C6 7A    DEC $7A
AE13 AO 1B    LDY #$1B     ; OFFSET VAN HIERARCHIEFLAG
AE15 85 4D    STA $4D     ; FLAG ZETTEN
AE17 DO D7    BNE $ADFO
AE19 D9 80 AO CMP $A080,Y  ; MET HIERARCHIEFLAG VERGELIJKEN
AE1C B0 48    BCS $AE66
AE1E 90 D9    BCC $ADF9
AE20 B9 82 AO LDA $A082,Y
AE23 48       PHA          ; ADRES VAN OPERATIE OP STACK
AE24 B9 81 AO LDA $A081,Y
AE27 48       PHA
AE28 20 33 AE JSR $AE33    ; OPERANDEN OP STACK
AE2B A5 4D    LDA $4D
AE2D 4C A9 AD JMP $ADA9    ; NAAR BEGIN VAN LUS
AE30 4C 08 AF JMP $AF08    ; LEVERT SYNTAX ERROR OP
AE33 A5 66    LDA $66      ; VOORTEKEN
AE35 BE 80 AO LDX $A080,Y  ; HIERARCHIEFLAG
AE38 A8       TAY
AE39 68       PLA
AE3A 85 22    STA $22
AE3C E6 22    INC $22     ; SPRONGADRES ONTHOUDEN
AE3E 68       PLA
AE3F 85 23    STA $23
AE41 98       TYA          ; VOORTEKEN
AE42 48       PHA
AE43 20 1B BC JSR $BC1B    ; FAC AFRONDEN
AE46 A5 65    LDA $65
AE48 48       PHA
AE49 A5 64    LDA $64
AE4B 48       PHA
AE4C A5 63    LDA $63
AE4E 48       PHA          ; FAC OP STACK
AE4F A5 62    LDA $62
AE51 48       PHA
AE52 A5 61    LDA $61
AE54 48       PHA
AE55 6C 22 00 JMP ($0022)  ; SPRONG NAAR OPERATIE
AE58 AO FF    LDY #$FF

```

```

AE5A 68      PLA
AE5B FO 23   BEQ $AE80
AE5D C9 64   CMP #$64
AE5F FO 03   BEQ $AE64
AE61 20 8D AD JSR $AD8D      ; NUMERIEK?
AE64 84 4B   STY $4B
AE66 68      PLA
AE67 4A      LSR A
AE68 85 12   STA $12
AE6A 68      PLA
AE6B 85 69   STA $69
AE6D 68      PLA
AE6E 85 6A   STA $6A
AE70 68      PLA      ; ARG VAN STACK HALEN
AE71 85 6B   STA $6B
AE73 68      PLA
AE74 85 6C   STA $6C
AE76 68      PLA
AE77 85 6D   STA $6D
AE79 68      PLA
AE7A 85 6E   STA $6E
AE7C 45 66   EOR $66
AE7E 85 6F   STA $6F
AE80 A5 61   LDA $61
AE82 60      RTS
***** VOLGEND ELEMENT VAN TERM
;
AE83 6C 0A 03 JMP ($030A) ; JMP $AE86
AE86 A9 00   LDA #$00
AE88 85 0D   STA $0D      ; TYPE-FLAG OP NUMERIEK
AE8A 20 73 00 JSR $0073    ; CHRGET: VOLGEND TEKEN HALEN
AE8D B0 03   BCS $AE92    ; CIJFER?
AE8F 4C F3 BC JMP $BCF3    ; VARIABELE NAAR FAC HALEN
AE92 20 13 B1 JSR $B113    ; LETTER?
AE95 90 03   BCC $AE9A    ; NEE
AE97 4C 28 AF JMP $AF28    ; VARIABELE HALEN
AE9A C9 FF   CMP #$FF     ; BASIC-CODE VOOR PI?
AE9C D0 0F   BNE $AEAD    ;
AE9E A9 A8   LDA #$A8     ; WIJZER OP CONSTATE PI
AEA0 A0 AE   LDY #$AE
AEA2 20 A2 BB JSR $BBA2    ; CONSTATE IN FAC HALEN
AEA5 4C 73 00 JMP $0073    ; CHRGET: VOLGEND TEKEN HALEN
*****
AEA8 82 49 0F DA A1      ; CONSTATE PI (3.14159265)
*****
AEAD C9 2E   CMP #$2E     ; PUNT
AEAF FO DE   BEQ $AE8F
AEB1 C9 AB   CMP #$AB     ; MIN
AEB3 FO 58   BEQ $AF0D    ; NAAR VOORTEKENWISSELING

```



```

AEB5 C9 AA    CMP #$AA      ; PLUS
AEB7 FO D1    BEQ $AE8A
AEB9 C9 22    CMP #$22      ; AANHALINGSTEKEN
AEBB DO OF    BNE $AECC
AEBD A5 7A    LDA $7A
AEBF A4 7B    LDY $7B      ; PROGRAMMAWIJZER HALEN
AEC1 69 00    ADC #$00
AEC3 90 01    BCC $AEC6
AEC5 C8       INY
AEC6 20 87 B4 JSR $B487     ; STRING OVERDRAGEN
AEC9 4C E2 B7 JMP $B7E2     ; PROGRAMMAWIJZER OP EINDE
;                               VAN STRING PLUS 1
AECC C9 A8    CMP #$A8      ; NOT-CODE
AECE DO 13    BNE $AEE3
AEDO AO 18    LDY #$18      ; OFFSET VAN HIERARCHIEFLAG IN TABEL
AED2 DO 3B    BNE $AFOF
***** BASIC-INSTRUCTIE NOT
AED4 20 BF B1 JSR $B1BF     ; FAC IN INTEGER VERANDEREN
AED7 A5 65    LDA $65
AED9 49 FF    EOR #$FF      ; ALLE BITS OMDRAAIEN
AEDB A8       TAY
AEDC A5 64    LDA $64
AEDE 49 FF    EOR #$FF
AEE0 4C 91 B3 JMP $B391     ; WEER IN REEEL VERANDEREN
*****
AEE3 C9 A5    CMP #$A5      ; FN-CODE
AEE5 DO 03    BNE $AEEA
AEE7 4C F4 B3 JMP $B3F4     ; FN UITVOEREN
*****
AEEA C9 B4    CMP #$B4      ; SGN-CODE
AEEC 90 03    BCC $AEF1     ; < (GEEN STRINGFUNCTIE)?
AEEE 4C A7 AF JMP $AFA7     ; HAALT STRING EN EERSTE PARAMETER
***** HAALT TERM TUSSEN HAAKJES
AEF1 20 FA AE JSR $AEFA     ; HAAKJE OPEN?
AEF4 20 9E AD JSR $AD9E     ; FRMEVL: HAALT TERM
***** TEKEN IN BASIC-TEKST?
AEF7 A9 29    LDA #$29      ; HAAKJE GESLOTEN
AEF9 2C       .BYTE $2C
AEFA A9 28    LDA #$28      ; HAAKJE OPEN
AEFC 2C       .BYTE $2C
AEFD A9 2C    LDA #$2C      ; KOMMA
AEFF AO 00    LDY #$00
AF01 D1 7A    CMP ($7A),Y   ; MET ACTUEEL TEKEN VERGELIJKEN
AF03 DO 03    BNE $AF08     ; NIET GELIJK?
AF05 4C 73 00 JMP $0073     ; CHRGET: VOLGEND TEKEN HALEN
AF08 A2 0B    LDX #$0B      ; NUMMER VOOR SYNTAX ERROR
AF0A 4C 37 A4 JMP $A437     ; FOUTMELDING UITVOEREN
AF0D AO 15    LDY #$15      ; OFFSET: HIERARCHIECODE VOOR
;                               WISSELING VAN VOORTEKEN

```

```

AFOF 68      PLA
AF10 68      PLA
AF11 4C FA AD JMP $ADFA      ; TER INTERPRETATIE
***** VARIABELE BINNEN BASIC?
AF14 38      SEC
AF15 A5 64    LDA $64
AF17 E9 00    SBC #$00      ; LIGT DESCRIPTOR ($64/$65)
AF19 A5 65    LDA $65      ; TUSSEN $A000 EN $E32A?
AF1B E9 A0    SBC #$A0
AF1D 90 08    BCC $AF27     ; JA? DAN C=1
AF1F A9 A2    LDA #$A2
AF21 E5 64    SBC $64
AF23 A9 E3    LDA #$E3
AF25 E5 65    SBC $65
AF27 60      RTS
***** VARIABELE HALEN
AF28 20 8B B0 JSR $B08B     ; VARIABELE ZOEKEN
AF2B 85 64    STA $64
AF2D 84 65    STY $65      ; WIJST OP VARIABELE OF
; STRINGDESCRIPTOR
AF2F A6 45    LDX $45
AF31 A4 46    LDY $46      ; VARIABELENAAM
AF33 A5 0D    LDA $0D      ; TYPE-FLAG
AF35 F0 26    BEQ $AF5D     ; NUMERIEK
AF37 A9 00    LDA #$00
AF39 85 70    STA $70
AF3B 20 14 AF JSR $AF14     ; DESCRIPTOR IN INTERPRETER?
AF3E 90 1C    BCC $AF5C     ; NEE
AF40 E0 54    CPX #$54      ; T
AF42 D0 18    BNE $AF5C
AF44 C0 C9    CPY #$C9      ; I$
AF46 D0 14    BNE $AF5C
AF48 20 84 AF JSR $AF84     ; TIJD HALEN
AF4B 84 5E    STY $5E
AF4D 88      DEY
AF4E 84 71    STY $71
AF50 A0 06    LDY #$06      ; LENGTE 6 VOOR TI$
AF52 84 5D    STY $5D
AF54 A0 24    LDY #$24
AF56 20 68 BE JSR $BE68     ;
AF59 4C 6F B4 JMP $B46F
AF5C 60      RTS
AF5D 24 0E    BIT $0E      ; FLAG VOOR INTEGER/REEEL
AF5F 10 0D    BPL $AF6E     ; REEEL?
***** INTEGERE VARIABELE HALEN
AF61 A0 00    LDY #$00
AF63 B1 64    LDA ($64),Y   ; INTEGER GETAL HALEN
AF65 AA      TAX
AF66 C8      INY

```

```

AF67 B1 64 LDA ($64),Y
AF69 A8 TAY
AF6A 8A TXA
AF6B 4C 91 B3 JMP $B391 ; VERANDEREN IN REEEL
***** REELE VARIABLE HALEN
AF6E 20 14 AF JSR $AF14 ; DESCRIPTOR IN INTERPRETER?
AF71 90 2D BCC $AFA0 ; NEE
AF73 E0 54 CPX #$54 ; T
AF75 D0 1B BNE $AF92
AF77 C0 49 CPY #$49 ; I
AF79 D0 25 BNE $AFA0
AF7B 20 84 AF JSR $AF84 ; TIME IN FAC HALEN
AF7E 98 TYA
AF7F A2 A0 LDX #$A0
AF81 4C 4F BC JMP $BC4F
***** TIJD HALEN
AF84 20 DE FF JSR $FFDE ; TIME HALEN
AF87 86 64 STX $64
AF89 84 63 STY $63
AF8B 85 65 STA $65
AF8D A0 00 LDY #$00 ; EN IN REEEL VERANDEREN
AF8F 84 62 STY $62
AF91 60 RTS
AF92 E0 53 CPX #$53 ; S
AF94 D0 0A BNE $AFA0
AF96 C0 54 CPY #$54 ; T
AF98 D0 06 BNE $AFA0
AF9A 20 B7 FF JSR $FFB7 ; STATUS HALEN
AF9D 4C 3C BC JMP $BC3C ; BYTE IN ACCU VERANDEREN
; IN REEEL
***** REELE VARIABLE HALEN
AFA0 A5 64 LDA $64
AFA2 A4 65 LDY $65 ; ADRES VAN VARIABLE
AFA4 4C A2 BB JMP $BBA2 ; VARIABLE NAAR FAC HALEN
***** FUNCTIEBEREKENING
AFA7 0A ASL A ; FUNCTIECODE MAAL 2
AFA8 48 PHA
AFA9 AA TAX
AFAA 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN
AFAD E0 8F CPX #$8F
AFAF 90 20 BCC $AFD1 ; NUMERIEKE FUNCTIE?
***** STRINGFUNCTIE, STRING
; EN EERSTE PARAMETER
AFB1 20 FA AE JSR $AEFA ; HAAKJE OPEN?
AFB4 20 9E AD JSR $AD9E ; FRMEVL: HAAL WILLEKEURIGE TERM
AFB7 20 FD AE JSR $AEFD ; KOMMA?
AFBA 20 8F AD JSR $AD8F ; STRING?
AFBD 68 PLA ; FUNCTIETOKEN VOOR
; LEFT$,RIGHT$,MID$

```

AFBE	AA	TAX	
AFBF	A5 65	LDA \$65	
AFC1	48	PHA	; ADRES VAN STRINGDESCRIPTOR
AFC2	A5 64	LDA \$64	
AFC4	48	PHA	
AFC5	8A	TXA	
AFC6	48	PHA	; TOKEN OP STACK
AFC7	20 9E B7	JSR \$B79E	; HAALT BYTE-WAARDE (TWEEDE PARAMETER VAN FUNCTIE)
AFCA	68	PLA	; TOKEN TERUGHALEN
AFCB	A8	TAY	
AFCC	8A	TXA	
AFCD	48	PHA	; BYTE-WAARDE OP STACK
AFCE	4C D6 AF	JMP \$AFD6	; ROUTINE UITVOEREN
***** NUMERIEKE FUNCTIE INTERPRETEREN			
AFD1	20 F1 AE	JSR \$AEF1	; HAALT TERM TUSSEN HAAKJES (FUNCTIEARGUMENT)
AFD4	68	PLA	; BASIC-CODE VOOR FUNCTIE
AFD5	A8	TAY	
AFD6	B9 EA 9F	LDA \$9FEA,Y	
AFD9	85 55	STA \$55	; ZET VECTOR VOOR
AFDB	B9 EB 9F	LDA \$9FEB,Y	; FUNCTIEBEREKENING
AFDE	85 56	STA \$56	
AFE0	20 54 00	JSR \$0054	; FUNCTIE UITVOEREN
AFE3	4C 8D AD	JMP \$AD8D	; NUMERIEK?
***** BASIC-INSTRUCTIE OR			
AFE6	A0 FF	LDY #\$FF	; FLAG VOOR OR
AFE8	2C	.BYTE \$2C	
***** BASIC-INSTRUCTIE AND			
AFE9	A0 00	LDY #\$00	; FLAG VOOR AND
AFEB	84 0B	STY \$0B	; FLAG ZETTEN
AFED	20 BF B1	JSR \$B1BF	; FAC VERANDEREN IN INTEGER
AFF0	A5 64	LDA \$64	
AFF2	45 0B	EOR \$0B	
AFF4	85 07	STA \$07	
AFF6	A5 65	LDA \$65	; MET FLAG VERBINDEN EN NAAR \$7/\$8
AFF8	45 0B	EOR \$0B	
AFFA	85 08	STA \$08	
AFFC	20 FC BB	JSR \$BBFC	; ARG NAAR FAC
AFFF	20 BF B1	JSR \$B1BF	; FAC NAAR INTEGER
B002	A5 65	LDA \$65	
B004	45 0B	EOR \$08	
B006	25 08	AND \$08	; LOGISCHE VERBINDING
B008	45 0B	EOR \$0B	
B00A	A8	TAY	
B00B	A5 64	LDA \$64	
B00D	45 0B	EOR \$0B	

```

BOOF 25 07    AND $07
BO11 45 0B    EOR $0B
BO13 4C 91 B3 JMP $B391    ; WEER IN REEL VERANDEREN
***** VERGELIJKEN
BO16 20 90 AD JSR $AD90    ; HETZELFDE TYPE VARIABLE?
BO19 B0 13    BCS $B02E    ; STRING, DAN VERDER
BO1B A5 6E    LDA $6E
BO1D 09 7F    ORA #$7F    ; ARG IN GEHEUGENFORMAAT
BO1F 25 6A    AND $6A
BO21 85 6A    STA $6A
BO23 A9 69    LDA #$69    ; ADRES VAN ARG
BO25 A0 00    LDY #$00
BO27 20 5B BC JSR $BC5B    ; ARG MET FAC VERGELIJKEN
BO2A AA      TAX
BO2B 4C 61 B0 JMP $B061    ; RESULTAAT IN FAC HALEN
***** VERGELIJKEN VAN STRINGS
BO2E A9 00    LDA #$00
BO30 85 0D    STA $0D    ; STRINGFLAG WISSEN
BO32 C6 4D    DEC $4D
BO34 20 A6 B6 JSR $B6A6    ; FRESTR
BO37 85 61    STA $61    ; STRINGLENGTE
BO39 86 62    STX $62
BO3B 84 63    STY $63    ; STRINGADRES
BO3D D5 6C    LDA $6C
BO3F A4 6D    LDY $6D    ; WIJZER OP TWEEDE STRING
BO41 20 AA B6 JSR $B6AA    ; FRESTR
BO44 86 6C    STX $6C
BO46 84 6D    STY $6D    ; STRINGADRES VAN TWEEDE STRING
BO48 AA      TAX
BO49 38      SEC
BO4A E5 61    SBC $61    ; LENGTES VERGELIJKEN
BO4C F0 08    BEQ $B056    ; GELIJK?
BO4E A9 01    LDA #$01
BO50 90 04    BCC $B056    ; TWEEDE STRING KORTER
BO52 A6 61    LDX $61    ; LENGTE VAN EERSTE STRING
BO54 A9 FF    LDA #$FF
BO56 85 66    STA $66
BO58 A0 FF    LDY #$FF
BO5A E8      INX
BO5B C8      INY
BO5C CA      DEX
BO5D D0 07    BNE $B066
BO5F A6 66    LDX $66
BO61 30 0F    BMI $B072
BO63 18      CLC
BO64 90 0C    BCC $B072
BO66 B1 6C    LDA ($6C),Y ; STRINGS VERGELIJKEN
BO68 D1 62    CMP ($62),Y ; TEKEN VOOR TEKEN
BO6A F0 EF    BEQ $B05B

```

```

B06C  A2 FF      LDX  #$FF
B06E  B0 02      BCS  $B072
B070  A2 01      LDX  #$01
B072  E8         INX
B073  8A         TXA
B074  2A         ROL
B075  25 12      AND  $12
B077  F0 02      BEQ  $B07B
B079  A9 FF      LDA  #$FF
B07B  4C 3C BC   JMP  $BC3C      ; RESULTAAT NAAR FAC HALEN
B07E  20 FD AE   JSR  $AEFD      ; CHCOM: KOMMA?
***** BASIC-INSTRUCTIE DIM
B081  AA         TAX
B082  20 90 B0   JSR  $B090      ; VARIABLE DIMENSIONEREN
B085  20 79 00   JSR  $0079      ; CHRGOT: LAATSTE TEKEN HALEN
B088  D0 F4      BNE  $B07E      ; NIET EINDE, DAN NAAR
;                               VOLGENDE VARIABLE
B08A  60         RTS
***** VARIABLE HALEN
B08B  A2 00      LDX  #$00      ; FLAG VOOR NIET DIMENSIONEREN
B08D  20 79 00   JSR  $0079      ; CHRGOT: LAATSTE TEKEN
B090  86 0C      STX  $0C      ; DIM-FLAG ZETTEN
B092  85 45      STA  $45      ; NAAM VAN VARIABLE
B094  20 79 00   JSR  $0079      ; CHRGOT: LAATSTE TEKEN
B097  20 13 B1   JSR  $B113      ; LETTER?
B09A  B0 03      BCS  $B09F      ; JA
B09C  4C 08 AF   JMP  $AF08      ; SYNTAX ERROR
B09F  A2 00      LDX  #$00
BOA1  86 0D      STX  $0D      ; STRINGFLAG WISSEN
BOA3  86 0E      STX  $0E      ; FLAG VOOR INTEGERE GETALLEN
;                               WISSEN
BOA5  20 73 00   JSR  $0073      ; CHRGET: VOLGEND TEKEN HALEN
BOA8  90 05      BCC  $BOAF      ; CIJFER?
BOAA  20 13 B1   JSR  $B113      ; LETTER?
BOAD  90 0B      BCC  $BOBA      ; NEE
BOAF  AA         TAX          ; TWEDE LETTER VAN NAAM
BOB0  20 73 00   JSR  $0073      ; CHRGET: VOLGEND TEKEN HALEN
BOB3  90 FB      BCC  $BOB0      ; CIJFER?
BOB5  20 13 B1   JSR  $B113      ; LETTER?
BOB8  B0 F6      BCS  $BOB0      ; JA, ANDERE TEKENS OVERSLAAN
BOBA  C9 24      CMP  #$24      ; $
BOBC  D0 06      BNE  $BOC4      ; NEE
BOBE  A9 FF      LDA  #$FF
BOC0  85 0D      STA  $0D      ; STRINGFLAG ZETTEN
BOC2  D0 10      BNE  $BOD4      ; SPRONG
BOC4  C9 25      CMP  #$25      ; %
BOC6  D0 13      BNE  $BODB      ; NEE
BOC8  A5 10      LDA  $10      ; INTEGER TOEGESTAAN?
BOCA  D0 D0      BNE  $B09C      ; NEE, SYNTAX ERROR

```

BOCC	A9 80	LDA #80	
BOCE	85 0E	STA \$0E	; FLAG VOOR INTEGERE GETALLEN ZETTEN
BODO	05 45	ORA \$45	; BIT 7 IN NAAM INSTELLEN
BOD2	85 45	STA \$45	
BOD4	8A	TXA	
BOD5	09 80	ORA #80	; TWEEDE LETTER VAN NAAM
BOD7	AA	TAX	
BOD8	20 73 00	JSR \$0073	; CHRGET: VOLGEND TEKEN HALEN
BODB	86 46	STX \$46	; TWEEDE LETTER OPSLAAN
BODD	38	SEC	
BODE	05 10	ORA \$10	
BOE0	E9 28	SBC #80	; HAAKJE OPEN
BOE2	D0 03	BNE \$BOE7	; GEEN HAAKJE OPEN?
BOE4	4C D1 B1	JMP \$B1D1	; GEDIMENSIONEERDE VARIABELE HALEN
BOE7	A0 00	LDY #800	
BOE9	84 10	STY \$10	; WIJZER OP BEGIN VAN VARIABELE
BOEB	A5 2D	LDA \$2D	
BOED	A6 2E	LDX \$2E	
BOEF	86 60	STX \$60	
BOF1	85 5F	STA \$5F	; VOOR ZOEKEN MARKEREN
BOF3	E4 30	CPX \$30	
BOF5	D0 04	BNE \$BOFB	
BOF7	C5 2F	CMP \$2F	; EIND VAN VARIABELE BEREIKT?
BOF9	F0 22	BEQ \$B11D	; JA, NIET GEVONDEN, OPNIEUW
BOFB	A5 45	LDA \$45	; EERSTE LETTER VAN NAAM
BOFD	D1 5F	CMP (\$5F),Y	; MET VARIABELENTABEL
			VERGELIJKEN
BOFF	D0 08	BNE \$B109	; NEE, VERDER ZOEKEN
B101	A5 46	LDA \$46	; TWEEDE LETTER
B103	C8	INY	
B104	D1 5F	CMP (\$5F),Y	; VERGELIJKEN
B106	F0 7D	BEQ \$B185	
B108	88	DEY	
B109	18	CLC	
B10A	A5 5F	LDA \$5F	
B10C	69 07	ADC #807	; WIJZER MET 7 VERHOGEN (2+5 BYTE REELE VARIABELE)
B10E	90 E1	BCC \$BOF1	
B110	E8	INX	
B111	D0 DC	BNE \$BOEF	; VERDER ZOEKEN
*****			LETTER?
B113	C9 41	CMP #841	; A
B115	90 05	BCC \$B11C	
B117	E9 5B	SBC #85B	; Z+1
B119	38	SEC	; JA, DAN C=1
B11A	E9 A5	SBC #8A5	; NEE, DAN C=0
B11C	60	RTS	

***** VARIABLE OPVOEREN

```

B11D 68      PLA
B11E 48      PHA          ; OPROEPADRES CONTROLLEREN
B11F C9 2A   CMP #$2A     ; OPROEP VAN FRMEVL?
B121 D0 05   BNE $B128    ; NEE, DAN OPNIEUW OPVOEREN
B123 A9 13   LDA #$13     ; WIJZER OP CONSTATE O
B125 A0 BF   LDY #$BF
B127 60      RTS
B128 A5 45   LDA $45
B12A A4 46   LDY $46      ; VARIABELNAAM
B12C C9 54   CMP #$54     ; T
B12E D0 0B   BNE $B13B
B130 C0 C9   CPY #$C9     ; I$
B132 F0 EF   BEQ $B123    ; JA, TI$
B134 C0 49   CPY #$49     ; I
B136 D0 03   BNE $B13B    ; NEE
B138 4C 08 AF JMP $AF08    ; SYNTAX ERROR
B13B C9 53   CMP #$53     ; S
B13D D0 04   BNE $B143
B13F C0 54   CPY #$54     ; T
B141 F0 F5   BEQ $B138    ; ST, DAN SYNTAX ERROR
B143 A5 2F   LDA $2F
B145 A4 30   LDY $30      ; WIJZER OP ARRAYTABEL
B147 85 5F   STA $5F
B149 84 60   STY $60      ; ONTHOUDEN
B14B A5 31   LDA $31
B14D A4 32   LDY $32      ; WIJZER OP EINDE ARRAYTABEL
B14F 85 5A   STA $5A
B151 84 5B   STY $5B      ; ONTHOUDEN
B153 18      CLC
B154 69 07   ADC #$07     ; 7 PLAATSEN VERSCHUIVEN VOOR
B156 90 01   BCC $B159    ; NIEUWE VARIABLE
B158 C8      INY
B159 85 58   STA $58
B15B 84 59   STY $59      ; NIEUW BLOKEINDE
B15D 20 B8 A3 JSR $A3B8    ; BLOK VERSCHUIVEN
B160 A5 58   LDA $58
B162 A4 59   LDY $59
B164 C8      INY
B165 85 2F   STA $2F      ; WIJZER OP ARRAYTABEL OPNIEUW
;                               INSTELLEN
B167 84 30   STY $30
B169 A0 00   LDY #$00
B16B A5 45   LDA $45      ; EERSTE LETTER VAN NAAM
B16D 91 5F   STA ($5F),Y
B16F C8      INY
B170 A5 46   LDA $46      ; EN TWEEDE LETTER OPSLAAN
B172 91 5F   STA ($5F),Y
B174 A9 00   LDA #$00

```



```

B176 C8      INY
B177 91 5F    STA ($5F),Y
B179 C8      INY
B17A 91 5F    STA ($5F),Y
B17C C8      INY      ; 5 MAAL 0 VOOR VARIABELEWAARDE
B17D 91 5F    STA ($5F),Y
B17F C8      INY
B180 91 5F    STA ($5F),Y
B182 C8      INY
B183 91 5F    STA ($5F),Y
B185 A5 5F    LDA $5F
B187 18      CLC
B188 69 02    ADC #$02      ; TWEE VOOR NAAM OPTELLEN
B18A A4 60    LDY $60      ; WIJZER OP VARIABELE
B18C 90 01    BCC $B18F
B18E C8      INY
B18F 85 47    STA $47      ; NAAR $47/$48
B191 84 48    STY $48
B193 60      RTS
*****
; BEREKENT WIJZER OP EERSTE
; ELEMENT VAN ARRAY
B194 A5 0B    LDA $0B      ; AANTAL DIMENSIES
B196 0A      ASL          ; MAAL TWEE
B197 69 05    ADC #$05     ; PLUS VIJF
B199 65 5F    ADC $5F
B19B A4 60    LDY $60      ; BIJ $5F/$60 OPTELLEN
B19D 90 01    BCC $B1A0
B19F C8      INY
B1A0 85 58    STA $58      ; WIJZER VOOR RESULTAAT
B1A2 84 59    STY $59
B1A4 60      RTS
*****
B1A5 90 80 00 00 00      ; CONSTATE -32768
*****
; FAC IN INTEGER VERANDEREN
B1AA 20 BF B1 JSR $B1BF   ; FAC IN INTEGER VERANDEREN
B1AD A5 64    LDA $64     ; LOW-BYTE
B1AF A4 65    LDY $65     ; HIGH-BYTE
B1B1 60      RTS
*****
; TERM HALEN EN IN INTEGER
; VERANDEREN
B1B2 20 73 00 JSR $0073   ; CHRGET: VOLGEND TEKEN HALEN
B1B5 20 9E AD JSR $AD9E   ; FRMEVL: TERM INTERPRETEREN
B1B8 20 8D AD JSR $AD8D   ; NUMERIEK
B1BB A5 66    LDA $66     ; VOORTEKEN?
B1BD 30 0D    BMI $B1CC   ; NEGATIEF, DAN ILLEGAL QUANTITY
B1BF A5 61    LDA $61     ; EXPONENT
B1C1 C9 90    CMP #$90    ; WAARDE > 32768
B1C3 90 09    BCC $B1CE   ; NEE
B1C5 A9 A5    LDA #$A5

```

B1C7	A0 B1	LDY #\$B1	; WIJZER OP CONSTATE -32768
B1C9	20 5B BC	JSR \$BC5B	; FAC MET CONSTATE VERGELIJKEN
B1CC	D0 7A	BNE \$B248	; ONGELIJK, ILLEGAL QUANTITY
B1CE	4C 9B BC	JMP \$BC9B	; REEEL IN INTEGER VERANDEREN
***** GEDIMENSIONEERDE VARIABLE HELEN			
B1D1	A5 0C	LDA \$0C	; DIM FLAG
B1D3	05 0E	ORA \$0E	; FLAG VOOR INTEGERE GETALLEN
B1D5	48	PHA	
B1D6	A5 0D	LDA \$0D	; STRING FLAG
B1D8	48	PHA	
B1D9	A0 00	LDY #\$00	; AANTAL INDICES
B1DB	98	TYA	
B1DC	48	PHA	
B1DD	A5 46	LDA \$46	; 2e LETTER VAN VARIABLENAAM
B1DF	48	PHA	
B1E0	A5 45	LDA \$45	; 1e LETTER VAN VARIABLENAAM
B1E2	48	PHA	
B1E3	20 B2 B1	JSR \$B1B2	; INDEX HELEN, VERANDEREN IN INTEGER
B1E6	68	PLA	
B1E7	85 45	STA \$45	
B1E9	68	PLA	; VARIABLENAAM TERUGHALEN
B1EA	85 46	STA \$46	
B1EC	68	PLA	; AANTAL INDICES
B1ED	A8	TAY	
B1EE	BA	TSX	
B1EF	BD 02 01	LDA \$0102,X	
B1F2	48	PHA	; FLAGS VAN STACK HELEN
B1F3	BD 01 01	LDA \$0101,X	
B1F6	48	PHA	
B1F7	A5 64	LDA \$64	
B1F9	9D 02 01	STA \$0102,X	
B1FC	A5 65	LDA \$65	; INDEX LOW EN HIGH OP STACK
B1FE	9D 01 01	STA \$0101,X	
B201	C8	INY	; AANTAL INDICES VERHOGEN
B202	20 79 00	JSR \$0079	; CHRGOT: LAATSTE TEKEN HELEN
B205	C9 2C	CMP #\$2C	; KOMMA?
B207	F0 D2	BEQ \$B1DB	; JA, DAN VOLGENDE INDEX
B209	84 0B	STY \$0B	; AANTAL INDICES OPSLAAN
B20B	20 F7 AE	JSR \$AEF7	; AANHALINGSTEKEN GESLOTEN?
B20E	68	PLA	
B20F	85 0D	STA \$0D	
B211	68	PLA	; FLAGS TERUGHALEN
B212	85 0E	STA \$0E	
B214	29 7F	AND #\$7F	
B216	85 0C	STA \$0C	
B218	A6 2F	LDX \$2F	
B21A	A5 30	LDA \$30	; WIJZER OP ARRAYTABEL
B21C	86 5F	STX \$5F	

B21E	85 60	STA \$60	; WIJZER ONTHOUDEN
B220	C5 32	CMP \$32	
B222	D0 04	BNE \$B228	
B224	E4 31	CPX \$31	; MET EINDE VAN TABEL VERGELIJKEN
B226	FO 39	BEQ \$B261	; JA, NIET GEVONDEN, DAN OPVOEREN
B228	AO 00	LDY #\$00	
B22A	B1 5F	LDA (\$5F),Y	; NAAM UIT TABEL
B22C	C8	INY	
B22D	C5 45	CMP \$45	; MET GEZOCHTE NAAM VERGELIJKEN
B22F	D0 06	BNE \$B237	
B231	A5 46	LDA \$46	
B233	D1 5F	CMP (\$5F),Y	; 2e LETTER
B235	FO 16	BEQ \$B24D	; GEVONDEN
B237	C8	INY	
B238	B1 5F	LDA (\$5F),Y	
B23A	18	CLC	; VELDLENGTE OPTELLEN
B23B	65 5F	ADC \$5F	
B23D	AA	TAX	
B23E	C8	INY	
B23F	B1 5F	LDA (\$5F),Y	
B241	65 60	ADC \$60	
B243	90 D7	BCC \$B21C	; EN VERDER ZOEKEN
B245	A2 12	LDX #\$12	; NUMMER VOOR BAD SUBSCRIPT
B247	2C	.BYTE \$2C	
B248	A2 0E	LDY #\$0E	; NUMMER VOOR ILLEGAL QUANTITY
B24A	4C 37 A4	JMP \$A437	; FOUTMELDING UITVOEREN
B24D	A2 13	LDX #\$13	; NUMMER VOOR REDIM'D ARRAY
B24F	A5 0C	LDA \$0C	; DIM FLAG NUL?
B251	D0 F7	BNE \$B24A	; NEE, DAN FOUTMELDING
B253	20 94 B1	JSR \$B194	; ZET WIJZER OP 1e ARRAYELEMENT
B256	A5 0B	LDA \$0B	; AANTAL GEVONDEN DIMENSIES
B258	AO 04	LDY #\$04	
B25A	D1 5F	CMP (\$5F),Y	; ARRAYDIMENSIES VERGELIJKEN
B25C	D0 E7	BNE \$B245	; ONGELIJK, DAN BAD SUBSCRIPT
B25E	4C EA B2	JMP \$B2EA	; ZOEKT GEWENSTE ARRAYELEMENT
***** ARRAYVARIABELE OPVOEREN *****			
B261	20 94 B1	JSR \$B194	; LENGTE VAN ARRAY HEADER
B264	20 08 A4	JSR \$A408	; GENOEG RUIMTE IN GEHEUGEN?
B267	AO 00	LDY #\$00	
B269	84 72	STY \$72	
B26B	A2 05	LDX #\$05	; DEFAULTWAARDE VOOR
			VARIABELENLENGTE (REEEL)
B26D	A5 45	LDA \$45	; 1e LETTER VAN NAAM
B26F	91 5F	STA (\$5F),Y	; IN ARRAYTABEL
B271	10 01	BPL \$B274	; NIET INTEGER?
B273	CA	DEX	
B274	C8	INY	
B275	A5 46	LDA \$46	; 2e LETTER
B277	91 5F	STA (\$5F),Y	; IN TABEL SCHRIJVEN

B279	10 02	BPL \$B27D	; GEEN STRING OF INTEGER?
B27B	CA	DEX	
B27C	CA	DEX	
B27D	86 71	STX \$71	; DEFINITIEVE VARIABELELENGTE
			2, 3 OF 5
B27F	A5 0B	LDA \$0B	; AANTAL DIMENSIES
B281	C8	INY	
B282	C8	INY	
B283	C8	INY	
B284	91 5F	STA (\$5F),Y	; OPSLAAN
B286	A2 0B	LDX #\$0B	; 11, DEFAULTTWAARDE VOOR
			DIMENSIONERING
B288	A9 00	LDA #\$00	
B28A	24 0C	BIT \$0C	; OPROEP DOOR DIM-INSTRUCTIE?
B28C	50 08	BVC \$B296	; NEE
B28E	68	PLA	; DIMENSIE VAN STACK HALEN
B28F	18	CLC	
B290	69 01	ADC #\$01	; 1 OPTELLEN
B292	AA	TAX	
B293	68	PLA	
B294	69 00	ADC #\$00	
B296	C8	INY	
B297	91 5F	STA (\$5F),Y	; EN OPSLAAN
B299	C8	INY	
B29A	8A	TXA	
B29B	91 5F	STA (\$5F),Y	
B29D	20 4C B3	JSR \$B34C	; PLAATS VOOR ANDERE
			DIMENSIES BEREKENEN
B2A0	86 71	STX \$71	
B2A2	85 72	STA \$72	; WIJZER OP EIND VAN VARIABELE"
			ONTHOUDEN
B2A4	A4 22	LDY \$22	
B2A6	C6 0B	DEC \$0B	; MEER DIMENSIES?
B2A8	D0 DC	BNE \$B286	; JA
B2AA	65 59	ADC \$59	
B2AC	B0 5D	BCS \$B30B	; VELDLENGTE+STARTADRES
B2AE	85 59	STA \$59	
B2B0	A8	TAY	
B2B1	8A	TXA	
B2B2	65 58	ADC \$58	
B2B4	90 03	BCC \$B2B9	
B2B6	C8	INY	
B2B7	F0 52	BEQ \$B30B	
B2B9	20 08 A4	JSR \$A408	; GENOEG GEHEUGENRUIMTE?
B2BC	85 31	STA \$31	
B2BE	84 32	STY \$32	; WIJZER OP EINDE ARRAYTABEL
B2C0	A9 00	LDA #\$00	; ARRAY MET NULLEN VULLEN
B2C2	E6 72	INC \$72	
B2C4	A4 71	LDY \$71	

```

B2C6  FO 05    BEQ $B2CD
B2C8  88      DEY
B2C9  91 58    STA ($58),Y
B2CB  D0 FB    BNE $B2C8
B2CD  C6 59    DEC $59
B2CF  C6 72    DEC $72
B2D1  D0 F5    BNE $B2C8
B2D3  E6 59    INC $59
B2D5  38      SEC
B2D6  A5 31    LDA $31
B2D8  E5 5F    SBC $5F
B2DA  A0 02    LDY #$02
B2DC  91 5F    STA ($5F),Y ; ARRAYLENGTE LOW
B2DE  A5 32    LDA $32
B2E0  C8      INY
B2E1  E5 60    SBC $60
B2E3  91 5F    STA ($5F),Y ; ARRAYLENGTE HIGH
B2E5  A5 0C    LDA $0C ; OPROEP VAN DIM-INSTRUCTIE
B2E7  D0 62    BNE $B34B ; JA, RTS
***** ARRAYELEMENT ZOEKEN
B2E9  C8      INY
B2EA  B1 5F    LDA ($5F),Y ; AANTAL DIMENSIES
B2EC  85 0B    STA $0B ; OPSLAAN
B2EE  A9 00    LDA #$00
B2F0  85 71    STA $71
B2F2  85 72    STA $72
B2F4  C8      INY
B2F5  68      PLA
B2F6  AA      TAX
B2F7  85 64    STA $64 ; INDEX VAN STACK HALEN
B2F9  68      PLA
B2FA  85 65    STA $65
B2FC  D1 5F    CMP ($5F),Y ; MET WAARDE IN ARRAY
; VERGELIJKEN
B2FE  90 0E    BCC $B30E ; KLEINER?

.
B300  D0 06    BNE $B308 ; GROTER, DAN BAD SUBSCRIPT
B302  C8      INY
B303  8A      TXA
B304  D1 5F    CMP ($5F),Y ; GELIJK, DAN LSB VERGELIJKEN
B306  90 07    BCC $B30F ; KLEINER, DAN VERDER
B308  4C 45 B2 JMP $B245 ; BAD SUBSCRIPT
B30B  4C 35 A4 JMP $A435 ; OUT OF MEMORY
***** BEREKENING VAN ADRES ARRAYELEMENT
B30E  C8      INY
B30F  A5 72    LDA $72
B311  05 71    ORA $71
B313  18      CLC

```

```

B314 FO OA      BEQ $B320
B316 20 4C B3 JSR $B34C      ; VERMENIGVULDIGEN
B319 8A         TXA
B31A 65 64      ADC $64
B31C AA         TAX
B31D 98         TYA
B31E A4 22      LDY $22
B320 65 65      ADC $65
B322 86 71      STX $71
B324 C6 0B      DEC $0B      ; AANTAL DIMENSIES
B326 D0 CA      BNE $B2F2    ; DOORGAAN MET VOLGENDE INDEX
B328 85 72      STA $72
B32A A2 05      LDX #$05     ; DEFAULT VOOR LENGTE VAN
;                               VARIABELE (5=REEEL)
B32C A5 45      LDA $45     ; 1e LETTER VAN NAAM
B32E 10 01      BPL $B331
B330 CA         DEX
B331 A5 46      LDA $46     ; 2e LETTER VAN NAAM
B333 10 02      BPL $B337
B335 CA         DEX
B336 CA         DEX
B337 86 28      STX $28     ; LENGTE VAN VARIABELEN 2, 3 OF 5
B339 A9 00      LDA #$00
B33B 20 55 B3 JSR $B355    ; OFFSET IN ARRAY BEREKENEN
B33E 8A         TXA
B33F 65 58      ADC $58
B341 85 47      STA $47
B343 98         TYA
B344 65 59      ADC $59
B346 85 48      STA $48
B348 A8         TAY
B349 A5 47      LDA $47
B34B 60         RTS
***** HULPROUTINE VOOR ARRAYBEREKENING
B34C 84 22      STY $22
B34E B1 5F      LDA ($5F),Y
B350 85 28      STA $28
B352 88         DEY
B353 B1 5F      LDA ($5F),Y
B355 85 29      STA $29
B357 A9 10      LDA #$10
B359 85 5D      STA $5D
B35B A2 00      LDX #$00
B35D A0 00      LDY #$00
B35F 8A         TXA
B360 0A         ASL
B361 AA         TAX
B362 98         TYA
B363 2A         ROL

```

```

B364 A8      TAY
B365 B0 A4   BCS $B30B      ; OUT OF MEMORY
B367 06 71   ASL $71
B369 26 72   ROL $72
B36B 90 0B   BCC $B378
B36D 18      CLC
B36E 8A      TXA
B36F 65 28   ADC $28
B371 AA      TAX
B372 98      TYA
B373 65 29   ADC $29
B375 A8      TAY
B376 B0 93   BCS $B30B      ; OUT OF MEMORY
B378 C6 5D   DEC $5D
B37A D0 E3   BNE $B35F
B37C 60      RTS
***** BASIC FUNCTIE FRE
B37D A5 0D   LDA $0D      ; TYPE-FLAG
B37F F0 03   BEQ $B384    ; GEEN STRING
B381 20 A6 B6 JSR $B6A6    ; FRESTR
B384 20 26 B5 JSR $B526    ; GARBAGE COLLECTION
B387 38      SEC
B388 A5 33   LDA $33
B38A E5 31   SBC $31      ; BEGIN VAN STRING
B38C A8      TAY
B38D A5 34   LDA $34
B38F E5 32   SBC $32      ; MIN VARIABLENEINDE
B391 A2 00   LDX #$00
B393 86 0D   STX $0D      ; FLAG OP NUMERIEK ZETTEN
B395 85 62   STA $62
B397 84 63   STY $63      ; UITKOMST ONTHOUDEN
B399 A2 90   LDX #$90
B39B 4C 44 BC JMP $BC44    ; IN REEEL VERANDEREN
***** BASIC FUNCTIE POS
B39E 38      SEC      ; C=1 CURSORPOSITIE HALEN
B39F 20 FO FF JSR $FFFO    ; CURSORPOSITIE HALEN
B3A2 A9 00   LDA #$00
B3A4 FO EB   BEQ $B391     ; VERDER ALS BOVEN
***** DIREKTE MODUS?
B3A6 A6 3A   LDX $3A
B3A8 E8      INX
B3A9 D0 A0   BNE $B34B     ; NEE, DAN RTS
B3AB A2 15   LDX #$15     ; NUMMER VOOR ILLEGAL DIRECT
B3AD 2C      .BYTE $2C
B3AE A2 1B   LDX #$1B     ; NUMMER VOOR UNDEF'D FUNCTION
B3B0 4C 37 A4 JMP $A437    ; FOUTMELDING UITVOEREN
***** BASIC INSTRUCTIE DEF FN
B3B3 20 E1 B3 JSR $B3E1    ; FN-SYNTAX?
B3B6 20 A6 B3 JSR $B3A6    ; DIREKTE MODUS?

```

```

B3B9 20 FA AE JSR $AEFA      ; HAAKJE OPEN?
B3BC A9 80   LDA #$80
B3BE 85 10   STA $10        ; BLOKKEERT INTEGERE VARIABELE
B3C0 20 8B B0 JSR $B08B     ; ZOEKT VARIABELE
B3C3 20 8D AD JSR $AD8D     ; NUMERIEK?
B3C6 20 F7 AE JSR $AEF7     ; HAAKJE GESLOTEN?
B3C9 A9 B2   LDA #$B2      ; BASIC-CODE VOOR =
B3CB 20 FF AE JSR $AEFF     ; =?
B3CE 48      PHA
B3CF A5 48   LDA $48
B3D1 48      PHA            ; ADRES VAN FN-VARIABELE OP STACK
B3D2 A5 47   LDA $47
B3D4 48      PHA
B3D5 A5 7B   LDA $7B
B3D7 48      PHA            ; PROGRAMMAWIJZER OP STACK
B3D8 A5 7A   LDA $7A
B3DA 48      PHA
B3DB 20 F8 A8 JSR $A8F8     ; PROGRAMMAWIJZER
;                               OP VOLGEND STATEMENT
B3DE 4C 4F B4 JMP $B44F     ; FN-VARIABELE VAN STACK HALEN
***** FN-SYNTAX?
B3E1 A9 A5   LDA #$A5      ; FN-CODE
B3E3 20 FF AE JSR $AEFF     ; FN-CODE?
B3E6 09 80   ORA #$80
B3E8 85 10   STA $10        ; BLOKKEERT INTEGERE VARIABELE
B3EA 20 92 B0 JSR $B092     ; ZOEKT VARIABELE
B3ED 85 4E   STA $4E
B3EF 84 4F   STY $4F        ; WIJZER VOOR FN-VARIABELE ZETTEN
B3F1 4C 8D AD JMP $AD8D     ; NUMERIEK?
***** BASIC-FUNCTIE FN
83F4 20 E1 B3 JSR $B3E1     ; FN-SYNTAX?
B3F7 A5 4F   LDA $4F
B3F9 48      PHA            ; WIJZER VOOR FN-VARIABELE OP STACK
B3FA A5 4E   LDA $4E
B3FC 48      PHA
B3FD 20 F1 AE JSR $AEF1     ; HAALT TERM TUSSEN HAAKJES
B400 20 8D AD JSR $AD8D     ; NUMERIEK?
B403 68      PLA
B404 85 4E   STA $4E
B406 68      PLA            ; WIJZER VOOR FN-VARIABELE
;                               TERUGHALEN
B407 85 4F   STA $4F
B409 A0 02   LDY #$02
B40B B1 4E   LDA ($4E),Y
B40D 85 47   STA $47
B40F AA      TAX
B410 C8      INY
B411 B1 4E   LDA ($4E),Y
B413 F0 99   BEQ $B3AE     ; GEEFT UNDEF'D FUNCTION

```



```

B415 85 48 STA $48
B417 C8 INY
B418 B1 47 LDA ($47),Y
B41A 48 PHA
B41B 88 DEY
B41C 10 FA BPL $B418
B41E A4 48 LDY $48
B420 20 D4 BB JSR $BBD4 ; FAC NAAR FN-VARIABELE
; OVERBRENGEN
B423 A5 7B LDA $7B
B425 48 PHA
B426 A5 7A LDA $7A
B428 48 PHA
B429 B1 4E LDA ($4E),Y
B42B 85 7A STA $7A
B42D C8 INY ; PROGRAMMAWIJZER OP FN-TERM
B42E B1 4E LDA ($4E),Y
B430 85 7B STA $7B
B432 A5 48 LDA $48
B434 48 PHA
B435 A5 47 LDA $47
B437 48 PHA
B438 20 8A AD JSR $AD8A ; FRMNUM: NUMERIEKE TERM HALEN
B43B 68 PLA
B43C 85 4E STA $4E
B43E 68 PLA
B43F 85 4F STA $4F
B441 20 79 00 JSR $0079 ; CHRGOT: LAATSTE TEKEN HALEN
B444 F0 03 BEQ $B449 ; GEEN ANDERE TEKENS?
B446 4C 08 AF JMP $AF08 ; GEEFT SYNTAX ERROR
B449 68 PLA
B44A 85 7A STA $7A
B44C 68 PLA ; PROGRAMMAWIJZER
B44D 85 7B STA $7B
B44F A0 00 LDY #$00
B451 68 PLA
B452 91 4E STA ($4E),Y
B454 68 PLA
B455 C8 INY
B456 91 4E STA ($4E),Y
B458 68 PLA
B459 C8 INY ; FN-VARIABELE VAN STACK HALEN
B45A 91 4E STA ($4E),Y
B45C 68 PLA
B45D C8 INY
B45E 91 4E STA ($4E),Y
B460 68 PLA
B461 C8 INY
B462 91 4E STA ($4E),Y

```

```

B464 60      RTS
***** BASIC-FUNCTIE STR$
B465 20 8D AD JSR $AD8D      ; NUMERIEK?
B468 A0 00    LDY #$00
B46A 20 DF BD JSR $BDDF      ; FAC IN ASCII VERANDEREN
B46D 68      PLA
B46E 68      PLA
B46F A9 FF    LDA #$FF      ; BEGINADRES STRING=$FF
B471 A0 00    LDY #$00
B473 FO 12    BEQ $B487
***** STRINGWIJZER BEREKENEN
B475 A6 64    LDX $64      : ACCU BEVAT STRINGLENGTE
B477 A4 65    LDY $65
B479 86 50    STX $50      ; WIJZER OP STRINGDESCRIPTOR
B47B 84 51    STY $51
B47D 20 F4 B4 JSR $B4F4      ; MAAKT PLAATS VOOR NIEUWE
;                               STRING, LENGTE IN ACCU
B480 86 62    STX $62      ; ADRES LOW
B482 84 63    STY $63      ; ADRES HIGH
B484 85 61    STA $61      ; LENGTE
B486 60      RTS
***** STRING HALEN, WIJZER IN A/Y
B487 A2 22    LDX #$22      ; AANHalingSTeKEN
B489 86 07    STX $07
B48B 86 08    STX $08
B48D 85 6F    STA $6F      ; BEGINADRES VAN STRING
B48F 84 70    STY $70
B491 85 62    STA $62
B493 84 63    STY $63
B495 A0 FF    LDY #$FF
B497 C8      INY           ; WIJZER VERHOGEN
B498 B1 6F    LDA ($6F),Y   ; VOLGEND TEKEN VAN STRING
B49A FO 0C    BEQ $B4A8      ; MARKERING VAN EINDE?
B49C C5 07    CMP $07
B49E FO 04    BEQ $B4A4
B4A0 C5 08    CMP $08
B4A2 D0 F3    BNE $B497
B4A4 C9 22    CMP #$22      ; AANHalingSTeKEN
B4A6 FO 01    BEQ $B4A9
B4A8 18      CLC
B4A9 84 61    STY $61      ; LENGTE VAN STRING
B4AB 98      TYA
B4AC 65 6F    ADC $6F
B4AE 85 71    STA $71      ; EINDADRES LOW+1
B4B0 A6 70    LDX $70
B4B2 90 01    BCC $B4B5
B4B4 E8      INX
B4B5 86 72    STX $72      ; EINDADRES HIGH+1
B4B7 A5 70    LDA $70      ; BEGINADRES HIGH

```

```

B4B9  F0 04      BEQ $B4BF      ; NUL?
B4BB  C9 02      CMP #$02       ; TWEE?
B4BD  DO 0B      BNE $B4CA      ; NEE
B4BF  98         TYA            ; LENGTE IN ACCU
B4C0  20 75 B4   JSR $B475      ; STRINGWIJZER BEREKENEN
B4C3  A6 6F      LDX $6F        ;
B4C5  A4 70      LDY $70        ; BEGINADRES HALEN
B4C7  20 88 B6   JSR $B688      ; STRING IN STRINGBEREIK
;                                     KOPIEREN
***** STRINGWIJZER OP DESCRIPTOR-
;                                     STACK LEGGEN
B4CA  A6 16      LDX $16        ; STRINGDESCRIPTOR-WIJZER
B4CC  EO 22      CPX #$22       ; STRINGSTACK VOL?
B4CE  DO 05      BNE $B4D5      ; NEE
B4D0  A2 19      LDX #$19       ; NUMMER VOOR
;                                     FORMULA TOO COMPLEX
B4D2  4C 37 A4   JMP $A437      ; FOUTMELDING UITVOEREN
B4D5  A5 61      LDA $61        ;
B4D7  95 00      STA $00,X      ; LENGTE STRING
B4D9  A5 62      LDA $62        ;
B4DB  95 01      STA $01,X      ; EN ADRES OP
B4DD  A5 63      LDA $63        ;
B4DF  95 02      STA $02,X      ; STRINGSTACK LEGGEN
B4E1  AO 00      LDY #$00       ;
B4E3  86 64      STX $64        ;
B4E5  84 65      STY $65        ; WIJZER NU OP DESCRIPTOR
B4E7  84 70      STY $70        ;
B4E9  88         DEY           ;
B4EA  84 OD      STY $OD        ; STRINGFLAG ZETTEN $FF
B4EC  86 17      STX $17        ;

B4EE  E8         INX           ;
B4EF  E8         INX           ; MET DRIE VERHOGEN
BF40  E8         INX           ;
B4F1  86 16      STX $16        ; ALS NIEUWE INDEX MARKEREN
B4F3  60         RTS           ;
***** PLAATS VOOR STRING RESERVEREN,
;                                     LENGTE IN A
B4F4  46 OF      LSR $OF        ; FLAG VOOR GARBAGE COLLECTION
;                                     TERUGZETTEN
B4F6  48         PHA           ; STRINGLENGTE
B4F7  49 FF      EOR #$FF       ;
B4F9  38         SEC           ;
B4FA  65 33      ADC $33        ;
B4FC  A4 34      LDY $34        ;
B4FE  B0 01      BCS $B501      ;
B500  88         DEY           ;
B501  C4 32      CPY $32        ;

```

```

B503 90 11    BCC $B516    ; TE WEINIG PLAATS,
;              DAN GARBAGE OPRUIMEN
B505 D0 04    BNE $B50B
B507 C5 31    CMP $31
B509 90 0B    BCC $B516
B50B 85 33    STA $33
B50D 84 34    STY $34
B50F 85 35    STA $35
B511 84 36    STY $36
B513 AA       TAX
B514 68       PLA          ; STRINGLENGTE TERUGHALEN
B515 60       RTS
B516 A2 10    LDX #$10     ; NUMMER VOOR OUT OF MEMORY
B518 A5 0F    LDA $0F     ; FLAG: ROMMEL AL
B51A 30 B6    BMI $B4D2   ; OPPERUIMD, DAN OUT OF MEMORY
B51C 20 26 B5 JSR $B526   ; GARBAGE OPRUIMEN
B51F A9 80    LDA #$80    ; FLAG ZETTEN
B521 85 0F    STA $0F
B523 68       PLA          ; STRINGLENGTE
B524 D0 D0    BNE $B4F6
***** GARBAGE COLLECTION
B526 A6 37    LDX $37     ; ONGELDIGE STRINGS WISSEN
B528 A5 38    LDA $38
B52A 86 33    STX $33
B52C 85 34    STA $34
B52E A0 00    LDY #$00
B530 84 4F    STY $4F
B532 84 4E    STY $4E
B534 A5 31    LDA $31
B536 A6 32    LDX $32
B538 85 5F    STA $5F
B53A 86 60    STX $60
B53C A9 19    LDA #$19
B53E A2 00    LDX #$00
B540 85 22    STA $22
B542 86 23    STX $23
B544 C5 16    CMP $16
B546 F0 05    BEQ $B54D
B548 20 C7 B5 JSR $B5C7
B54B F0 F7    BEQ $B544
B54D A9 07    LDA #$07
B54F 85 53    STA $53
B551 A5 2D    LDA $2D
B553 A6 2E    LDX $2E
B555 85 22    STA $22
B557 86 23    STX $23
B559 E4 30    CPX $30
B55B D0 04    BNE $B561
B55D C5 2F    CMP $2F

```

B55F	F0	05		BEQ	\$B566
B561	20	BD	B5	JSR	\$B5BD
B564	F0	F3		BEQ	\$B559
B566	85	58		STA	\$58
B568	86	59		STX	\$59
B56A	A9	03		LDA	#\$03
B56C	85	53		STA	\$53
B56E	A5	58		LDA	\$58
B570	A6	59		LDX	\$59
B572	E4	32		CPX	\$32
B574	D0	07		BNE	\$B57D
B576	C5	31		CMP	\$31
B578	D0	03		BNE	\$B57D
B57A	4C	06	B6	JMP	\$B606
B57D	85	22		STA	\$22
B57F	86	23		STX	\$23
B581	A0	00		LDY	#\$00
B583	B1	22		LDA	(\$22),Y
B585	AA			TAX	
B586	C8			INY	
B587	B1	22		LDA	(\$22),Y
B589	08			PHP	
B58A	C8			INY	
B58B	B1	22		LDA	(\$22),Y
B58D	65	58		ADC	\$58
B58F	85	58		STA	\$58
B591	C8			INY	
B592	B1	22		LDA	(\$22),Y
B594	65	59		ADC	\$59
B596	85	59		STA	\$59
B598	28			PLP	
B599	10	D3		BPL	\$B56E
B59B	8A			TXA	
B59C	30	D0		BMI	\$B56E
B59E	C8			INY	
B59F	B1	22		LDA	(\$22),Y
B5A1	A0	00		LDY	#\$00
B5A3	0A			ASL	
B5A4	69	05		ADC	#\$05
B5A6	65	22		ADC	\$22
B5A8	85	22		STA	\$22
B5AA	90	02		BCC	\$B5AE
B5AC	E6	23		INC	\$23
B5AE	A6	23		LDX	\$23
B5B0	E4	59		CPX	\$59
B5B2	D0	04		BNE	\$B5B8
B5B4	C5	58		CMP	\$58
B5B6	F0	BA		BEQ	\$B572
B5B8	20	C7	B5	JSR	\$B5C7

B5BB F0 F3 BEQ \$B5B0
 ***** KAN ER GEWIST WORDEN?

B5BD B1 22 LDA (\$22),Y
 B5BF 30 35 BMI \$B5F6
 B5C1 C8 INY
 B5C2 B1 22 LDA (\$22),Y
 B5C4 10 30 BPL \$B5F6
 B5C6 C8 INY
 B5C7 B1 22 LDA (\$22),Y
 B5C9 F0 2B BEQ \$B5F6
 B5CB C8 INY
 B5CC B1 22 LDA (\$22),Y
 B5CE AA TAX
 B5CF C8 INY
 B5D0 B1 22 LDA (\$22),Y
 B5D2 C5 34 CMP \$34
 B5D4 90 06 BCC \$B5DC
 B5D6 D0 1E BNE \$B5F6
 B5D8 E4 33 CPX \$33
 B5DA B0 1A BCS \$B5F6
 B5DC C5 60 CMP \$60
 B5DE 90 16 BCC \$B5F6
 B5E0 D0 04 BNE \$B5E6
 B5E2 E4 5F CPX \$5F
 B5E4 90 10 BCC \$B5F6
 B5E6 86 5F STX \$5F
 B5E8 85 60 STA \$60
 B5EA A5 22 LDA \$22
 B5EC A6 23 LDX \$23
 B5EE 85 4E STA \$4E
 B5F0 86 4F STX \$4F
 B5F2 A5 53 LDA \$53
 B5F4 85 55 STA \$55
 B5F6 A5 53 LDA \$53
 B5F8 18 CLC
 B5F9 65 22 ADC \$22
 B5FB 85 22 STA \$22
 B5FD 90 02 BCC \$B601
 B5FF E6 23 INC \$23

B601 A6 23 LDX \$23
 B603 A0 00 LDY #\$00
 B605 60 RTS

***** STRINGS KOPPELEN

B606 A5 4F LDA \$4F
 B608 05 4E ORA \$4E
 B60A F0 F5 BEQ \$B601
 B60C A5 55 LDA \$55
 B60E 29 04 AND #\$04

```

B610 4A      LSR
B611 A8      TAY
B612 85 55   STA $55
B614 B1 4E   LDA ($4E),Y
B616 65 5F   ADC $5F      ; $5F/$60=WIJZER OP
                          BEGIN VAN OUD BLOK
;
B618 85 5A   STA $5A
B61A A5 60   LDA $60
B61C 69 00   ADC #$00
B61E 85 5B   STA $5B      ; $5A/$5B=WIJZER OP
                          EIND VAN OUD BLOK
;
B620 A5 33   LDA $33
B622 A6 34   LDX $34
B624 85 58   STA $58      ; $58/59=WIJZER OP
                          EIND VAN NIEUW BLOK
;
B626 86 59   STX $59
B628 20 BF A3 JSR $A3BF    ; STRINGS VERSCHUIVEN
B62B A4 55   LDY $55
B62D C8      INY
B62E A5 58   LDA $58
B630 91 4E   STA ($4E),Y
B632 AA      TAX
B633 E6 59   INC $59
B635 A5 59   LDA $59
B637 C8      INY
B638 91 4E   STA ($4E),Y
B63A 4C 2A B5 JMP $B52A
***** STRINGS VERBINDEN MET +
B63D A5 65   LDA $65
B63F 48      PHA          ; DESCRIPTOR VAN 1e STRING
                          ONTHOUDEN
;
B640 A5 64   LDA $64
B642 48      PHA
B643 20 83 AE JSR $AE83    ; 2e STRING HALEN
B646 20 8F AD JSR $AD8F    ; STRINGVARIABELE?
B649 68      PLA
B64A 85 6F   STA $6F
B64C 68      PLA          ; DESCRIPTOR TERUGHALEN
B64D 85 70   STA $70
B64F A0 00   LDY #$00
B651 B1 6F   LDA ($6F),Y  ; LENGTE VAN 1e STRING
B653 18      CLC
B654 71 64   ADC ($64),Y  ; PLUS LENGTE VAN 2e STRING
B656 90 05   BCC $B65D    ; < 256
B658 A2 17   LDX #$17     ; NUMMER VOOR STRING TOO LONG
B65A 4C 37 A4 JMP $A437    ; FOUTMELDING UITVOEREN
B65D 20 75 B4 JSR $B475    ; PLAATS RESERVEREN VOOR
                          VERBONDEN STRING
;
B660 20 7A B6 JSR $B67A    ; 1e STRING DAAR NAARTOE OVERDRAGEN

```

```

B663 A5 50 LDA $50
B665 A4 51 LDY $51 ; WIJZER OP 2e STRINGDESCRIPTOR
B667 20 AA B6 JSR $B6AA ; FRESTR
B66A 20 8C B6 JSR $B68C ; 2e STRING AAN 1e KOPPELEN
B66D A5 6F LDA $6F
B66F A4 70 LDY $70
B671 20 AA B6 JSR $B6AA ; FRESTR
B674 20 CA B4 JSR $B4CA ; DESCRIPTOR IN STRINGSTACK
B677 4C B8 AD JMP $ADB8 ; TERUG NAAR INTERPRETATIE VAN
; FORMULE
***** STRING IN GERESERVEERDE
; BEREIK OVERDRAGEN
B67A A0 00 LDY #$00
B67C B1 6F LDA ($6F),Y ; STRINGLENGTE ONTHOUDEN
B67E 48 PHA
B67F C8 INY
B680 B1 6F LDA ($6F),Y ; STRINGADRES LOW
B682 AA TAX
B683 C8 INY
B684 B1 6F LDA ($6F),Y ; STRINGADRES HIGH
B686 A8 TAY
B687 68 PLA ; STRINGLENGTE
B688 86 22 STX $22
B68A 84 23 STY $23 ; WIJZER OP STRING
B68C A8 TAY
B68D F0 0A BEQ $B699 ; LENGTE NUL, DAN KLAAR
B68F 48 PHA
B690 88 DEY
B691 B1 22 LDA ($22),Y ; STRING OVERDRAGEN
B693 91 35 STA ($35),Y
B695 98 TYA
B696 D0 F8 BNE $B690
B698 68 PLA
B699 18 CLC
B69A 65 35 ADC $35
B69C 85 35 STA $35 ; WIJZER+STRINGLENGTE
B69E 90 02 BCC $B6A2
B6A0 E6 36 INC $36
B6A2 60 RTS
***** FRESTR: STRINGBEHEER
B6A3 20 8F AD JSR $AD8F ; STRINGVARIABLE?
B6A6 A5 64 LDA $64
B6A8 A4 65 LDY $65 ; WIJZER OP STRINGDESCRIPTOR
B6AA 85 22 STA $22
B6AC 84 23 STY $23
B6AE 20 DB B6 JSR $B6DB ; DESCRIPTOR UIT STRINGSTACK
; VERWIJDEREN
B6B1 08 PHP
B6B2 A0 00 LDY #$00

```



```

B6B4 B1 22 LDA ($22),Y
B6B6 48 PHA
B6B7 C8 INY
B6B8 B1 22 LDA ($22),Y
B6BA AA TAX
B6BB C8 INY
B6BC B1 22 LDA ($22),Y
B6BE A8 TAY
B6BF 68 PLA
B6C0 28 PLP
B6C1 D0 13 BNE $B6D6 ; STRING NIET IN STRINGSTACK
B6C3 C4 34 CPY $34
B6C5 D0 0F BNE $B6D6
B6C7 E4 33 CPX $33
B6C9 D0 0B BNE $B6D6
B6CB 48 PHA
B6CC 18 CLC
B6CD 65 33 ADC $33
B6CF 85 33 STA $33 ; $33/$34 WIJST NU OP
; STRINGBEGIN
B6D1 90 02 BCC $B6D5
B6D3 E6 34 INC $34
B6D5 68 PLA
B6D6 86 22 STX $22
B6D8 84 23 STY $23
B6DA 60 RTS
***** STRINGWIJZER VERWIJDEREN UIT
; DESCRIPTORSTACK
B6DB C4 18 CPY $18
B6DD D0 0C BNE $B6EB
B6DF C5 17 CMP $17 ; STRINGDESCRIPTOR IN
; STRINGSTACK?
B6E1 D0 08 BNE $B6EB
B6E3 85 16 STA $16
B6E5 E9 03 SBC #$03 ; JA, DAN VERWIJDEREN
B6E7 85 17 STA $17
B6E9 A0 00 LDY #$00
B6EB 60 RTS
***** BASIC-FUNCTIE CHR$
B6EC 20 A1 B7 JSR $B7A1 ; HAALT BYTE-WAARDE (0-255)
B6EF 8A TXA ; CODE IN ACCU
B6F0 48 PHA
B6F1 A9 01 LDA #$01 ; STRINGLENGTE=1
B6F3 20 7D B4 JSR $B47D ; PLAATS VOOR STRING RESERVEREN
B6F6 68 PLA ; ASCII-CODE TERUGHALEN
B6F7 A0 00 LDY #$00
B6F9 91 62 STA ($62),Y ; OPSLAAN ALS STRINGTEKEN
B6FB 68 PLA
B6FC 68 PLA

```

```

B6FD 4C CA B4 JMP $B4CA ; DESCRIPTOR OP STRINGSTACK
***** BASIC-FUNCTIE LEFT$
B700 20 61 B7 JSR $B761 ; STRINGPARAMETER EN -LENGTE
; VAN STACK HALEN
B703 D1 50 CMP ($50),Y ; LENGTE MET LEFT$-PARAMETER
; VERGELIJKEN
B705 98 TYA
B706 90 04 BCC $B70C ; LEFT$-PARAMETER < STRINGLENGTE?
B708 B1 50 LDA ($50),Y ; STRINGLENGTE
B70A AA TAX
B70B 98 TYA
B70C 48 PHA
B70D 8A TXA
B70E 48 PHA
B70F 20 7D B4 JSR $B47D ; PLAATS VOOR NIEUWE STRING
; RESERVEREN
B712 A5 50 LDA $50
B714 A4 51 LDY $51 ; WIJZER OP DESCRIPTOR
B716 20 AA B6 JSR $B6AA ; FRESTR
B719 68 PLA
B71A A8 TAY
B71B 68 PLA ; LENGTE VAN NIEUWE STRING
B71C 18 CLC
B71D 65 22 ADC $22 ; PLUS ADRES VAN OUDE STRING
B71F 85 22 STA $22
B721 90 02 BCC $B725
B723 E6 23 INC $23
B725 98 TYA
B726 20 8C B6 JSR $B68C ; NIEUWE STRING IN STRINGBEREIK
; OVERDRAGEN
B729 4C CA B4 JMP $B4CA ; DESCRIPTOR OP STRINGSTACK
***** BASIC-FUNCTIE RIGHT$
B72C 20 61 B7 JSR $B761 ; STRINGPARAMETER EN -LENGTE
; VAN STACK HALEN
B72F 18 CLC
B730 F1 50 SBC ($50),Y ; VAN STRINGLENGTE AFTREKKEN
B732 49 FF EOR #$FF ; NUMMER VAN 1e ELEMENT
; IN OUDE STRING
B734 4C 06 B7 JMP $B706 ; VERDER ALS LEFT$
***** BASIC-FUNCTIE MID$
B737 A9 FF LDA #$FF
B739 85 65 STA $65
B73B 20 79 00 JSR $0079 ; CHRGET: LAATSTE TEKEN HALEN
B73E C9 29 CMP #$29 ; HAAKJE SLUITEN
B740 F0 06 BEQ $B748
B742 20 FD AE JSR $AEFD ; KOMMA?
B745 20 9E B7 JSR $B79E ; HAALT BYTE-WAARDE, 2e PARAMETER
B748 20 61 B7 JSR $B761 ; STRINGPARAMETER EN
; BEGINPOSITIE HALEN

```

```

B74B FO 4B BEQ $B798 ; 1e PARAMETER=0, ILLEGAL
; QUANTITY
B74D CA DEX
B74E 8A TXA
B74F 48 PHA ; NUMMER VAN 1e ELEMENT
; IN OUDE STRING
B750 18 CLC
B751 A2 00 LDX #$00
B753 F1 50 SBC ($50),Y ; LENGTE VAN OUDE STRING
B755 B0 B6 BCS $B70D ; < 1e MID-PARAMETER
B757 49 FF EOR #$FF ; NIEUWE STRINGLENGTE
B759 C5 65 CMP $65
B75B 90 B1 BCC $B70E
B75D A5 65 LDA $65
B75F B0 AD BCS $B70E ; ABSOLUTE SPRONG
***** STRINGPARAMETER NUMERIEKE
; WAARDE VAN STACK HALEN
B761 20 F7 AE JSR $AEF7 ; HAAKJE GESLOTEN?
B764 68 PLA
B765 A8 TAY ; OPROEPADRES ONTHOUDEN
B766 68 PLA
B767 85 55 STA $55
B769 68 PLA
B76A 68 PLA
B76B 68 PLA ; 1e PARAMETER
B76C AA TAX
B76D 68 PLA ; ADRES LOW EN HIGH VAN
; STRINGDESCRIPTOR
B76E 85 50 STA $50
B770 68 PLA
B771 85 51 STA $51
B773 A5 55 LDA $55
B775 48 PHA
B776 98 TYA ; OPROEPADRES WEER OP STACK
B777 48 PHA
B778 A0 00 LDY #$00
B77A 8A TXA ; LENGTE, 2e PARAMETER
B77B 60 RTS
***** BASIC-FUNCTIE LEN
B77C 20 82 B7 JSR $B782 ; FRESTR, STRINGLENGTE HALEN
B77F 4C A2 B3 JMP $B3A2 ; BYTE-WAARDE IN REEL VERANDEREN
***** STRINGPARAMETER HALEN
B782 20 A3 B6 JSR $B6A3 ; FRESTR, STRING HALEN
; LENGTE IN ACCU
B785 A2 00 LDX #$00
B787 86 0D STX $0D ; TYPE-FLAG OP NUMERIEK ZETTEN
B789 A8 TAY ; LENGTE IN Y
B78A 60 RTS
***** BASIC-FUNCTIE ASC

```

```

B78B 20 82 B7 JSR $B782 ; STRING HALEN, WIJZER
; IN $22/$23, LENGTE IN Y
B78E FO 08 BEQ $B798 ; LENGTE=0, ILLEGAL QUANTITY
B790 AO 00 LDY #$00
B792 B1 22 LDA ($22),Y ; 1e TEKEN HALEN
B794 A8 TAY ; ASCII-CODE
B795 4C A2 B3 JMP $B3A2 ; IN REEEL VERANDEREN
B798 4C 48 B2 JMP $B248 ; ILLEGAL QUANTITY
***** HAALT BYTE-WAARDE NAAR X
B79B 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
B79E 20 8A AD JSR $AD8A ; FRMNUM: NUMERIEKE WAARDE
; NAAR FAC HALEN
B7A1 20 B8 B1 JSR $B1B8 ; CONTROLEERT BEREIK EN
; VERANDERT IN INTEGER
B7A4 A6 64 LDX $64 ; MSB
B7A6 D0 F0 BNE $B798 ; ONGELIJK 0, DAN ILLEGAL QUANTITY
B7A8 A6 65 LDX $65
B7AA 4C 79 00 JMP $0079 ; CHRGOT: LAATSTE TEKEN HALEN
***** BASIC-FUNCTIE VAL
B7AD 20 82 B7 JSR $B782 ; ADRES EN LENGTE VAN STRING HALEN
B7B0 D0 03 BNE $B7B5 ; STRINGLENGTE ONGELIJK 0?
B7B2 4C F7 B8 JMP $B8F7 ; 0 IN FAC
B7B5 A6 7A LDX $7A
B7B7 A4 7B LDY $7B ; PROGRAMMAWIJZER WEGZETTEN
B7B9 86 71 STX $71
B7BB 84 72 STY $72
B7BD A6 22 LDX $22
B7BF 86 7A STX $7A ; BEGINADRES VAN STRING IN
; STRINGWIJZER ZETTEN
B7C1 18 CLC
B7C2 65 22 ADC $22
B7C4 85 24 STA $24
B7C6 A6 23 LDX $23 ; EINDADRES VAN STRING+1
B7C8 86 7B STX $7B
B7CA 90 01 BCC $B7CD
B7CC E8 INX
B7CD 86 25 STX $25
B7CF AO 00 LDY #$00
B7D1 B1 24 LDA ($24),Y ; 1e BYTE NA STRING
B7D3 48 PHA ; WEGZETTEN
B7D4 98 TYA
B7D5 91 24 STA ($24),Y ; EN DOOR 0 VERVANGEN
B7D7 20 79 00 JSR $0079 ; CHRGOT: LAATSTE TEKEN HALEN
B7DA 20 F3 BC JSR $BCF3 ; STRING IN REEEL VERANDEREN
B7DD 68 PLA ; TEKEN NA STRING
B7DE AO 00 LDY #$00
B7E0 91 24 STA ($24),Y ; WEER TERUGZETTEN
B7E2 A6 71 LDX $71
B7E4 A4 72 LDY $72

```

```

B7E6 86 7A STX $7A ; PROGRAMMAWIJZER TERUGHALEN
B7E8 84 7B STY $7B
B7EA 60 RTS
***** GETADR EN GETBYT: HALEN 16-
; EN 8-BITS WAARDE
B7EB 20 8A AD JSR $AD8A ; FRMNUM: HAALT NUMERIEKE WAARDE
B7EE 20 F7 B7 JSR $B7F7 ; FAC IN ADRESFORMAAT
; VERANDEREN $14/$15
B7F1 20 FD AE JSR $AEFD ; CHKCOM: KOMMA?
B7F4 4C 9E B7 JMP $B79E ; HAALT BYTE-WAARDE NAAR X
***** GETADR: FAC IN POSITIEF
; 16-BITS GETAL VERANDEREN
B7F7 A5 66 LDA $66 ; VOORTEKEN
B7F9 30 9D BMI $B798 ; NEGATIEF, DAN ILLEGAL QUANTITY
B7FB A5 61 LDA $61 ; EXPONENT
B7FD C9 91 CMP #$91 ; GETAL MET 65536 VERGELIJKEN
B7FF B0 97 BCS $B798 ; GROTER, DAN ILLEGAL QUANTITY

B801 20 9B BC JSR $BC9B ; FAC IN ADRESFORMAAT VERANDEREN
B804 A5 64 LDA $64
B806 A4 65 LDY $65 ; WAARDE HALEN
B808 84 14 STY $14 ; EN NAAR $14/$15
B80A 85 15 STA $15
B80C 60 RTS
***** BASIC-FUNCTIE PEEK
B80D A5 15 LDA $15
B80F 48 PHA ; ADRES $14/$15 WEGZETTEN
B810 A5 14 LDA $14
B812 48 PHA
B813 20 F7 B7 JSR $B7F7 ; FAC NAAR ADRESFORMAAT
B816 A0 00 LDY #$00
B818 B1 14 LDA ($14),Y ; PEEK-WAARDE HALEN
B81A A8 TAY ; NAAR Y
B81B 68 PLA
B81C 85 14 STA $14
B81E 68 PLA ; ADRES TERUGHALEN
B81F 85 15 STA $15
B821 4C A2 B3 JMP $B3A2 ; Y IN REEEL VERANDEREN
***** BASIC-INSTRUCTIE POKE
B824 20 EB B7 JSR $B7EB ; POKE-ADRES EN -WAARDE HALEN
B827 8A TXA ; POKE-WAARDE IN ACCU
B828 A0 00 LDY #$00
B82A 91 14 STA ($14),Y ; EN IN GEHEUGEN SCHRIJVEN
B82C 60 RTS
***** BASIC-INSTRUCTIE WAIT
B82D 20 EB B7 JSR $B7EB ; ADRES EN WAARDE HALEN
B830 86 49 STX $49
B832 A2 00 LDX #$00 ; DEFAULT VOOR 3e PARAMETER
B834 20 79 00 JSR $0079 ; CHRGTOT: LAATSTE TEKEN

```

```

B837 FO 03      BEQ $B83C      ; GEEN 3e PARAMETER?
B839 20 F1 B7 JSR $B7F1      ; CONTROLEERT OP KOMMA
;
B83C 86 4A      STX $4A
B83E A0 00      LDY #$00
B840 B1 14      LDA ($14),Y    ; WAIT-ADRES
B842 45 4A      EOR $4A
B844 25 49      AND $49        ; LOGISCHE VERBINDING
B846 FO F8      BEQ $B840      ; BLIJVEN WACHTEN
B848 60         RTS
***** REKENROUTINES
***** FAC=FAC+0.5
B849 A9 11      LDA #$11
B84B A0 BF      LDY #$BF        ; WIJZER OP CONSTATE 0.5
B84D 4C 67 B8 JMP $B867        ; FAC=FAC+CONSTANTE (A/Y)
***** FAC=CONSTANTE (A/Y)-FAC
B850 20 8C BA JSR $BA8C        ; CONSTATE (A/Y) NAAR FAC
***** FAC=ARG-FAC
B853 A5 66      LDA $66
B855 49 FF      EOR #$FF        ; VOORTEKEN OMDRAAIEN
B857 85 66      STA $66
B859 45 6E      EOR $6E
B85B 85 6F      STA $6F
B85D A5 61      LDA $61
B85F 4C 6A B8 JMP $B86A        ; FAC=FAC+ARG
*****
B862 20 99 B9 JSR $B999        ; EXPONENTEN VAN FAC EN ARG
;
B865 90 3C      BCC $B8A3
***** PLUS FAC=CONSTANTE (A/Y)+FAC
B867 20 8C BA JSR $BA8C        ; CONSTATE(A/Y) NAAR ARG
***** PLUS FAC=FAC+ARG
B86A D0 03      BNE $B86F        ; FAC ONGELIJK NUL?
B86C 4C FC BB JMP $BBFC        ; NEE, DAN FAC=ARG
B86F A6 70      LDX $70
B871 86 56      STX $56
B873 A2 69      LDX #$69
B875 A5 69      LDA $69
B877 A8         TAY
B878 FO CE      BEQ $B848
B87A 38         SEC
B87B E5 61      SBC $61
B87D FO 24      BEQ $B8A3
B87F 90 12      BCC $B893
B881 84 61      STY $61
B883 A4 6E      LDY $6E
B885 84 66      STY $66
B887 49 FF      EOR #$FF

```

B889	69	00	ADC	#\$00
B88B	A0	00	LDY	#\$00
B88D	84	56	STY	\$56
B88F	A2	61	LDX	#\$61
B891	D0	04	BNE	\$B897
B893	A0	00	LDY	#\$00
B895	84	70	STY	\$70
B897	C9	F9	CMP	#\$F9
B899	30	C7	BMI	\$B862
B89B	A8		TAY	
B89C	A5	70	LDA	\$70
B89E	56	01	LSR	\$01,X
B8A0	20	B0	B9 JSR	\$B9B0
B8A3	24	6F	BIT	\$6F
B8A5	10	57	BPL	\$B8FE
B8A7	A0	61	LDY	#\$61
B8A9	E0	69	CPX	#\$69
B8AB	F0	02	BEQ	\$B8AF
B8AD	A0	69	LDY	#\$69
B8AF	38		SEC	
B8B0	49	FF	EOR	#\$FF
B8B2	65	56	ADC	\$56
B8B4	85	70	STA	\$70
B8B6	B9	04	00 LDA	\$0004,Y
B8B9	F5	04	SBC	\$04,X
B8BB	85	65	STA	\$65
B8BD	B9	03	00 LDA	\$0003,Y
B8C0	F5	03	SBC	\$03,X
B8C2	85	64	STA	\$64
B8C4	B9	02	00 LDA	\$0002,Y
B8C7	F5	02	SBC	\$02,X
B8C9	85	63	STA	\$63
B8CB	B9	01	00 LDA	\$0001,Y
B8CE	F5	01	SBC	\$01,X
B8D0	85	62	STA	\$62
B8D2	B0	03	BCS	\$B8D7
B8D4	20	47	B9 JSR	\$B947
B8D7	A0	00	LDY	#\$00
B8D9	98		TYA	
B8DA	18		CLC	
B8DB	A6	62	LDX	\$62
B8DD	D0	4A	BNE	\$B929
B8DF	A6	63	LDX	\$63
B8E1	86	62	STX	\$62
B8E3	A6	64	LDX	\$64
B8E5	86	63	STX	\$63
B8E7	A6	65	LDX	\$65
B8E9	86	64	STX	\$64
B8EB	A6	70	LDX	\$70

; MANTISSE VAN FAC INVERTEREN

B8ED	86 65	STX \$65
B8EF	84 70	STY \$70
B8F1	69 08	ADC #\$08
B8F3	C9 20	CMP #\$20
B8F5	D0 E4	BNE \$B8DB
B8F7	A9 00	LDA #\$00
B8F9	85 61	STA \$61
B8FB	85 66	STA \$66
B8FD	60	RTS
B8FE	65 56	ADC \$56

B900	85 70	STA \$70
B902	A5 65	LDA \$65
B904	65 6D	ADC \$6D
B906	85 65	STA \$65
B908	A5 64	LDA \$64
B90A	65 6C	ADC \$6C
B90C	85 64	STA \$64
B90E	A5 63	LDA \$63
B910	65 6B	ADC \$6B
B912	85 63	STA \$63
B914	A5 62	LDA \$62
B916	65 6A	ADC \$6A
B918	85 62	STA \$62
B91A	4C 36 B9	JMP \$B936
B91D	69 01	ADC #\$01
B91F	06 70	ASL \$70
B921	26 65	ROL \$65
B923	26 64	ROL \$64
B925	26 63	ROL \$63
B927	26 62	ROL \$62
B929	10 F2	BPL \$B91D
B92B	38	SEC
B92C	E5 61	SBC \$61
B92E	B0 C7	BCS \$B8F7
B930	49 FF	EOR #\$FF
B932	69 01	ADC #\$01
B934	85 61	STA \$61
B936	90 0E	BCC \$B946
B938	E6 61	INC \$61
B93A	F0 42	BEQ \$B97E
B93C	66 62	ROR \$62
B93E	66 63	ROR \$63
B940	66 64	ROR \$64
B942	66 65	ROR \$65
B944	66 70	ROR \$70
B946	60	RTS

***** MANTISSE VAN FAC INVERTEREN

B947	A5 66	LDA \$66	
B949	49 FF	EOR #\$FF	
B94B	85 66	STA \$66	
B94D	A5 62	LDA \$62	
B94F	49 FF	EOR #\$FF	
B951	85 62	STA \$62	
B953	A5 63	LDA \$63	
B955	49 FF	EOR #\$FF	
B957	85 63	STA \$63	
B959	A5 64	LDA \$64	
B95B	49 FF	EOR #\$FF	
B95D	85 64	STA \$64	
B95F	A5 65	LDA \$65	
B961	49 FF	EOR #\$FF	
B963	85 65	STA \$65	
B965	A5 70	LDA \$70	
B967	49 FF	EOR #\$FF	
B969	85 70	STA \$70	
B96B	E6 70	INC \$70	
B96D	D0 0E	BNE \$B97D	
B96F	E6 65	INC \$65	
B971	D0 0A	BNE \$B97D	
B973	E6 64	INC \$64	
B975	D0 06	BNE \$B97D	; OVERFLOWS IN ACHT NEMEN
B977	E6 63	INC \$63	
B979	D0 02	BNE \$B97D	
B97B	E6 62	INC \$62	
B97D	60	RTS	
B97E	A2 0F	LDX #\$0F	; NUMMER VOOR OVERFLOW
B980	4C 37 A4	JMP \$A437	; FOUTMELDING UITVOEREN
*****			REGISTER NAAR RECHTS
B983	A2 25	LDX #\$25	; OFFSET-WIJZER OP REGISTER
B985	B4 04	LDY \$04,X	
B987	84 70	STY \$70	
B989	B4 03	LDY \$03,X	
B98B	94 04	STY \$04,X	
B98D	B4 02	LDY \$02,X	
B98F	94 03	STY \$03,X	
B991	B4 01	LDY \$01,X	
B993	94 02	STY \$02,X	
B995	A4 68	LDY \$68	
B997	94 01	STY \$01,X	
B999	69 08	ADC #\$08	
B99B	30 E8	BMI \$B985	
B99D	F0 E6	BEQ \$B985	
B99F	E9 08	SBC #\$08	
B9A1	A8	TAY	
B9A2	A5 70	LDA \$70	

```

B9A4  B0 14      BCS $B9BA
B9A6  16 01      ASL $01,X
B9A8  90 02      BCC $B9AC
B9AA  F6 01      INC $01,X
B9AC  76 01      ROR $01,X
B9AE  76 01      ROR $01,X
B9B0  76 02      ROR $02,X
B9B2  76 03      ROR $03,X
B9B4  76 04      ROR $04,X
B9B6  6A         ROR
B9B7  C8         INY
B9B8  DO EC      BNE $B9A6
B9BA  18         CLC
B9BB  60         RTS

```

*****CONSTANTEN VOOR LOG

```

B9BC  81 00 00 00 00      1
B9C1  03                  POLYNOOMGRAAD=3,
;                          DAN 4 COEFFICIENTEN
B9C2  7F 5E 56 CB 79      .434255942
B9C7  80 13 9B 0B 64      .576584541
B9CC  80 76 38 93 16      .961800759
B9D1  82 38 AA 3B 20      2.88539007
B9D5  80 35 04 F3 34      .707106781=1/SQR(2)
B9DB  81 35 04 F3 34      1.41421356=SQR(2)
B9E0  80 80 00 00 00      -.5
B9E5  80 31 72 17 F8      .693147181=LOG(2)

```

*****BASIC-FUNCTIE LOG

```

B9EA  20 2B BC JSR $BC2B    ; VOORTEKEN HALEN
B9ED  FO 02   BEQ $B9F1    ; NUL, DAN KLAAR
B9EF  10 03   BPL $B9F4    ; POSITIEF, DAN OK
B9F1  4C 48 B2 JMP $B248    ; ILLEGAL QUANTITY
B9F4  A5 61   LDA $61      ; EXPONENT
B9F6  E9 7F   SBC #$7F     ; NORMALISEREN
B9F8  48      PHA          ; EN ONTHOUDEN
B9F9  A9 80   LDA #$80     ; GETAL TUSSEN 0.5 EN 1
B9FB  85 61   STA $61      ; BRENGEN
B9FD  A9 D6   LDA #$D6
B9FF  A0 B9   LDY #$B9     ; WIJZER OP CONSTATE 1/SQR(2)
BA00  20 67 B8 JSR $B876    ; BIJ FAC OPTELLEN
BA04  A9 DB   LDA #$DB
BA06  A0 B9   LDY #$B9     ; WIJZER OP CONSTATE SQR(2)
BA08  20 0F BB JSR $BBOF    ; SQR(2) DOOR FAC DELEN
BA0B  A9 BC   LDA #$BC
BA0D  A0 B9   LDY #$B9     ; WIJZER OP CONSTATE 1
BA0F  20 50 B8 JSR $B850    ; 1 MINUS FAC
BA12  A9 C1   LDA #$C1
BA14  A0 B9   LDY #$B9     ; WIJZER OP POLYNOOMCOEFFICIENT
BA16  20 43 EO JSR $EO43    ; POLYNOOMBEREKENING
BA19  A9 EO   LDA #$EO

```

```

BA1B  A0 B9    LDY  #$B9      ; WIJZER OP CONSTATE -0.5
BA1D  20 67 B8 JSR  $B867     ; BIJ FAC OPTELLEN
BA20   68      PLA            ; EXPONENT TERUGHALEN
BA21  20 7E BD JSR  $BD7E     ; FAC=FAC+FAC
BA24  A9 E5    LDA  #$E5
BA26  A0 B9    LDY  #$B9      ; WIJZER OP CONSTATE LOG(2)
***** FAC=CONSTANTE (A/Y)*FAC
BA28  20 8C BA JSR  $BA8C     ; CONSTATE NAAR ARG
***** FAC=ARG*FAC
BA2B  D0 03    BNE  $BA30     ; ONGELIJK NUL?
BA2D  4C 8B BA JMP  $BA8B     ; RTS
BA30  20 B7 BA JSR  $BAB7     ; EXPONENT BEREKENEN
BA33  A9 00    LDA  #$00
BA35  85 26    STA  $26
BA37  85 27    STA  $27
BA39  85 28    STA  $28      ; FUNCTIEREGISTER WISSEN
BA3B  85 29    STA  $29
BA3D  A5 70    LDA  $70
BA3F  20 59 BA JSR  $BA59     ; BIT VOOR BIT VERMENIGVULDIGEN
BA42  A5 65    LDA  $65
BA44  20 59 BA JSR  $BA59     ; BIT VOOR BIT VERMENIGVULDIGEN
BA47  A5 64    LDA  $64
BA49  20 59 BA JSR  $BA59     ; BIT VOOT BIT VERMENIGVULDIGEN
BA4C  A5 63    LDA  $63
BA4E  20 59 BA JSR  $BA59     ; BIT VOOR BIT VERMENIGVULDIGEN
BA51  A5 62    LDA  $62
BA53  20 5E BA JSR  $BA5E     ; BIT VOOR BIT VERMENIGVULDIGEN
BA56  4C 8F BB JMP  $BB8F     ; REGISTER NAAR FAC, LINKERKANTLIJN
; RECHT MAKEN
***** BIT VOOR BIT VERMENIGVULDIGEN
BA59  D0 03    BNE  $BA5E
BA5B  4C 83 B9 JMP  $B983     ; REGISTER NAAR RECHTS VERSCHUIVEN
BA5E   4A      LSR
BA5F  09 80    ORA  #$80
BA61  A8      TAY
BA62  90 19    BCC  $BA7D
BA64   18      CLC
BA65  A5 29    LDA  $29
BA67  65 6D    ADC  $6D
BA69  85 29    STA  $29
BA6B  A5 28    LDA  $28
BA6D  65 6C    ADC  $6C
BA6F  85 28    STA  $28
BA71  A5 27    LDA  $27
BA73  65 6B    ADC  $6B
BA75  85 27    STA  $27
BA77  A5 26    LDA  $26
BA79  65 6A    ADC  $6A
BA7B  85 26    STA  $26

```

BA7D	66	26	ROR	\$26	
BA7F	66	27	ROR	\$27	
BA81	66	28	ROR	\$28	
BA83	66	29	ROR	\$29	
BA85	66	70	ROR	\$70	
BA87	98		TYA		
BA88	4A		LSR		
BA89	D0	D6	BNE	\$BA61	
BA8B	60		RTS		
***** ARG=CONSTANTE (A/Y)					
BA8C	85	22	STA	\$22	
BA8E	84	23	STY	\$23	; WIJZER
BA90	A0	04	LDY	#\$04	
BA92	B1	22	LDA	(\$22),Y	
BA94	85	6D	STA	\$6D	
BA96	88		DEY		; CONSTANTE NAAR ARG
BA97	B1	22	LDA	(\$22),Y	
BA99	85	6C	STA	\$6C	
BA9B	88		DEY		
BA9C	B1	22	LDA	(\$22),Y	
BA9E	85	6B	STA	\$6B	
BAA0	88		DEY		
BAA1	B1	22	LDA	(\$22),Y	
BAA3	85	6E	STA	\$6E	
BAA5	45	66	EOR	\$66	
BAA7	85	6F	STA	\$6F	
BAA9	A5	6E	LDA	\$6E	
BAAB	09	80	ORA	#\$80	; VOORTEKEN
BAAD	85	6A	STA	\$6A	
BAAF	88		DEY		
BAB0	B1	22	LDA	(\$22),Y	
BAB2	85	69	STA	\$69	; EXPONENT
BAB4	A5	61	LDA	\$61	
BAB6	60		RTS		
BAB7	A5	69	LDA	\$69	
BAB9	F0	1F	BEQ	\$BADA	
BABB	18		CLC		
BABC	65	61	ADC	\$61	
BABE	90	04	BCC	\$BAC4	
BACO	30	1D	BMI	\$BADF	
BAC2	18		CLC		
BAC3	2C	10 14	BIT	\$1410	
BAC6	69	80	ADC	#\$80	
BAC8	85	61	STA	\$61	
BACA	D0	03	BNE	\$BACF	
BACC	4C	FB B8	JMP	\$B8FB	; FAC=0
BACF	A5	6F	LDA	\$6F	
BAD1	85	66	STA	\$66	
BAD3	60		RTS		

```

BAD4  A5 66      LDA $66
BAD6  49 FF      EOR #$FF
BAD8  30 05      BMI $BADF
BADA  68         PLA
BADB  68         PLA
BADC  4C F7 B8   JMP $B8F7      ; FAC=0
BADF  4C 7E B9   JMP $B97E      ; OVERFLOW ERROR
***** FAC=FAC*10
BAE2  20 0C BC   JSR $BCOC      ; FAC AFRONDEN EN NAAR ARG
BAE5  AA         TAX
BAE6  F0 10      BEQ $BAF8      ; FAC=0, DAN KLAAR
BAE8  18         CLC
BAE9  69 02      ADC #$02      ; EXP+2 KOMT OVEREEN MET *4
BAEB  B0 F2      BCS $BADF      ; OVERFLOW
BAED  A2 00      LDX #$00
BAEF  86 6F      STX $6F
BAF1  20 77 B8   JSR $B877      ; FAC=FAC+ARG KOMT OVEREEN MET *5
BAF4  E6 61      INC $61        ; EXPONENT VERHOGEN KOMT OVEREEN
;                                     MET *2
BAF6  F0 E7      BEQ $BADF      ; OVERFLOW
BAF8  60         RTS
*****
BAF9  84 20 00 00 00      ; CONSTATE: REELE GETAL 10
***** FAC=FAC/10
BAFE  20 0C BC   JSR $BCOC      ; FAC AFRONDEN EN NAAR ARG
BB01  A9 F9      LDA $F9
BB03  A0 BA      LDY #$BA      ; WIJZER OP CONSTATE 10
BB05  A2 00      LDX #$00
BB07  86 6F      STX $6F
BB09  20 A2 BB   JSR $BBA2      ; CONSTATE 10 NAAR FAC
BB0C  4C 12 BB   JMP $BB12      ; FAC=ARG/FAC
***** FAC=CONSTATE (A/Y)/FAC
BBOF  20 8C BA   JSR $BA8C      ; CONSTATE (A/Y) NAAR ARG
***** FAC=ARG/FAC
BB12  F0 76      BEQ $BB8A      ; FAC=0, DAN DIVISION BY ZERO
BB14  20 1B BC   JSR $BC1B      ; FAC AFRONDEN
BB17  A9 00      LDA #$00
BB19  38         SEC
BB1A  E5 61      SBC $61
BB1C  85 61      STA $61
BB1E  20 B7 BA   JSR $BAB7      ; EXP VAN RESULTAAT BEPALEN
BB21  E6 61      INC $61
BB23  F0 BA      BEQ $BADF      ; OVERFLOW VAN EXPONENT
BB25  A2 FC      LDX #$FC      ; WIJZER OP FUNCTIEREGISTER
BB27  A9 01      LDA #$01
BB29  A4 6A      LDY $6A
BB2B  C4 62      CPY $62

```

BB2D	DO 10	BNE \$BB3F	
BB2F	A4 6B	LDY \$6B	
BB31	C4 63	CPY \$63	; ARG BIT VOOR BIT MET FAC
			VERGELIJKEN
BB33	DO 0A	BNE \$BB3F	
BB35	A4 6C	LDY \$6C	
BB37	C4 64	CPY \$64	
BB39	DO 04	BNE \$BB3F	
BB3B	A4 6D	LDY \$6D	
BB3D	C4 65	CPY \$65	
BB3F	08	PHP	; STATUS WEGZETTEN
BB40	2A	ROL	
BB41	90 09	BCC \$BB4C	
BB43	E8	INX	
BB44	95 29	STA \$29,X	
BB46	F0 32	BEQ \$BB7A	
BB48	10 34	BPL \$BB7E	
BB4A	A9 01	LDA #\$01	
BB4C	28	PLP	
BB4D	B0 0E	BCS \$BB5D	
BB4F	06 6D	ASL \$6D	
BB51	26 6C	ROL \$6C	
BB53	26 6B	ROL \$6B	
BB55	26 6A	ROL \$6A	
BB57	B0 E6	BCS \$BB3F	
BB59	30 CE	BMI \$BB29	
BB5B	10 E2	BPL \$BB3F	
BB5D	A8	TAY	
BB5E	A5 6D	LDA \$6D	
BB60	E5 65	SBC \$65	
BB62	85 6D	STA \$6D	
BB64	A5 6C	LDA \$6C	
BB66	E5 64	SBC \$64	
BB68	85 6C	STA \$6C	
BB6A	A5 6B	LDA \$6B	
BB6C	E5 63	SBC \$63	
BB6E	85 6B	STA \$6B	
BB70	A5 6A	LDA \$6A	
BB72	E5 62	SBC \$62	
BB74	85 6A	STA \$6A	
BB76	98	TYA	
BB77	4C 4F BB	JMP \$BB4F	
BB7A	A9 40	LDA #\$40	
BB7C	DO CE	BNE \$BB4C	
BB7E	0A	ASL	
BB7F	0A	ASL	
BB80	0A	ASL	; ACCU*64
BB81	0A	ASL	
BB82	0A	ASL	

```

BB83 0A      ASL
BB84 85 70   STA $70
BB86 28      PLP
BB87 4C 8F BB JMP $BB8F      ; HULPREGISTER NAAR FAC
*****
BB8A A2 14   LDX #$14      ; NUMMER VOOR DIVISION BY ZERO
BB8C 4C 37 A4 JMP $A437    ; FOUTMELDING UITVOEREN
***** HULPREGISTER ($26-$29)
; NAAR FAC
BB8F A5 26   LDA $26
BB91 85 62   STA $62
BB93 A5 27   LDA $27
BB95 85 63   STA $63
BB97 A5 28   LDA $28
BB99 85 64   STA $64
BB9B A5 29   LDA $29
BB9D 85 65   STA $65
BB9F 4C D7 B8 JMP $B8D7    ; LINKERKANTLIJN VAN FAC
; RECHT MAKEN
***** CONSTATE (A/Y) NAAR FAC
BBA2 85 22   STA $22
BBA4 84 23   STY $23      ; WIJZER ZETTEN
BBA6 A0 04   LDY #$04
BBA8 B1 22   LDA ($22),Y
BBAA 85 65   STA $65
BBAC 88      DEY          ; MANTISSE
BBAD B1 22   LDA ($22),Y
BBAF 85 64   STA $64
BBB1 88      DEY
BBB2 B1 22   LDA ($22),Y
BBB4 85 63   STA $63
BBB6 88      DEY
BBB7 B1 22   LDA ($22),Y
BBB9 85 66   STA $66
BBBB 09 80   ORA #$80      ; VOORTEKEN MANTISSE
BBBD 85 62   STA $62
BBBF 88      DEY
BBC0 B1 22   LDA ($22),Y
BBC2 85 61   STA $61      ; EXPONENT
BBC4 84 70   STY $70
BBC6 60      RTS
***** FAC NAAR ACCU#4
BBC7 A2 5C   LDX #$5C      ; ADRES LOW ACCU#4
BBC9 2C      .BYTE $2C
***** FAC NAAR ACCU#3
BBCA A2 57   LDX #$57      ; ADRES LOW ACCU#3
BBCC A0 00   LDY $00      ; ADRES HIGH
BBCE FO 04   BEQ $BBD4    ; ABSOLUTE SPRONG
***** FAC NAAR VARIABLE

```

```

BBDO A6 49 LDX $49
BBD2 A4 4A LDY $4A ; ADRES VAN VARIABLE
BBD4 20 1B BC JSR $BC1B ; FAC AFRONDEN
BBD7 86 22 STX $22
BBD9 84 23 STY $23 ; WIJZER OP DOELADRES
BBDB A0 04 LDY #$04
BBDD A5 65 LDA $65
BBDF 91 22 STA ($22),Y
BBE1 88 DEY
BBE2 A5 64 LDA $64
BBE4 91 22 STA ($22),Y
BBE6 88 DEY
BBE7 A5 63 LDA $63
BBE9 91 22 STA ($22),Y
BBEB 88 DEY
BBEC A5 66 LDA $66
BBEE 09 7F ORA #$7F
BBFO 25 62 AND $62 ; VOORTEKEN OP GEHEUGENFORMAAT
; BRENGEN
BBF2 91 22 STA ($22),Y
BBF4 88 DEY
BBF5 A5 61 LDA $61
BBF7 91 22 STA ($22),Y
BBF9 84 70 STY $70
BBFB 60 RTS
***** ARG NAAR FAC
BBFC A5 6E LDA $6E
BBFE 85 66 STA $66

BC00 A2 05 LDX #$05 ; 5 BYTES
BC02 B5 68 LDA $68,X
BC04 95 60 STA $60,X
BC06 CA DEX
BC07 D0 F9 BNE $BC02
BC09 86 70 STX $70
BC0B 60 RTS
***** FAC NAAR ARG
BC0C 20 1B BC JSR $BC1B ; FAC AFRONDEN
BC0F A2 06 LDX #$06
BC11 B5 60 LDA $60,X
BC13 95 68 STA $68,X
BC15 CA DEX
BC16 D0 F9 BNE $BC11
BC18 86 70 STX $70
BC1A 60 RTS
***** FAC AFRONDEN
BC1B A5 61 LDA $61 ; EXP=0, DAN KLAAR
BC1D F0 FB BEQ $BC1A
BC1F 06 70 ASL $70 ; AFRONDINGSPLAATS > $7F

```



```

BC21  90 F7      BCC $BC1A      ; NEE, DAN KLAAR
BC23  20 6F B9 JSR $B96F      ; MANTISSE MET 1 VERHOGEN
BC26  D0 F2      BNE $BC1A      ; NU NUL?
BC28  4C 38 B9 JMP $B938      ; NAAR RECHTS VERSCHUIVEN,
;                               EXPONENT VERHOGEN
***** VOORTEKEN VAN FAC HALEN
BC2B  A5 61      LDA $61        ; NUL?
BC2D  F0 09      BEQ $BC38
BC2F  A5 66      LDA $66
BC31  2A         ROL
BC32  A9 FF      LDA #$FF      ; NEGATIEF
BC34  B0 02      BCS $BC38
BC36  A9 01      LDA #$01      ; POSITIEF
BC38  60         RTS
***** BASIC-FUNCTIE SGN
BC39  20 2B BC JSR $BC2B      ; VOORTEKEN HALEN
BC3C  85 62      STA $62
BC3E  A9 00      LDA #$00
BC40  85 63      STA $63
BC42  A2 88      LDX #$88
BC44  A5 62      LDA $62
BC46  49 FF      EOR #$FF
BC48  2A         ROL
BC49  A9 00      LDA #$00
BC4B  85 65      STA $65
BC4D  85 64      STA $64
BC4F  86 61      STX $61
BC51  85 70      STA $70
BC53  85 66      STA $66
BC55  4C D2 B8 JMP $B8D2
***** BASIC-FUNCTIE ABS
BC58  46 66      LSR $66      ; VOORTEKENBIT WISSEN
BC5A  60         RTS
***** CONSTATE (A/Y) MET FAC
;                               VERGELIJKEN
BC5B  85 24      STA $24
BC5D  84 25      STY $25      ; WIJZER OP CONSTATE
BC5F  A0 00      LDY #$00
BC61  B1 24      LDA ($24),Y  ; EXPONENT
BC63  C8         INY
BC64  AA         TAX
BC65  F0 C4      BEQ $BC2B      ; 0, VOORTEKEN VAN FAC HALEN
BC67  B1 24      LDA ($24),Y
BC69  45 66      EOR $66
BC6B  30 C2      BMI $BC2F      ; VERSCHILLENDE VOORTEKENS
BC6D  E4 61      CPX $61
BC6F  D0 21      BNE $BC92
BC71  B1 24      LDA ($24),Y  ; 1e BYTE VERGELIJKEN
BC73  09 80      ORA #$80

```

```

BC75 C5 62    CMP $62
BC77 D0 19    BNE $BC92
BC79 C8       INY
BC7A B1 24    LDA ($24),Y ; 2e BYTE VERGELIJKEN
BC7C C5 63    CMP $63
BC7E D0 12    BNE $BC92
BC80 C8       INY
BC81 B1 24    LDA ($24),Y ; 3e BYTE VERGELIJKEN
BC83 C5 64    CMP $64
BC85 D0 0B    BNE $BC92
BC87 C8       INY
BC88 A9 7F    LDA #$7F
BC8A C5 70    CMP $70
BC8C B1 24    LDA ($24),Y ; 4e BYTE VERGELIJKEN
BC8E E5 65    SBC $65
BC90 F0 28    BEQ $BCBA
BC92 A5 66    LDA $66
BC94 90 02    BCC $BC98
BC96 49 FF    EOR #$FF ; UITKOMST KLEINER,
; DAN INVERTEREN
BC98 4C 31 BC JMP $BC31 ; FLAG VOOR UITKOMST ZETTEN
***** REEEL VERANDEREN IN INTEGER *****
BC9B A5 61    LDA $61 ; EXPONENT
BC9D F0 4A    BEQ $BCE9 ; NUL?
BC9F 38       SEC
BCA0 E9 A0    SBC #$A0
BCA2 24 66    BIT $66
BCA4 10 09    BPL $BCAF
BCA6 AA       TAX
BCA7 A9 FF    LDA #$FF
BCA9 85 68    STA $68
BCAB 20 4D B9 JSR $B94D ; MANTISSE VAN FAC INVERTEREN
BCAE 8A       TXA
BCAF A2 61    LDX #$61
BCB1 C9 F9    CMP #$F9
BCB3 10 06    BPL $BCBB
BCB5 20 99 B9 JSR $B999 ; FAC NAAR RECHTS VERSCHUIVEN
BCB8 84 68    STY $68
BCBA 60       RTS
BCBB A8       TAY
BCBC A5 66    LDA $66
BCBE 29 80    AND #$80
BCC0 46 62    LSR $62
BCC2 05 62    ORA $62
BCC4 85 62    STA $62
BCC6 20 B0 B9 JSR $B9B0 ; FAC BIT VOOR BIT NAAR RECHTS
; VERSCHUIVEN
;
BCC9 84 68    STY $68
BCCB 60       RTS

```

```

***** BASIC-FUNCTIE INT
BCCC A5 61 LDA $61 ; EXPONENT
BCCE C9 A0 CMP #$A0 ; GEHEEL GETAL?
BCD0 B0 20 BCS $BCF2 ; JA, DAN KLAAR
BCD2 20 9B BC JSR $BC9B ; FAC IN INTEGER VERANDEREN
BCD5 84 70 STY $70
BCD7 A5 66 LDA $66
BCD9 84 66 STY $66
BCDB 49 80 EOR #$80
BCDD 2A ROL
BCDE A9 A0 LDA #$A0
BCE0 85 61 STA $61
BCE2 A5 65 LDA $65
BCE4 85 07 STA $07
BCE6 4C D2 B8 JMP $B8D2 ; LINKERKANTLIJN VAN FAC RECHT
BCE9 85 62 STA $62 ; MANTISSE MET NULLEN VULLEN
BCEB 85 63 STA $63
BCED 85 64 STA $64
BCEF 85 65 STA $65
BCF1 A8 TAY
BCF2 60 RTS
***** ASCII IN REEEL VERANDEREN
BCF3 A0 00 LDY #$00
BCF5 A2 0A LDX #$0A ; BEREIK $5D TOT $66 WISSEN
BCF7 94 5D STY $5D,X
BCF9 CA DEX
BCFA 10 FB BPL $BCF7
BCFC 90 0F BCC $BD0D
BCFE C9 2D CMP #$2D ; VERBINDINGSSTREEPJE
BD00 D0 04 BNE $BD06
BD02 86 67 STX $67 ; FLAG VOOR NEGATIEF
BD04 F0 04 BEQ $BDOA
BD06 C9 2B CMP #$2B ; PLUS
BD08 D0 05 BNE $BD0F
BD0A 20 73 00 JSR $0073 ; CHRGET VOLGEND TEKEN HALEN
BD0D 90 5B BCC $BD6A
BD0F C9 2E CMP #$2E ; PUNT
BD11 F0 2E BEQ $BD41
BD13 C9 45 CMP #$45 ; E
BD15 D0 30 BNE $BD47
BD17 20 73 00 JSR $0073 ; CHRGET: VOLGEND TEKEN HALEN
BD1A 90 17 BCC $BD33
BD1C C9 AB CMP #$AB ; BASIC-CODE VOOR
; VERBINDINGSSTREEPJE
BD1E F0 0E BEQ $BD2E
BD20 C9 2D CMP #$2D ; VERBINDINGSSTREEPJE
BD22 F0 0A BEQ $BD2E
BD24 C9 AA CMP #$AA ; BASIC-CODE VOOR PLUS
BD26 F0 08 BEQ $BD30

```

BD28	C9 2B	CMP #\$2B	; PLUS
BD2A	F0 04	BEQ \$BD30	
BD2C	D0 07	BNE \$BD35	
BD2E	66 60	ROR \$60	; BIT 7 ZETTEN
BD30	20 73 00	JSR \$0073	; CHRGET: VOLGEND TEKEN HALEN
BD33	90 5C	BCC \$BD91	
BD35	24 60	BIT \$60	; BIT 7 GEZET?
BD37	10 0E	BPL \$BD47	; NEE
BD39	A9 00	LDA #\$00	
BD3B	38	SEC	
BD3C	E5 5E	SBC \$5E	
BD3E	4C 49 BD	JMP \$BD49	
BD41	66 5F	ROR \$5F	; OPROEP DOOR DECIMALE PUNT
BD43	24 5F	BIT \$5F	
BD45	50 C3	BVC \$BDOA	
BD47	A5 5E	LDA \$5E	
BD49	38	SEC	
BD4A	E5 5D	SBC \$5D	
BD4C	85 5E	STA \$5E	
BD4E	F0 12	BEQ \$BD62	
BD50	10 09	BPL \$BD5B	
BD52	20 FE BA	JSR \$BAFE	; FAC=FAC/10
BD55	E6 5E	INC \$5E	
BD57	D0 F9	BNE \$BD52	
BD59	F0 07	BEQ \$BD62	
BD5B	20 E2 BA	JSR \$BAE2	; FAC=FAC*10
BD5E	C6 5E	DEC \$5E	
BD60	D0 F9	BNE \$BD5B	
BD62	A5 67	LDA \$67	
BD64	30 01	BMI \$BD67	
BD66	60	RTS	
BD67	4C B4 BF	JMP \$BFB4	; WISSELING VAN VOORTEKEN: FAC=-FAC
BD6A	48	PHA	
BD6B	24 5F	BIT \$5F	
BD6D	10 02	BPL \$BD71	
BD6F	E6 5D	INC \$5D	
BD71	20 E2 BA	JSR \$BAE2	; FAC=FAC*10
BD74	68	PLA	
BD75	38	SEC	
BD76	E9 30	SBC #\$30	; 0 AFTREKKEN LEVERT HEX OP
BD78	20 7E BD	JSR \$BD7E	; TELT VOLGENDE PLAATS
			BIJ FAC OP
			; VOLGEND TEKEN
BD7B	4C 0A BD	JMP \$BDOA	
BD7E	48	PHA	
BD7F	20 0C BC	JSR \$BCOC	; FAC NAAR ARG
BD82	68	PLA	
BD83	20 3C BC	JSR \$BC3C	
BD86	A5 6E	LDA \$6E	
BD88	45 66	EOR \$66	

```

BD8A 85 6F STA $6F
BD8C A6 61 LDX $61
BD8E 4C 6A B8 JMP $B86A ; FAC=FAC+ARG
BD91 A5 5E LDA $5E ; OPROEP DOOR E
BD93 C9 0A CMP #$0A
BD95 90 09 BCC $BDA0
BD97 A9 64 LDA #$64
BD99 24 60 BIT $60
BD9B 30 11 BMI $BDAE
BD9D 4C 7E B9 JMP $B97E ; OVERFLOW ERROR
BDA0 0A ASL
BDA1 0A ASL
BDA2 18 CLC
BDA3 65 5E ADC $5E
BDA5 0A ASL
BDA6 18 CLC
BDA7 A0 00 LDY #$00
BDA9 71 7A ADC ($7A),Y
BDAB 38 SEC
BDAC E9 30 SBC #$30 ; 0
BDAE 85 5E STA $5E
BDB0 4C 30 BD JMP $BD30 ; VOLGEND TEKEN HALEN
*****CONSTANTE VOOR REEEL NAAR ASCII
BDB3 9B 3E BC 1F FD ; 99999999.9
BDB8 9E 6E 6B 27 FD ; 999999999
BDBD 9E 6E 6B 28 00 ; 1E9
*****UITVOER VAN REGELNUMMER
; BIJ FOUTMELDING
BDC2 A9 71 LDA #$71
BDC4 A0 A3 LDY #$A3 ; WIJZER OP IN
BDC6 20 DA BD JSR $BDDA ; STRING UITVOEREN
BDC9 A5 3A LDA $3A
BDCB A6 39 LDX $39 ; ACTUELE REGELNUMMER HALEN
*****POSITIEF INTEGER GETAL IN A/X
; UITVOEREN
BDCD 85 62 STA $62
BDCF 86 63 STX $63 ; VOOR WIJZIGING IN FAC ZETTEN
BDD1 A2 90 LDX #$90
BDD3 38 SEC
BDD4 20 49 BC JSR $BC49 ; INTEGER IN REEEL VERANDEREN
BDD7 20 DF BD JSR $BDDF ; FAC IN ASCII VERANDEREN
BDDA 4C 1E AB JMP $AB1E ; STRING UITVOEREN
*****FAC IN ASCII-FORMAAT
; VERANDEREN EN NAAR $100
BDDD A0 01 LDY #$01
BDDF A9 20 LDA #$20 ; SPATIE VOOR POSITIEF GETAL
BDE1 24 66 BIT $66 ; VOORTEKEN
BDE3 10 02 BPL $BDE7 ; POSITIEF?
BDE5 A9 2D LDA #$2D ; MINTEKEN VOOR NEGATIEF GETAL

```

BDE7	99	FF	OO	STA	\$00FF,Y	; IN BUFFERBEREIK SCHRIJVEN
BDEA	85	66		STA	\$66	
BDEC	84	71		STY	\$71	
BDEE	C8			INY		
BDEF	A9	30		LDA	#\$30	; 0
BDF1	A6	61		LDX	\$61	; EXPONENENT
BDF3	D0	03		BNE	\$BDF8	; GETAL ONGELIJK NUL?
BDF5	4C	04	BF	JMP	\$BF04	; JA, KLAAR
BDF8	A9	00		LDA	#\$00	
BDFa	E0	80		CPX	#\$80	; FAC MET 1 VERGELIJKEN
BDFC	F0	02		BEQ	\$BE00	
BDFE	B0	09		BCS	\$BE09	; FAC GROTER DAN 1

BE00	A9	BD		LDA	#\$BD	
BE02	A0	BD		LDY	#\$BD	; WIJZER OP CONSTATE 1E9
BE04	20	28	BA	JSR	\$BA28	; CONSTATE(WIJZER A/Y)*FAC
BE07	A9	F7		LDA	#\$F7	
BE09	85	5D		STA	\$5D	
BE0B	A9	B8		LDA	#\$B8	
BE0D	A0	BD		LDY	#\$BD	; WIJZER OP CONSTATE 999999999
BE0F	20	5B	BC	JSR	\$BC5B	; CONSTATE(WIJZER A/Y)=FAC?
BE12	F0	1E		BEQ	\$BE32	; GELIJK
BE14	10	12		BPL	\$BE28	
BE16	A9	B3		LDA	#\$B3	
BE18	A0	BD		LDY	#\$BD	; WIJZER OP CONSTATE 99999999.9
BE1A	20	5B	BC	JSR	\$BC5B	; CONSTATE(WIJZER A/Y)=FAC?
BE1D	F0	02		BEQ	\$BE21	
BE1F	10	0E		BPL	\$BE2F	
BE21	20	E2	BA	JSR	\$BAE2	; FAC=FAC*10
BE24	C6	5D		DEC	\$5D	
BE26	D0	EE		BNE	\$BE16	
BE28	20	FE	BA	JSR	\$BAFE	; FAC=FAC/10
BE2B	E6	5D		INC	\$5D	
BE2D	D0	DC		BNE	\$BE0B	
BE2F	20	49	B8	JSR	\$B849	; FAC=FAC+0.5, AFRONDEN
BE32	20	9B	BC	JSR	\$BC9B	; FAC NAAR INTEGER
BE35	A2	01		LDX	#\$01	
BE37	A5	5D		LDA	\$5D	
BE39	18			CLC		
BE3A	69	0A		ADC	#\$0A	
BE3C	30	09		BMI	\$BE47	; GETAL < 0.1?
BE3E	C9	0B		CMP	#\$0B	
BE40	B0	06		BCS	\$BE48	; GETAL > 1E9
BE42	69	FF		ADC	#\$FF	
BE44	AA			TAX		
BE45	A9	02		LDA	#\$02	
BE47	38			SEC		

BE48	E9	02	SBC	#\$02	
BE4A	85	5E	STA	\$5E	
BE4C	86	5D	STX	\$5D	
BE4E	8A		TXA		
BE4F	F0	02	BEQ	\$BE53	
BE51	10	13	BPL	\$BE66	
BE53	A4	71	LDY	\$71	
BE55	A9	2E	LDA	#\$2E	; PUNT
BE57	C8		INY		
BE58	99	FF 00	STA	\$00FF,Y	
BE5B	8A		TXA		
BE5C	F0	06	BEQ	\$BE64	
BE5E	A9	30	LDA	#\$30	; NUL
BE60	C8		INY		
BE61	99	FF 00	STA	\$00FF,Y	
BE64	84	71	STY	\$71	
BE66	A0	00	LDY	#\$00	; BEREKENING VAN AFZONDERLIJKE CIJFERS
BE68	A2	80	LDX	#\$80	
BE6A	A5	65	LDA	\$65	
BE6C	18		CLC		
BE6D	79	19 BF	ADC	\$BF19,Y	
BE70	85	65	STA	\$65	
BE72	A5	64	LDA	\$64	
BE74	79	18 BF	ADC	\$BF18,Y	
BE77	85	64	STA	\$64	
BE79	A5	63	LDA	\$63	
BE7B	79	17 BF	ADC	\$BF17,Y	
BE7E	85	63	STA	\$63	
BE80	A5	62	LDA	\$62	
BE82	79	16 BF	ADC	\$BF16,Y	
BE85	85	62	STA	\$62	
BE87	E8		INX		
BE88	B0	04	BCS	\$BE8E	
BE8A	10	DE	BPL	\$BE6A	
BE8C	30	02	BMI	\$BE90	
BE8E	30	DA	BMI	\$BE6A	
BE90	8A		TXA		
BE91	90	04	BCC	\$BE97	
BE93	49	FF	EOR	#\$FF	
BE95	69	0A	ADC	#\$0A	; 10
BE97	69	2F	ADC	#\$2F	; 0-1
BE99	C8		INY		
BE9A	C8		INY		
BE9B	C8		INY		
BE9C	C8		INY		
BE9D	84	47	STY	\$47	
BE9F	A4	71	LDY	\$71	
BEA1	C8		INY		

BEA2	AA		TAX	
BEA3	29	7F	AND	#\$7F
BEA5	99	FF 00	STA	\$00FF,Y
BEA8	C6	5D	DEC	\$5D
BEAA	DO	06	BNE	\$BEB2
BEAC	A9	2E	LDA	#\$2E ; PUNT
BEAE	C8		INY	
BEAF	99	FF 00	STA	\$00FF,Y
BEB2	84	71	STY	\$71
BEB4	A4	47	LDY	\$47
BEB6	8A		TXA	
BEB7	49	FF	EOR	#\$FF
BEB9	29	80	AND	#\$80
BEBB	AA		TAX	
BEEC	CO	24	CPY	#\$24 ; EINDE VAN TABEL
				BIJ VERANDEREN FAC
;				
BEBE	FO	04	BEQ	\$BEC4
BECO	CO	3C	CPY	#\$3C ; EINDE VAN TABEL
				BIJ BEREKENING TI\$
;				
BEC2	DO	A6	BNE	\$BE6A
BEC4	A4	71	LDY	\$71
BEC6	B9	FF 00	LDA	\$00FF,Y
BEC9	88		DEY	
BECA	C9	30	CMP	#\$30 ; 0
BECC	FO	F8	BEQ	\$BEC6
BECE	C9	2E	CMP	#\$2E ; PUNT
BEDO	FO	01	BEQ	\$BED3
BED2	C8		INY	
BED3	A9	2B	LDA	#\$2B ; PLUS
BED5	A6	5E	LDX	\$5E
BED7	FO	2E	BEQ	\$BF07
BED9	10	08	BPL	\$BEE3
BEDB	A9	00	LDA	#\$00
BEDD	38		SEC	
BEDE	E5	5E	SBC	\$5E
BEE0	AA		TAX	
BEE1	A9	2D	LDA	#\$2D ; MIN
BEE3	99	01 01	STA	\$0101,Y
BEE6	A9	45	LDA	#\$45 ; E
BEE8	99	00 01	STA	\$0100,Y
BEEB	8A		TXA	
BEEC	A2	2F	LDX	#\$2F ; 0-1
BEEE	38		SEC	
BEEF	E8		INX	
BEFO	E9	0A	SBC	#\$0A ; 10
BEF2	BO	FB	BCS	\$BEEF
BEF4	69	3A	ADC	#\$3A ; G+1
BEF6	99	03 01	STA	\$0103,Y
BEF9	8A		TXA	


```

BEFA 99 02 01 STA $0102,Y
BEFD A9 00 LDA #$00 ; BUFFER MET $0 AFSLUITEN
BEFF 99 04 01 STA $0104,Y
BF02 F0 08 BEQ $BFOC
BF04 99 FF 00 STA $00FF,Y
BF07 A9 00 LDA #$00 ; BUFFER MET $0 AFSLUITEN
BF09 99 00 01 STA $0100,Y
BFOC A9 00 LDA #$00 ; WIJZER OP BUFFER $100
BFOE A0 01 LDY #$01
BF10 60 RTS
*****
BF11 80 00 00 00 00 ; CONSTATE 0.5 VOOR SQR-FUNCT.
*****CONSTANTEN VOOR REEL NAAR ASCII
; 32-BITS BINAIRE GETALLEN
; MET VOORTEKEN
BF16 FA 0A 1F 00 ; -100 000 000
BF1A 00 98 96 80 ; 10 000 000
BF1E FF F0 BD C0 ; -1 000 000
BF22 00 01 86 A0 ; 100 000
BF26 FF FF D8 F0 ; -10 000
BF2A 00 00 03 E8 ; 1 000
BF2E FF FF FF 9C ; -100
BF32 00 00 00 0A ; 10
BF26 FF FF FF FF ; -1
*****CONSTANTEN VOOR VERANDEREN
; VAN TI NAAR TI$
BF3A FF DF 0A 80 ; -2 160 000
BF3E 00 03 4B C0 ; 216 000
BF42 FF FF 73 60 ; -36 000
BF46 00 00 0E 10 ; 3 600
BF4A FF FF FD A8 ; -600
BF4E 00 00 00 3C ; 60
BF52 EC
BF53 AA ...
BF70 ... AA
*****BASIC-FUNCTIE SQR
BF71 20 0C BC JSR $BCOC ; FAC AFRONDEN EN NAAR ARG
BF74 A9 11 LDA #$11
BF76 A0 BF LDY #$BF ; WIJZER OP CONSTATE 0.5
*****MACHTSVERHEFFEN FAC=ARG TOT
; DE MACHT CONSTATE(A/Y)
BF78 20 A2 BB JSR $BBA2 ; CONSTATE NAAR FAC
*****MACHTSVERHEFFEN FAC=ARG^FAC
BF7B F0 70 BEQ $BFED
BF7D A5 69 LDA $69 ; EXPONENT ARG=BASIS
BF7F D0 03 BNE $BF84 ; ONGELIJK NUL?
BF81 4C F9 B8 JMP $BF8F ; KLAAR
BF84 A2 4E LDY #$4E
BF86 A0 00 LDY #$00 ; WIJZER OP HULPACCU

```

```

BF88 20 D4 BB JSR $BBD4 ; FAC NAAR HULPACCU
BF8B A5 6E LDA $6E ; EXPONENT FAC = MACHTSEXPOONENT
BF8D 10 0F BPL $BF9E ; < 1?
BF8F 20 CC BC JSR $BCCC ; INT-FUNCTIE
BF92 A9 4E LDA #$4E
BF94 A0 00 LDY #$00 ; WIJZER OP HULPACCU
BF96 20 5B BC JSR $BC5B ; MET FAC VERGELIJKEN
BF99 D0 03 BNE $BF9E
BF9B 98 TYA
BF9C A4 07 LDY $07
BF9E 20 FE BB JSR $BBFE ; ARG NAAR FAC
BFA1 98 TYA
BFA2 48 PHA
BFA3 20 EA B9 JSR $B9EA ; LOG-FUNCTIE
BFA6 A9 4E LDA #$4E
BFA8 A0 00 LDY #$00 ; WIJZER OP HULPACCU
BFAA 20 28 BA JSR $BA28 ; MET FAC VERMENIGVULDIGEN
BFAD 20 ED BF JSR $BFED ; EXP-FUNCTIE
BFBO 68 PLA
BFB1 4A LSR A
BFB2 90 0A BCC $BFBE
***** VOORTEKEN VERANDEREN
BFB4 A5 61 LDA $61 ; EXPONENT
BFB6 F0 06 BEQ $BFBE ; GETAL=0, DAN KLAAR
BFB8 A5 66 LDA $66
BFBA 49 FF EOR #$FF ; VOORTEKEN INVERTEREN
BFBC 85 66 STA $66
BFBE 60 RTS
***** CONSTANTEN VOOR EXP
BFBF 81 38 AA 3B 29 ; 1.44269504=1/LOG(2)
BFC4 07 ; 7=POLYNOOMGRAAD
; 8 COEFFICIENTEN
BFC5 71 34 58 3E 56 ; 2.14987637E-5
BFCA 74 16 7E B3 1B ; 1.4352314E-4
BFCF 77 2F EE E3 85 ; 1.34226348E-3
BFD4 7A 1D 84 1C 2A ; 9.614011701E-3
BFD9 7C 63 59 58 0A ; .0555051269
BFDE 7E 75 FD E7 C6 ; .240226385
BFE3 80 31 72 18 10 ; .693147186
BFE8 81 00 00 00 00 ; 1
***** BASIC-FUNCTIE EXP
BFED A9 BF LDA #$BF
BFEF A0 BF LDY #$BF ; WIJZER OP CONSTATTE 1/LOG(2)
BFF1 20 28 BA JSR $BA28 ; MET FAC VERMENIGVULDIGEN
BFF4 A5 70 LDA $70
BFF6 69 50 ADC #$50
BFF8 90 03 BCC $BFFD
BFFA 20 23 BC JSR $BC23 ; MANTISSE VAN FAC MET 1
; VERHOGEN

```

```

BFFD 4C 00 EO JMP $E000
E000 85 56 STA $56
E002 20 0F BC JSR $BCOF ; FAC NAAR ARG
E005 A5 61 LDA $61 ; EXPONENT
E007 C9 88 CMP #$88 ; GETAL > 128?
E009 90 03 BCC $E00E
E00B 20 D4 BA JSR $BAD4 ; POSITIEF, DAN OVERFLOW
E00E 20 CC BC JSR $BCCC ; INTEGER-FUNCTIE
E011 A5 07 LDA $07
E013 18 CLC
E014 69 81 ADC #$81
E016 F0 F3 BEQ $E00B ; GELIJK AAN 127?
E018 38 SEC
E019 E9 01 SBC #$01
E01B 48 PHA
E01C A2 05 LDX #$05
E01E B5 69 LDA $69,X
E020 B4 61 LDY $61,X ; FAC EN ARG VERWISSELEN
E022 95 61 STA $61,X
E024 94 69 STY $69,X
E026 CA DEX
E027 10 F5 BPL $E01E
E029 A5 56 LDA $56
E02B 85 70 STA $70
E02D 20 53 B8 JSR $B853 ; ARG-FAC
E030 20 B4 BF JSR $BFB4 ; VERANDERING VAN VOORTEKEN
E033 A9 C4 LDA #$C4
E035 A0 BF LDY #$BF ; WIJZER OP POLYNOMCOEFFICIENT
E037 20 59 EO JSR $E059 ; POLYNOM BEREKENEN
E03A A9 00 LDA #$00
E03C 85 6F STA $6F
E03E 68 PLA
E03F 20 B9 BA JSR $BAB9 ; EXPONENTEN VAN FAC EN ARG
; OPTELLEN
E042 60 RTS
***** POLYNOMBEREKENING
; Y=A1*X+A2*X^3+A3*X^5+...
E043 85 71 STA $71
E045 84 72 STY $72 ; WIJZER OP POLYNOMCOEFFICIENTEN
E047 20 CA BB JSR $BBCA ; FAC NAAR ACCU#3 BRENGEN
E04A A9 57 LDA #$57 ; WIJZER OP ACCU#3
E04C 20 28 BA JSR $BA28 ; FAC*ACCU#3 (KWADRATEREN)
E04F 20 5D EO JSR $E05D ; POLYNOMBEREKENING
E052 A9 57 LDA #$57
E054 A0 00 LDY #$00 ; WIJZER OP ACCU#3
E056 4C 28 BA JMP $BA28 ; FAC=FAC*ACCU#3
***** POLYNOMBEREKENING
; Y=A0+A1*X+A2*X^2+A3+...
E059 85 71 STA $71

```

```

EO5B 84 72 STY $72 ; WIJZER OP POLYNOMGRAAD
EO5D 20 C7 BB JSR $BBC7 ; FAC NAAR ACCU#4 BRENGEN
EO60 B1 71 LDA ($71),Y ; POLYNOMGRAAD
EO62 85 67 STA $67 ; ALS TELLER
EO64 A4 71 LDY $71
EO66 C8 INY
EO67 98 TYA ; WIJZER VERHOGEN, WIJST DAN
EO68 D0 02 BNE $EO6C ; OP 1e COEFFICIENTEN
EO6A E6 72 INC $72
EO6C 85 71 STA $71
EO6E A4 72 LDY $72
EO70 20 28 BA JSR $BA28 ; FAC=FAC*CONSTANTE(WIJZER A/Y)
EO73 A5 71 LDA $71
EO75 A4 72 LDY $72
EO77 18 CLC
EO78 69 05 ADC #$05 ; WIJZER MET 5 VERHOGEN -
; VOLGEND GETAL
EO7A 90 01 BCC $EO7D
EO7C C8 INY
EO7D 85 71 STA $71
EO7F 84 72 STY $72
EO81 20 67 B8 JSR $B867 ; FAC=FAC+CONSTANTE(WIJZER A/Y)
EO84 A9 5C LDA #$5C
EO86 A0 00 LDY #$00 ; WIJZER OP ACCU#4
EO88 C6 67 DEC $67 ; TELLER VERLAGEN
EO8A D0 E4 BNE $EO70
EO8C 60 RTS
***** CONSTANTE VOOR RND
EO8D 98 35 44 7A 00 ; 11879546
EO92 68 28 B1 46 00 ; 3.92767774E-4
***** BASIC-FUNCTIE RND
EO97 20 2B BC JSR $BC2B ; VOORTEKEN HALEN
EO9A 30 37 BMI $EOD3 ; NEGATIEF?
EO9C D0 20 BNE $EOBE
EO9E 20 F3 FF JSR $FFF3 ; BASIS-ADRES VAN CIA HALEN
EOA1 86 22 STX $22
EOA3 84 23 STY $23 ; ALS WIJZER OPSLAAN
EOA5 A0 04 LDY #$04
EOA7 B1 22 LDA ($22),Y ; TIMER A LOW
EOA9 85 62 STA $62
EOAB C8 INY
EOAC B1 22 LDA ($22),Y ; TIMER A HIGH
EOAE 85 64 STA $64
EOB0 A0 08 LDY #$08
EOB2 B1 22 LDA ($22),Y ; TOD 1/10 SEC
EOB4 85 63 STA $63
EOB6 C8 INY
EOB7 B1 22 LDA ($22),Y ; TOD SEC
EOB9 85 65 STA $65

```

EOBB	4C E3	EO	JMP	\$EOE3	
EOBE	A9 8B		LDA	#\$8B	
EOCO	A0 00		LDY	#\$00	; WIJZER OP LAATSTE RND-WAARDE
EOC2	20 A2	BB	JSR	\$BBA2	; NAAR FAC HALEN
EOC5	A9 8D		LDA	#\$8D	
EOC7	A0 E0		LDY	#\$E0	
EOC9	20 28	BA	JSR	\$BA28	
EOCC	A9 92		LDA	#\$92	
EOCE	A0 E0		LDY	#\$E0	; WIJZER OP CONSTATE
EODO	20 67	B8	JSR	\$B867	; FAC=FAC*CONSTATE
EOD3	A6 65		LDX	\$65	
EOD5	A5 62		LDA	\$62	
EOD7	85 65		STA	\$65	
EOD9	86 62		STX	\$62	; PLAATSEN IN FAC VERWISSELEN
EODB	A6 63		LDX	\$63	
EODD	A5 64		LDA	\$64	
EODF	85 63		STA	\$63	
EOE1	86 64		STX	\$64	
EOE3	A9 00		LDA	#\$00	
EOE5	85 66		STA	\$66	
EOE7	A5 61		LDA	\$61	
EOE9	85 70		STA	\$70	
EOEB	A9 80		LDA	#\$80	; EXPONENT
EOED	85 61		STA	\$61	
EOEF	20 D7	B8	JSR	\$B8D7	; LINKER KANTLIJN VAN FAC
					RECHT MAKEN
EOF2	A2 8B		LDX	#\$8B	
EOF4	A0 00		LDY	#\$00	; WIJZER OP LAATSTE RND-WAARDE
EOF6	4C D4	BB	JMP	\$BBD4	; FAC AFRONDEN EN OPSLAAN
*****					INTERPRETATIE VAN FOUTEN NA
					I/O-ROUTINES
EOF9	C9 F0		CMP	#\$F0	; RS-232 OPEN OF CLOSE
EOFB	D0 07		BNE	\$E104	; NEE
EOFD	84 38		STY	\$38	; EINDE VAN BASIC-RAM
					OPNIEUW INSTELLEN
EOFF	86 37		STX	\$37	
E101	4C 63	A6	JMP	\$A663	; EN NAAR CLR-INSTRUCTIE
E104	AA		TAX		; FOUTNUMMER NAAR X
E105	D0 02		BNE	\$E109	; ONGELIJK AAN O?
E107	A2 1E		LDX	#\$1E	; ANDERS NUMMER VOOR BREAK
E109	4C 37	A4	JMP	\$A437	; FOUTMELDING UITVOEREN
*****					BASIC BSOUT
E10C	20 D2	FF	JSR	\$FFD2	; TEKEN UITVOEREN
E10F	B0 E8		BCS	\$EOF9	; FOUT?
E111	60		RTS		
*****					BASIC BASIN
E112	20 CF	FF	JSR	\$FFCF	; TEKEN HALEN
E115	B0 E2		BCS	\$EOF9	; FOUT?
E117	60		RTS		

```

***** BASIC CKOUT
E118 20 AD E4 JSR $E4AD ; UITVOERAPPARAAT
E11B BO DC BCS $EOF9 ; FOUT?
E11D 60 RTS
***** BASIC CHKIN
E11E 20 C6 FF JSR $FFC6 ; INVOERAPPARAAT ZETTEN
E121 BO D6 BCS $EOF9 ; FOUT?
E123 60 RTS
***** BASIC GETIN
E124 20 E4 FF JSR $FFE4 ; TEKEN HALEN
E127 BO D0 BCS $EOF9 ; FOUT HALEN
E129 60 RTS
***** SYS-INSTRUCTIE
E12A 20 8A AD JSR $AD8A ; FRMNUM: NUMERIEKE TERM HALEN
E12D 20 F7 B7 JSR $B7F7 ; IN ADRESFORMAAT VERANDEREN,
; NAAR $14/$15
E130 A9 E1 LDA #$E1
E132 48 PHA ; SPRONGADRES OP STACK
E133 A9 46 LDA #$46
E135 48 PHA
E136 AD OF 03 LDA $030F ; STATUS
E139 48 PHA
E13A AD OC 03 LDA $030C ; ACCU,
E13D AE OD 03 LDX $030D ; X-REGISTER EN
E140 AC OE 03 LDY $030E ; Y-REGISTER OVERDRAGEN
E143 28 PLP ; STATUS ZETTEN
E144 6C 14 00 JMP ($0014) ; ROUTINE OPROEPEN
E147 08 PHP ; STATUS OPSLAAN
E148 8D OC 03 STA $030C ; ACCU,
E14B 8E OD 03 STX $030D ; X-REGISTER,
E14E 8C OE 03 STY $030E ; Y-REGISTER EN
E151 68 PLA
E152 8D OF 03 STA $030F ; STATUS WEER OPSLAAN
E155 60 RTS
***** SAVE-INSTRUCTIE
E156 20 D4 E1 JSR $E1D4 ; PARAMETERS (FILENAAM,
; PRIM. EN SEC. ADRES)
E159 A6 2D LDX $2D ; EINDADRES = EIND BASIC-PROGRAMMA
E15B A4 2E LDY $2E
E15D A9 2B LDA #$2B ; STARTADRES = WIJZER OP
; BASIC-START
E15F 20 D8 FF JSR $FFD8 ; SAVE-ROUTINE
E162 BO 95 BCS $EOF9 ; FOUT?
E164 60 RTS
***** VERIFY-INSTRUCTIE
E165 A9 01 LDA #$01 ; VERIFY-FLAG
E167 2C .BYTE $2C
***** LOAD-INSTRUCTIE
E168 A9 00 LDA #$00 ; LOAD-FLAG

```

```

E16A 85 0A STA $0A ; OPSLAAN
E16C 20 F6 E4 JSR $E4F6 ; PARAMETER HALEN
E16F A5 0A LDA $0A ; FLAG
E171 A6 2B LDX $2B ; STARTADRES = BASIC-START
E173 A4 2C LDY $2C
E175 20 D5 FF JSR $FFD5 ; LOAD-ROUTINE
E178 B0 57 BCS $E1D1 ; FOUT?
E17A A5 0A LDA $0A ; LOAD/VERIFY-FLAG
E17C F0 17 BEQ $E195 ; LOAD?
E17E A2 1C LDX #$1C ; OFFSET VOOR VERIFY ERROR
E180 20 B7 FF JSR $FFB7 ; STATUS HALEN
E183 29 10 AND #$10 ; FOUT-BIT ISOLEREN
E185 D0 17 BNE $E19E ; STATUSBIT GEZET, DAN MOET FOUT
E187 A5 7A LDA $7A ; HI-BYTE $7B ZIJN
E189 C9 02 CMP #$02 ; DIREKTE MODUS?
E18B F0 07 BEQ $E194 ; JA, DAN KLAAR
E18D A9 64 LDA #$64 ; WIJZER OP OK
E18F A0 A3 LDY #$A3
E191 4C 1E AB JMP $AB1E ; UITVOEREN
E194 60 RTS
E195 20 B7 FF JSR $FFB7 ; STATUS HALEN
E198 29 BF AND #$BF ; EOF-BIT WISSEN
E19A F0 05 BEQ $E1A1 ; GEEN FOUT
E19C A2 1D LDX #$1D ; OFFSET VOOR LOAD ERROR
E19E 4C 37 A4 JMP $A437 ; FOUTMELDING UITVOEREN
E1A1 A5 7B LDA $7B
E1A3 C9 02 CMP #$02 ; DIREKTE MODUS?
E1A5 D0 0E BNE $E1B5 ; NEE, DAN VERDER
E1A7 86 2D STX $2D ; EINDADRES = EINDE VAN PROGRAMMA
E1A9 84 2E STY $2E
E1AB A9 76 LDA #$76 ; WIJZER OP READY
E1AD A0 A3 LDY #$A3
E1AF 20 1E AB JSR $AB1E ; STRING UITVOEREN
E1B2 4C 2A A5 JMP $A52A ; PROGRAMMAREGELS OPNIEUW
; KOPPELEN, CLR
E1B5 20 8E A6 JSR $A68E ; CHRGET-WIJZER OP BEGIN VAN
; PROGRAMMA
E1B8 20 33 A5 JSR $A533 ; PROGRAMMAREGELS OPNIEUW
; KOPPELEN
E1BB 4C 77 A6 JMP $A677 ; RESTORE, BASIC INITIALISEREN
***** BASIC-INSTRUCTIE OPEN
E1BE 20 19 E2 JSR $E219 ; PARAMETER HALEN
E1C1 20 C0 FF JSR $FFC0 ; OPEN-ROUTINE
E1C4 B0 0B BCS $E1D1 ; FOUT?
E1C6 60 RTS
***** BASIC-INSTRUCTIE CLOSE
E1C7 20 19 E2 JSR $E219 ; PARAMETER HALEN
E1CA A5 49 LDA $49 ; FILENUMMER
E1CC 20 C3 FF JSR $FFC3 ; CLOSE-ROUTINE

```

```

E1CF 90 C3      BCC $E194      ; GEEN FOUT, RTS
E1D1 4C F9 EO   JMP $EOF9      ; NAAR FOUT-INTERPRETATIE
*****          ; PARAMETER VOOR LOAD EN SAVE HALEN
E1D4 A9 00      LDA #$00       ; DEFAULT VOOR LENGTE FILENAAM
E1D6 20 BD FF   JSR $FFBD      ; FILENAAMPARAMETER ZETTEN
E1D9 A2 08      LDX #$08       ; DEFAULT VOOR DEVICE-NUMMER
E1DB A0 01      LDY #$01       ; DEFAULT VOOR SECUNDAIRE ADRES
E1DD 20 38 F4   JSR $F438      ; FILEPARAMETER ZETTEN
E1E0 20 06 E2   JSR $E206      ; MEER TEKENS?
E1E3 20 57 E2   JSR $E257      ; FILENAAM HALEN
E1E6 20 EC E4   JSR $E4EC      ; MEER TEKENS
E1E9 20 00 E2   JSR $E200      ; DEVICE-NUMMER HALEN
E1EC A0 00      LDY #$00       ; SECUNDAIRE ADRES
E1EE 86 49      STX $49        ; DEVICE-NUMMER
E1FO 20 BA FF   JSR $FFBA      ; FILEPARAMETER ZETTEN
E1F3 20 EC E4   JSR $E4EC      ; MEER TEKENS?
E1F6 20 00 E2   JSR $E200      ; SECUNDAIRE ADRES HALEN
E1F9 8A         TXA
E1FA A8         TAY            ; SECUNDAIRE ADRES
E1FB A6 49      LDX $49        ; DEVICE-NUMMER
E1FD 4C BA FF   JMP $FFBA      ; FILEPARAMETERS ZETTEN
*****
E200 20 0E E2   JSR $E20E      ; KOMMA? ANDERE TEKENS?
E203 4C 9E B7   JMP $B79E      ; HAALT BYTE-WAARDE NAAR X
*****          ; ANDERE TEKENS?
E206 20 79 00   JSR $0079      ; CHRGOT: LAATSTE TEKEN
E209 D0 02      BNE $E20D      ; ANDERE TEKENS, DAN TERUG
E20B 68         PLA            ; ANDERS NAAR HOGERE ROUTINE
E20C 68         PLA
E20D 60         RTS
*****
E20E 20 FD AE   JSR $AEFD      ; KOMMA?
E211 20 79 00   JSR $0079      ; CHRGOT: LAATSTE TEKEN HALEN
E214 D0 F7      BNE $E20D      ; ANDERE TEKENS, DAN TERUG
E216 4C 08 AF   JMP $AF08      ; SYNTAX ERROR
*****          ; PARAMETER VOOR OPEN HALEN
E219 A9 00      LDA #$00       ; DEFAULT VOOR LENGTE FILENAAM
E21B 20 BD FF   JSR $FFBD      ; PARAMETER VOOR FILENAAM ZETTEN
E21E 20 11 E2   JSR $E211      ; ANDERE TEKENS
;E221 20 9E B7   JSR $B79E      ; HAALT LOGISCHE FILENUMMER
;                                ; NAAR X
E224 86 49      STX $49
E226 8A         TXA            ; LOGISCHE FILENUMMER
E227 A2 08      LDX #$08       ; DEFAULT VOOR DEVICE-NUMMER
E229 A0 0F      LDY #$0F       ; DEFAULT VOOR SECUNDAIRE ADRES
E22B 20 38 F4   JSR $F438      ; FILEPARAMETERS ZETTEN
E22E 20 06 E2   JSR $E206      ; ANDERE TEKENS?
E231 20 00 E2   JSR $E200      ; HAALT DEVICE-NUMMER
E234 86 4A      STX $4A

```



```

E236 A0 00 LDY #$00 ; SECUNDAIRE ADRES
E238 A5 49 LDA $49 ; LOGISCH FILENUMMER
E23A E0 03 CPX #$03 ; DEVICE-NUMMER < 3
E23C D0 01 BNE $E23F ; JA
E23E 88 DEY ; ANDERS SECUNDAIRE ADRES OP 255
; (GEEN SECUNDAIRE ADRESSEN))
E23F 20 BA FF JSR $FFBA ; FILEPARAMETERS ZETTEN
E242 20 06 E2 JSR $E206 ; ANDERE TEKENS?
E245 20 00 E2 JSR $E200 ; HAALT SECUNDAIRE ADRES
E248 8A TXA
E249 A8 TAY ; SECUNDAIRE ADRES
E24A A6 4A LDX $4A ; DEVICE-NUMMER
E24C A5 49 LDA $49 ; LOGISCHE FILENUMMER
E24E 20 BA FF JSR $FFBA ; FILEPARAMETERS ZETTEN
E251 20 06 E2 JSR $E206 ; ANDERE TEKENS?
E254 20 0E E2 JSR $E20E ; KOMMA?
E257 20 9E AD JSR $AD9E ; FREMVL: TERM HALEN
E25A 20 A3 B6 JSR $B6A3 ; HAALT STRINGPARAMETER, FRESTR
E25D A6 22 LDX $22
E25F A4 23 LDY $23 ; ADRES VAN FILENAAM
E261 4C BD FF JMP $FFBD ; FILENAAMPARAMETERS ZETTEN
***** BASIC-FUNCTIE COS
E264 A9 E0 LDA #$E0
E266 A0 E2 LDY #$E2 ; WIJZER OP CONSTATE PI/2
E268 20 67 B8 JSR $B867 ; BIJ FAC OPTELLEN
***** BASIC-FUNCTIE SIN
E26B 20 0C BC JSR $BC0C ; FAC AFRONDEN EN NAAR ARG
E26E A9 E5 LDA #$E5
E270 A0 E2 LDY #$E2 ; WIJZER OP CONSTATE PI*2
E272 A6 6E LDX $6E
E274 20 07 BB JSR $BB07 ; FAC DOOR 2*PI DELEN
E277 20 0C BC JSR $BC0C ; FAC AFRONDEN EN NAAR ARG
E27A 20 CC BC JSR $BCCC ; INT-FUNCTIE
E27D A9 00 LDA #$00
E27F 85 6F STA $6F
E281 20 53 B8 JSR $B853 ; ARG-FAC
E284 A9 EA LDA #$EA
E286 A0 E2 LDY #$E2 ; WIJZER OP CONSTATE 0.25
E288 20 50 B8 JSR $B850 ; 0.25-FAC
E28B A5 66 LDA $66
E28D 48 PHA ; VOORTEKEN OP STACK
E28E 10 0D BPL $E29D ; POSITIEF?
E290 20 49 B8 JSR $B849 ; FAC+0.5
E293 A5 66 LDA $66 ; VOORTEKEN
E295 30 09 BMI $E2A0 ; NEGATIEF?
E297 A5 12 LDA $12
E299 49 FF EOR #$FF ; FLAG OMDRAAIEN
E29B 85 12 STA $12
E29D 20 B4 BF JSR $BFB4 ; VOORTEKEN VERANDEREN

```

```

E2AO  A9 EA    LDA #$EA
E2A2  A0 E2    LDY #$E2      ; WIJZER OP CONSTATE 0.25
E2A4  20 67 B8 JSR $B867     ; FAC+0.25
E2A7  68      PLA           ; VOORTEKEN HALEN
E2A8  10 03    BPL $E2AD     ; POSITIEF?
E2AA  20 B4 BF JSR $BFB4     ; VOORTEKEN WISSELEN
E2AD  A9 EF    LDA #$EF
E2AF  A0 E2    LDY #$E2      ; WIJZER OP POLYNOOMCOEFFICIENT
E2B1  4C 43 EO JMP $E043     ; POLYNOOM BEREKENEN
***** BASIC-FUNCTIE TAN
E2B4  20 CA BB JSR $BBCA     ; FAC NAAR ACCU#3
E2B7  A9 00    LDA #$00
E2B9  85 12    STA $12       ; FLAG ZETTEN
E2BB  20 6B E2 JSR $E26B     ; SIN BEREKENEN
E2BE  A2 4E    LDX #$4E
E2C0  A0 00    LDY 7$00      ; WIJZER OP HULPACCU
E2C2  20 F6 EO JSR $EOF6     ; FAC NAAR HULPACCU
E2C5  A9 57    LDA #$57
E2C7  A0 00    LDY #$00      ; WIJZER OP ACCU#3
E2C9  20 A2 BB JSR $BBA2     ; ACCU#3 NAAR FAC
E2CC  A9 00    LDA #$00
E2CE  85 66    STA $66       ; VOORTEKEN
E2D0  A5 12    LDA $12       ; FLAG
E2D2  20 DC E2 JSR $E2DC     ; COS BEREKENEN
E2D5  A9 4E    LDA #$4E
E2D7  A0 00    LDY #$00      ; WIJZER OP HULPACCU (SIN)
E2D9  4C 0F BB JMP $BBOF     ; DOOR FAC DELEN
E2DC  48      PHA
E2DD  4C 9D E2 JMP $E29D     ; COS BEREKENEN
***** CONSTANTEN VOOR SIN EN COS
E2EO  81 49 0F DA A2        ; 1.57079633    PI/2
E2E5  83 49 0F DA A2        ; 6.28318531    2*PI
E2EA  7F 00 00 00 00        ; .25
E2EF  05                    ; 5=POLYNOOMGRAAD, 6 COEFFICIENTEN
E2FO  84 E6 1A 2D 1B        ; -14.3813907
E2F5  86 28 07 FB F8        ; 42.0077971
E2FA  87 99 68 89 01        ; -76.7041703
E2FF  87 23 35 DF E1        ; 81.6052237
E304  86 A5 5D E7 28        ; -41.3147021
E309  83 49 0F DA A2        ; 6.28318531    2*PI
***** BASIC-FUNCTIE ATN
E30E  A5 66    LDA #$66     ; VOORTEKEN
E30F  48      PHA           ; WEGZETTEN
E311  10 03    BPL $E316     ; POSITIEF?
E313  20 B4 BF JSR $BFB4     ; VOORTEKEN VERANDEREN
E316  A5 61    LDA $61       ; EXPONENT
E318  48      PHA           ; WEGZETTEN
E319  C9 81    CMP #$81      ; GETAL MET 1 VERGELIJKEN
E31B  90 07    BCC $E324     ; KLEINER?

```

```

E31D A9 BC LDA # $BC
E31F A0 B9 LDY # $B9 ; WIJZER OP CONSTATE 1
E321 20 0F BB JSR $BBOF ; 1 DOOR FAC DELEN
E324 A9 3E LDA # $3E
E326 A0 E3 LDY # $E3 ; WIJZER OP POLYNOOMCOEFFICIENT
E328 20 43 E0 JSR $E043 ; POLYNOOOM BEREKENEN
E32B 68 PLA ; EXPONENT TERUGHALEN
E32C C9 81 CMP # $81 ; WAS GETAL < 1?
E32E 90 07 BCC $E337
E330 A9 E0 LDA # $E0
E332 A0 E2 LDY # $E2 ; WIJZER OP CONSTATE PI/2
E334 20 50 B8 JSR $B850 ; PI/2 MINUS FAC
E337 68 PLA ; VOORTEKEN HALEN
E338 10 03 BPL $E33D ; POSITIEF?
E33A 4C B4 BF JMP $BFB4 ; VOORTEKEN WISSELEN
E33D 60 RTS
***** CONSTATE: REELE GETALLEN
; VOOR ATN-FUNCTIE
E33E 0B ; 11 = POLYNOOMGRAAD, DAN
; 12 COEFFICIENTEN
E33F 76 B3 83 BD D3 ; -6.84793912E-04
E344 79 1E F4 A6 F5 ; 4.85094216E-03
E349 7B 83 FC B0 10 ; -.0161117015
E34E 7C 0C 1F 67 CA ; .034209638
E353 7C DE 53 CB C1 ; -.054279133
E358 7D 14 64 70 4C ; .0724571965
E35D 7D B7 EA 51 7A ; -.0898019185
E362 7D 63 30 88 7E ; .110932413
E367 7E 92 44 99 3A ; -.142839808
E36C 7E 4C CC 91 C7 ; .19999912
E371 7F AA AA AA 13 ; -.333333316
E376 81 00 00 00 00 ; 1
***** BASIC NMI-SPRONG
E37B 20 CC FF JSR $FFCC ; CLRCH
E37E A9 00 LDA # $00
E380 85 13 STA $13 ; INVOERAPPARAAT = TOETSENBOARD
E382 20 7A A6 JSR $A67A ; BASIC INITIALISEREN
E385 58 CLI
E386 A2 80 LDX # $80 ; FLAG VOOR GEEN FOUT
E388 6C 00 03 JMP ($0300) ; VECTOR VOOR BASIC WARME START
; JMP $E38E
E38B 8A TXA ; FOUTNUMBER IN ACCU
E38C 30 03 BMI $E391 ; GEEN FOUT, DAN READY
E38E 4C 3A A4 JMP $A43A ; FOUTMELDING UITVOEREN
E391 4C 74 A4 JMP $A474 ; READY-MODUS
***** BASIC KOUDE START
E394 20 53 E4 JSR $E453 ; BASIC-VECTOREN ZETTEN
E397 20 BF E3 JSR $E3BF ; RAM INITIALISREN
E39A 20 22 E4 JSR $E422 ; INSCHAKELMELDING UITVOEREN

```

```

E39D A2 FB LDX #$FB ; STACKPOINTER ZETTEN
E39F 9A TXS
E3A0 D0 E4 BNE $E386 ; NAAR WARME START
***** KOPIE VAN CHRGET-ROUTINE
E3A2 E6 7A INC $7A
E3A4 D0 02 BNE $E3A8 ; WIJZER IN BASIC-TEKST
; VERHOGEN
E3A6 E6 7B INC $7B
E3A8 AD 60 EA LDA $EA60
E3AB C9 3A CMP #$3A ; DUBBELE PUNT
E3AD B0 0A BCS $E3B9
E3AF C9 20 CMP #$20 ; SPATIE
E3B1 F0 EF BEQ $E3A2
E3B3 38 SEC
E3B4 E9 30 SBC #$30 ; TEST OP CIJFER, DAN C=1
E3B6 38 SEC
E3B7 E9 D0 SBC #$D0
E3B9 60 RTS
***** BEGINWAARDE VOOR RND-FUNCTIE
E3BA 80 4F C7 52 58 ; .811635157
***** RAM VOOR BASIC INITIALISEREN
E3BF A9 4C LDA #$4C ; JMP
E3C1 85 54 STA $54 ; VOOR FUNCTIES
E3C3 8D 10 03 STA $0310 ; VOOR USR-FUNCTIE
E3C6 A9 48 LDA #$48 ; WIJZER OP ILLEGAL QUANTITY
E3C8 A0 B2 LDY #$B2
E3CA 8D 11 03 STA $0311 ; ALS USR-VECTOR OPSLAAN
E3CD 8C 12 03 STY $0312
E3D0 A9 91 LDA #$91 ; $B391
E3D2 A0 B3 LDY #$B3
E3D4 85 05 STA $05 ; ALS VECTOR VOOR VERANDERING
; VAN INTEGER IN REEEL
E3D6 84 06 STY $06
E3D8 A9 AA LDA #$AA ; $B1AA
E3DA A0 B1 LDY #$B1
E3DC 85 03 STA $03 ; ALS VECTOR VOOR VERANDERING
; VAN INTEGER IN REEEL
E3DE 84 04 STY $04
E3E0 A2 1C LDX #$1C
E3E2 BD A2 E3 LDA $E3A2,X ; CHRGET-ROUTINE
E3E5 95 73 STA $73,X ; IN RAM KOPIEREN
E3E7 CA DEX
E3E8 10 F8 BPL $E3E2
E3EA A9 03 LDA #$03
E3EC 85 53 STA $53 ; STAPGROOTTE BIJ
; GARBAGE COLLECTION
E3EE A9 00 LDA #$00
E3F0 85 68 STA $68
E3F2 85 13 STA $13 ; INVOERAPPARAAT = TOETSENBORD

```

```

E3F4 85 18 STA $18
E3F6 A2 01 LDX #$01
E3F8 8E FD 01 STX $01FD ; DUMMIES VOOR LINKADRES BIJ
; TUSSENVOEGEN VAN REGELS
E3FB 8E FC 01 STX $01FC
E3FE A2 19 LDX #$19

```

```

E400 86 16 STX $16 ; WIJZER VOOR STRINGBEHEER
E402 38 SEC
E403 20 9C FF JSR $FF9C ; BEGIN VAN RAM HALEN
E406 86 2B STX $2B
E408 84 2C STY $2C ; ALS BASIC-START OPSLAAN
E40A 38 SEC
E40B 20 99 FF JSR $FF99 ; EINDE VAN RAM HALEN
E40E 86 37 STX $37
E410 84 38 STY $38 ; ALS BASIC-EINDE OPSLAAN
E412 86 33 STX $33
E414 84 34 STY $34
E416 A0 00 LDY #$00
E418 98 TYA
E419 91 2B STA ($2B),Y ; $00 BIJ BASIC-START
E41B E6 2B INC $2B
E41D D0 02 BNE $E421 ; BASIC-START+1
E41F E6 2C INC $2C
E421 60 RTS

```

```

E422 A5 2B LDA $2B
E424 A4 2C LDY $2C ; WIJZER OP BEGIN VAN BASIC-RAM
E426 20 08 A4 JSR $A408 ; PLAATS IN HET GEHEUGEN?
E429 A9 73 LDA #$73 ; WIJZER OP INSCHAKELMELDING
E42B A0 E4 LDY #$E4
E42D 20 1E AB JSR $AB1E ; STRING UITVOEREN
E430 A5 37 LDA $37
E432 38 SEC ; BASIC-EINDE
E433 E5 2B SBC $2B
E435 AA TAX ; MINUS BASIC-START
E436 A5 38 LDA $38
E438 E5 2C SBC $2C ; IS AANTAL BYTES FREE
E43A 20 CD BD JSR $BDCD ; AANTAL UITVOEREN
E43D A9 60 LDA #$60 ; WIJZER OP BASIC BYTES FREE
E43F A0 E4 LDY #$E4
E441 20 1E AB JSR $AB1E ; STRING UITVOEREN
E444 4C 44 A6 JMP $A644 ; NAAR NEW-INSTRUCTIE
***** TABEL VAN BASIC-VECTOREN

```

```

E447 8B E3 83 A4 7C A5 1A A7
E44F E4 A7 86 AE

```

***** BASIC-VECTOREN LADEN

```
E453 A2 OB LDX #$0B
E455 BD 47 E4 LDA $E447,X
E458 9D 00 03 STA $0300,X
E45B CA DEX
E45C 10 F7 BPL $E455
E45E 60 RTS
```

***** BEDRIJFSSYSTEEM
SYSTEEM-MELDINGEN

```
;
E45F 00 20 42 41 53 49 43 20; BASIC BYTES FREE
E467 42 59 54 45 53 20 46 52
E46F 45 45 0D 00
E473 93 0D 20 20 20 20 2A 2A; (CLR) *** COMMODORE 64 BASIC V2 ***
E47B 2A 2A 20 43 4F 4D 4D 4F; (CR) (CR) 64K RAM SYSTEM
E483 44 4F 52 45 20 36 34 20
E48B 42 41 53 49 43 20 56 32
E493 20 2A 2A 2A 2A 0D 0D 20
E49B 36 34 4B 20 52 41 4D 20
E4A3 53 59 53 54 45 4D 20 20
E4AB 00
E4AC 5C
```

***** BASIC-CKOUT ROUTINE

```
E4AD 48 PHA
E4AE 20 C9 FF JSR $FFC9 ; CKOUT: UITVOERAPPARAAT INSTELLEN
E4B1 AA TAX ; FOUTNUMMER NAAR X
E4B2 68 PLA
E4B3 90 01 BCC $E4B6 ; GEEN FOUT?
E4B5 8A TXA ; FOUTNUMMER WEER IN ACCU
E4B6 60 RTS
E4B7 AA .....
E4D9 ..... AA
```

***** ACHTERGRONDKLEUR INSTELLEN

```
E4DA AD 86 02 LDA $0286 ; KLEUR HALEN
E4DD 91 F3 STA ($F3),Y ; IN KLEUR-RAM SCHRIJVEN
E4DF 60 RTS
```

***** WACHT OP COMMODORE-TOETS

```
E4E0 69 02 ADC #$02 ; 2*256/60=8.5 SECONDEN WACHTEN
E4E2 A4 91 LDY $91 ; FLAG TESTEN
E4E4 C8 INY
E4E5 D0 04 BNE $E4EB ; TOETS INGEDRUKT?
E4E7 C5 A1 CMP $A1 ; TIJD NOG NIET VOORBIJ?
E4E9 D0 F7 BNE $E4E2
E4EB 60 RTS
```

*****TIMERCONSTANTEN VOOR RS-232 BAUD RATE, PAL-VERSIE

```
E4EC 19 26 $2619 =9753 50 BAUD
E4EE 44 19 $1944 =6468 75 BAUD
E4F0 1A 11 $111A =4378 110 BAUD
E4F2 E8 0D $0DE8 =3560 134.5 BAUD
E4F4 70 0C $0C70 =3184 150 BAUD
```

```

E4F6 06 06 $0606 =1542 300 BAUD
E4F8 D1 02 $02D1 =736 600 BAUD
E4FA 37 01 $0137 =311 1200 BAUD
E4FC AE 00 $00AE =174 1800 BAUD
E4FE 69 00 $0069 =105 2400 BAUD
***** BASISADRESSEN VAN CIA'S HALEN
E500 A2 00 LDX #$00
E502 A0 DC LDY #$DC ; $DC00
E504 60 RTS
***** HAALT HET AANTAL REGELS
; EN KOLOMMEN
E505 A2 28 LDX #$28 ; 40 KOLOMMEN
E507 A0 19 LDY #$19 ; 25 REGELS
E509 60 RTS
***** CURSOR INSTELLEN (C=0),
; HALEN(C=1)
E50A B0 07 BCS $E513
E50C 86 D6 STX $D6 ; REGEL
E50E 84 D3 STY $D3 ; KOLOM
E510 20 6C E5 JSR $E56C ; CURSOR INSTELLEN
E513 A6 D6 LDX $D6
E515 A4 D3 LDY $D3
E517 60 RTS
***** BEELDSCHERM RESET
E518 20 A0 E5 JSR $E5A0 ; VIDEOCONTROLLER INITIALISEREN
E51B A9 00 LDA #$00
E51D 8D 91 02 STA $0291 ; SHIFT-COMMODORE
; MOGELIJK MAKEN
E520 85 CF STA $CF ; CURSOR NIET IN KNIPPERMODUS
E522 A9 48 LDA #$48
E524 8D 8F 02 STA $028F ; ($028F)=$EB48
E527 A9 EB LDA #$EB
E529 8D 90 02 STA $0290 ; =WIJZER OP ADRESSEN VOOR
; TOETSENBORD-DECODERING
E52C A9 0A LDA #$0A ; 10
E52E 8D 89 02 STA $0289 ; MAXIMALE LENGTE VAN
; TOETSENBORDBUFFER
E531 8D 8C 02 STA $028C ; TELLER VOOR REPEAT-SNELHEID
E534 A9 0E LDA $OE ; LICHTBLAUW
E536 8D 86 02 STA $0286 ; ACTUELE KLEUR
E539 A9 04 LDA #$04
E53B 8D 8B 02 STA $028B ; REPEAT-SNELHEID
E53E A9 0C LDA #$0C
E540 85 CD STA $CD ; KNIPPERDUUR VAN CURSOR
E542 85 CC STA $CC ; KNIPPERFLAG VOOR CURSOR
***** BEELDSCHERM WISSEN
E544 AD 88 02 LDA $0288 ; GEHEUGENPAGINA VOOR
; BEELDSCHERM-RAM
E547 09 80 ORA #$80

```

```

E549 A8      TAY
E54A A9 00   LDA #$00
E54C AA      TAX
E54D 94 D9   STY $D9,X      ; ADRESSEN VAN BEELDSCHERMREGELS
E54F 18      CLC
E550 69 28   ADC #$28      ; PLUS 40 (1 REGEL)
E552 90 01   BCC $E555
E554 C8      INY
E555 E8      INX
E556 E0 1A   CPX #$1A      ; 26, ALLE REGELS?
E558 D0 F3   BNE $E54D
E55A A9 FF   LDA #$FF
E55C 95 D9   STA $D9,X
E55E A2 18   LDX #$18      ; 24, AANTAL REGELS MIN 1
E560 20 FF E9 JSR $E9FF     ; BEELDSCHERMREGEL WISSEN
E563 CA      DEX
E564 10 FA   BPL $E560
***** CURSOR HOME
E566 A0 00   LDY #$00
E568 84 D3   STY $D3      ; CURSORKOLOM
E56A 84 D6   STY $D6      ; CURSORREGEL
***** CURSORPOSITIE BEREKENEN,
; BEELDSCHERMWIJZER ZETTEN
E56C A6 D6   LDX $D6      ; CURSORREGEL
E56E A5 D3   LDA $D3      ; CURSORKOLOM
E570 B4 D9   LDY $D9,X
E572 30 08   BMI $E57C
E574 18      CLC
E575 69 28   ADC #$28      ; +40
E577 85 D3   STA $D3
E579 CA      DEX
E57A 10 F4   BPL $E570
E57C B5 D9   LDA $D9,X      ; MSB VAN BEGINADRES VAN REGEL
E57E 29 03   AND #$03
E580 0D 88 02 ORA $0288     ; VIDEO-RAM PAGE (HI-BYTE)
E583 85 D2   STA $D2
E585 BD F0 EC LDA $ECF0,X   ; LSB VAN BEGINADRES VAN REGEL
E588 85 D1   STA $D1      ; ALS WIJZER NAAR $D1/$D2
E58A A9 27   LDA #$27      ; 39
E58C E8      INX
E58D B4 D9   LDY $D9,X
E58F 30 06   BMI $E597
E591 18      CLC
E592 69 28   ADC #$28      ; +40
E594 E8      INX
E595 10 F6   BPL $E58D
E597 85 D5   STA $D5
E599 60      RTS

```



```

*****
E59A 20 A0 E5 JSR $E5A0 ; VIDEOCONTROLLER INITIALISEREN
E59D 4C 66 E5 JMP $E566 ; CURSOR HOME
***** VIDEOCONTROLLER INITIALISEREN
E5A0 A9 03 LDA #03
E5A2 85 9A STA $9A ; UITVOER OP BEELDSCHERM
E5A4 A9 00 LDA #$00
E5A6 85 99 STA $99 ; INVOER VAN TOETSENBORD
E5A8 A2 2F LDX #2F ; 2F
E5AA BD B8 EC LDA $ECB8,X ; CONSTANTEN
E5AD 9D FF CF STA $CFFF,X ; IN VIDEOCONTROLLER SCHRIJVEN
E5B0 CA DEX
E5B1 D0 F7 BNE $E5AA
E5B3 60 RTS
***** TEKEN UIT TOETSENBORDBUFFER
;
E5B4 AC 77 02 LDY $0277 ; 1e TEKEN HALEN
E5B7 A2 00 LDX #$00
E5B9 BD 78 02 LDA $0278,X ; BUFFER NAAR VOREN
E5BC 9D 77 02 STA $0277,X
E5BF E8 INX
E5C0 E4 C6 CPX $C6 ; MET AANTAL TEKENS VERGELIJKEN
E5C2 D0 F5 BNE $E5B9
E5C4 C6 C6 DEC $C6 ; TEKEN GETAL VERLAGEN
E5C6 98 TYA ; TEKEN IN ACCU HALEN
E5C7 58 CLI
E5C8 18 CLC
E5C9 60 RTS
***** WACHTIJS VOOR INVOER
;
E5CA 20 16 E7 JSR $E716 ; TEKEN OP BEELDSCHERM BRENGEN
E5CD A5 C6 LDA $C6 ; AANTAL INGEDRUKTE TOETSEN
E5CF 85 CC STA $CC
E5D1 8D 92 02 STA $0292
E5D4 F0 F7 BEQ $E5CD ; GEEN TOETS GEDRUKT,
; DAN WACHTEN
E5D6 78 SEI
E5D7 A5 CF LDA $CF ; CURSOR IN KNIPPERMODUS?
E5D9 F0 0C BEQ $E5E7 ; NEE
E5DB A5 CE LDA $CE ; TEKEN ONDER CURSOR
E5DD AE 87 02 LDX $0287 ; KLEUR ONDER CURSOR
E5E0 A0 00 LDY #$00
E5E2 84 CF STY $CF ; CURSOR NIET IN KNIPPERMODUS
E5E4 20 13 EA JSR $EA13 ; TEKEN EN KLEUR INSTELLEN
E5E7 4C C4 FE JMP $FEC4 ; TEKEN UIT TOETSENBORDBUFFER
;
E5EA C9 83 CMP #$83 ; CODE VOOR SHIFT/RUN
E5EC D0 10 BNE $E5FE
E5EE A2 09 LDX #$09 ; 9 TEKENS

```

```

E5F0 78      SEI
E5F1 86 C6    STX $C6      ; AANTAL TEKENS ONTHOUDEN
E5F3 BD E6 EC LDA $ECE6,X  ; LOAD (CR) RUN (CR)
E5F6 9D 76 02 STA $0276,X ; IN TOETSENBORDBUFFER HALEN
E5F9 CA       DEX
E5FA D0 F7    BNE $E5F3
E5FC FO CF    BEQ $E5CD    ; EN VERWERKEN
E5FE C9 OD    CMP #$0D     ; CR
E600 D0 C8    BNE $E5CA    ; NEE, DAN TERUG NAAR WACHTLUS
E602 A4 D5    LDY $D5      ; LENGTE VAN BEELDSCHERMREGEL
E604 84 D0    STY $D0      ; CR-FLAG ZETTEN
E606 B1 D1    LDA ($D1),Y  ; TEKEN VAN BEELDSCHERM HALEN
E608 C9 20    CMP #$20     ; SPATIE AAN HET EINDE
;              VAN DE REGEL WISSEN
E60A D0 03    BNE $E60F
E60C 88       DEY
E60D D0 F7    BNE $E606
E60F C8       INY
E610 84 C8    STY $C8      ; POSITIE ALS INDEX ONTHOUDEN
E612 A0 00    LDY #$00     ; CURSORKOLOM=0
E614 8C 92 02 STY $0292    ; AANHALINGSTEKEN-FLAG WISSEN
E617 84 D3    STY $D3
E619 84 D4    STY $D4
E61B A5 C9    LDA $C9
E61D 30 1B    BMI $E63A
E61F A6 D6    LDX $D6
E621 20 91 E5 JSR $E591
E624 E4 C9    CPX $C9
E626 D0 12    BNE $E63A
E628 A5 CA    LDA $CA      ; LAATSTE KOLOM
E62A 85 D3    STA $D3      ; IN KOLOMWIJZER BRENGEN
E62C C5 C8    CMP $C8      ; MET INDEX VERGELIJKEN
E62E 90 0A    BCC $E63A
E630 B0 2B    BCS $E65D
*****
E632 98       TYA
E633 48       PHA          ; REGISTER WEGZETTEN
E634 8A       TXA
E635 48       PHA
E636 A5 D0    LDA $D0      ; CR-FLAG
E638 FO 93    BEQ $E5CD    ; NEE, DAN NAAR WACHTLUS
E63A A4 D3    LDY $D3      ; KOLOM
E63C B1 D1    LDA ($D1),Y  ; TEKEN VAN BEELDSCHERM HALEN
E63E 85 D7    STA $D7
E640 29 3F    AND #$3F
E642 Q6 D7    ASL $D7      ; EN VERANDEREN IN ASCII
E644 24 D7    BIT $D7
E646 10 02    BPL $E64A
E648 09 80    ORA #$80

```

E64A	90 04	BCC	\$E650	
E64C	A6 D4	LDX	\$D4	
E64E	D0 04	BNE	\$E654	
E650	70 02	BVS	\$E654	
E652	09 40	ORA	#\$40	
E654	E6 D3	INC	\$D3	; CURSOR EEN PLAATS VERSCHUIVEN
E656	20 84 E6	JSR	\$E684	; AANHALINGSTEKEN?
E659	C4 C8	CPY	\$C8	; CURSOR IN LAATSTE KOLOM?
E65B	D0 17	BNE	\$E674	
E65D	A9 00	LDA	#\$00	
E65F	85 D0	STA	\$D0	; CR-FLAG
E661	A9 0D	LDA	#\$0D	
E663	A6 99	LDX	\$99	
E665	E0 03	CPX	#\$03	; INVOER VAN BEELDSCHERM?
E667	F0 06	BEQ	\$E66F	; JA
E669	A6 9A	LDX	\$9A	; UITVOER OP BEELDSCHERM
E66B	E0 03	CPX	#\$03	; JA
E66D	F0 03	BEQ	\$E672	
E66F	20 16 E7	JSR	\$E716	; TEKEN OP BEELDSCHERM SCHRIJVEN
E672	A9 0D	LDA	#\$0D	
E674	85 D7	STA	\$D7	
E676	68	PLA		
E677	AA	TAX		; REGISTER TERUGHALEN
E678	68	PLA		
E679	A8	TAY		
E67A	A5 D7	LDA	\$D7	; BEELDSCHERM-CODE
E67C	C9 DE	CMP	#\$DE	; MET CODE VOOR PI VERGELIJKEN
E67E	D0 02	BNE	\$E682	
E680	A9 FF	LDA	#\$FF	; JA, DOOR BASIC-CODE VOOR PI VERVANGEN
E682	18	CLC		
E683	60	RTS		
***** AANHALINGSTEKEN?				
E684	C9 22	CMP	#\$22	; AANHALINGSTEKEN?
E686	D0 08	BNE	\$E690	; NEE, DAN KLAAR
E688	A5 D4	LDA	\$D4	
E68A	49 01	EOR	#\$01	; FLAG VOOR AANHALINGSTEKEN OMDRAAIEN
E68C	85 D4	STA	\$D4	
E68E	A9 22	LDA	#\$22	; AANHALINGSTEKEN-CODE WEER INSTELLEN
E690	60	RTS		
***** TEKEN OP BEELDSCHERM UITVOEREN				
E691	09 40	ORA	#\$40	
E693	A6 C7	LDX	\$C7	; RVS?
E695	F0 02	BEQ	\$E699	; VERANDEREN IN BEELDSCHERM CODE
E697	09 80	ORA	#\$80	; JA, DAN BIT 7 ZETTEN
E699	A6 D8	LDX	\$D8	

```

E69B F0 02    BEQ $E69F
E69D 46 D4    LSR $D4
E69F AE 86 02 LDX $0286      ; KLEURCODE
E6A2 20 13 EA JSR $EA13      ; TEKEN IN BEELDSCHERM-RAM
;                               SCHRIJVEN
E6A5 20 B6 E6 JSR $E6B6      ; TABEL VAN BEGIN VAN REGELS
;                               ACTUALISEREN
E6A8 68        PLA
E6A9 A8        TAY
E6AA A5 D8     LDA $D8
E6AC F0 02     BEQ $E6B0
E6AF 46 D4     LSR $D4      ; AANHALINGSTEKEN-MODUS WISSEN
E6B0 68        PLA
E6B1 AA        TAX
E6B2 68        PLA
E6B3 18        CLC
E6B4 58        CLI
E6B5 60        RTS
***** MSB VOOR REGELBEGIN OPNIEUW
;       BEREKENEN
E6B6 20 B3 E8 JSR $E8B3
E6B9 E6 D3     INC $D3
E6BB A5 D5     LDA $D5
E6BD C5 D3     CMP $D3
E6BF B0 3F     BCS $E700
E6C1 C9 4F     CMP #$4F      ; 79 TEKENS (DUBBELLE REGEL?)
E6C3 F0 32     BEQ $E6F7
E6C5 AD 92 02 LDA $0292
E6C8 F0 03     BEQ $E6CD
E6CA 4C 67 E9 JMP $E967
E6CD A6 D6     LDX $D6      ; REGEL
E6CF E0 19     CPX #$19     ; 25?
E6D1 90 07     BCC $E6DA
E6D3 20 EA E8 JSR $E8EA
E6D6 C6 D6     DEC $D6
E6D8 A6 D6     LDX $D6
E6DA 16 D9     ASL $D9,X
E6DC 56 D9     LSR $D9,X
E6DE E8        INX
E6DF B5 D9     LDA $D9,X
E6E1 09 80     ORA #$80
E6E3 95 D9     STA $D9,X
E6E5 CA        DEX
E6E6 A5 D5     LDA $D5
E6E8 18        CLC
E6E9 69 28     ADC #$28     ; +40
E6EB 85 D5     STA $D5
E6ED B5 D9     LDA $D9,X
E6EF 30 03     BMI $E6F4

```

```

E6F1  CA      DEX
E6F2  DO F9    BNE $E6ED
E6F4  4C FO E9 JMP $E9FO      ; WIJZER OP KLEURRAM VOOR
;                                REGEL X
E6F7  C6 D6    DEC $D6        ; REGEL VERLAGEN
E6F9  20 7C E8 JSR $E87C
E6FC  A9 00    LDA #$00
E6FE  85 D3    STA $D3        ; KOLOM=0
E700  60      RTS
***** TERUG NAAR VORIGE REGEL
E701  A6 D6    LDX $D6        ; CURSORREGEL
E703  DO 06    BNE $E70B      ; NUL?
E705  86 D3    STX $D3        ; CURSORKOLOM
E707  68      PLA
E708  68      PLA
E709  DO 9D    BNE $E6A8
E70B  CA      DEX
E70C  86 D6    STX $D6        ; REGELNUMMER VERLAGEN
E70E  20 6C E5 JSR $E56C      ; CURSORPOSITIE BEREKENEN
E711  A4 D5    LDY $D5
E713  84 D3    STY $D3
E715  60      RTS
***** UITVOER OP BEELDSCHERM
E716  48      PHA
E717  85 D7    STA $D7        ; TEKEN ONTHOUDEN
E719  8A      TXA
E71A  48      PHA            ; REGISTER WEGZETTEN
E71B  98      TYA
E71C  48      PHA
E71D  A9 00    LDA #$00
E71F  85 DO    STA $DO
E721  A4 D3    LDY $D3
E723  A5 D7    LDA $D7
E725  10 03    BPL $E72A
E727  4C D4 E7 JMP $E7D4      ; TEKEN > $7F BEHANDELEN
E72A  C9 OD    CMP #$0D      ; CARRIAGE RETURN?
E72C  DO 03    BNE $E731
E72E  4C 91 E8 JMP $E891      ; RETURN UITVOEREN
E731  C9 20    CMP #$20      ; SPATIE
E733  90 10    BCC $E745      ; PRINTBARE TEKENS?
E735  C9 60    CMP #$60
E737  90 04    BCC $E73D
E739  29 DF    AND #$DF
E73B  DO 02    BNE $E73F
E73D  29 3F    AND #$3F
E73F  20 84 E6 JSR $E684      ; AANHALINGSTEKEN?
E742  4C 93 E6 JMP $E693      ; NAAR UITVOER, ASCII-CODE
;                                VERANDEREN IN BEELDSCHERMCODE
E745  A6 D8    LDX $D8

```

E747	FO 03	BEQ \$E74C	
E749	4C 97 E6	JMP \$E697	
E74C	C9 14	CMP #\$14	; DEL?
E74E	DO 2E	BNE \$E77E	
E750	98	TYA	
E751	DO 06	BNE \$E759	
E753	20 01 E7	JSR \$E701	; TERUG NAAR VORIGE REGEL
E756	4C 73 E7	JMP \$E773	
E759	20 A1 E8	JSR \$E8A1	
E75C	88	DEY	
E75D	84 D3	STY \$D3	
E75F	20 24 EA	JSR \$EA24	; WIJZER OP KLEURRAM BEREKENEN
E762	C8	INY	
E763	B1 D1	LDA (\$D1),Y	; TEKEN VAN BEELDSCHERM
E765	88	DEY	
E766	91 D1	STA (\$D1),Y	; 1 KOLOM NAAR LINKS SCHUIVEN
E768	C8	INY	
E769	B1 F3	LDA (\$F3),Y	; KLEUR
E76B	88	DEY	
E76C	91 F3	STA (\$F3),Y	; 1 KOLOM NAAR LINKS SCHUIVEN
E76E	C8	INY	
E76F	C4 D5	CPY \$D5	
E771	DO EF	BNE \$E762	
E773	A9 20	LDA #\$20	; SPATIE
E775	91 D1	STA (\$D1),Y	
E777	AD 86 02	LDA \$0286	
E77A	91 F3	STA (\$F3),Y	; KLEURCODE INSTELLEN
E77C	10 4D	BPL \$E7CB	; KLAAR
E77E	A6 D4	LDX \$D4	; AANHALINGSMODUS
E780	FO 03	BEQ \$E785	; NEE
E782	4C 97 E6	JMP \$E697	; TEKEN REVERSE UITVOEREN
E785	C9 12	CMP #\$12	; RVS ON?
E787	DO 02	BNE \$E78B	; NEE
E789	85 C7	STA \$C7	; FLAG VOOR RVS ZETTEN
E78B	C9 13	CMP #\$13	; HOME
E78D	DO 03	BNE \$E792	; NEE
E78F	20 66 E5	JSR \$E566	; JA, CURSOR HOME
E792	C9 1D	CMP #\$1D	; CURSOR RIGHT?
E794	DO 17	BNE \$E7AD	; NEE
E796	C8	INY	
E797	20 B3 E8	JSR \$E8B3	
E79A	84 D3	STY \$D3	
E79C	88	DEY	
E79D	C4 D5	CPY \$D5	
E79F	90 09	BCC \$E7AA	
E7A1	C6 D6	DEC \$D6	
E7A3	20 7C E8	JSR \$E87C	
E7A6	A0 00	LDY #\$00	
E7A8	84 D3	STY \$D3	; KOLOM=0

```

E7AA 4C AE E8 JMP $E6A8 ; KLAAR
E7AD C9 11 CMP #$11 ; CURSOR DOWN?
E7AF D0 1D BNE $E7CE ; NEE
E7B1 18 CLC
E7B2 98 TYA
E7B3 69 28 ADC #$28
E7B5 A8 TAY
E7B6 E6 D6 INC $D6
E7B8 C5 D5 CMP $D5
E7BA 90 EC BCC $E7A8
E7BC F0 EA BEQ $E7A8
E7BE C6 D6 DEC $D6
E7C0 E9 28 SBC #$28 ; -40
E7C2 90 04 BCC $E7C8
E7C4 85 D3 STA $D3
E7C6 D0 F8 BNE $E7C0
E7C8 20 7C E8 JSR $E87C
E7CB 4C AE E8 JMP $E6A8 ; KLAAR
E7CE 20 CB E8 JSR $E8CB ; KLEUR-CODE?
E7D1 4C 44 EC JMP $EC44 ; ANDERE SPECIALE TEKENS?
***** TEKEN GROTER DAN $127
E7D4 29 7F AND #$7F ; CODE > 127, DAN BIT 7 WISSEN
E7D6 C9 7F CMP #$7F ; PI?
E7D8 D0 02 BNE $E7DC
E7DA A9 5E LDA #$5E ; BEELDSCHERMCODE VOOR PI
E7DC C9 20 CMP #$20 ; STUURTEKEN?
E7DE 90 03 BCC $E7E3 ; JA
E7E0 4C 91 E6 JMP $E691 ; PRINTBARE TEKENS UITVOEREN
E7E3 C9 0D CMP #$0D ; SHIFT/RETURN?
E7E5 D0 03 BNE $E7EA
E7E7 4C 91 E8 JMP $E891 ; NIEUWE REGEL
E7EA A6 D4 LDX $D4 ; AANHALINGSMODUS?
E7EC D0 3F BNE $E82D ; JA, STUURTEKEN REVERSE
; UITVOEREN
E7EE C9 14 CMP #$14 ; INS?
E7F0 D0 37 BNE $E829
E7F2 A4 D5 LDY $D5 ; REGELLENTE
E7F4 B1 D1 LDA ($D1),Y ; LAATSTE TEKEN IN REGEL
E7F6 C9 20 CMP #$20 ; GELIJK AAN SPATIE?
E7F8 D0 04 BNE $E7FE
E7FA C4 D3 CPY $D3 ; CURSOR IN LAATSTE KOLOM?
E7FC D0 07 BNE $E805
E7FE C0 4F CPY #$4F ; 79? MAXIMALE REGELLENTE
E800 F0 24 BEQ $E826 ; LAATSTE KOLOM, DAN GEEN ACTIE
E802 20 65 E9 JSR $E965 ; SPATIE INVOREN
E805 A4 D5 LDY $D5 ; REGELLENTE
E807 20 24 EA JSR $EA24 ; WIJZER OP KLEURRAM BEREKENEN
E80A 88 DEY
E80B B1 D1 LDA ($D1),Y ; TEKEN VAN BEELDSCHERM

```

```

E80D C8      INY
E80E 91 D1   STA ($D1),Y ; 1 NAAR RECHTS
E810 88      DEY
E811 B1 F3   LDA ($F3),Y ; EN KLEUR
E813 C8      INY
E814 91 F3   STA ($F3),Y ; VERSCHUIVEN
E816 88      DEY
E817 C4 D3   CPY $D3      ; EN NAAR ACTUELE POSITIE
E819 D0 EF   BNE $E80A
E81B A9 20   LDA #$20     ; SPATIE
E81D 91 D1   STA ($D1),Y ; OP ACTUELE POSITIE SCHRIJVEN
E81F AD 86 02 LDA $0286    ; KLEUR
E822 91 F3   STA ($F3),Y ; ZETTEN
E824 E6 D8   INC $D8      ; AANTAL INSERTS VERHOGEN
E826 4C AE E6 JMP $E6AE
E829 A6 D8   LDX $D8
E82B F0 05   BEQ $E832
E82D 09 40   ORA #$40
E82F 4C 97 E6 JMP $E697
*****
E832 C9 11   CMP #$11     ; CURSOR UP
E834 D0 16   BNE $E84C
E836 A6 D6   LDX $D6      ; REGEL
E838 F0 37   BEQ $E871    ; 0, DAN KLAAR
E83A C6 D6   DEC $D6      ; REGELNUMMER 1 VERLAGEN
E83C A5 D3   LDA $D3      ; KOLOM
E83E 38      SEC
E83F E9 28   SBC #$28     ; -40
E841 90 04   BCC $E847
E843 85 D3   STA $D3      ; CURSOR KOLOM
E845 10 2A   BPL $E871    ; POSITIEF, OK
E847 20 6C E5 JSR $E56C   ; BEELDSCHERMWIJZER
;                               OPNIEUW ZETTEN
E84A D0 25   BNE $E871
E84C C9 12   CMP #$12     ; RVS OFF
E84E D0 04   BNE $E854
E850 A9 00   LDA #$00
E852 85 C7   STA $C7      ; RVS-FLAG WISSEN
E854 C9 1D   CMP #$1D     ; CURSOR LEFT
E856 D0 12   BNE $E86A
E858 98      TYA
E859 F0 09   BEQ $E864
E85B 20 A1 E8 JSR $E8A1
E85E 88      DEY
E85F 84 D3   STY $D3      ; CURSORKOLOM
E861 4C AE E6 JMP $E6AE   ; KLAAR
E864 20 01 E7 JSR $E701   ; TERUG NAAR VORIGE REGEL
E867 4C AE E6 JMP $E6AE   ; KLAAR
E86A C9 13   CMP #$13     ; CLR SCREEN?

```


E86C	DO	06	BNE	\$E874	
E86E	20	44	E5	JSR	\$E544 ; BEELDSCHERM WISSEN
E871	4C	AE	E6	JMP	\$E6AE ; KLAAR
E874	09	80	ORA	#\$80	; BIT 7 WEER INSTELLEN
E876	20	CB	E8	JSR	\$E8CB ; KLEURCODE?
E879	4C	4F	EC	JMP	\$EC4F ; VAN TEKST NAAR GRAFIEK?
E87C	46	C9	LSR	\$C9	
E87E	A6	D6	LDX	\$D6	
E880	E8		INX		
E881	E0	19	CPX	#\$19	; 25, LAATSTE REGEL
E883	DO	03	BNE	\$E888	
E885	20	EA	E8	JSR	\$E8EA ; BEELDSCHERM SCROLLLEN
E888	B5	D9	LDA	\$D9,X	
E88A	10	F4	BPL	\$E880	
E88C	86	D6	STX	\$D6	
E88E	4C	6C	E5	JMP	\$E56C ; CURSORPOSITIE BEREKENEN
E891	A2	00	LDX	#\$00	
E893	86	D8	STX	\$D8	
E895	86	C7	STX	\$C7	; FLAGS WISSEN
E897	86	D4	STX	\$D4	
E899	86	D3	STX	\$D3	
E89B	20	7C	E8	JSR	\$E87C
E89E	4C	AE	E6	JMP	\$E6AE ; KLAAR
E8A1	A2	02	LDX	#\$02	
E8A3	A9	00	LDA	#\$00	
E8A5	C5	D3	CMP	\$D3	
E8A7	F0	07	BEQ	\$E8B0	
E8A9	18		CLC		
E8AA	69	28	ADC	#\$28	; +40 (1 REGEL)
E8AC	CA		DEX		
E8AD	DO	F6	BNE	\$E8A5	
E8AF	60		RTS		
E8B0	C6	D6	DEC	\$D6	
E8B2	60		RTS		
E8B3	A2	02	LDX	#\$02	
E8B5	A9	27	LDA	#\$27	; 39 (LAATSTE KOLOM)
E8B7	C5	D3	CMP	\$D3	
E8B9	F0	07	BEQ	\$E8C2	
E8BB	18		CLC		
E8BC	69	28	ADC	#\$28	; +40
E8BE	CA		DEX		
E8BF	DO	F6	BNE	\$E8B7	
E8C1	60		RTS		
E8C2	A6	D6	LDX	\$D6	
E8C4	E0	19	CPX	#\$19	; 25
E8C6	F0	02	BEQ	\$E8CA	
E8C8	E6	D6	INC	\$D6	
E8CA	60		RTS		

```

***** KLEURCODE?
E8CB A2 OF LDX #$OF ; AANTAL CODES
E8CD DD DA E8 CMP $E8DA,X ; MET TABEL VOOR KLEURCODES
; VERGELIJKEN
E8D0 F0 04 BEQ $E8D6 ; GEVONDEN
E8D2 CA DEX
E8D3 10 F8 BPL $E8CD
E8D5 60 RTS
E8D6 8E 86 02 STX $0286 ; KLEURCODE INSTELLEN
E8D9 60 RTS
***** KLEURCODETABEL
E8DA 90 05 1C 9F 9C 1E 1F 9E
E8E2 81 95 96 97 98 99 9A 9B
***** BEELDSCHERM SCROLLEN
E8EA A5 AC LDA $AC
E8EC 48 PHA
E8ED A5 AD LDA $AD
E8EF 48 PHA ; WIJZER WEGZETTEN
E8F0 A5 AE LDA $AE
E8F2 48 PHA
E8F3 A5 AF LDA $AF
E8F5 48 PHA
E8F6 A2 FF LDX #$FF ; BIJ REGEL 0 BEGINNEN
E8F8 C6 D6 DEC $D6 ; REGELNUMMER VERLAGEN
E8FA C6 C9 DEC $C9
E8FC CE A5 A2 DEC $02A5
E8FF E8 INX
E900 20 F0 E9 JSR $E9F0 ; WIJZER OP VIDEO-RAM
; VOOR REGEL X
E903 E0 18 CPX #$18 ; 24
E905 B0 0C BCS $E913 ; REEDS ALLE TEKENS?
E907 BD F1 EC LDA $ECF1,X ; LSB HALEN
E90A 85 AC STA $AC
E90C B5 DA LDA $DA,X ; MSB
E90E 20 C8 E9 JSR $E9C8 ; BEELDSCHERMREGEL 1 OMHOOG
E911 30 EC BMI $E8FF ; VOLGENDE REGEL
E913 20 FF E9 JSR $E9FF ; ONDERSTE REGEL WISSEN
E916 A2 00 LDX #$00
E918 B5 D9 LDA $D9,X
E91A 29 7F AND #$7F
E91C B4 DA LDY $DA,X
E91E 10 02 BPL $E922
E920 09 80 ORA #$80
E922 95 D9 STA $D9,X
E924 E8 INX
E925 E0 18 CPX #$18 ; 24
E927 D0 EF BNE $E918
E929 A5 F1 LDA $F1
E92B 09 80 ORA #$80

```

```

E92D 85 F1 STA $F1
E92F A5 D9 LDA $D9
E931 10 C3 BPL $E8F6
E933 E6 D6 INC $D6
E935 EE A5 02 INC $02A5
E938 A9 7F LDA #$7F
E93A 8D 00 DC STA $DC00
E93D AD 01 DC LDA $DC01
E940 C9 FB CMP #$FB ; CTRL INGEDRUKT?
E942 08 PHP
E943 A9 7F LDA #$7F
E945 8D 00 DC STA $DC00
E948 28 PLP
E949 D0 0B BNE $E956 ; NEE
E94B A0 00 LDY #$00
E94D EA NOP
E94E CA DEX
E94F D0 FC BNE $E94D ; VERTRAGINGSCLUS
E951 88 DEY
E952 D0 F9 BNE $E94D
E954 84 C6 STY $C6 ; AANTAL INGEDRUKTE TOETSEN=0
E956 A6 D6 LDX $D6
E958 68 PLA
E959 85 AF STA $AF
E95B 68 PLA
E95C 85 AE STA $AE ; WIJZER TERUGHALEN
E95E 68 PLA
E95F 85 AD STA $AD
E961 68 PLA
E962 85 AC STA $AC
E964 60 RTS
***** INVOEGEN VOORTZETTINGSREGEL
E965 A6 D6 LDX $D6 ; REGELNUMMER
E967 E8 INX
E968 B5 D9 LDA $D9,X
E96A 10 FB BPL $E967
E96C 8E A5 02 STX $02A5
E96F E0 18 CPX #$18 ; 24
E971 F0 0E BEQ $E981
E973 90 0C BCC $E981
E975 20 EA E8 JSR $E8EA ; BEELDSCHERM SCROLLLEN
E978 AE A5 02 LDX $02A5
E97B CA DEX
E97C C6 D6 DEC $D6 ; REGELNUMMER VERLAGEN
E97E 4C DA E6 JMP $E6DA
*****
E981 A5 AC LDA $AC
E983 48 PHA
E984 A5 AD LDA $AD

```

```

E986 48      PHA
E987 A5 AE   LDA $AE
E989 48      PHA
E98A A5 AF   LDA $AF
E98C 48      PHA
E98D A2 19   LDX #$19      ; 25
E98F CA      DEX          ; REGELNUMMER
E990 20 F0 E9 JSR $E9F0    ; WIJZER OP KLEUR-RAM BEREKENEN
E993 EC A5 02 CPX $02A5
E996 90 OE   BCC $E9A6
E998 F0 OC   BEQ $E9A6
E99A BD EF EC LDA $ECEF,X  ; LSB VAN REGELBEGIN INSTELLEN
E99D 85 AC   STA $AC
E99F B5 D8   LDA $D8,X     ; MSB INSTELLEN
E9A1 20 C8 E9 JSR $E9C8    ; REGEL 1 OMHOOG
E9A4 30 E9   BMI $E98F
E9A6 20 FF E9 JSR $E9FF    ; BEELDSCHERMREGEL WISSEN
E9A9 A2 17   LDX #$17     ; 23
E9AB EC A5 02 CPX $02A5
E9AE 90 OF   BCC $E9BF
E9B0 B5 DA   LDA $DA,X
E9B2 29 7F   AND #$7F
E9B4 B4 D9   LDY $D9,X
E9B6 10 02   BPL $E9BA
E9B8 09 80   ORA #$80
E9BA 95 DA   STA $DA,X
E9BC CA      DEX
E9BD D0 EC   BNE $E9AB
E9BF AE A5 02 LDX $02A5
E9C2 20 DA E6 JSR $E6DA    ; MSB OPNIEUW BEREKENEN
E9C5 4C 58 E9 JMP $E958    ; REGISTER TERUGHALEN, RTS
***** REGEL NAAR BOVEN VERSCHUIVEN
E9C8 29 03   AND #$03
E9CA OD 88 02 ORA $0288    ; BEELDSCHERMWIJZER VOOR
;                               NIEUWE REGEL
E9CD 85 AD   STA $AD
E9CF 20 E0 E9 JSR $E9E0    ; WIJZER VOOR NIEUWE REGEL
;                               BEREKENEN
E9D2 A0 27   LDY #$27     ; 39 TEKENS
E9D4 B1 AC   LDA ($AC),Y
E9D6 91 D1   STA ($D1),Y  ; TEKEN
E9D8 B1 AE   LDA ($AE),Y
E9DA 91 F3   STA ($F3),Y  ; EN KLEUR OVERDRAGEN
E9DC 88      DEY
E9DD 10 F5   BPL $E9D4    ; ALLE KOLOMMEN?
E9DF 60      RTS
***** BEELDSCHERMREGEL VOOR
; SCROLLREGEL BEREKENEN
E9E0 20 24 EA JSR $EA24    ; WIJZER OP KLEUR-RAM BEREKENEN

```

```

E9E3  A5 AC      LDA $AC
E9E5  85 AE      STA $AE
E9E7  A5 AD      LDA $AD
E9E9  29 03      AND #$03      ; WIJZER IN $AE/$AF
E9EB  09 D8      ORA #$D8
E9ED  85 AF      STA $AF
E9EF  60          RTS
*****
;                               WIJZER OP VIDEO-RAM
;                               VOOR REGEL X
E9F0  BD F0 EC LDA $ECF0,X      ; LSB HALEN
E9F3  85 D1      STA $D1
E9F5  B5 D9      LDA $D9,X      ; MSB HALEN
E9F7  29 03      AND #$03
E9F9  OD 88 02  ORA $0288      ; MSB VAN VIDEO-RAM
E9FC  85 D2      STA $D2
E9FE  60          RTS
*****
;                               BEELDSCHERMREGEL X WISSEN
E9FF  AO 27      LDY #$27      ; 39 KOLOMMEN
EA01  20 F0 E9 JSR $E9F0      ; WIJZER OP VIDEO-RAM ZETTEN
EA04  20 24 EA JSR $EA24      ; WIJZER OP KLEUR-RAM ZETTEN
EA07  20 DA E4 JSR $E4DA      ; ACHTERGRONDKLEUR INSTELLEN
EA0A  A9 20      LDA #$20      ; SPATIE
EA0C  91 D1      STA ($D1),Y   ; ZETTEN
EA0E  88          DEY
EA0F  10 F6      BPL $EA07      ; AL KOLOM 0?
EA11  60          RTS
*****
EA13  A8          TAY
EA14  A9 02      LDA #$02
EA16  85 CD      STA $CD      ; KNIPPERTELLER DOOR REPEAT-
;                               FUNCTIE INSTELLEN
EA18  20 24 EA JSR $EA24      ; ADRES OP KLEUR-RAM BEREKENEN
EA1B  98          TYA
*****
;                               TEKEN EN KLEUR OP BEELDSCHERM
EA1C  A4 D3      LDY $D3      ; KOLOPOSITIE
EA1E  91 D1      STA ($D1),Y   ; TEKEN IN ACCU OP BEELDSCHERM
EA20  8A          TXA      ; KLEURCODE IN X
EA21  91 F3      STA ($F3),Y   ; IN KLEUR-RAM SCHRIJVEN
EA23  60          RTS
*****
;                               WIJZER OP KLEUR-RAM BEREKENEN
EA24  A5 D1      LDA $D1      ; $D1/$D2=WIJZER OP POSITIE
;                               VAN VIDEO-RAM
EA26  85 F3      STA $F3
EA28  A5 D2      LDA $D2
EA2A  29 03      AND #$03
EA2C  09 D8      ORA #$D8      ; MSB=$D8
EA2E  85 F4      STA $F4      ; $F3/$F4=WIJZER OP KLEUR-RAM
EA30  60          RTS
*****
;                               INTERRUPTROUTINE

```

EA31	20	EA	FF	JSR	\$FFEA	; STOPTOETS, TIJD VERHOGEN
EA34	A5	CC		LDA	\$CC	; MAG CURSOR KNIPPEN?
EA36	D0	29		BNE	\$EA61	; ONGELIJK NUL, DAN NIET
						KNIPPEN
EA38	C6	CD		DEC	\$CD	; VERLAAG KNIPPERTELLER
EA3A	D0	25		BNE	\$EA61	; ONGELIJK NUL, DAN VERDER
EA3C	A9	14		LDA	#\$14	; KNIPPERTELLER WEER OP 20
EA3E	85	CD		STA	\$CD	
EA40	A4	D3		LDY	\$D3	; CURSORKOLOM
EA42	46	CF		LSR	\$CF	; KNIPPERSCHAKELAAR=0, DAN C=1
EA44	AE	87	02	LDX	\$0287	; KLEUR ONDER CURSOR
EA47	B1	D1		LDA	(\$D1),Y	; KLEURCODE INSTELLEN
EA49	BO	11		BCS	\$EA5C	; KNIPPERSCHAKELAAR WAS AAN,
						DAN VERDER
EA4B	E6	CF		INC	\$CF	; KNIPPERSCHAKELAAR AAN
EA4D	85	CE		STA	\$CE	; TEKEN ONDER CURSOR ONTHOUDEN
EA4F	20	24	EA	JSR	\$EA24	; WIJZER OP KLEUR-RAM BEREKENEN
EA52	B1	F3		LDA	(\$F3),Y	; KLEURCODE HALEN
EA54	8D	87	02	STA	\$0287	; EN ONTHOUDEN
EA57	AE	86	02	LDX	\$0286	; KLEURCODE ONDER CURSOR
EA5A	A5	CE		LDA	\$CE	; TEKEN ONDER CURSOR
EA5C	49	80		EOR	#\$80	; RVS-BIT OMDRAAIEN
EA5E	20	1C	EA	JSR	\$EA1C	; TEKEN EN KLEUR BEPALEN
EA61	A5	01		LDA	\$01	
EA63	29	10		AND	#\$10	; IS EEN TOETS VAN RECORDER
EA65	FO	0A		BEQ	\$EA71	; INGEDRUKT?
EA67	A0	00		LDY	#\$00	
EA69	84	CO		STY	\$CO	; RECORDER-FLAG ZETTEN
EA6B	A5	01		LDA	\$01	
EA6D	09	20		ORA	#\$20	; RECORDERMOTOR UIT
EA6F	D0	08		BNE	\$EA79	
EA71	A5	CO		LDA	\$CO	; RECORDER-FLAG
EA73	D0	06		BNE	\$EA7B	
EA75	A5	01		LDA	\$01	
EA77	29	1F		AND	#\$1F	; RECORDERMOTOR AAN
EA79	85	01		STA	\$01	
EA7B	20	87	EA	JSR	\$EA87	; TOETSENBOARD UITLEZEN
EA7E	AD	0D	DC	LDA	\$DCOD	; IRQ-FLAG WISSEN
EA81	68			PLA		
EA82	A8			TAY		
EA83	68			PLA		; REGISTERS HERSTELLEN
EA84	AA			TAX		
EA85	68			PLA		; VERLAAT INTERRUPT
EA86	40			RTI		
***** TOETSENBOARD UITLEZEN						
EA87	A9	00		LDA	#\$00	
EA89	8D	8D	02	STA	\$028D	; SHIFT/CTRL-FLAG WISSEN
EA8C	A0	40		LDY	#\$40	; \$40=GEEN TOETS INGEDRUKT
EA8E	84	CB		STY	\$CB	; CODE VOOR INGEDRUKTE TOETS

EA90	8D 00 DC	STA \$DC00	; ELKE MATRIXREGEL TESTEN
EA93	AE 01 DC	LDX \$DC01	
EA96	E0 FF	CPX #\$FF	; GEEN TOETS INGEDRUKT?
EA98	F0 61	BEQ \$EAFB	; DAN STOPPEN
EA9A	A8	TAY	
EA9B	A9 81	LDA #\$81	
EA9D	85 F5	STA \$F5	
EA9F	A9 EB	LDA #\$EB	
EAA1	85 F6	STA \$F6	; WIJZER OP TABEL 1 (\$EB81)
EAA3	A9 FE	LDA #\$FE	
EAA5	8D 00 DC	STA \$DC00	
EAA8	A2 08	LDX #\$08	; 8 MATRIXREGELS
EAAA	48	PHA	
EAAB	AD 01 DC	LDA \$DC01	
EAAE	CD 01 DC	CMP \$DC01	; TOETSENBORD VRIJGEVEN
EAB1	D0 F8	BNE \$EAAB	
EAB3	4A	LSR A	; BITS NA ELKAAR IN CARRY
			VERSCHUIVEN
EAB4	B0 16	BCS \$EACC	; 1=NIET GEDRUKT
EAB6	48	PHA	
EAB7	B1 F5	LDA (\$F5),Y	; ASCII-CODE UIT TABEL HALEN
EAB9	C9 05	CMP #\$05	
EABB	B0 0C	BCS \$EAC9	; GROTER DAN 5?
EABD	C9 03	CMP #\$03	
EABF	F0 08	BEQ \$EAC9	; STOP-CODE
EAC1	0D 8D 02	ORA \$028D	
EAC4	8D 8D 02	STA \$028D	; FLAG ZETTEN
EAC7	10 02	BPL \$EACB	
EAC9	84 CB	STY \$CB	; NUMMER VAN TOETS ONTHOUDEN
EACB	68	PLA	
EACC	C8	INY	
EACD	C0 41	CPY #\$41	
EACF	B0 0B	BCS \$EADC	; > \$40?
EAD1	CA	DEX	
EAD2	D0 DF	BNE \$EAB3	; VOLGENDE MATRIXKOLOM
EAD4	38	SEC	
EAD5	68	PLA	
EAD6	2A	ROL	
EAD7	8D 00 DC	STA \$DC00	
EADA	D0 CC	BNE \$EAA8	; VOLGENDE MATRIX-REGEL
EADC	68	PLA	
EADD	6C 8F 02	JMP (\$028F)	; JMP \$EB48 ZET WIJZER OP TABEL
EAE0	A4 CB	LDY \$CB	; NUMMER VAN TOETS
EAE2	B1 F5	LDA (\$F5),Y	; ASCII-WAARDE UIT TABEL HALEN
EAE4	AA	TAX	
EAE5	C4 C5	CPY \$C5	; MET LAATSTE TOETS VERGELIJKEN
EAE7	F0 07	BEQ \$EAFO	

EAE9	AO 10	LDY #\$10	
EAE8	8C 8C 02	STY \$028C	; START WACHTTIJD VOOR REPETEREN
EAAE	DO 36	BNE \$EB26	
EAF0	29 7F	AND #\$7F	; BIT 7 WISSEN
EAF2	2C 8A 02	BIT \$028A	; REPEAT-FUNCTIE VOOR ALLE TOETSEN?
EAF5	30 16	BMI \$EB0D	; BIT 7 GEZET, DAN ALLE TOETSEN
EAF7	70 49	BVS \$EB42	; BIT 6 GEZET, DAN ALLEEN DE
EAF9	C9 7F	CMP #\$7F	; VOLGENDE TOETSEN HERHALEN
EAFB	F0 29	BEQ \$EB26	
EAFD	C9 14	CMP #\$14	; DEL-, INST-CODE
EAFF	F0 0C	BEQ \$EB0D	
EB01	C9 20	CMP #\$20	; SPATIE
EB03	F0 08	BEQ \$EB0D	
EB05	C9 1D	CMP #\$1D	; CURSOR RIGHT, LEFT
EB07	F0 04	BEQ \$EB0D	
EB09	C9 11	CMP #\$11	; CURSOR DOWN, UP
EB0B	DO 35	BNE \$EB42	
EB0D	AC 8C 02	LDY \$028C	; VERTRAGINGSTELLER VOOR
			REPEAT-FUNCTIE
EB10	F0 05	BEQ \$EB17	
EB12	CE 8C 02	DEC \$028C	; AFTELLEN
EB15	DO 2B	BNE \$EB42	
EB17	CE 8B 02	DEC \$028B	; TELLER VOOR REPEAT-SNELHEID
EB1A	DO 26	BNE \$EB42	
EB1C	AO 04	LDY #\$04	
EB1E	8C 8B 02	STY \$028B	; TELLER OPNIEUW INSTELLEN
EB21	A4 C6	LDY \$C6	; AANTAL TEKENS IN
			TOETSENBORDBUFFER
EB23	88	DEY	
EB24	10 1C	BPL \$EB42	; > 1, DAN NEGEREN
EB26	A4 CB	LDY \$CB	
EB28	84 C5	STY \$C5	
EB2A	AC 8D 02	LDY \$028D	
EB2D	8C 8E 02	STY \$028E	
EB30	E0 FF	CPX #\$FF	; TOETSENBORDCODE ONJUIST?
EB32	F0 0E	BEQ \$EB42	; JA, DAN NEGEREN
EB34	8A	TXA	
EB35	A6 C6	LDX \$C6	; AANTAL TEKENS IN
			TOETSENBORDBUFFER
EB37	EC 89 02	CPX \$0289	; MET MAXIMALE WAARDE
			VERGELIJKEN
EB3A	B0 06	BCS \$EB42	; BUFFER VOL, DAN TEKEN NEGEREN
EB3C	9D 77 02	STA \$0277,X	; TEKEN IN TOETSENBORDBUFFER
			SCHRIJVEN
EB3F	E8	INX	
EB40	86 C6	STX \$C6	; MAXIMALE WAARDE VERHOGEN
EB42	A9 7F	LDA #\$7F	; UITLEZEN VAN TOETSENBORDMATRIX
			OP DEFAULT
EB44	8D 00 DC	STA \$DC00	


```

EB47 60      RTS
***** CONTROLE VAN SHIFT, CTRL, C=
EB48 AD 8D 02 LDA $028D      ; FLAG VOOR SHIFT/CTRL
EB4B C9 03    CMP #$03
EB4D DO 15    BNE $EB64      ; WIJZER OP DECODERINGTABEL
; BEREKENEN
EB4F CD 8E 02 CMP $028E
EB52 FO EE    BEQ $EB42
EB54 AD 91 02 LDA $0291      ; SHIFT/COMMODORE TOEGESTAAN
EB57 30 1D    BMI $EB76      ; NEE, TERUG NAAR DECODERING
EB59 AD 18 DO LDA $D018      ; SHIFT/COMMODORE
EB5C 49 02    EOR #$02      ; OMSCHAKELING VAN KLEINE
; LETTERS NAAR HOOFDLETTERS
EB5E 8D 18 DO STA $D018
EB61 4C 76 EB JMP $EB76      ; KLAAR
EB64 0A      ASL
EB65 C9 08    CMP #$08
EB67 90 02    BCC $EB6B
EB69 A9 06    LDA #$06
EB6B AA      TAX
EB6C BD 79 EB LDA $EB79,X
EB6F 85 F5    STA $F5      ; WIJZER OP DECODEERTABEL
; VAN TOETSENBOARD LADEN
EB71 BD 7A EB LDA $EB7A,X
EB74 85 F6    STA $F6
EB76 4C EO EA JMP $EAEO      ; TERUG NAAR DECODERING
***** WIJZER OP DECODEERTABELLEN
; VAN TOETSENBOARD
EB79 81 EB C2 EB 03 EC 78 EC
***** DECODEERTABEL 1, ZONDER SHIFT
EB81 14 OD 1D 88 85 86 87 11
EB89 33 57 41 34 5A 53 45 01
EB91 35 52 44 36 43 46 54 58
EB99 37 59 47 38 42 48 55 56
EBA1 39 49 4A 30 4D 4B 4F 4E
EBA9 2B 50 4C 2D 2E 3A 40 2C
EBB1 5C 2A 3B 13 01 3D 5E 2F
EBB9 31 5F 04 32 20 02 51 03
EBC1 FF
***** DECODEERTABEL 2, MET SHIFT
EBC2 94 8D 9D 8C 89 8A 8B 91
EBC9 23 D7 C1 24 DA D3 C5 01
EBD2 25 D2 C4 26 C3 C6 D4 D8
EBDA 27 D9 C7 28 C2 C8 D5 D6
EBE2 29 C9 CA 30 CD CB CF CE
EBEA DB DO CC DD 3E 5B BA 3C
EBF2 A9 CO 5D 93 01 3D DE 3F
EBFA 21 5F 04 22 A0 02 D1 83

```

EC02 FF

C

***** TOETSENBOARD-DECODEERTABEL 3
MET 'C=' TOETS

;

EC03 94 8D 9D 8C 89 8A 8B 91
EC0B 96 B3 B0 97 AD AE B1 01
EC13 98 B2 AC 99 BC BB A3 BD
EC1B 9A B7 A5 9B BF B4 B8 BE
EC23 29 A2 B5 30 A7 A1 B9 AA
EC2B A6 AF B6 DC 3E 5B A4 3C
EC33 A8 DF 5D 93 01 3D DE 3F
EC3B 81 5F 04 95 A0 02 AB 83
EC43 FF

***** CONTROLEERD OP STUURTEKENS

EC44 C9 0E CMP m\$0E ; CHR\$(14)
EC46 D0 07 BNE \$EC4F
EC48 AD 18 D0 LDA \$D018 ; CHARACTER-GENERATOR
EC4B 09 02 ORA m\$02 ; OP HOOFDLETTER-MODUS
EC4D D0 09 BNE \$EC58
EC4F C9 8E CMP m\$8E ; CHR\$(142)
EC51 D0 0B BNE \$EC5E
EC53 AD 18 D0 LDA \$D018
EC56 29 FD AND m\$FD ; KLEINSCHRIFT-MODUS
EC58 8D 18 D0 STA \$D018 ; ZETTEN
EC5B 4C A8 E6 JMP \$E6A8
EC5E C9 08 CMP m\$08 ; CHR\$(8)
EC60 D0 07 BNE \$EC69
EC62 A9 80 LDA m\$80
EC64 0D 91 02 ORA \$0291 ; SHIFT-COMMODORE VERHINDEREN
EC67 30 09 BMI \$EC72
EC69 C9 09 CMP m\$09 ; CHR\$(9)
EC6B D0 EE BNE \$EC5B
EC6D A9 7F LDA m\$7F
EC6F 2D 91 02 AND \$0291 ; SHIFT-COMMODORE TOEGESTAAN
EC72 8D 91 02 STA \$0291
EC75 4C A8 E6 JMP \$E6A8

***** TOETSENBOARD-DECODEERTABEL 4
MET CTRL-TOETS

;

EC78 FF FF FF FF FF FF FF
EC80 1C 17 01 9F 1A 13 05 FF
EC88 9C 12 04 1E 03 06 14 18
EC90 1F 19 07 9E 02 18 15 16
EC98 12 09 0A 92 0D 0B 0F 0E
ECA0 FF 10 0C FF FF 1B 00 FF
ECA8 1C FF 1D FF FF 1F 1E FF
ECB0 90 06 FF 05 FF FF 11 FF
ECB8 FF

***** CONSTANTEN VOOR
VIDEOCONTROLLER

;

```

ECB9 00 00 00 00 00 00 00 00
ECC1 00 00 00 00 00 00 00 00
ECC9 00 9B 37 00 00 00 08 00
ECD1 14 0F 00 00 00 00 00 00
ECD9 0E 06 01 02 03 04 00 01
ECE1 02 03 04 05 06 07
***** TEKST NA INDRUKKEN
; SHIFT RUN/STOP
ECE7 4C 4F 41 44 0D 52 55 4E; LOAD (CR) RUN (CR)
ECEA 0D
***** TABEL LSB VAN
; BEELDSCHERMREGEL BEGIN
ECF0 00 28 50 78 A0 C8 F0 18
ECF8 40 68 90 B8 E0 08 30 58
ED00 80 A8 D0 F8 20 48 70 98
ED08 C0
***** TALK ZENDEN
ED09 09 40 ORA #$40 ; BIT VOOR TALK ZENDEN
ED0B 2C .BYTE 2C
***** LISTEN ZENDEN
ED0C 09 20 ORA #$20 ; BIT VOOR LISTEN ZENDEN
ED0E 20 A4 F0 JSR $FOA4 ; EINDE VAN OVERDRACHT VIA
; RS-232 AFWACHTEN
ED11 48 PHA
ED12 24 94 BIT $94 ; NOG EEN BYTE?
ED14 10 0A BPL $ED20 ; NEE
ED16 38 SEC
ED17 66 A3 ROR $A3
ED19 20 40 ED JSR $ED40 ; BYTE OP SERIELE BUS UITVOEREN
ED1C 46 94 LSR $94
ED1E 46 A3 LSR $A3
ED20 68 PLA
ED21 85 95 STA $95 ; IN UITVOERBUFFER
ED23 78 SEI
ED24 20 97 EE JSR $EE97 ; DATA HIGH
ED27 C9 3F CMP #$3F
ED29 D0 03 BNE $ED2E
ED2B 20 85 EE JSR $EE85 ; KLOK HIGH
ED2E AD 00 DD LDA $DD00
ED31 09 08 ORA #$08 ; ATN LOW MAKEN
ED33 8D 00 DD STA $DD00
ED36 78 SEI
ED37 20 8E EE JSR $EE8E ; KLOK LOW
ED3A 20 97 EE JSR $EE97 ; DATA HIGH
ED3D 20 B3 EE JSR $EEB3 ; 1 MILLISECONDE WACHTEN
***** 1 BYTE OP SERIELE BUS UITVOEREN
ED40 78 SEI
ED41 20 97 EE JSR $EE97 ; DATA HIGH
ED44 20 A9 EE JSR $EEA9 ; DATA IN CARRY

```

ED47	BO 64	BCS \$EDAD	; DATA LOW, DAN
;			DEVICE NOT PRESENT
ED49	20 85 EE	JSR \$EE85	; KLOK HIGH
ED4C	24 A3	BIT \$A3	
ED4E	10 0A	BPL \$ED5A	
ED50	20 A9 EE	JSR \$EEA9	; DATA IN CARRY
ED53	90 FB	BCC \$ED50	; WACHTEN OP DATA LOW
ED55	20 A9 EE	JSR \$EEA9	; DATA IN CARRY
ED58	BO FB	BCS \$ED55	; WACHTEN OP DATA HIGH
ED5A	20 A9 EE	JSR \$EEA9	; DATA IN CARRY
ED5D	90 FB	BCC \$ED5A	; WACHTEN OP DATA LOW
ED5F	20 8E EE	JSR \$EE8E	; KLOK HIGH
ED62	A9 08	LDA #\$08	
ED64	85 A5	STA \$A5	; BITTELLER VOOR SERIELE
;			UITVOER INSTELLEN
ED66	AD 00 DD	LDA \$DD00	
ED69	CD 00 DD	CMP \$DD00	
ED6C	DO F8	BNE \$ED66	
ED6E	0A	ASL A	; DATA IN CARRY
ED6F	90 3F	BCC \$EDB0	; DATA HIGH, DAN TIME OUT
ED71	66 95	ROR \$95	; VOLGEND BIT VOOR UITVOER
;			VOORBEREIDEN
ED73	BO 05	BCS \$ED7A	; BIT GEZET?
ED75	20 A0 EE	JSR \$EEA0	; NEE, DAN DATA LOW UITVOEREN
ED78	DO 03	BNE \$ED7D	
ED7A	20 97 EE	JSR \$EE97	; DATA HIGH UITVOEREN
ED7D	20 85 EE	JSR \$EE85	; KLOK HIGH
ED80	EA	NOP	
ED81	EA	NOP	
ED82	EA	NOP	
ED83	EA	NOP	
ED84	AD 00 DD	LDA \$DD00	
ED87	29 DF	AND #\$DF	; DATA HIGH
ED89	09 10	ORA #\$10	; EN KLOK LOW
ED8B	8D 00 DD	STA \$DD00	; UITVOEREN
ED8E	C6 A5	DEC \$A5	; VOLGEND BIT
ED90	DO D4	BNE \$ED66	
ED92	A9 04	LDA #\$04	
ED94	8D 07 DC	STA \$DC07	; TIMER HIGH, CA. 1 MILLISECONDE
ED97	A9 19	LDA #\$19	
ED99	8D 0F DC	STA \$DC0F	; TIMER STARTEN
ED9C	AD 0D DC	LDA \$DC0D	
ED9F	AD 0D DC	LDA \$DC0D	
EDA2	29 02	AND #\$02	; TIMER AFGELOPEN?
EDA4	DO 0A	BNE \$EDB0	; JA, DAN TIME OUT
EDA6	20 A9 EE	JSR \$EEA9	; DATA IN CARRY
EDA9	BO F4	BCS \$ED9F	; WACHTEN OP DATA HIGH
EDAB	58	CLI	
EDAC	60	RTS	

```

*****
EDAD A9 80 LDA #$80 ; DEVICE NOT PRESENT
EDAF 2C .BYTE $2C
EDB0 A9 03 LDA #$03 ; TIME OUT
EDB2 20 1C FE JSR $FE1C ; STATUS ZETTEN
EDB5 58 CLI
EDB6 18 CLC
EDB7 90 4A BCC $EE03
***** SECUNDAIRE ADRES NA LISTEN
; UITVOEREN
EDB9 85 95 STA $95 ; SECUNDAIRE ADRES OPSLAAN
EDBB 20 36 ED JSR $ED36 ; BIJ ATN LOW UITVOEREN
EDBE AD 00 DD LDA $DD00
EDC1 29 F7 AND #$F7 ; ATN TERUGZETTEN, HIGH
EDC3 8D 00 DD STA $DD00
EDC6 60 RTS
***** SECUNDAIRE ADRES NAAR TALK
; UITVOEREN
EDC7 85 95 STA $95 ; SECUNDAIRE ADRES OPSLAAN
EDC9 20 36 ED JSR $ED36 ; BIJ ATN UITVOEREN
EDCC 78 SEI
EDCD 20 A0 EE JSR $EEA0 ; DATA LOW
EDDO 20 BE ED JSR $EDBE ; ATN TERUGZETTEN, HIGH
EDD3 20 85 EE JSR $EE85 ; KLOK HIGH
EDD6 20 A9 EE JSR $EEA9 ; DATA HALEN
EDD9 30 FB BMI $EDD6 ; OP DATA HIGH WACHTEN
EDDB 58 CLI
EDDC 60 RTS
***** IECOUT: 1 BYTE OP IEC-BUS
; UITVOEREN
EDDD 24 94 BIT $94 ; NOG EEN BYTE UITVOEREN?
EDDF 30 05 BMI $EDE6 ; JA
EDE1 38 SEC
EDE2 66 94 ROR $94 ; FLAG VOOR GEBUFFERDE BYTE
; ZETTEN
EDE4 D0 05 BNE $EDEB ; CONDITONELE SPRONG
EDE6 48 PHA ; BYTE ONTHOUDEN
EDE7 20 40 ED JSR $ED40 ; TEKEN UITVOEREN
EDEA 68 PLA ; BYTE TERUGHALEN
EDEB 85 95 STA $95 ; EN IN UITVOERREGISTER ZETTEN
EDED 18 CLC
EDEE 60 RTS
***** UNTALK ZENDEN
EDEF 78 SEI
EDF0 20 8E EE JSR $EE8E ; KLOK LOW
EDF3 AD 00 DD LDA $DD00
EDF6 09 08 ORA #$08 ; ATN LOW MAKEN
EDF8 8D 00 DD STA $DD00
EDFB A9 5F LDA #$5F

```

```

EDFD 2C          .BYTE 2C
***** UNLISTEN ZENDEN
EDFE A9 3F      LDA #$3F
EE00 20 11 ED JSR $ED11      ; UITVOEREN
EE03 20 BE ED JSR $EDBE      ; ATN TERUGZETTEN, HI
EE06 8A          TXA
EE07 A2 0A      LDX #$0A
EE09 CA          DEX          ; CA. 40 MICROSECONDEN WACHTEN
EEOA D0 FD      BNE $EE09
EEOC AA          TAX
EEOD 20 85 EE JSR $EE85      ; KLOK HIGH
EE10 4C 97 EE JMP $EE97      ; DATA HIGH
***** IECIN 1 TEKEN VAN
; IEC-BUS HALEN
EE13 78          SEI
EE14 A9 00      LDA #$00
EE16 85 A5      STA $A5
EE18 20 85 EE JSR $EE85      ; KLOK HIGH
EE1B 20 A9 EE JSR $EEA9      ; DATA HALEN
EE1E 10 FB      BPL $EE1B    ; WACHTEN OP DATA HIGH
EE20 A9 01      LDA #$01
EE22 8D 07 DC STA $DC07      ; TIMER HIGH
EE25 A9 19      LDA #$19
EE27 8D 0F DC STA $DC0F      ; TIMER STARTEN
EE2A 20 97 EE JSR $EE97      ; DATA HIGH
EE2D AD 0D DC LDA $DC0D
EE30 AD 0D DC LDA $DC0D      ; TIMER
EE33 29 02      AND #$02      ; AFGELOPEN?
EE35 D0 07      BNE $EE3E      ; JA, TIME OUT
EE37 20 A9 EE JSR $EEA9      ; DATA HALEN
EE3A 30 F4      BMI $EE30      ; WACHTEN OP DATA LOW
EE3C 10 18      BPL $EE56
EE3E A5 A5      LDA $A5
EE40 F0 05      BEQ $EE47
EE42 A9 02      LDA #$02      ; TIME OUT
EE44 4C B2 ED JMP $EDB2      ; STATUS ZETTEN
EE47 20 A0 EE JSR $EEA0      ; DATA LOW
EE4A 20 85 EE JSR $EE85      ; KLOK HIGH
EE4D A9 40      LDA #$40      ; EOF
EE4F 20 1C FE JSR $FE1C      ; STATUS ZETTEN
EE52 E6 A5      INC $A5
EE54 D0 CA      BNE $EE20
EE56 A9 08      LDA #$08
EE58 85 A5      STA $A5      ; BITTELLER ZETTEN
EE5A AD 00 DD LDA $DD00      ; GEGEVENSBIT HALEN
EE5D CD 00 DD CMP $DD00
EE60 D0 F8      BNE $EE5A
EE62 0A          ASL A          ; IN CARRY SCHUIVEN
EE63 10 F5      BPL $EE5A      ; BIT 0

```

```

EE65 66 A4 ROR $A4 ; NEE
EE67 AD 00 DD LDA $DD00
EE6A CD 00 DD CMP $DD00 ; GEGEVENS HALEN EN VRIJGEVEN
EE6D DO F8 BNE $EE67
EE6F 0A ASL
EE70 30 F5 BMI $EE67
EE72 C6 A5 DEC $A5 ; VOLGEND BIT
EE74 DO E4 BNE $EE5A
EE76 20 A0 EE JSR $EEA0 ; DATA LOW
EE79 24 90 BIT $90 ; STATUS
EE7B 50 03 BVC $EE80 ; GEEN EOF?
EE7D 20 06 EE JSR $EE06 ; WACHTEN EN BIT 1 ZENDEN
EE80 A5 A4 LDA $A4 ; GEGEVENSBYTE IN ACCU
EE82 58 CLI
EE83 18 CLC
EE84 60 RTS
***** KLOK HIGH
EE85 AD 00 DD LDA $DD00
EE88 29 EF AND #$EF ; BIT 4 WISSEN
EE8A 8D 00 DD STA $DD00
EE8D 60 RTS
***** KLOK LOW
EE8E AD 00 DD LDA $DD00
EE91 09 10 ORA #$10 ; BIT 4 ZETTEN
EE93 8D 00 DD STA $DD00
EE96 60 RTS
***** DATA HIGH
EE97 AD 00 DD LDA $DD00
EE9A 29 DF AND #$DF
EE9C 8D 00 DD STA $DD00 ; BIT 5 WISSEN
EE9F 60 RTS
***** DATA LOW
EEA0 AD 00 DD LDA $DD00
EEA3 09 20 ORA #$20
EEA5 8D 00 DD STA $DD00 ; BIT 5 ZETTEN
EEA8 60 RTS
***** BIT VAN IEC-BUS IN CARRY
EEA9 AD 00 DD LDA $DD00
EEAC CD 00 DD CMP $DD00 ; GEGEVENSREGISTER
EEAF DO F8 BNE $EEA9
EEB1 0A ASL A ; GEGEVENSBIT IN CARRY
EEB2 60 RTS
***** VERTRAGING VAN 1 MILLISECONDE
EEB3 8A TXA
EEB4 A2 B8 LDX #$B8 ; 184
EEB6 CA DEX
EEB7 DO FD BNE $EEB6
EEB9 AA TAX
EEBA 60 RTS

```

```

***** UITVOER VIA RS-232
EEBB A5 B4 LDA $B4 ; AANTAL TE ZENDEN BITS
EEBD FO 47 BEQ $EFO6 ; BYTE AL OVERGEDRAGEN?
EEBF 30 3F BMI $EFO0 ; STOPBITS NOODZAKELIJK?
EEC1 46 B6 LSR $B6
EEC3 A2 00 LDX #$00
EEC5 90 01 BCC $EEC8 ; BIT GEWIST?
EEC7 CA DEX ; NEE, DAN $FF
EEC8 8A TXA
EEC9 45 BD EOR $BD ; BEREKEN PARITEIT VAN
; TE STUREN BYTE
EECB 85 BD STA $BD
EECD C6 B4 DEC $B4 ; BITTELLER VERLAGEN
EECF FO 06 BEQ $EED7 ; ALLE BITS VERZONDEN?
EED1 8A TXA
EED2 29 04 AND #$04 ; BIT 2 ISOLEREN
EED4 85 B5 STA $B5 ; EN IN UITVOERREGISTER ZETTEN
EED6 60 RTS
EED7 A9 20 LDA #$20 ; BIT 5 (PARITEIT)
EED9 2C 94 02 BIT $0294 ; INSTRUCTIeregister VOOR RS-232
EEDC FO 14 BEQ $EEF2 ; ZONDER PARITEIT?
EEDE 30 1C BMI $EEFC ; VASTE PARITEIT?
EEEE 70 14 BVS $EEF6 ; ONEVEN PARITEIT?
EEE2 A5 BD LDA $BD ; PARITEIT=1?
EEE4 DO 01 BNE $EEE7 ; JA
EEE6 CA DEX ; PARITEIT $FF
EEE7 C6 B4 DEC $B4 ; BITTELLER OP $FF
EEE9 AD 93 02 LDA $0293 ; CONTROLEREGISTER VOOR RS-232
EEEC 10 E3 BPL $EED1 ; 2 STOPBITS?
EEEE C6 B4 DEC $B4 ; BITTELLER OP $FE
EEFO DO DF BNE $EED1 ; VERDER BIJ BEREKENING
; VAN STOPBITS
EEF2 E6 B4 INC $B4 ; BITTELLER VERHOGEN,
; GEEN PARITEIT
EEF4 DO FO BNE $EEE6 ; VERDER BIJ BEREKENING
; VAN STOPBITS
EEF6 A5 BD LDA $BD ; PARITEIT
EEF8 FO ED BEQ $EEE7 ; GELIJK AAN 0, DAN BIT 0
; UITVOEREN
EEFA DO EA BNE $EEE6 ; ANDERS BIT 1 UITVOEREN
EEFC 70 E9 BVS $EEE7 ; BIT 0 UITVOEREN
EEFE 50 E6 BVC $EEE6 ; ANDERS BIT 1 UITVOEREN
; (VASTE PARITEIT)
EFO0 E6 B4 INC $B4 ; BITTELLER VERHOGEN
EFO2 A2 FF LDX #$FF ; STOPBIT
EFO4 DO CB BNE $EED1 ; CONDITIELE SPRONG
EFO6 AD 94 02 LDA $0294 ; INSTRUCTIeregister VOOR RS-232
EFO9 4A LSR A ; BIT 0 IN CARRY
EFOA 90 07 BCC $EF13 ; 3-LIJNS HANDSHAKE,

```



```

; DAN INLEZEN OVERSLAAN
EFOC 2C 01 DD BIT $DD01
EFOF 10 1D BPL $EF2E ; MIST DSR?
EF11 50 1E BVC $EF31 ; MIST CTS?
EF13 A9 00 LDA #$00
EF15 85 BD STA $BD ; PARITEITSREGISTER WISSEN
EF17 85 B5 STA $B5 ; REGISTER VOOR TE ZENDEN BIT
; (STARTBIT)
EF19 AE 98 02 LDX $0298 ; AANTAL OVER TE DRAGEN BITS
EF1C 86 B4 STX $B4 ; ALS BITTELLER
EF1E AC 9D 02 LDY $029D ; ALLE BYTES VERZONDEN?
EF21 CC 9E 02 CPY $029E
EF24 FO 13 BEQ $EF39 ; ALLE TEKENS VERZONDEN, DAN
; AFSLUITEN
EF26 B1 F9 LDA ($F9),Y ; GEGEVENSBYTE UIT RS-232 BUFFER
EF28 85 B6 STA $B6 ; NAAR KARAKTERBUFFER
EF2A EE 9D 02 INC $029D ; BUFFERWIJZER VERHOGEN
EF2D 60 RTS
EF2E A9 40 LDA #$40 ; DSR (DATA SET READY) MIST
EF30 2C .BYTE $2C
EF31 A9 40 LDA #$10 ; CTS (CLEAR TO SEND) MIST
EF33 0D 97 02 ORA $0297 ; STATUS ZETTEN
EF36 8D 97 02 STA $0297
EF39 A9 01 LDA #$01
EF3B 8D 0D DD STA $DD0D ; NMI VOOR TIMER A WISSEN
EF3E 4D A1 02 EOR $02A1
EF41 09 80 ORA #$80 ; FLAG VOOR RS-232 OMDRAAIEN
EF43 8D A1 02 STA $02A1
EF46 8D 0D DD STA $DD0D ; OVERIGE NMI'S TOELATEN
EF49 60 RTS
***** AANTAL GEGEVENSBITS VOOR
; RS-232 BEREKENEN
EF4A A2 09 LDX #$09
EF4C A9 20 LDA #$20
EF4E 2C 93 02 BIT $0293 ; CONTROLEWOORD
EF51 FO 01 BEQ $EF54 ; BIT 5 GEWIST?
EF53 CA DEX
EF54 50 02 BVC $EF58 ; BIT 6 GEWIST?
EF56 CA DEX
EF57 CA DEX ; X=AANTAL GEGEVENSBITS
EF58 60 RTS
***** ONTVANGEN BIT VERWERKEN
EF59 A6 A9 LDX $A9 ; STARTBIT?
EF5B D0 33 BNE $EF90 ; JA
EF5D C6 A8 DEC $A8 ; BITTELLER VERLAGEN
EF5F FO 36 BEQ $EF97 ; ALLE BITS ONTVANGEN?
EF61 30 0D BMI $EF70 ; NOG STOPBITS TE VERWACHTEN
EF63 A5 A7 LDA $A7 ; ONTVANGEN BIT
EF65 45 AB EOR $AB ; PARITEIT BEREKENEN

```

EF67	85	AB	STA \$AB	
EF69	46	A7	LSR \$A7	; ONTVANGEN BIT IN CARRY
EF6B	66	AA	ROR \$AA	; EN IN ONTVANGREGISTER ZETTEN
EF6D	60		RTS	
EF6E	C6	A8	DEC \$A8	; BITTELLER VERLAGEN
EF70	A5	A7	LDA \$A7	; STOPBIT
EF72	F0	67	BEQ \$EFDB	; GELIJK AAN NUL?
EF74	AD	93 02	LDA \$0293	; CONTROLEWOORD
EF77	0A		ASL	; BIT 7 (AANTAL STOPBITS) IN CARRY
EF78	A9	01	LDA #\$01	
EF7A	65	A8	ADC \$A8	; AANTAL BITS EN STOPBITS OPTELLEN
EF7C	DO	EF	BNE \$EF6D	; NOG NIET ALLE STOPBITS ONTVANGEN?
EF7E	A9	90	LDA #\$90	; RXD VIA FLAG ONTVANGEN
EF80	8D	OD DD	STA \$DDOD	; NMI VRIJGEVEN
EF83	OD	A1 02	ORA \$02A1	
EF86	8D	A1 02	STA \$02A1	; FLAG VOOR RS-232-NMI'S
EF89	85	A9	STA \$A9	; FLAG VOOR STARTBIT
EF8B	A9	02	LDA #\$02	
EF8D	4C	3B EF	JMP \$EF3B	; NMI VOOR TIMER B WISSEN
EF90	A5	A7	LDA \$A7	; STARTBIT
EF92	DO	EA	BNE \$EF7E	; ONGELIJK 0
EF94	4C	D3 E4	JMP \$E4D3	; FLAG VOOR STARTBIT TERUGZETTEN
*****				ONTVANGEN BYTE VERWERKEN
EF97	AC	9B 02	LDY \$029B	; BUFFERWIJZER
EF9A	C8		INY	; VERHOGEN
EF9B	CC	9C 02	CPY \$029C	; ONTVANGBUFFER VOL?
EF9E	F0	2A	BEQ \$EFCA	; JA, STATUS ZETTEN
EFA0	8C	9B 02	STY \$029B	; BUFFERWIJZER OPSLAAN
EFA3	88		DEY	
EFA4	A5	AA	LDA \$AA	; ONTVANGEN BYTE
EFA6	AE	98 02	LDX \$0298	; AANTAL GEGEVENSBITS
EFA9	EO	09	CPX #\$09	; 8 BITS+1 STOPBIT?
EFAB	F0	04	BEQ \$EFB1	; JA, OK
EFAD	4A		LSR A	; ANDERS BITS IN GOEDE POSITIE SCHUIVEN
EFAE	E8		INX	
EFAF	DO	F8	BNE \$EFA9	
EFB1	91	F7	STA (\$F7),Y	; BYTE IN RS-232-BUFFER SCHRIJVEN
EFB3	A9	20	LDA #\$20	
EFB5	2C	94 02	BIT \$0294	; BIT 5 IN INSTRUCTIE
EFB8	F0	B4	BEQ \$EF6E	; OVERDRACHT ZONDER PARITEIT?
EFBA	30	B1	BMI \$EF6D	; VASTE BIT IN PLAATS VAN PARITEIT?
EFBC	A5	A7	LDA \$A7	; ONTVANGEN PARITEITSBYTE
EFBE	45	AB	EOR \$AB	; MET BEREKENDE PARITEIT VERGELIJKEN

```

EFC0 FO 03    BEQ $EFC5    ; GELIJK?
EFC2 70 A9    BVS $EF6D    ; EVEN PARITEIT, DAN OK
EFC4 2C       .BYTE $2C
EFC5 BVC      BVC $EF6D    ; ONEVEN PARITEIT, DAN OK
EFC7 A9 01    LDA #$01     ; PARITEITSFOUT
EFC9 2C       .BYTE $2C
EFCA A9 04    LDA #$04     ; ONTVANGSTBUFFER VOL
EFCC 2C       . BYTE $2C
EFCD A9 80    LDA #$80     ; BREAK-INSTRUCTIE ONTVANGEN
EFCF 2C       .BYTE $2C
EFD0 A9 02    LDA #$02     ; KADERFOUT
EFD2 0D 97 02 ORA $0297
EFD5 8D 97 02 STA $0297    ; STATUS ZETTEN
EFD8 4C 7E EF JMP $EF7E    ; VOOR ONTVANGST VOLGEND BYTE
EFDB A5 AA    LDA $AA      ; ONTVANGEN BYTE
EFDD DO F1    BNE $EFD0    ; KADERFOUT
EFDF FO EC    BEQ $EFC5    ; BREAK-INSTRUCTIE ONTVANGEN
***** RS-232 CKOUT: UITVOER VIA RS-232
EFE1 85 9A    STA $9A      ; DEVICENUMMER OPSLAAN
EFE3 AD 94 02 LDA $0294    ; RS-232-INSTRUCTIE
EFE6 4A       LSR A        ; BIT 0
EFE7 90 29    BCC $F012    ; 3 LIJNS-HANDSHAKE?
EFE9 A9 02    LDA #$02
EFEB 2C 01 DD BIT $DD01    ; DATA SET READY UITLEZEN
EFEE 10 1D    BPL $FOOD    ; NEE, DAN FOUT
EFF0 DO 20    BNE $F012    ; GEEN REQUEST TO SEND?
EFF2 AD A1 02 LDA $02A1    ; NMI-FLAG
EFF5 29 02    AND #$02     ; GEGEVENSONTVANGST ACTIEF?
EFF7 DO F9    BNE $EFF2    ; WACHTEN TOT ONTVANGST IS
; BEEINDIGD
EFF9 2C 01 DD BIT $DD01
EFFC 70 FB    BVS $EFF9    ; OP CLEAR TO SEND ZETTEN
EFEE AD 01 DD LDA $DD01
FO01 09 02    ORA #$02     ; REQUEST TO SEND ZETTEN
FO03 8D 01 DD STA $DD01
FO06 2C 01 DD BIT $DD01
FO09 70 07    BVS $F012    ; OP CLEAR TO SEND WACHTEN
FO0B 30 F9    BMI $FO06    ; DATA SET READY UITLEZEN
FOOD A9 40    LDA #$40
FOOF 8D 97 02 STA $0297    ; DSR SIGNAAL MIST, STATUS ZETTEN
FO12 18       CLC
FO13 60       RTS
***** UITVOER IN RS-232 BUFFER
FO14 20 28 FO JSR $F028    ; INDIEN NOODZAKELIJK
; OVERDRACHT STARTEN
FO17 AC 9E 02 LDY $029E    ; SCHRIJFWIJZER IN INVOERBUFFER
FO1A C8       INY          ; VERHOGEN
FO1B CC 9D 02 CPY $029D    ; EN LEESWIJZER VERGELIJKEN
FO1E FO F4    BEQ $F014    ; BUFFER VOL, DAN WACHTEN

```

```

F020 8C 9E 02 STY $029E ; NIEUWE WAARDE VOOR
; SCHRIJFWIJZER ONTHOUDEN
F023 88      DEY
F024 A5 9E    LDA $9E ; UIT TE VOEREN BYTE
F026 91 F9    STA ($F9),Y ; IN BUFFER SCHRIJVEN
F028 AD A1 02 LDA $02A1 ; NMI-FLAG VOOR RS-232
F02B 4A      LSR A ; BIT 0?
F02C B0 1E    BCS $F04C ; AL AAN HET ZENDEN?
F02E A9 10    LDA #$10 ; TIMER A
F030 8D 0E DD STA $DDOE ; STARTEN
F033 AD 99 02 LDA $0299
F036 8D 04 DD STA $DD04
F039 AD 9A 02 LDA $029A ; TIMER VOOR BAUD-RATE BIJ ZENDEN
; OPNIEUW INSTELLEN
F03C 8D 05 DD STA $DD05
F03F A9 81    LDA #$81
F041 20 3B EF JSR $EF3B ; NMI BIJ ONDERLOOP TIMER A
F044 20 06 EF JSR $EF06 ; CTS EN DSR CONTROLEREN,
; OVERDRACHT VRIJGEVEN
F047 A9 11    LDA #$11
F049 8D 0E DD STA $DDOE ; TIMER A STARTEN
F04C 60      RTS
***** RS-232 CHKIN, INVOER VIA RS-232
F04D 85 99    STA $99 ; DEVICE-NUMMER
F04F AD 94 02 LDA $0294 ; RS-232-INSTRUCTIE
F052 4A      LSR A ; BIT 0
F053 90 28    BCC $F07D ; 3 LIJNS-HANDSHAKE?
F055 29 08    AND #$08
F057 F0 24    BEQ $F07D
F059 A9 02    LDA #$02
F05B 2C 01 DD BIT $DD01 ; DATA SET READY UITLEZEN
F05E 10 AD    BPL $F00D ; NEE
F060 F0 22    BEQ $F084 ; READY TO SEND UITLEZEN
F062 AD A1 02 LDA $02A1 ; NMI-FLAG VOOR RS-232
F065 4A      LSR ; AAN HET ZENDEN?
F066 B0 FA    BCS $F062 ; JA, WACHTEN OP EINDE
F068 AD 01 DD LDA $DD01
F06B 29 FD    AND #$FD ; REQUEST TO SEND
F06D 8D 01 DD STA $DD01
F070 AD 01 DD LDA $DD01
F073 29 04    AND #$04 ; DATA TERMINAL READY
F075 F0 F9    BEQ $F070 ; NEE, WACHTEN
F077 A9 90    LDA #$90 ; NMI-MASKER VOOR FLAG
F079 18      CLC
F07A 4C 3B EF JMP $EF3B ; NMI VRIJGEVEN
F07D AD A1 02 LDA $02A1 ; NMI-FLAG VOOR RS-232
F080 29 12    AND #$12 ; RS-232 NIET ACTIEF,
F082 F0 F3    BEQ $F077 ; DAN STARTEN
F084 18      CLC

```

```

F085 60          RTS
***** GET VAN RS-232
F086 AD 97 02 LDA $0297          ; STATUS
F089 AC 9C 02 LDY $029C          ; BUFFERWIJZER
F08C CC 9B 02 CPY $029B          ; GELIJK
F08F FO 0B      BEQ $F09C          ; BUFFER LEEG?
F091 29 F7      AND #$F7          ; BIT 3 (BUFFER LEEG)
F093 8D 97 02 STA $0297          ; IN STATUS WISSEN
F096 B1 F7      LDA ($F7),Y        ; BYTE UIT BUFFER HALEN
F098 EE 9C 02 INC $029C          ; BUFFERWIJZER VERHOGEN
F09B 60          RTS
F09C 09 08      ORA #$08          ; BUFFER LEEG
F09E 8D 97 02 STA $0297          ; STATUS ZETTEN
FOA1 A9 00      LDA #$00          ; 0 OVERDRAGEN
FOA3 60          RTS
***** WACHTEN OP EINDE VAN
; RS-232-OVERDRACHT
FOA4 48          PHA
FOA5 AD A1 02 LDA $02A1          ; NMI-FLAG VOOR RS-232
FOA8 FO 11      BEQ $FOBB          ; NIET GEZET, DAN OK
FOAA AD A1 02 LDA $02A1
FOAD 29 03      AND #$03          ; BIT 0 (ZENDEN) EN BIT 1
; (ONTVANGEN)
FOAF DO F9      BNE $FOAA          ; WACHTEN OP EINDE
FOB1 A9 10      LDA #$10
FOB3 8D 0D DD STA $DD0D          ; INTERRUPT OVER FLAG-LIJN
FOB6 A9 00      LDA #$00
FOB8 8D A1 02 STA $02A1          ; FLAG TERUG
FOBB 68          PLA
FOBC 60          RTS
***** SYSTEEMMELDINGEN
FOBD OD 49 2F 4F 20 45 52 4F; I/O ERROR #
FOC6 52 20 A3
FOC9 OD 53 45 41 52 43 48 49; SEARCHING
FOD1 4E 47 A0
FOD4 46 4F 52 A0          ; FOR
FOD8 OD 50 52 45 53 53 20 50; PRESS PLAY ON TAPE
FOE0 4C 41 59 20 4F 4E 20 54
FOE8 41 50 C5
FOEB 50 52 45 53 53 20 52 45; PRESS RECORD & PLAY ON TAPE
FOF3 43 4F 52 44 20 26 20 50
FOFB 47 41 59 20 4F 4E 20 54
F103 41 50 C5
F106 OD 4C 4F 41 44 49 4E C7; LOADING
F10E OD 53 41 56 49 4E 47 A0; SAVING
F116 OD 56 45 52 49 46 59 49; VERIFYING
F11E 4E C7
F120 OD 46 4F 55 4E 44 A0      ; FOUND
F127 OD 4F 4B 8D          ; OK

```

```

***** SYSTEEMMELDINGEN UITVOEREN
F12B 24 9D BIT $9D ; FLAG VOOR DIREKTE MODUS
F12D 10 0D BPL $F13C ; PROGRAMMA, DAN OVERSLAAN
F12F B9 BD FO LDA $FOBD,Y ; OFFSET VANMELDING IN Y
F132 08 PHP
F133 29 7F AND #$7F ; BIT 7 WISSEN
F135 20 D2 FF JSR $FFD2 ; UITVOEREN
F138 C8 INY
F139 28 PLP
F13A 10 F3 BPL $F12F ; NOG MEER LETTERS?
F13C 18 CLC
F13D 60 RTS
***** GETIN
F13E A5 99 LDA $99 ; INVOERAPPARAAT
F140 D0 24 BNE $F166
F142 A5 C6 LDA $C6 ; AANTAL TEKENEN IN TOETSENBORD-
; BUFFER
F144 FO OF BEQ $F155 ; GEEN TEKEN?
F146 78 SEI
F147 4C B4 E5 JMP $E5B4 ; TEKEN UIT TOETSENBORDBUFFER
; HALEN
F14A C9 02 CMP #$02
F14C D0 18 BNE $F166 ; NIET RS-232, DAN NAAR
; BASIN-ROUTINE
F14E 84 97 STY $97
F150 20 86 FO JSR $F086 ; GET VAN RS-232
F153 A4 97 LDY $97
F155 18 CLC
F156 60 RTS
***** BASIN: INVOER VAN TEKEN
F157 A5 99 LDA $99 ; DEVICE-NUMMER
F159 D0 0B BNE $F166 ; GEEN TOETSENBORD?
F15B A5 D3 LDA $D3 ; CURSORPOSITIE
F15D 85 CA STA $CA
F15F A5 D6 LDA $D6 ; VOOR TOETSENBORDINVOER
; INSTELLEN
F161 85 C9 STA $C9
F163 4C 32 E6 JMP $E632 ; INVOER VAN BEELDSCHERM
F166 C9 03 CMP #$03
F168 D0 09 BNE $F173
***** VAN BEELDSCHERM
F16A 85 D0 STA $D0
F16C A5 D5 LDA $D5
F16E 85 C8 STA $C8
F170 4C 32 E6 JMP $E632
F173 B0 38 BCS $F1AD ; VAN IEC-BUS
F175 C9 02 CMP #$02
F177 FO 19 BEQ $F192
***** VAN BAND

```

```

F179 86 97 STX $97 ; X-REGISTER ONTHOUDEN
F17B 20 99 F1 JSR $F199 ; TEKEN VAN BAND HALEN
F17E B0 16 BCS $F196
F180 48 PHA
F181 20 99 F1 JSR $F199 ; TEKEN VAN BAND HALEN
F184 B0 0D BCS $F193 ; FOUT?
F186 D0 05 BNE $F18D ; LAATSTE TEKEN
F188 A9 40 LDA #$40 ; EOF
F18A 20 1C FE JSR $FE1C ; STATUS ZETTEN
F18D C6 A6 DEC $A6 ; CASSETTEBUFFERWIJZER
; VERLAGEN
F18F A6 97 LDX $97 ; X-REGISTER TERUGHALEN
F191 68 PLA
F192 60 RTS
F193 AA TAX ; FOUTNUMMER NAAR X
F194 68 PLA ; TEKEN TERUGHALEN
F195 8A TXA ; FOUTNUMMER IN ACCU
F196 A6 97 LDX $97 ; X-REGISTER TERUGHALEN
F198 60 RTS
***** TEKEN VAN BAND HALEN
F199 20 0D F8 JSR $F80D ; CASSETTEBUFFERWIJZER
; VERHOGEN
F19C D0 0B BNE $F1A9 ; ZIJN ER NOG TEKENS
; IN DE BUFFER?
F19E 20 41 F8 JSR $F841 ; NEE, VOLGEND BLOK VAN BAND
; HALEN
F1A1 B0 11 BCS $F1B4 ; STOPTOETS, DAN STOPPEN
F1A3 A9 00 LDA #$00
F1A5 85 A6 STA $A6 ; BUFFERWIJZER OP 0, TEKEN HALEN
F1A7 F0 F0 BEQ $F199
F1A9 B1 B2 LDA ($B2),Y ; TEKEN UIT BUFFER LEZEN
F1AB 18 CLC
F1AC 60 RTS
F1AD A5 90 LDA $90 ; STATUS TESTEN
F1AF F0 04 BEQ $F1B5 ; OK
F1B1 A9 0D LDA #$0D ; CR-CODE UITVOEREN
F1B3 18 CLC
F1B4 60 RTS
***** IEC-INVOER
F1B5 4C 13 EE JMP $EE13 ; 1 BYTE VAN IEC-BUS HALEN
***** INVOER VIA RS-232
F1B8 A5 54 LDA $54 ; BYTE VAN RS-232 HALEN
F1BA C9 4C CMP #$4C ; FOUT?
F1BC D0 03 BNE $F1C1 ; BYTE 0?
F1BE 4C 66 FE JMP $FE66 ; NEE, DAN OK
F1C1 AD 97 02 LDA $0297 ; STATUS
F1C4 29 60 AND #$60 ; MIST DSR?
F1C6 D0 E9 BNE $F1B1 ; JA, CR TERUGGEVEN
F1C8 F0 EE BEQ $F1B8 ; NEE, NIEUWE POGING

```

```

***** BSOUT: UITVOER VAN TEKEN
F1CA 48 PHA
F1CB A5 9A LDA $9A ; DEVICE-NUMMER VOOR UITVOER
F1CD C9 03 CMP #$03 ; BEELDSCHERM?
F1CF D0 04 BNE $F1D5 ; NEE
F1D1 68 PLA
F1D2 4C 16 E7 JMP $E716 ; TEKEN OP SCHERM
F1D5 90 04 BCC $F1DB
F1D7 68 PLA
F1D8 4C DD ED JMP $EDDD ; BYTE OP IEC-BUS
F1DB 4A LSR A
F1DC 68 PLA
F1DD 85 9E STA $9E ; UIT TE VOEREN TEKEN ONTHOUDEN
F1DF 8A TXA
F1E0 48 PHA
F1E1 98 TYA
F1E2 48 PHA
F1E3 90 17 BCC $F1FC ; UITVOER VIA RS-232
F1E5 20 0D F8 JSR $F80D ; CASSETTEBUFFERWIJZER VERHOGEN
F1E8 D0 0E BNE $F1F8 ; BUFFER VOL?
F1EA 20 64 F8 JSR $F864 ; BUFFER OP BAND SCHRIJVEN
F1ED B0 0E BCS $F1FD ; STOPTOETS, DAN STOPPEN
F1EF A9 02 LDA #$02 ; CONTROLEBYTE VOOR GEGEVENSBLK
F1F1 A0 00 LDY #$00
F1F3 91 B2 STA ($B2),Y ; IN BUFFER SCHRIJVEN
F1F5 C8 INY
F1F6 84 A6 STY $A6 ; BUFFERWIJZER VERHOGEN EN
; ONTHOUDEN
F1F8 A5 9E LDA $9E
F1FA 91 B2 STA ($B2),Y ; TEKEN IN BUFFER SCHRIJVEN
F1FC 18 CLC
F1FD 68 PLA
F1FE A8 TAY
F1FF 68 PLA
F200 AA TAX
F201 A5 9E LDA $9E ; TEKEN TERUGHALEN
F203 90 02 BCC $F207 ; OK?
F205 A9 00 LDA #$00 ; FLAG VOOR STOPTOETS INGEDRUKT
F207 60 RTS
***** UITVOER VIA RS-232
F208 20 17 F0 JSR $F017 ; TEKEN IN RS-232-BUFFER ZETTEN
F20B 4C FC F1 JMP $F1FC
***** CHKIN: INVOERAPPARAAT INSTELLEN
F20E 20 0F F3 JSR $F30F ; ZOEKT LOGISCHE FILENUMMER
F211 F0 03 BEQ $F216 ; GEVONDEN?
F213 4C 01 F7 JMP $F701 ; FILE NOT OPEN
F216 20 1F F3 JSR $F31F ; ZET FILEPARAMETER
F219 A5 BA LDA $BA ; DEVICE-NUMMER
F21B F0 16 BEQ $F233 ; 0, TOETSENBOORD

```



```

F21D C9 03    CMP #$03
F21F FO 12    BEQ $F233      ; 3, BEELDSCHERM
F221 BO 14    BCS $F237      ; IEC-BUS
F223 C9 02    CMP #$02       ; RS-232?
F225 DO 03    BNE $F22A      ; NEE, DAN BAND
F227 4C 4D FO JMP $F04D      ; JA
***** BAND ALS INVOERAPPARAAT INSTELLEN
F22A A6 B9    LDX $B9        ; SECUNDAIR ADRES
F22C EO 60    CPX #$60       ; 0?
F22E FO 03    BEQ $F233
F230 4C 0A F7 JMP $F70A      ; NOT INPUT FILE
F233 85 99    STA $99        ; DEVICE-NUMMER VOOR UITVOER
F235 18       CLC
F236 60       RTS
F237 AA       TAX
F238 20 09 ED JSR $ED09      ; TALK ZENDEN
F23B A5 B9    LDA $B9        ; SECUNDAIR ADRES
F23D 10 06    BPL $F245
F23F 20 CC ED JSR $EDCC      ; WACHT OP KLOKSIGNAAL
F242 4C 48 F2 JMP $F248
F245 20 C7 ED JSR $EDC7      ; SECUNDAIR ADRES VOOR TALK
;                               ZENDEN
F248 8A       TXA
F249 24 90    BIT $90        ; STATUS
F24B 10 E6    BPL $F233      ; OK?
F24D 4C 07 F7 JMP $F707      ; DEVICE NOT PRESENT
***** CKOUT: UITVOERAPPARAAT INSTELLEN
F250 20 0F F3 JSR $F30F      ; ZOEKT LOGISCHE FILENUMMER
F253 FO 03    BEQ $F258      ; GEVONDEN
F255 4C 01 F7 JMP $F701      ; FILE NOT OPEN
F258 20 1F F3 JSR $F31F      ; ZET FILEPARAMETER
F25B A5 BA    LDA $BA        ; DEVICE-NUMMER
F25D DO 03    BNE $F262      ; ONGELIJK NUL?
F25F 4C OD F7 JMP $F70D      ; NOT INPUT FILE
F262 C9 03    CMP #$03       ; BEELDSCHERM?
F264 FO 0F    BEQ $F275      ; JA
F266 BO 11    BCS $F279      ; IEC-BUS
F268 C9 02    CMP #$02       ; RS-232?
F26A DO 03    BNE $F26F
F26C 4C E1 EF JMP $EFE1      ; JA
***** BAND ALS UITVOERAPPARAAT INSTELLEN
F26F A6 B9    LDX $B9        ; SECUNDAIR ADRES
F271 EO 60    CPX #$60       ; 0?
F273 FO EA    BEQ $F25F      ; BANDFILE OM TE LEZEN,
;                               NOT OUTPUT FILE
F275 85 9A    STA $9A        ; NUMMER VAN UITVOERAPPARAAT
;                               INSTELLEN
F277 18       CLC
F278 60       RTS

```

```

***** UITVOER OP IEC-BUS ZETTEN
F279 AA TAX
F27A 20 OC ED JSR $EDOC ; LISTEN ZENDEN
F27D A5 B9 LDA $B9 ; SECUNDAIR ADRES
F27F 10 05 BPL $F286
F281 20 BE ED JSR $EDBE ; ATN TERUGZETTEN
F284 DO 03 BNE $F289
F286 20 B9 ED JSR $EDB9 ; SECUNDAIR ADRES VOOR
; LISTEN ZENDEN
F289 8A TXA
F28A 24 90 BIT $90 ; STATUS
F28C 10 E7 BPL $F275 ; OK
F28E 4C 07 F7 JMP $F707 ; DEVICE NOT PRESENT
***** CLOSE: LOGISCHE FILENUMMER IN A
F291 20 14 F3 JSR $F314 ; ZOEKT LOGISCHE FILENUMMER
F294 FO 02 BEQ $F298
F296 18 CLC ; FILE NIET VOORHANDEN,
; DAN KLAAR
F297 60 RTS ; FILEPARAMETER ZETTEN
F298 20 1F F3 JSR $F31F
F29B 8A TXA
F29C 48 PHA
F29D A5 BA LDA $BA ; DEVICE-NUMMER
F29F FO 50 BEQ $F2F1 ; TOETSENBOARD?
F2A1 C9 03 CMP #$03
F2A3 FO 4C BEQ $F2F1 ; BEELDSCHERM?
F2A5 B0 47 BCS $F2EE ; IEC?
F2A7 C9 02 CMP #$02 ; RS-232?
F2A9 DO 1D BNE $F2C8 ; BAND
***** RS-232-FILE SLUITEN
F2AB 68 PLA
F2AC 20 F2 F2 JSR $F2F2 ; FILE-ENTRY IN TABEL WISSEN
F2AF 20 83 F4 JSR $F483 ; CIA'S VOOR I/O TERUGZETTEN
F2B2 20 27 FE JSR $FE27 ; MEMORY TOP HALEN
F2B5 A5 F8 LDA $F8 ; INVOERBUFFER
F2B7 FO 01 BEQ $F2BA
F2B9 C8 INY
F2BA A5 FA LDA $FA ; UITVOERBUFFER
F2BC FO 01 BEQ $F2BF
F2BE C8 INY
F2BF A9 00 LDA #$00
F2C1 85 F8 STA $F8 ; BUFFER VRIJGEVEN
F2C3 85 FA STA $FA
F2C5 4C 7D F4 JMP $F47D ; MEMORY TOP OPNIEUW INSTELLEN
***** BANDFILE SLUITEN
F2C8 A5 B9 LDA $B9 ; SECUNDAIR ADRES
F2CA 29 OF AND #$0F
F2CC FO 23 BEQ $F2F1 ; FILE OM TE LEZEN?
F2CE 20 DO F7 JSR $F7D0 ; BEGINADRES VAN CASSETTEBUFFER

```

```

;
F2D1 A9 00 LDA #$00
F2D3 38 SEC
F2D4 20 DD F1 JSR $F1DD
F2D7 20 64 F8 JSR $F864 ; BUFFER OP BAND SCHRIJVEN
F2DA 90 04 BCC $F2E0
F2DC 68 PLA
F2DD A9 00 LDA #$00
F2DF 60 RTS
F2E0 A5 B9 LDA $B9 ; SECUNDAIR ADRES
F2E2 C9 62 CMP #$62 ; GELIJK AAN 2
F2E4 D0 0B BNE $F2F1
F2E6 A9 05 LDA #$05 ; CONTROLEBYTE VOOR EOT-HEADER
F2E8 20 6A F7 JSR $F76A ; BLOK OP BAND SCHRIJVEN
F2EB 4C F1 F2 JMP $F2F1
F2EE 20 42 F6 JSR $F642 ; IEC-FILE SLUITEN
F2F1 68 PLA
F2F2 AA TAX
F2F3 C6 98 DEC $98 ; AANTAL OPEN FILES VERLAGEN
F2F5 E4 98 CPX $98
F2F7 F0 14 BEQ $F30D ; GELIJK AAN 0, DAN KLAAR
F2F9 A4 98 LDY $98
F2FB B9 59 02 LDA $0259,Y
F2FE 9D 59 02 STA $0259,X
F301 B9 63 02 LDA $0263,Y ; TABEL VOOR FILEPARAMETERS OPENEN
F304 9D 63 02 STA $0263,X
F307 B9 6D 02 LDA $026D,Y
F30A 9D 6D 02 STA $026D,X
F30D 18 CLC
F30E 60 RTS
***** ZOEKT LOGISCHE FILENUMMER (IN X)
F30F A9 00 LDA #$00
F311 85 90 STA $90 ; STATUS WISSEN
F313 8A TXA
F314 A6 98 LDX $98 ; AANTAL OPEN FILES
F316 CA DEX
F317 30 15 BMI $F32E
F319 DD 59 02 CMP $0259,X ; ZOEKT ENTRY IN TABEL
F31C D0 F8 BNE $F316
F31E 60 RTS
***** ZET FILEPARAMETERS
F31F BD 59 02 LDA $0259,X
F322 85 B8 STA $B8 ; LOGISCHE FILENUMMER
F324 BD 63 02 LDA $0263,X
F327 85 BA STA $BA ; DEVICE-NUMMER
F329 BD 6D 02 LDA $026D,X
F32C 85 B9 STA $B9 ; SECUNDAIR ADRES
F32E 60 RTS
***** CLALL: SLUIT ALLE I/O-KANALEN

```

```

F32F A9 00 LDA #$00
F331 85 98 STA $98 ; AANTAL OPEN FILES = 0
***** CLRCH: SLUIT ACTIEVE I/O-KANAAL
F333 A2 03 LDX #$03
F335 E4 9A CPX $9A ; NUMMER VAN UITVOERAPPARAAT
F337 B0 03 BCS $F33C ; KLEINER DAN 3
F339 20 FE ED JSR $EDFE ; IEC, UNTALK ZENDEN
F33C E4 99 CPX $99 ; NUMMER VAN INVOERAPPARAAT
F33E B0 03 BCS $F343 ; KLEINER DAN 3
F340 20 EF ED JSR $EDEF ; IEC, UNTALK ZENDEN
F343 86 9A STX $9A ; UITVOER WEER OP SCHERM
F345 A9 00 LDA #$00
F347 85 99 STA $99 ; INVOER WEER VIA TOETSENBOARD
F349 60 RTS
***** OPEN
F34A A6 B8 LDX $B8 ; FILENUMBER
F34C D0 03 BNE $F351 ; ONGELIJK AAN 0
F34E 4C 0A F7 JMP $F70A ; NOT INPUT FILE
F351 20 0F F3 JSR $F30F ; ZOEKT LOGISCHE FILENUMBER
F354 D0 03 BNE $F359 ; GEVONDEN?
F356 4C FE F6 JMP $F6FE ; FILE OPEN
F359 A6 98 LDX $98 ; AANTAL OPEN FILES
F35B E0 0A CPX #$0A ; VERGELIJKEN MET 10
F35D 90 03 BCC $F362
F35F 4C FB F6 JMP $F6FB ; TOO MANY FILES
F362 E6 98 INC $98 ; AANTAL VERHOGEN
F364 A5 B8 LDA $B8 ; LOGISCHE FILENUMBER
F366 9D 59 02 STA $0259,X
F369 A5 B9 LDA $B9 ; SECUNDAIR ADRES
F36B 09 60 ORA #$60
F36D 85 B9 STA $B9
F36F 9D 6D 02 STA $026D,X
F372 A5 BA LDA $BA ; DEVICE-NUMMER
F374 9D 63 02 STA $0263,X ; IN DE TABEL SCHRIJVEN
F377 F0 5A BEQ $F3D3 ; TOETSENBOARD?
F379 C9 03 CMP #$03
F37B F0 56 BEQ $F3D3 ; BEELDSCHERM
F37D 90 05 BCC $F384
F37F 20 D5 F3 JSR $F3D5 ; FILE OP IEC-BUS OPENEN
F382 90 4F BCC $F3D3
F384 C9 02 CMP #$02 ; BAND?
F386 D0 03 BNE $F38B ; NEE
F388 4C 09 F4 JMP $F409 ; RS-232 OPEN
F38B 20 D0 F7 JSR $F7D0 ; BEGINADRES VAN CASSETTEBUFFER
; HALEN
F38E B0 03 BCS $F393
F390 4C 13 F7 JMP $F713 ; ILLEGAL DEVICE NUMBER
F393 A5 B9 LDA $B9 ; SECUNDAIR ADRES
F395 29 0F AND #$0F

```

F397	DO 1F	BNE \$F3B8	; ONGELIJK AAN O, DAN SCHRIJVEN
F399	20 17 F8	JSR \$F817	; WACHT OP PLAY-TOETS
F39C	B0 36	BCS \$F3D4	; STOPTOETS GEDRUKT?
F39E	20 AF F5	JSR \$F5AF	; SEARCHING (FOR NAME) UITVOEREN
F3A1	A5 B7	LDA \$B7	; LENGTE VAN FILENAAM
F3A3	F0 0A	BEQ \$F3AF	; GEEN FILENAAM, DAN VERDER
F3A5	20 EA F7	JSR \$F7EA	; ZOEKT GEWENSTE BANDHEADER
F3A8	90 18	BCC \$F3C2	; GEVONDEN
F3AA	F0 28	BEQ \$F3D4	; OK
F3AC	4C 04 F7	JMP \$F704	; EOT, FILE NOT FOUND
			UITVOEREN
F3AF	20 2C F7	JSR \$F72C	; VOLGENDE BANDHEADER ZOEKEN
F3B2	F0 20	BEQ \$F3D4	; EOT, FOUT
F3B4	90 0C	BCC \$F3C2	; GEVONDEN
F3B6	B0 F4	BCS \$F3AC	; PRG-FILE, VERDER ZOEKEN
F3B8	20 38 F8	JSR \$F838	; WACHT OP RECORD- & PLAY-TOETS
F3BB	B0 17	BCS \$F3D4	; STOPTOETS, DAN STOPPEN
F3BD	A9 04	LDA #\$04	; CONTROLEBYTE VOOR GEGEVENS- HEADER
F3BF	20 6A F7	JSR \$F76A	; HEADER OP BAND SCHRIJVEN
F3C2	A9 BF	LDA #\$BF	; WIJZER OP EINDE VAN CASSETTEBUFFER
F3C4	A4 B9	LDY \$B9	; SECUNDAIR ADRES
F3C6	C0 60	CPY #\$60	
F3C8	F0 07	BEQ \$F3D1	; GELIJK AAN O, DAN VERDER
F3CA	A0 00	LDY #\$00	
F3CC	A9 02	LDA #\$02	; CONTROLEBYTE VOOR GEGEVENSBLK
F3CE	91 B2	STA (\$B2),Y	; IN CASSETTEBUFFER SCHRIJVEN
F3D0	98	TYA	
F3D1	85 A6	STA \$A6	; WIJZER IN CASSETTEBUFFER
F3D3	18	CLC	
F3D4	60	RTS	
***** FILE OP IEC-BUS OPENEN			
F3D5	A5 B9	LDA \$B9	; SECUNDAIR ADRES
F3D7	30 FA	BMI \$F3D3	
F3D9	A4 B7	LDY \$B7	; LENGTE VAN FILENAAM
F3DB	F0 F6	BEQ \$F3D3	; GELIJK AAN O, DAN KLAAR
F3DD	A9 00	LDA #\$00	
F3DF	85 90	STA \$90	; STATUS WISSEN
F3E1	A5 BA	LDA \$BA	; DEVICE-NUMMER
F3E3	20 0C ED	JSR \$EDOC	; LISTEN
F3E6	A5 B9	LDA \$B9	; SECUNDAIR ADRES
F3E8	09 F0	ORA #\$F0	
F3EA	20 B9 ED	JSR \$EDB9	; ZENDEN
F3ED	A5 90	LDA \$90	; STATUS?
F3EF	10 05	BPL \$F3F6	; OK
F3F1	68	PLA	
F3F2	68	PLA	
F3F3	4C 07 F7	JMP \$F707	; DEVICE NOT PRESENT

```

F3F6 A5 B7 LDA $B7 ; LENGTE VAN FILENAAM
F3F8 F0 OC BEQ $F406
F3FA A0 00 LDY #$00
F3FC B1 BB LDA ($BB),Y ; FILENAAM
F3FE 20 DD ED JSR $EDDD ; OP IEC-BUS UITVOEREN
F401 C8 INY
F402 C4 B7 CPY $B7
F404 D0 F6 BNE $F3FC
F406 4C 54 F6 JMP $F654 ; UNLISTEN, RETURN
***** RS-232 OPEN
F409 20 83 F4 JSR $F483 ; CIA'S INSTELLEN
F40C 8C 97 02 STY $0297 ; RS-232 STATUS ZETTEN
F40F C4 B7 CPY $B7 ; LENGTE FILENAAM
F411 F0 0A BEQ $F41D
F413 B1 BB LDA ($BB),Y ; EERSTE 4 TEKENS OPSLAAN
F415 99 93 02 STA $0293,Y
F418 C8 INY
F419 C0 04 CPY #$04
F41B D0 F2 BNE $F40F
F41D 20 4A EF JSR $EF4A ; AANTAL GEGEVENSBITS BEREKENEN
F420 8E 98 02 STX $0298 ; EN OPSLAAN
F423 AD 93 02 LDA $0293 ; CONTROLEREGISTER
F426 29 0F AND #$0F ; BITS VOOR BAUD RATE ISOLEREN
F428 F0 1C BEQ $F446
F42A 0A ASL A ; *2 VOOR TABEL
F42B AA TAX
F42C AD A6 02 LDA $02A6 ; NTSC-VERSIE
F42F D0 09 BNE $F43A ; NEE
F431 BC C1 FE LDY $FEC1,X ; BAUD RATE, HIGH
; VOOR NTSC-TIMING
F434 BD C0 FE LDA $FEC0,X ; BAUD RATE, LOW
F437 4C 40 F4 JMP $F440
F43A BC EB E4 LDY $E4EB,X ; BAUD RATE, HIGH VOOR
; PAL-TIMING
F43D BD EA E4 LDA $E4EA,X ; BAUD RATE, LOW
F440 8C 96 02 STY $0296
F443 8D 95 02 STA $0295 ; OPSLAAN
F446 AD 95 02 LDA $0295
F449 0A ASL A
F44A 20 2E FF JSR $FF2E ; CODE VOOR BAUD RATE OPVRAGEN
F44D AD 94 02 LDA $0294
F450 4A LSR
F451 90 09 BCC $F45C
F453 AD 01 DD LDA $DD01 ; MIST DSR?
F456 0A ASL A
F457 B0 03 BCS $F45C ; NEE
F459 20 0D F0 JSR $F00D ; STATUS VOOR DSR ZETTEN
F45C AD 9B 02 LDA $029B
F45F 8D 9C 02 STA $029C ; BUFFERWIJZER INSTELLEN VOOR

```

```

;
F462 AD 9E 02 LDA $029E
F465 8D 9D 02 STA $029D
F468 20 27 FE JSR $FE27 ; MEMORY TOP HALEN
F46B A5 F8 LDA $F8
F46D D0 05 BNE $F474 ; INVOERBUFFER AL AANGELEGD?
F46F 88 DEY
F470 84 F8 STY $F8 ; WIJZER VOOR RS-232-INVOERBUFFER
F472 86 F7 STX $F7
F474 A5 FA LDA $FA
F476 D0 05 BNE $F47D ; UITVOERBUFFER AL AANGELEGD?
F478 88 DEY
F479 84 FA STY $FA ; WIJZER VOOR UITVOER VIA RS-232
F47B 86 F9 STX $F9
F47D 38 SEC
F47E A9 FO LDA #$FO ; FLAG ZETTEN
F480 4C 2D FE JMP $FE2D ; MEMORY TOP OPNIEUW INSTELLEN
***** CIA'S TERUG NAAR RS-232
F483 A9 7F LDA #$7F
F485 8D 0D DD STA $DD0D ; IRQ'S TERUGZETTEN
F488 A9 06 LDA #$06 ; BIT 1 EN 2 UITGANG
F48A 8D 03 DD STA $DD03 ; PORT B RICHTING
F48D 8D 01 DD STA $DD01 ; PORT A RICHTING
F490 A9 04 LDA #$04
F492 0D 00 DD ORA $DD00 ; BIT 2 = TXD
F495 8D 00 DD STA $DD00
F498 A0 00 LDY #$00
F49A 8C A1 02 STY $02A1 ; NMI-FLAG
F49D 60 RTS
***** LOAD-ROUTINE
F49E 86 C3 STX $C3 ; BEGINADRES OPSLAAN
F4A0 84 C4 STY $C4
F4A2 6C 30 03 JMP ($0330) ; JMP $4A5 LOAD-VECTOR
F4A5 85 93 STA $93 ; LOAD/VERIFY FLAG
F4A7 A9 00 LDA #$00
F4A9 85 90 STA $90 ; STATUS WISSEN
F4AB A5 BA LDA $BA ; DEVICE-NUMMER
F4AD D0 03 BNE $F4B2 ; ONGELIJK 0, DAN VERDER
F4AF 4C 13 F7 JMP $F713 ; ILLEGAL DEVICE NUMBER
F4B2 C9 03 CMP #$03 ; BEELDSCHERM?
F4B4 F0 F9 BEQ $F4AF ; JA, FOUT
F4B6 90 7B BCC $F533 ; KLEINER 3, DAN VAN BAND HALEN
***** IEC-LOAD
F4B8 A4 B7 LDY $B7 ; LENGTE FILENAAM
F4BA D0 03 BNE $F4BF ; ONGELIJK 0, DAN VERDER
F4BC 4C 10 F7 JMP $F710 ; MISSING FILENAME
F4BF A6 B9 LDX $B9 ; SECUNDAIR ADRES
F4C1 20 AF F5 JSR $F5AF ; SEARCHING FOR FILENAME
F4C4 A9 60 LDA #$60 ; SECUNDAIR ADRES 0

```

F4C6	85 B9	STA \$B9	
F4C8	20 D5 F3	JSR \$F3D5	; FILE OP IEC-BUS OPENEN
F4CB	A5 BA	LDA \$BA	; DEVICE-NUMMER
F4CD	20 09 ED	JSR \$ED09	; TALK ZENDEN
F4D0	A5 B9	LDA \$B9	
F4D2	20 C7 ED	JSR \$EDC7	; SECUNDAIR ADRES ZENDEN
F4D5	20 13 EE	JSR \$EE13	; BYTE VAN IEC-BUS HALEN
F4D8	85 AE	STA \$AE	; ALS BEGINADRES LOW OPSLAAN
F4DA	A5 90	LDA \$90	; STATUS
F4DC	4A	LSR A	
F4DD	4A	LSR A	
F4DE	B0 50	BCS \$F530	; TIME OUT, DAN FOUT
F4E0	20 13 EE	JSR \$EE13	; BEGINADRES HIGH HALEN
F4E3	85 AF	STA \$AF	
F4E5	8A	TXA	
F4E6	D0 08	BNE \$F4F0	; SECUNDAIR ADRES ONGELIJK O?
F4E8	A5 C3	LDA \$C3	
F4EA	85 AE	STA \$AE	; NEE, DAN OP
			AANGEGEVEN ADRES LADEN
F4EC	A5 C4	LDA \$C4	
F4EE	85 AF	STA \$AF	
F4F0	20 D2 F5	JSR \$F5D2	; LOADING/VERIFYING
			UITVOEREN
F4F3	A9 FD	LDA #\$FD	
F4F5	25 90	AND \$90	; TIME OUT-BIT WISSEN
F4F7	85 90	STA \$90	
F4F9	20 E1 FF	JSR \$FFE1	; STOPTOETS INLEZEN
F4FC	D0 03	BNE \$F501	; NIET INGEDRUKT, DAN VERDER
F4FE	4C 33 F6	JMP \$F633	; FILE SLUITEN
F501	20 13 EE	JSR \$EE13	; PROGRAMMABYTE VAN BUS HALEN
F504	AA	TAX	
F505	A5 90	LDA \$90	; STATUS?
F507	4A	LSR A	
F508	4A	LSR A	
F509	B0 E8	BCS \$F4F3	; FOUT, DAN STOPPEN
F50B	8A	TXA	
F50C	A4 93	LDY \$93	; LOAD/VERIFY-FLAG?
F50E	F0 0C	BEQ \$F51C	; 0, DAN LOAD
F510	A0 00	LDY #\$00	
F512	D1 AE	CMP (\$AE),Y	; VERIFY, DAN VERGELIJKEN
F514	F0 08	BEQ \$F51E	
F516	A9 10	LDA #\$10	; ONGELIJK, DAN STATUS ZETTEN
F518	20 1C FE	JSR \$FE1C	; STATUS ZETTEN
F51B	2C	.BYTE \$2C	
F51C	91 AE	STA (\$AE),Y	; BYTE OPSLAAN
F51E	E6 AE	INC \$AE	
F520	D0 02	BNE \$F524	; ADRES VERHOGEN
F522	E6 AF	INC \$AF	
F524	24 90	BIT \$90	; STATUS


```

F526 50 CB      BVC $F4F3      ; NOG GEEN EOF?
F528 20 EF ED   JSR $EDEF      ; UNTALK ZENDEN
F52B 20 42 F6   JSR $F642      ; FILE SLUITEN
F52E 90 79      BCC $F5A9      ; GEEN FOUT?
F530 4C 04 F7   JMP $F704      ; FILE NOT FOUND
*****
F533 4A          LSR A          ; DEVICE-NUMMER
F534 B0 03      BCS $F539      ; 1 (BAND), DAN VERDER
F536 4C 13 F7   JMP $F713      ; RS-232, ILLEGAL DEVICE NUMBER
F539 20 D0 F7   JSR $F7D0      ; BEGINADRES VAN CASSETTEBUFFER
;                               HALEN
F53C B0 03      BCS $F541
F53E 4C 13 F7   JMP $F713      ; ILLEGAL DEVICE NUMBER
F541 20 17 F8   JSR $F817      ; WACHT OP PLAY-TOETS
F544 B0 68      BCS $F5AE      ; STOPTOETS, DAN STOPPEN
F546 20 AF F5   JSR $F5AF      ; SEARCHING (FOR NAME)
;                               UITVOEREN
F549 A5 B7      LDA $B7        ; LENGTE VAN FILENAAM
F54B F0 09      BEQ $F556      ; 0, DAN VERDER
F54D 20 EA F7   JSR $F7EA      ; JUISTE BANDHEADER ZOEKEN
F550 90 0B      BCC $F55D      ; GEVONDEN?
F552 F0 5A      BEQ $F5AE      ; STOPTOETS, DAN STOPPEN
F554 B0 DA      BCS $F530      ; EOT, DAN FILE NOT FOUND
F556 20 2C F7   JSR $F72C      ; VOLGENDE BANDHEADER ZOEKEN
F559 F0 53      BEQ $F5AE      ; STOPTOETS, DAN STOPPEN
F55B B0 D3      BCS $F530      ; EOT, DAN FILE NOT FOUND
F55D A5 90      LDA $90        ; STAUS HALEN
F55F 29 10      AND #$10       ; EOF-BIT UITSCHAKELEN
F561 38          SEC
F562 D0 4A      BNE $F5AE      ; ANDERE FOUT
F564 E0 01      CPX #$01       ; HEADER-TYPE 1 = BASIC-PROGRAMMA
;                               (VERSCHUIFBAAR)
F566 F0 11      BEQ $F579
F568 E0 03      CPX #$03      ; HEADER-TYPE 2 = MACHINETAAL-
;                               PROGRAMMA (ABSOLUUT)
F56A D0 DD      BNE $F549
F56C A0 01      LDY #$01
F56E B1 B2      LDA ($B2),Y    ; BEGINADRES LOW
F570 85 C3      STA $C3
F572 C8          INY
F573 B1 B2      LDA ($B2),Y    ; BEGINADRES HIGH
F575 85 C4      STA $C4
F577 B0 04      BCS $F57D
F579 A5 B9      LDA $B9        ; SECUNDAIR ADRES
F57B D0 EF      BNE $F56C      ; ONGELIJK 0, DAN
;                               ABSOLUUT LADEN
F57D A0 03      LDY #$03
F57F B1 B2      LDA ($B2),Y
F581 A0 01      LDY #$01      ; EINDADRES MIN

```

```

F583 F1 B2      SBC ($B2),Y
F585 AA        TAX
F586 A0 04     LDY #$04
F588 B1 B2     LDA ($B2),Y
F58A A0 02     LDY #$02      ; BEGINADRES
F58C F1 B2     SBC ($B2),Y
F58E A8        TAY
F58F 18        CLC          ; GELIJK AAN PROGRAMMALENGTE
F590 8A        TXA
F591 65 C3     ADC $C3
F593 85 AE     STA $AE      ; PROGRAMMALENGTE + BEGINADRES
F595 98        TYA
F596 65 C4     ADC $C4      ; GELIJK AAN EINDADRES
F598 85 AF     STA $AF
F59A A5 C3     LDA $C3
F59C 85 C1     STA $C1      ; BEGINADRES NAAR $C1/$C2
F59E A5 C4     LDA $C4
F5A0 85 C2     STA $C2
F5A2 20 D2 F5 JSR $F5D2     ; LOADING/VERIFYING UITVOEREN
F5A5 20 4A F8 JSR $F84A     ; PROGRAMMA VAN BAND LADEN
F5A8 24 18     BIT $18
F5AA A6 AE     LDX $AE      ; EINDADRES NAAR X/Y
F5AC A4 AF     LDY $AF
F5AE 60        RTS
***** SEARCHING (FOR FILENAME)
;
F5AF A5 9D     LDA $9D      ; DIREKTE MODUS?
F5B1 10 1E     BPL $F5D1    ; NEE, DAN OVERSLAAN
F5B3 A0 0C     LDY #$0C     ; OFFSET VOOR SEARCHING
F5B5 20 2F F1 JSR $F12F     ; MELDING UITVOEREN
F5B8 A5 B7     LDA $B7      ; LENGTE VAN FILENAAM
F5BA F0 15     BEQ $F5D1    ; 0, DAN KLAAR
F5BC A0 17     LDY #$17     ; OFFSET VOOR FOR
F5BE 20 2F F1 JSR $F12F     ; MELDING UITVOEREN
F5C1 A4 B7     LDY $B7      ; LENGTE VAN FILENAAM
F5C3 F0 0C     BEQ $F5D1    ; 0, DAN KLAAR
F5C5 A0 00     LDY #$00
F5C7 B1 BB     LDA ($BB),Y  ; FILENAAM HALEN
F5C9 20 D2 FF JSR $FFD2     ; EN UITVOEREN
F5CC C8        INY
F5CD C4 B7     CPY $B7      ; LOADING/VERIFYING UITVOEREN
F5CF D0 F6     BNE $F5C7    ; OFFSET VOOR LOADING
F5D1 60        RTS          ; LOAD/VERIFY-FLAG?
F5D2 A0 49     LDY #$49     ; LOAD, DAN UITVOEREN
F5D4 A5 93     LDA $93      ; OFFSET VOOR VERIFYING
F5D6 F0 02     BEQ $F5DA    ; MELDING UITVOEREN
F5D8 A0 59     LDY #$59
***** SAVE-ROUTINE
F5DA 4C 2B F1 JMP $F12B     ; EINDADRES LOW

```

```

F5DD 86 AE STX $AE ; EINDADRES HIGH
F5DF 84 AF STY $AF
F5E1 AA TAX
F5E2 B5 00 LDA $00,X
F5E4 85 C1 STA $C1 ; BEGINADRES LOW
F5E6 B5 01 LDA $01,X
F5E8 85 C2 STA $C2 ; BEGINADRES HIGH
F5EA 6C 32 03 JMP ($0332) ; SAVE-VECTOR, JMP $F5ED
F5ED A5 BA LDA $BA ; DEVICE-NUMMER
F5EF D0 03 BNE $F5F4
F5F1 4C 13 F7 JMP $F713 ; ILLEGAL DEVICE NUMBER
F5F4 C9 03 CMP #$03
F5F6 F0 F9 BEQ $F5F1 ; BEELDSCHERM, FOUT
F5F8 90 5F BCC $F659 ; BAND
***** OPSLAAN OP IEC-BUS
F5FA A9 61 LDA #$61 ; SECUNDAIR ADRES OP 1
F5FC 85 B9 STA $B9 ; ZETTEN
F5FE A4 B7 LDY $B7 ; LENGTE VAN FILENAAM
F600 D0 03 BNE $F605 ; ONGELIJK AAN 0, DAN OK
F602 4C 10 F7 JMP $F710 ; MISSING FILENAME
F605 20 D5 F3 JSR $F3D5 ; FILENAAM OP IEC-BUS
F608 20 8F F6 JSR $F68F ; SAVING UITVOEREN
F60B A5 BA LDA $BA ; DEVICE-NUMMER
F60D 20 0C ED JSR $EDOC ; LISTEN ZENDEN
F610 A5 B9 LDA $B9 ; SECUNDAIR ADRES
F612 20 B9 ED JSR $EDB9 ; VOOR LISTEN ZENDEN
F615 A0 00 LDY #$00
F617 20 8E FB JSR $FB8E ; BEGINADRES NAAR $AC/AD
F61A A5 AC LDA $AC ; BEGINADRES LOW
F61C 20 DD ED JSR $EDDD ; ZENDEN
F61F A5 AD LDA $AD ; EN HIGH
F621 20 DD ED JSR $EDDD ; ZENDEN
F624 20 D1 FC JSR $FCD1 ; EINDADRES AL BEREIKT
F627 B0 16 BCS $F63F ; JA, DAN KLAAR
F629 B1 AC LDA ($AC),Y ; PROGRAMMABYTES
F62B 20 DD ED JSR $EDDD ; OP IEC-BUS UITVOEREN
F62E 20 E1 FF JSR $FFE1 ; STOPTOETS INLEZEN
F631 D0 07 BNE $F63A ; NIET INGEDRUKT, DAN VERDER
F633 20 42 F6 JSR $F642 ; IEC-BUS-KANAAL SLUITEN
F636 A9 00 LDA #$00
F638 38 SEC ; FLAG ZETTEN VOOR UITVOER
; VAN BREAK
F639 60 RTS
F63A 20 DB FC JSR $FCDB ; ACTUEEL ADRES
F63D D0 E5 BNE $F624 ; CONDITIONELE SPRONG
F63F 20 FE ED JSR $EDFE ; UNLISTEN ZENDEN
***** FILE OP IEC-BUS SLUITEN
F642 24 B9 BIT $B9 ; SECUNDAIR ADRES
F644 30 11 BMI $F657 ; GEEN SECUNDAIR ADRES?

```

```

F646 A5 BA LDA $BA ; DEVICE-NUMMER
F648 20 OC ED JSR $EDOC ; LISTEN ZENDEN
F64B A5 B9 LDA $B9 ; SECUNDAIR ADRES
F64D 29 EF AND #$EF
F64F 09 EO ORA #$EO ; SECUNDAIR ADRES VOOR CLOSE
F651 20 B9 ED JSR $EDB9 ; UITVOEREN
F654 20 FE ED JSR $EDFE ; UNLISTEN ZENDEN
F657 18 CLC
F658 60 RTS
F659 4A LSR A ; DEVICE-NUMMER GEDEELD DOOR 2
F65A B0 03 BCS $F65F ; BAND
F65C 4C 13 F7 JMP $F713 ; RS-232, ILLEGAL DEVICE NUMBER
F65F 20 D0 F7 JSR $F7D0 ; BEGINADRES VAN CASSETTEBUFFER
;
F662 90 8D BCC $F5F1 ; ILLEGAL DEVICE NUMBER
F664 20 38 F8 JSR $F838 ; WACHT OP RECORD- & PLAY-TOETS
F667 B0 25 BCS $F68E ; STOPPEN?
F669 20 8F F6 JSR $F68F ; SAVING NAME UITVOEREN
F66C A2 03 LDX #$03 ; HEADER-TYPE 3 = MASCHINETAAL-
; PROGRAMMA (ABSOLUUT)
F66E A5 B9 LDA $B9 ; SECUNDAIR ADRES
F670 29 01 AND #$01 ; BIT 0 GEZET (1 OF 3)
F672 D0 02 BNE $F676 ; JA, DAN MACHINETAALPROGRAMMA
F674 A2 01 LDX #$01 ; HEADER-TYPE 1 = BASIC-PROGRAMMA
F676 8A TXA
F677 20 6A F7 JSR $F76A ; HEADER OP BAND SCHRIJVEN
F67A B0 12 BCS $F68E ; STOPPEN?
F67C 20 67 F8 JSR $F867 ; SECUNDAIR ADRES
F67F B0 0D BCS $F68E ; BIT 1 GEZET (2 OF 3)
F681 A5 B9 LDA $B9 ; NEE, DAN KLAAR
F683 29 02 AND #$02 ; EOT, CONTROLEBYTE
F685 F0 06 BEQ $F68D ; BLOK OP BAND SCHRIJVEN
F687 A9 05 LDA #$05
F689 20 6A F7 JSR $F76A
F68C 24 18 BIT $18
F68E 60 RTS
***** SAVING UITVOEREN
F68F A5 9D LDA $9D ; DIREKTE MODUS?
F691 10 FB BPL $F68E ; NEE, DAN KLAAR
F693 A0 51 LDY #$51 ; OFFSET VOOR SAVING
F695 20 2F F1 JSR $F12F ; MELDING UITVOEREN
F698 4C C1 F5 JMP $F5C1 ; FIENAAM UITVOEREN
***** UDTIM: TIME VERHOGEN
F69B A2 00 LDX #$00
F69D E6 A2 INC $A2
F69F D0 06 BNE $F6A7 ; TIJD VERHOGEN
F6A1 E6 A1 INC $A1
F6A3 D0 02 BNE $F6A7
F6A5 E6 A0 INC $A0

```

```

F6A7 38      SEC
F6A8 A5 A2   LDA $A2
F6AA E9 01   SBC #$01
F6AC A5 A1   LDA $A1      ; MET WAARDE VOOR 24 UUR
                          VERGELIJKEN
;
F6AE E9 1A   SBC #$1A
F6B0 A5 A0   LDA $A0
F6B2 E9 4F   SBC #$4F
F6B4 90 06   BCC $F6BC    ; KLEINER, DAN OK
F6B6 86 A0   STX $A0
F6B8 86 A1   STX $A1      ; TIJD OP 0 ZETTEN
F6BA 86 A2   STX $A2
F6BC AD 01 DC LDA $DC01
F6BF CD 01 DC CMP $DC01   ; TOETS INGEDRUKT?
F6C2 D0 F8   BNE $F6BC
F6C4 AA      TAX
F6C5 30 13   BMI $F6DA    ; NEE
F6C7 A2 BD   LDX #$BD
F6C9 8E 00 DC STX $DC00   ; KIES RIJEN VOOR STOPTOETS
F6CC AE 01 DC LDX $DC01
F6CF EC 01 DC CPX $DC01   ; STOPTOETS?
F6D2 D0 F8   BNE $F6CC
F6D4 8D 00 DC STA $DC00
F6D7 E8      INX
F6D8 D0 02   BNE $F6DC
F6DA 85 91   STA $91      ; FLAG VOOR STOPTOETS
F6DC 60      RTS
***** TIME HALEN
F6DD 78      SEI
F6DE A5 A2   LDA $A2
F6E0 A6 A1   LDX $A1
F6E2 A4 A0   LDY $A0
***** TIME INSTELLEN
F6E4 78      SEI
F6E5 85 A2   STA $A2
F6E7 86 A1   STX $A1
F6E9 84 A0   STY $A0
F6EB 58      CLI
F6EC 60      RTS
***** STOPTOETS INLEZEN
F6ED A5 91   LDA $91      ; FLAG
F6EF C9 7F   CMP #$7F     ; STOPCODE ?
F6F1 D0 07   BNE $F6FA
F6F3 08      PHP
F6F4 20 CC FF JSR $FFCC    ; CLRCH
F6F7 85 C6   STA $C6      ; AANTAL INGEDRUKTE TOETSEN
F6F9 28      PLP
F6FA 60      RTS
***** MELDINGEN VAN BEDRIJFSSYSTEEM

```

```

;
F6FB A9 01 LDA #$01 ; UITVOEREN
F6FD 2C .BYTE $2C ; TOO MANY FILES
F6FE A9 02 LDA $02 ; FILE OPEN
F700 2C .BYTE $2C
F701 A9 03 LDA $03 ; FILE NOT OPEN
F703 2C .BYTE $2C
F704 A9 04 LDA $04 ; FILE NOT FOUND
F706 2C .BYTE $2C
F707 A9 05 LDA $05 ; DEVICE NOT PRESENT
F709 2C .BYTE $2C
F70A A9 06 LDA $06 ; NOT INPUT FILE
F70C 2C . BYTE $2C
F70D A9 07 LDA $07 ; NOT OUTPUT FILE
F70F 2C . BYTE $2C
F710 A9 08 LDA $08 ; MISSING FILENAME
F713 2C . BYTE $2C
F714 A9 09 LDA $09 ; ILLEGAL DEVICE NUMBER
F715 48 PHA ; FOUTNUMMER ONTHOUDEN
F716 20 CC FF JSR $FFCC ; CLRCH
F719 A0 00 LDY #$00
F71B 24 9D BIT $9D ; FLAG VOOR ERROR-UITVOER?
F71D 50 0A BVC $F729 ; NIET GEZET, DAN VERDER
F71F 20 2F F1 JSR $F12F ; I/O-ERROR # UITVOEREN
F722 68 PLA
F723 48 PHA ; FOUTNUMMER HALEN
F724 09 30 ORA #$30 ; NAAR ASCII
F726 20 D2 FF JSR $FFD2 ; EN UITVOEREN
F729 68 PLA
F72A 38 SEC
F72B 60 RTS
***** PROGRAMMA-HEADER VAN
; BAND LEZEN
F72C A5 93 LDA $93 ; LOAD/VERIFY-FLAG REDDEN
F72E 48 PHA
F72F 20 41 F8 JSR $F841 ; BLOK VAN BAND LEZEN
F732 68 PLA
F733 85 93 STA $93 ; LOAD/VERIFY-FLAG TERUGHALEN
F735 B0 32 BCS $F769 ; FOUT, DAN STOPPEN
F737 A0 00 LDY #$00
F739 B1 B2 LDA ($B2),Y ; HEADER-TYPE ?
F73B C9 05 CMP #$05 ; EOT?
F73D F0 2A BEQ $F769
F73F C9 01 CMP #$01 ; BASIC-PROGRAMMA?
F741 F0 08 BEQ $F74B
F743 C9 03 CMP #$03 ; MACHINETAALPROGRAMMA?
F745 F0 04 BEQ $F74B
F747 C9 04 CMP #$04 ; GEGEVENS-HEADER?
F749 D0 E1 BNE $F72C ; NEE

```

```

F74B AA TAX
F74C 24 9D BIT $9D ; DIREKTE MODUS
F74E 10 17 BPL $F767 ; NEE, DAN VERDER
F750 A0 63 LDY #$63
F752 20 2F F1 JSR $F12F ; FOUND UITVOEREN
F755 A0 05 LDY #$05 ; OFFSET VAN FILENAAM
F757 B1 B2 LDA ($B2),Y
F759 20 D2 FF JSR $FFD2 ; FILENAAM UITVOEREN
F75C C8 INY
F75D C0 15 CPY #$15
F75F D0 F6 BNE $F757
F761 A5 A1 LDA $A1 ; MID-BYTE VOOR KLOK
; IN ACCU LADEN
F763 20 E0 E4 JSR $E4E0 ; WACHT OP C=-TOETS, OF WACHTLUS
F766 EA NOP
F767 18 CLC ; BIJ FOUT IS C=1
F768 88 DEY
F769 60 RTS
***** HEADER GENEREREN EN
; OP BAND SCHRIJVEN
F76A 85 9E STA $9E ; HEADER-TYPE
F76C 20 D0 F7 JSR $F7D0 ; BEGINADRES VAN CASSETTEBUFFER
; HALEN
F76F 90 5E BCC $F7CF
F771 A5 C2 LDA $C2
F773 48 PHA ; BEGINADRES
F774 A5 C1 LDA $C1
F776 48 PHA ; EN
F777 A5 AF LDA $AF
F779 48 PHA ; EINDADRES ONTHOUDEN
F77A A5 AE LDA $AE
F77C 48 PHA
F77D A0 BF LDY #$BF ; BUFFERLENGTE - 1
F77F A9 20 LDA #$20
F781 91 B2 STA ($B2),Y ; CASSETTEBUFFER WISSEN
F783 88 DEY
F784 D0 FB BNE $F781
F786 A5 9E LDA $9E
F788 91 B2 STA ($B2),Y ; HEADER-TYPE
F78A C8 INY
F78B A5 C1 LDA $C1
F78D 91 B2 STA ($B2),Y ; BEGINADRES
F78F C8 INY
F790 A5 C2 LDA $C2
F792 91 B2 STA ($B2),Y
F794 C8 INY
F795 A5 AE LDA $AE
F797 91 B2 STA ($B2),Y ; EINDADRES
F799 C8 INY

```

```

F79A A5 AF LDA $AF
F79C 91 B2 STA ($B2),Y
F79E C8 INY
F79F 84 9F STY $9F
F7A1 A0 00 LDY #$00
F7A3 84 9E STY $9E ; TELLER VOOR LENGTE VAN FILENAAM
F7A5 A4 9E LDY $9E
F7A7 C4 B7 CPY $B7 ; MET LENGTE VERGELIJKEN
F7A9 F0 0C BEQ $F7B7 ; ALLE LETTERS, DAN VERDER
F7AB B1 BB LDA ($BB),Y ; FILENAAM HALEN
F7AD A4 9F LDY $9F
F7AF 91 B2 STA ($B2),Y ; IN HEADER SCHRIJVEN
F7B1 E6 9E INC $9E
F7B3 E6 9F INC $9F
F7B5 D0 EE BNE $F7A5
F7B7 20 D7 F7 JSR $F7D7 ; BEGINADRES- EN EINDADRES
; IN CASSETTEBUFFER
F7BA A9 69 LDA #$69
F7BC 85 AB STA $AB ; PROEFSOM VOOR HEADER/
; DATABLOK = $69
F7BE 20 6B F8 JSR $F86B ; BLOK OP BAND SCHRIJVEN
F7C1 A8 TAY
F7C2 68 PLA
F7C3 85 AE STA $AE
F7C5 68 PLA ; EIND-
F7C6 85 AF STA $AF
F7C8 68 PLA ; EN
F7C9 85 C1 STA $C1
F7CB 68 PLA ; BEGINADRES TERUGHALEN
F7CC 85 C2 STA $C2
F7CE 98 TYA
F7CF 60 RTS
***** BEGINADRES VAN CASSETTEBUFFER
; HALEN
F7D0 A6 B2 LDX $B2
F7D2 A4 B3 LDY $B3
F7D4 C0 02 CPY #$02 ; ADRES < $200?
F7D6 60 RTS
*****
F7D7 20 D0 F7 JSR $F7D0 ; ADRES VAN CASSETTEBUFFER HALEN
F7DA 8A TXA
F7DB 85 C1 STA $C1 ; BEGINADRES = START VAN
; CASSETTEBUFFER
F7DD 18 CLC
F7DE 69 C0 ADC #$C0
F7E0 85 AE STA $AE
F7E2 98 TYA ; EINDADRES = BEGINADRES +
; LENGTE (192)
F7E3 85 C2 STA $C2

```



```

F7E5 69 00    ADC #$00
F7E7 85 AF    STA $AF
F7E9 60      RTS
***** ZOEK BAND AF NAAR NAAM
F7EA 20 2C F7 JSR $F72C    ; VOLGENDE BANDHEADER ZOEKEN
F7ED B0 1D    BCS $F80C    ; EOT, DAN KLAAR
F7EF A0 05    LDY #$05     ; OFFSET VOOR FILENAAM
;                                     IN HEADER
F7F1 84 9F    STY $9F
F7F3 A0 00    LDY #$00
F7F5 84 9E    STY $9E     ; TELLER VOOR LENGTE VAN FILENAAM
F7F7 C4 B7    CPY $B7     ; MET LENGTE VAN GEZOCHTE
;                                     FILENAAM VERGELIJKEN
F7F9 F0 10    BEQ $F80B    ; GELIJK, DAN GEVONDEN
F7FB B1 BB    LDA ($BB),Y  ; LETTERS VAN FILENAAM
F7FD A4 9F    LDY $9F
F7FF D1 B2    CMP ($B2),Y  ; MET FILENAAM IN HEADER
;                                     VERGELIJKEN
F801 D0 E7    BNE $F7EA    ; ONGELIJK, DAN VOLGENDE
;                                     HEADER?
F803 E6 9E    INC $9E
F805 E6 9F    INC $9F     ; TELLER VERHOGEN
F807 A4 9E    LDY $9E
F809 D0 EC    BNE $F7F7    ; VOLGENDE LETTERS VERGELIJKEN
F80B 18      CLC
F80C 60      RTS
***** CASSETTEBUFFERWIJZER
; VERHOGEN
F80D 20 D0 F7 JSR $F7D0    ; ADRES VAN CASSETTEBUFFER
; HALEN
F810 E6 A6    INC $A6     ; WIJZER VERHOGEN
F812 A4 A6    LDY $A6
F814 C0 C0    CPY #$C0     ; MET MAXIMALE WAARDE
;                                     (192) VERGELIJKEN
F816 60      RTS
***** WACHT OP TOETS VAN CASSETTE
F817 20 2E F8 JSR $F82E    ; LEEST CASSETTETOETS UIT
F81A F0 1A    BEQ $F836    ; INGEDRUKT, DAN KLAAR
F81C A0 1B    LDY #$1B     ; OFFSET VOOR PRESS PLAY ON TAPE
F81E 20 2F F1 JSR $F12F    ; UITVOEREN
F821 20 D0 F8 JSR $F8D0    ; STOPTOETS?
F824 20 2E F8 JSR $F82E    ; LEEST CASSETTETOETS UIT
F827 D0 F8    BNE $F821
F829 A0 6A    LDY #$6A     ; OFFSET VOOR OK
F82B 4C 2F F1 JMP $F12F    ; UITVOEREN
***** CASSETTETOETS INGEDRUKT?
F82E A9 10    LDA #$10     ; BIT 4?
F830 24 01    BIT $01     ; TOETS INGEDRUKT?

```

```

F832 DO 02      BNE $F836      ; NEE
F834 24 01      BIT $01        ; NOGMAALS UITLEZEN
F836 18         CLC            ; JA, DAN Z=1, ANDERS Z=0
F837 60         RTS
***** WACHT OP CASSETTETOETS
; VOOR SCHRIJVEN
F838 20 2E F8 JSR $F82E        ; LEEST CASSETTETOETS IN
F83B FO F9      BEQ $F836      ; INGEDRUKT, DAN KLAAR
F83D AO 2E      LDY #$2E       ; OFFSET VOOR
; PRESS RECORD & PLAY ON TAPE
F83F DO DD      BNE $F81E      ; VERDER ALS BOVEN
***** BLOK VAN BAND LEZEN
F841 A9 00      LDA #$00
F843 85 90      STA $90        ; STATUS EN
F845 85 93      STA $93        ; VERIFY-FLAG WISSEN
F847 20 D7 F7 JSR $F7D7        ; ADRES VAN CASSETTEBUFFER HALEN
***** PROGRAMMA VAN BAND LADEN
F84A 20 17 F8 JSR $F817        ; WACHT OP PLAY-TOETS
F84D B0 1F      BCS $F86E      ; STOPTOETS INGEDRUKT?
F84F 78         SEI
F850 A9 00      LDA #$00
F852 85 AA      STA $AA
F854 85 B4      STA $B4
F856 85 B0      STA $B0        ; RAM VOOR IRQ-ROUTINE WISSEN
F858 85 9E      STA $9E
F85A 85 9F      STA $9F
F85C 85 9C      STA $9C
F85E A9 90      LDA #$90        ; IRQ AAN PIN-FLAG
F860 A2 OE      LDX #$0E        ; NUMMER VAN IRQ-VECTOR, $F92C
F862 DO 11      BNE $F875
***** ADRES VAN CASSETTEBUFFER
; BAND SCHRIJVEN
F864 20 D7 F7 JSR $F7D7        ; ADRES VAN CASSETTEBUFFER HALEN
F867 A9 14      LDA #$14        ; LENGTE VAN WRITE-HEADER
F869 85 AB      STA $AB        ; OPSLAAN
***** BLOK/PROGRAMMA OP BAND ZETTEN
F86B 20 38 F8 JSR $F838        ; WACHT OP REC- & PLAY-TOETS
F86E B0 6C      BCS $F8DC      ; STOPTOETS INGEDRUKT?
F870 78         SEI
F871 A9 82      LDA #$82        ; IRQ BIJ ONDERLOOP VAN TIMER B
F873 A2 08      LDX #$08        ; NUMMER VAN IRQ-VECTOR, $F6CA
F875 A0 7F      LDY #$7F
F877 8C OD DC STY $DCOD        ; IRQ-MASKER WISSEN
F87A 8D OD DC STA $DCOD        ; EN OPNIEUW INSTELLEN
F87D AD OE DC LDA $DCOE        ; CRA
F880 09 19      ORA #$19        ; TIMER B LADEN,
; ONE SHOT, STARTEN
F882 8D OF DC STA $DCOF        ; CRB, IRQ EN TIMER B
F885 29 91      AND #$91

```

F887	8D A2 02	STA \$02A2	; TIMER A CONTROLE-FLAG
F88A	20 A4 F0	JSR \$FOA4	; WACHTEN OP EINDE VAN OVERDRACHT
			VIA RS-232
F88D	AD 11 D0	LDA \$D011	
F890	29 EF	AND #\$EF	; BEELDSCHERM UIT
F892	8D 11 D0	STA \$D011	
F895	AD 14 03	LDA \$0314	; IRQ-VECTOR
F898	8D 9F 02	STA \$029F	
F89B	AD 15 03	LDA \$0315	; NA \$29F/\$240 OPSLAAN
F89E	8D A0 02	STA \$02A0	
F8A1	20 BD FC	JSR \$FCBD	; IRQ-VECTOR VOOR BAND-I/O
			ZETTEN (X-GEINDICEERD)
F8A4	A9 02	LDA #\$02	
F8A6	85 BE	STA \$BE	; AANTAL TE LEZEN BLOKS
F8A8	20 97 FB	JSR \$FB97	; SERIELE UITVOER VOORBEREIDEN
			BITTELLER INSTELLEN
F8AB	A5 01	LDA \$01	
F8AD	29 1F	AND #\$1F	; FLAG VOOR CASSETTEBUFFER
			ZETTEN
F8AF	85 01	STA \$01	
F8B1	85 C0	STA \$C0	
F8B3	A2 FF	LDX #\$FF	
F8B5	A0 FF	LDY #\$FF	
F8B7	88	DEY	
F8B8	D0 FD	BNE \$F8B7	; WACHTLUS VOOR AANLOOPTIJD
F8BA	CA	DEX	
F8BB	D0 F8	BNE \$F8B5	
F8BD	58	CLI	; INTERRUPT VOOR BAND-I/O
			VRIJGEVEN
*****			I/O-AFSLUITING AFWACHTEN
F8BE	AD A0 02	LDA \$02A0	; IRQ-VECTOR WEER OP
			STANDAARDWAARDE?
F8C1	CD 15 03	CMP \$0315	
F8C4	18	CLC	
F8C5	F0 15	BEQ \$F8DC	; JA, DAN KLAAR
F8C7	20 D0 F8	JSR \$F8D0	; STOPTOETS?
F8CA	20 BC F6	JSR \$F6BC	; BIJ INGEDRUKTE STOPTOETS
			FLAG ZETTEN
F8CD	4C BE F8	JMP \$F8BE	; WACHTEN
*****			STOPTOETS?
F8D0	20 E1 FF	JSR \$FFE1	; STOPTOETS INLEZEN
F8D3	18	CLC	
F8D4	D0 0B	BNE \$F8E1	; NEE, DAN TERUG
F8D6	20 93 FC	JSR \$FC93	; CASSETTEMOTOR UIT, NORMALE
			IRQ HERSTELLEN
F8D9	38	SEC	; KENMERK VOOR BREAK
F8DA	68	PLA	
F8DB	68	PLA	; SPRONGADRES WISSEN
F8DC	A9 00	LDA #\$00	

```

F8DE 8D A0 02 STA $02A0      ; KENMERK VOOR NOMALE IRQ
F8E1 60          RTS
***** BAND VOOR LEZEN VOORBEREIDEN
F8E2 86 B1      STX $B1
F8E4 A5 B0      LDA $B0
F8E6 0A         ASL
F8E7 0A         ASL
F8E8 18         CLC
F8E9 65 B0      ADC $B0
F8EB 18         CLC
F8EC 65 B1      ADC $B1
F8EE 85 B1      STA $B1
F8F0 A9 00      LDA #$00
F8F2 24 B0      BIT $B0
F8F4 30 01      BMI $F8F7
F8F6 2A         ROL
F8F7 06 B1      ASL $B1
F8F9 2A         ROL
F8FA 06 B1      ASL $B1
F8FC 2A         ROL
F8FD AA         TAX
F8FE AD 06 DC LDA $DC06      ; TIMER B LOW
F901 C9 16      CMP #$16
F903 90 F9      BCC $F8FE
F905 65 B1      ADC $B1
F907 8D 04 DC STA $DC04      ; TIMER A LOW
F90A 8A         TXA
F90B 6D 07 DC ADC $DC07      ; TIMER B HIGH
F90E 8D 05 DC STA $DC05      ; TIMER A HIGH
F911 AD A2 02 LDA $02A2
F914 8D 0E DC STA $DC0E
F917 8D A4 02 STA $02A4
F91A AD 0D DC LDA $DC0D      ; INPUT VAN BAND LEZEN: PIN-FLAG
F91D 29 10      AND #$10      ; BIT ISOLEREN
F91F F0 09      BEQ $F92A
F921 A9 F9      LDA #$F9
F923 48         PHA            ; SPRONGADRES OP STACK
F924 A9 2A      LDA #$2A
F926 48         PHA
F927 4C 43 FF JMP $FF43      ; NAAR INTERRUPT
F92A 58         CLI
F92B 60          RTS
***** INTERRUPTROUTINE VOOR BAND LEZEN
F92C AE 07 DC LDX $DC07      ; TIMER B HIGH
F92F A0 FF      LDY #$FF
F931 98         TYA
F932 ED 06 DC SBC $DC06      ; TIMER B LOW
F935 EC 07 DC CPX $DC07      ; TIMER B HIGH
F938 D0 F2      BNE $F92C

```

F93A	86	B1	STX	\$B1	
F93C	AA		TAX		
F93D	8C	06	DC	STY	\$DC06 ; TIMER B LOW
F940	8C	07	DC	STY	\$DC07 ; TIMER B HIGH
F943	A9	19	LDA	#19	
F945	8D	0F	DC	STA	\$DCOF ; IRQ VAN TIMER B
F948	AD	0D	DC	LDA	\$DCOD ; INPUT VAN BAND: PIN-FLAG
F94B	8D	A3	02	STA	\$02A3
F94E	98		TYA		
F94F	E5	B1	SBC	\$B1	
F951	86	B1	STX	\$B1	
F953	4A		LSR		
F954	66	B1	ROR	\$B1	
F956	4A		LSR		
F957	66	B1	ROR	\$B1	
F959	A5	B0	LDA	\$B0	
F95B	18		CLC		
F95C	69	3C	ADC	#\$3C	
F95E	C5	B1	CMP	\$B1	
F960	B0	4A	BCS	\$F9AC	
F962	A6	9C	LDX	\$9C	
F964	F0	03	BEQ	\$F969	
F966	4C	60	FA	JMP	\$FA60
F969	A6	A3	LDX	\$A3	
F96B	30	1B	BMI	\$F988	
F96D	A2	00	LDX	#\$00	
F96F	69	30	ADC	#\$30	
F971	65	B0	ADC	\$B0	
F973	C5	B1	CMP	\$B1	
F975	B0	1C	BCS	\$F993	
F977	E8		INX		
F978	69	26	ADC	#\$26	
F97A	65	B0	ADC	\$B0	
F97C	C5	B1	CMP	\$B1	
F97E	B0	17	BCS	\$F997	
F980	69	2C	ADC	#\$2C	
F982	65	B0	ADC	\$B0	
F984	C5	B1	CMP	\$B1	
F986	90	03	BCC	\$F98B	
F988	4C	10	FA	JMP	\$FA10
F98B	A5	B4	LDA	\$B4	
F98D	F0	1D	BEQ	\$F9AC	
F98F	85	A8	STA	\$A8	
F991	D0	19	BNE	\$F9AC	
F993	E6	A9	INC	\$A9	
F995	B0	02	BCS	\$F999	
F997	C6	A9	DEC	\$A9	
F999	38		SEC		
F99A	E9	13	SBC	#\$13	

F99C	E5 B1	SBC \$B1
F99E	65 92	ADC \$92
F9A0	85 92	STA \$92
F9A2	A5 A4	LDA \$A4
F9A4	49 01	EOR #\$01
F9A6	85 A4	STA \$A4
F9A8	F0 2B	BEQ \$F9D5
F9AA	86 D7	STX \$D7
F9AC	A5 B4	LDA \$B4
F9AE	F0 22	BEQ \$F9D2
F9B0	AD A3 02	LDA \$02A3
F9B3	29 01	AND #\$01
F9B5	D0 05	BNE \$F9BC
F9B7	AD A4 02	LDA \$02A4
F9BA	D0 16	BNE \$F9D2
F9BC	A9 00	LDA #\$00
F9BE	85 A4	STA \$A4
F9C0	8D A4 02	STA \$02A4
F9C3	A5 A3	LDA \$A3
F9C5	10 30	BPL \$F9F7
F9C7	30 BF	BMI \$F988
F9C9	A2 A6	LDX #\$A6
F9CB	20 E2 F8	JSR \$F8E2
F9CE	A5 9B	LDA \$9B
F9D0	D0 B9	BNE \$F98B
F9D2	4C BC FE	JMP \$FEBC
F9D5	A5 92	LDA \$92
F9D7	F0 07	BEQ \$F9E0
F9D9	30 03	BMI \$F9DE
F9DB	C6 B0	DEC \$B0
F9DD	2C E6 B0	BIT \$BOE6
F9E0	A9 00	LDA #\$00
F9E2	85 92	STA \$92
F9E4	E4 D7	CPX \$D7
F9E6	D0 0F	BNE \$F9F7
F9E8	8A	TXA
F9E9	D0 A0	BNE \$F98B
F9EB	A5 A9	LDA \$A9
F9ED	30 BD	BMI \$F9AC
F9EF	C9 10	CMP #\$10
F9F1	90 B9	BCC \$F9AC
F9F3	85 96	STA \$96
F9F5	B0 B5	BCS \$F9AC
F9F7	8A	TXA
F9F8	45 9B	EOR \$9B
F9FA	85 9B	STA \$9B
F9FC	A5 B4	LDA \$B4
F9FE	F0 D2	BEQ \$F9D2
FA00	C6 A3	DEC \$A3

; TERUGKEER VAN INTERRUPT

FA02	30	C5	BMI	\$F9C9	
FA04	46	D7	LSR	\$D7	
FA06	66	BF	ROR	\$BF	
FA08	A2	DA	LDX	#\$DA	
FA0A	20	E2	F8 JSR	\$F8E2	
FA0D	4C	BC	FE JMP	\$FEBC	; TERUGKEER VAN INTERRUPT
FA10	A5	96	LDA	\$96	
FA12	F0	04	BEQ	\$FA18	
FA14	A5	B4	LDA	\$B4	
FA16	F0	07	BEQ	\$FA1F	
FA18	A5	A3	LDA	\$A3	
FA1A	30	03	BMI	\$FA1F	
FA1C	4C	97	F9 JMP	\$F997	
FA1F	46	B1	LSR	\$B1	
FA21	A9	93	LDA	#\$93	
FA23	38		SEC		
FA24	E5	B1	SBC	\$B1	
FA26	65	B0	ADC	\$B0	
FA28	0A		ASL		
FA29	AA		TAX		
FA2A	20	E2	F8 JSR	\$F8E2	
FA2D	E6	9C	INC	\$9C	
FA2F	A5	B4	LDA	\$B4	
FA31	D0	11	BNE	\$FA44	
FA33	A5	96	LDA	\$96	
FA35	F0	26	BEQ	\$FA5D	
FA37	85	A8	STA	\$A8	
FA39	A9	00	LDA	#\$00	
FA3B	85	96	STA	\$96	
FA3D	A9	81	LDA	#\$81	
FA3F	8D	0D	DC STA	\$DCOD	; IRQ BIJ ONDERLOOP TIMER A
FA42	85	B4	STA	\$B4	
FA44	A5	96	LDA	\$96	
FA46	85	B5	STA	\$B5	
FA48	F0	09	BEQ	\$FA53	
FA4A	A9	00	LDA	#\$00	
FA4C	85	B4	STA	\$B4	
FA4E	A9	01	LDA	#\$01	
FA50	8D	0D	DC STA	\$DCOD	; IRQ-FLAG WEER WISSEN
FA53	A5	BF	LDA	\$BF	
FA55	85	BD	STA	\$BD	
FA57	A5	A8	LDA	\$A8	
FA59	05	A9	ORA	\$A9	
FA5B	85	B6	STA	\$B6	
FA5D	4C	BC	FE JMP	\$FEBC	; TERUGKEER VAN INTERRUPT
FA60	20	97	FB JSR	\$FB97	; BITTELLER ZETTEN VOOR SERIELE UITVOER
FA63	85	9C	STA	\$9C	
FA65	A2	DA	LDX	#\$DA	

FA67	20	E2	F8	JSR	\$F8E2	
FA6A	A5	BE		LDA	\$BE	
FA6C	F0	02		BEQ	\$FA70	
FA6E	85	A7		STA	\$A7	
FA70	A9	0F		LDA	#\$0F	
FA72	24	AA		BIT	\$AA	
FA74	10	17		BPL	\$FA8D	
FA76	A5	B5		LDA	\$B5	
FA78	D0	0C		BNE	\$FA86	
FA7A	A6	BE		LDX	\$BE	
FA7C	CA			DEX		
FA7D	D0	0B		BNE	\$FA8A	
FA7F	A9	08		LDA	#\$08	; LONG BLOCK ERROR
FA81	20	1C	FE	JSR	\$FE1C	; STATUS ZETTEN
FA84	D0	04		BNE	\$FA8A	
FA86	A9	00		LDA	#\$00	
FA88	85	AA		STA	\$AA	
FA8A	4C	BC	FE	JMP	\$FEBC	; TERUGKEER VAN INTERRUPT
FA8D	70	31		BVS	\$FAC0	
FA8F	D0	18		BNE	\$FAA9	
FA91	A5	B5		LDA	\$B5	
FA93	D0	F5		BNE	\$FA8A	
FA95	A5	B6		LDA	\$B6	
FA97	D0	F1		BNE	\$FA8A	
FA99	A5	A7		LDA	\$A7	
FA9B	4A			LSR		
FA9C	A5	BD		LDA	\$BD	
FA9E	30	03		BMI	\$FAA3	
FAA0	90	18		BCC	\$FABA	
FAA2	18			CLC		
FAA3	B0	15		BCS	\$FABA	
FAA5	29	0F		AND	#\$0F	
FAA7	85	AA		STA	\$AA	
FAA9	C6	AA		DEC	\$AA	
FAAB	D0	DD		BNE	\$FA8A	
FAAD	A9	40		LDA	#\$40	
FAAF	85	AA		STA	\$AA	
FAB1	20	8E	FB	JSR	\$FB8E	
FAB4	A9	00		LDA	#\$00	
FAB6	85	AB		STA	\$AB	
FAB8	F0	D0		BEQ	\$FA8A	
FABA	A9	80		LDA	#\$80	
FABC	85	AA		STA	\$AA	
FABE	D0	CA		BNE	\$FA8A	
FAC0	A5	B5		LDA	\$B5	
FAC2	F0	0A		BEQ	\$FACE	
FAC4	A9	04		LDA	#\$04	; SHORT BLOK ERROR
FAC6	20	1C	FE	JSR	\$FE1C	; STATUS ZETTEN
FAC9	A9	00		LDA	#\$00	

FACB	4C 4A	FB JMP	\$FB4A	
FACE	20 D1	FC JSR	\$FCD1	; EINDADRES AL BEREIKT?
FAD1	90 03	BCC	\$FAD6	; NEE
FAD3	4C 48	FB JMP	\$FB48	
FAD6	A6 A7	LDX	\$A7	
FAD8	CA	DEX		
FAD9	F0 2D	BEQ	\$FB08	
FADB	A5 93	LDA	\$93	; LOAD/VERIFY-FLAG
FADD	F0 0C	BEQ	\$FAEB	; LOAD?
FADF	A0 00	LDY	#\$00	
FAE1	A5 BD	LDA	\$BD	; GELEZEN BYTE
FAE3	D1 AC	CMP (\$AC),Y		; VERGELIJKEN
FAE5	F0 04	BEQ	\$FAEB	; GELIJK?
FAE7	A9 01	LDA	#\$01	
FAE9	85 B6	STA	\$B6	; FLAG ZETTEN
FAEB	A5 B6	LDA	\$B6	
FAED	F0 4B	BEQ	\$FB3A	; BYTE IN ORDE?
FAEF	A2 3D	LDX	#\$3D	
FAF1	E4 9E	CPX	\$9E	
FAF3	90 3E	BCC	\$FB33	
FAF5	A6 9E	LDX	\$9E	
AF7	A5 AD	LDA	\$AD	
FAF9	9D 01 01	STA	\$0101,X	
FAFC	A5 AC	LDA	\$AC	; GEGEVENS VOOR CORRECTIE BIJ SECOND PASS
FAFE	9D 00 01	STA	\$0100,X	
FB01	E8	INX		
FB02	E8	INX		
FB03	86 9E	STX	\$9E	
FB05	4C 3A	FB JMP	\$FB3A	
FB08	A6 9F	LDX	\$9F	
FBOA	E4 9E	CPX	\$9E	
FBOC	F0 35	BEQ	\$FB43	
FBOE	A5 AC	LDA	\$AC	
FB10	DD 00 01	CMP	\$0100,X	; FOUTCORRECTIE BIJ PASS 2
FB13	D0 2E	BNE	\$FB43	
FB15	A5 AD	LDA	\$AD	
FB17	DD 01 01	CMP	\$0101,X	
FB1A	D0 27	BNE	\$FB43	
FB1C	E6 9F	INC	\$9F	; CORRECTIETELLER MET 2 VERHOGEN
FB1E	E6 9F	INC	\$9F	; VERIFY?
FB20	A5 93	LDA	\$93	; NEE
FB22	F0 0B	BEQ	\$FB2F	; INGELEZEN BYTE
FB24	A5 BD	LDA	\$BD	
FB26	A0 00	LDY	#\$00	; MET GEHEUGENINHOUD VERGELIJKEN
FB28	D1 AC	CMP (\$AC),Y		; GELIJK, DAN VOLGENDE BYTE
FB2A	F0 17	BEQ	\$FB43	
FB2C	C8	INY		
FB2D	84 B6	STY	\$B6	; FLAG VOOR FOUT ZETTEN

FB2F	A5 B6	LDA \$B6	; FOUTFLAG ?
FB31	F0 07	BEQ \$FB3A	; GEEN FOUT?
FB33	A9 10	LDA #\$10	; SECOND PASS ERROR
FB35	20 1C FE	JSR \$FE1C	; STATUS ZETTEN
FB38	D0 09	BNE \$FB43	; EN VOLGENDE BYTE VERWERKEN
FB3A	A5 93	LDA \$93	; VERIFY?
FB3C	D0 05	BNE \$FB43	; JA
FB3E	A8	TAY	
FB3F	A5 BD	LDA \$BD	; INGELEZEN BYTE
FB41	91 AC	STA (\$AC),Y	; OPSLAAN
FB43	20 DB FC	JSR \$FCDB	; ADRESWIJZER VERHOGEN
FB46	D0 43	BNE \$FB8B	; TERUGKEER VAN INTERRUPT
FB48	A9 80	LDA #\$80	
FB4A	85 AA	STA \$AA	; FLAG VOOR LEZEN OP EINDE
FB4C	78	SEI	
FB4D	A2 01	LDX #\$01	
FB4F	8E OD DC	STX \$DCOD	; IRQ VAN TIMER A VERHINDEREN
FB52	AE OD DC	LDX \$DCOD	; IRQ-FLAG WISSEN
FB55	A6 BE	LDX \$BE	; PASS-TELLER
FB57	CA	DEX	; VERLAGEN
FB58	30 02	BMI \$FB5C	
FB5A	86 BE	STX \$BE	; PASS-TELLER ONTHOUDEN
FB5C	C6 A7	DEC \$A7	; FOUT IN PASS?
FB5E	F0 08	BEQ \$FB68	
FB60	A5 9E	LDA \$9E	; FOUT IN PASS 1?
FB62	D0 27	BNE \$FB8B	; JA, TERUGKEER VAN INTERRUPT
FB64	85 BE	STA \$BE	; GEEN BLOK MEER VERWERKEN
FB66	F0 23	BEQ \$FB8B	; TERUGKEER VAN INTERRUPT
FB68	20 93 FC	JSR \$FC93	; PASS BEEINDIGD
FB6B	20 8E FB	JSR \$FB8E	; ADRES WEER OP PROGRAMMA BEGIN
FB6E	A0 00	LDY #\$00	
FB70	84 AB	STY \$AB	; PROEFSOM WISSEN
FB72	B1 AC	LDA (\$AC),Y	; PROGRAMMA PROEFSOM BEREKENEN
FB74	45 AB	EOR \$AB	
FB76	85 AB	STA \$AB	; EN OPSLAAN
FB78	20 DB FC	JSR \$FCDB	; ADRESWIJZER VERHOGEN
FB7B	20 D1 FC	JSR \$FCD1	; EINDADRES AL BEREIKT?
FB7E	90 F2	BCC \$FB72	; NEE, VERDER VERGELIJKEN
FB80	A5 AB	LDA \$AB	; BEREKENDE PROEFSOM
FB82	45 BD	EOR \$BD	; MET PROEFSOM VAN BAND
			VERGELIJKEN
FB84	F0 05	BEQ \$FB8B	; PROEFSOM OK?
FB86	A9 20	LDA #\$20	; PROEFSOM ERROR
FB88	20 1C FE	JSR \$FE1C	; STATUS ZETTEN
FB8B	4C BC FE	JMP \$FEBC	; TERUGKEER VAN INTERRUPT
*****			ACTULE WIJZER OP
			BEGIN VAN PROGRAMMA
FB8E	A5 C2	LDA \$C2	
FB90	85 AD	STA \$AD	; BEGINADRES \$C1/\$C2

```

FB92  A5 C1    LDA $C1          ; NAAR $AC/$AD
FB94  85 AC    STA $AC
FB96  60       RTS
***** BITTELLER VOOR SERIELE
; UITVOER ZETTEN
FB97  A9 08    LDA #$08
FB99  85 A3    STA $A3          ; 8 BITS
FB9B  A9 00    LDA #$00
FB9D  85 A4    STA $A4
FB9F  85 A8    STA $A8          ; WERKGEHEUGEN WISSEN
FBA1  85 9B    STA $9B
FBA3  85 A9    STA $A9
FBA5  60       RTS
***** BIT OP BAND SCHRIJVEN
FBA6  A5 BD    LDA $BD          ; BIT IN $BD
FBA8  4A       LSR A            ; BIT 0 IN CARRY
FBA9  A9 60    LDA #$60          ; TIJD VOOR BIT 0
FBAB  90 02    BCC $FBAF
FBAD  A9 B0    LDA #$B0          ; TIJD VOOR BIT 1
FBAF  A2 00    LDX #$00
FBB1  8D 06 DC STA $DC06          ; TIMER B LOW
FBB4  8E 07 DC STX $DC07          ; TIMER B HIGH
FBB7  AD 0D DC LDA $DC0D          ; INTERRUPT-FLAG WISSEN
FBBA  A9 19    LDA #$19
FBBC  8D 0F DC STA $DC0F          ; TIMER B STARTEN
FBBF  A5 01    LDA $01
FBC1  49 08    EOR #$08          ; UITVOER-BIT VOOR BAND
; INVERTEREN
FBC3  85 01    STA $01
FBC5  29 08    AND #$08          ; ACTUELE PULS ONTHOUDEN
FBC7  60       RTS
FBC8  38       SEC
FBC9  66 B6    ROR $B6
FBCB  30 3C    BMI $FC09          ; TERUGKEER VAN INTERRUPT
***** INTERRUPTROUTINE VOOR BAND
; SCHRIJVEN
FBCD  A5 A8    LDA $A8
FBCF  D0 12    BNE $FBE3
FBD1  A9 10    LDA #$10          ; TIMER OP $110 (272)
FBD3  A2 01    LDX #$01
FBD5  20 B1 FB JSR $FBB1          ; FREQUENTIE OP BAND SCHRIJVEN
FBD8  D0 2F    BNE $FC09          ; TERUGKEER VAN INTERRUPT
FBDA  E6 A8    INC $A8
FBDC  A5 B6    LDA $B6
FBDE  10 29    BPL $FC09          ; TERUGKEER VAN INTERRUPT
FBE0  4C 57 FC JMP $FC57          ; TWEEDE BLOK SCHRIJVEN
FBE3  A5 A9    LDA $A9
FBE5  D0 09    BNE $FBF0
FBE7  20 AD FB JSR $FBAD          ; BIT SCHRIJVEN

```

FBEA	D0 1D	BNE \$FC09	; TERUGKEER VAN INTERRUPT
FBEC	E6 A9	INC \$A9	
FBEE	D0 19	BNE \$FC09	; TERUGKEER VAN INTERRUPT
FBF0	20 A6 FB	JSR \$FBA6	; BIT OP BAND SCHRIJVEN
FBF3	D0 14	BNE \$FC09	; TERUGKEER VAN INTERRUPT
FBF5	A5 A4	LDA \$A4	
FBF7	49 01	EOR #\$01	
FBF9	85 A4	STA \$A4	
FBFB	F0 0F	BEQ \$FC0C	
FBFD	A5 BD	LDA \$BD	
FBFF	49 01	EOR #\$01	; BIT VOOR UITVOER INVERTEREN
FC01	85 BD	STA \$BD	
FC03	29 01	AND #\$01	
FC05	45 9B	EOR \$9B	
FC07	85 9B	STA \$9B	
FC09	4C BC FE	JMP \$FEBC	; TERUGKEER VAN INTERRUPT
FC0C	46 BD	LSR \$BD	; VOLGENDE BIT IN POSITIE 0.
FC0E	C6 A3	DEC \$A3	; BITTELLER VERLAGEN
FC10	A5 A3	LDA \$A3	
FC12	F0 3A	BEQ \$FC4E	; VOLGENDE BIT UITVOEREN
FC14	10 F3	BPL \$FC09	; TERUGKEER VAN INTERRUPT
FC16	20 97 FB	JSR \$FB97	; BITTELLER WEER OP 8 ZETTEN
FC19	58	CLI	
FC1A	A5 A5	LDA \$A5	
FC1C	F0 12	BEQ \$FC30	
FC1E	A2 00	LDX #\$00	
FC20	86 D7	STX \$D7	
FC22	C6 A5	DEC \$A5	
FC24	A6 BE	LDX \$BE	
FC26	E0 02	CPX #\$02	
FC28	D0 02	BNE \$FC2C	; TERUGKEER VAN INTERRUPT
FC2A	09 80	ORA #\$80	; EINDADRES AL BEREIKT?
FC2C	85 BD	STA \$BD	
FC2E	D0 D9	BNE \$FC09	
FC30	20 D1 FC	JSR \$FCD1	
FC33	90 0A	BCC \$FC3F	
FC35	D0 91	BNE \$FBC8	
FC37	E6 AD	INC \$AD	
FC39	A5 D7	LDA \$D7	
FC3B	85 BD	STA \$BD	
FC3D	B0 CA	BCS \$FC09	; TERUGKEER VAN INTERRUPT
FC3F	A0 00	LDY #\$00	
FC41	B1 AC	LDA (\$AC),Y	; TE SCHRIJVEN BYTE
FC43	85 BD	STA \$BD	
FC45	45 D7	EOR \$D7	
FC47	85 D7	STA \$D7	
FC49	20 DB FC	JSR \$FCDB	; ADRESWIJZER VERHOGEN
FC4C	D0 BB	BNE \$FC09	; TERUGKEER VAN INTERRUPT
FC4E	A5 9B	LDA \$9B	

```

FC50 49 01    EOR #$01
FC52 85 BD    STA $BD
FC54 4C BC FE JMP $FEBC      ; TERUGKEER VAN INTERRUPT
FC57 C6 BE    DEC $BE        ; TELLER VOOR BLOKKEN VERLAGEN
FC59 D0 03    BNE $FC5E      ; NOG EEN BLOK?
FC5B 20 CA FC JSR $FCCA      ; NEE, CASSETTEMOTOR UIT
FC5E A9 50    LDA #$50       ; 80
FC60 85 A7    STA $A7        ; TELLER VOOR IMPULSEN
FC62 A2 08    LDX #$08
FC64 78       SEI
FC65 20 BD FC JSR $FCBD      ; IRQ OP $FC6A
FC68 D0 EA    BNE $FC54      ; TERUGKEER VAN INTERRUPT
***** INTERRUPTROUTINE VOOR BAND
; SCHRIJVEN
FC6A A9 78    LDA #$78      ; 120
FC6C 20 AF FB JSR $FBAF      ; BIT OP BAND SCHRIJVEN
FC6F D0 E3    BNE $FC54      ; TERUGKEER VAN INTERRUPT
FC71 C6 A7    DEC $A7        ; TELLER VERLAGEN
FC73 D0 DF    BNE $FC54      ; ONGELIJK 0, DAN TERUGKEER
; VAN INTERRUPT
FC75 20 97 FB JSR $FB97      ; BITTELLER VOOR SERIELE
; UITVOER INSTELLEN
FC78 C6 AB    DEC $AB
FC7A 10 D8    BPL $FC54      ; TERUGKEER VAN INTERRUPT
FC7C A2 0A    LDX #$0A
FC7E 20 BD FC JSR $FCBD      ; IRQ OP $FB97
FC81 58       CLI
FC82 E6 AB    INC $AB
FC84 A5 BE    LDA $BE        ; TELLER VOOR AANTAL BLOKKEN
FC86 F0 30    BEQ $FCB8      ; ALLE BLOKKEN GESCHREVEN?
FC88 20 8E FB JSR $FB8E      ; ADRES OP BEGIN ZETTEN
FC8B A2 09    LDX #$09
FC8D 86 A5    STX $A5        ; TELLER VOOR SYNCHRONISATIE
FC8F 86 B6    STX $B6        ; FLAG VOOR BLOK GESCHREVEN
FC91 D0 83    BNE $FC16
***** STOPPEN MET GEBRUIK VAN RECORDER
FC93 08       PHP           ; STATUS ONTHOUDEN
FC94 78       SEI
FC95 AD 11 DO LDA $D011
FC98 09 10    ORA #$10      ; BEELDSCHERM INSCHAKELEN
FC9A 8D 11 DO STA $D011
FC9D 20 CA FC JSR $FCCA      ; CASSETTEMOTOR UIT
FCA0 A9 7F    LDA #$7F
FCA2 8D 0D DC STA $DCOD      ; INTERRUPTMOGELIJKHEDEN WISSEN
FCA5 20 DD FD JSR $FDDD      ; CIA WEER OP STANDAARDWAARDEN,
; 1/60 SECONDE TIMING
FCA8 AD A0 02 LDA $02A0      ; INTERRUPTVECTOR AL OP
; STANDAARDWAARDE?
FCAB F0 09    BEQ $FCB6      ; JA, KLAAR

```

```

FCAD 8D 15 03 STA $0315
FCBO AD 9F 02 LDA $029F      ; IRQ TERUGHALEN
FCB3 8D 14 03 STA $0314
FCB6 28      PLP              ; STATUS TERUGHALEN
FCB7 60      RTS
***** IRQ-VECTOR ZETTEN,
; X-GEINDICEERD
FCB8 20 93 FC JSR $FC93      ; IRQ OP DEFAULT
FCBB FO 97    BEQ $FC54
FCBD BD 93 FD LDA $FD93,X
FCCO 8D 14 03 STA $0314      ; IRQ-VECTOR UIT TABEL
; INSTELLEN
FCC3 BD 94 FD LDA $FD94,X
FCC6 8D 15 03 STA $0315
FCC9 60      RTS
*****
FCCA A5 01    LDA $01
FCCC 09 20    ORA #$20        ; CASSETTEMOTOR UITSCHAKELEN
FCCE 85 01    STA $01
FCDO 60      RTS
***** EINDADRES BEREIKT?
FCD1 38      SEC
FCD2 A5 AC    LDA $AC          ; ACTUELE ADRES $AE/$AF
FCD4 E5 AE    SBC $AE
FCD6 A5 AD    LDA $AD          ; EINDADRES $AE/$AF
FCD8 E5 AF    SBC $AF
FCDA 60      RTS
***** ADRESWIJZER VERHOGEN
FCDB E6 AC    INC $AC
FCDD DO 02    BNE $FCE1
FCDF E6 AD    INC $AD
FCE1 60      RTS
***** RESET
FCE2 A2 FF    LDX #$FF
FCE4 78      SEI
FCE5 9A      TXS
FCE6 D8      CLD
FCE7 20 02 FD JSR $FD02        ; ROM IN $8000
FCEA DO 03    BNE $FCEF        ; GEEN AUTOSTART-MODUUL?
FCEC 6C 00 80 JMP ($8000)      ; SPRONG NAAR BEGIN VAN MODUUL
FCEF 8E 16 DO STX $D016        ; VIDEOCONTROLLER VOOR STUURREGISTER 2
FCF2 20 A3 FD JSR $FDA3        ; INTERRUPT VOORBEREIDEN
FCF5 20 50 FD JSR $FD50        ; WERKGEHEUGEN INITIALISEREN
FCF8 20 15 FD JSR $FD15        ; HARDWARE EN I/O-VECTOREN INSTELLEN
FCFB 20 5B FF JSR $FF5B        ; VIDEO RESET
FCFE 58      CLI
FCFF 6C 00 A0 JMP ($A000)      ; NAAR BASIC KOUDE START
***** ROM IN $8000?
FD02 A2 05    LDX m$05

```

```

FD04 BD 0F FD LDA $FDOF,X
FD07 DD 03 80 CMP $8003,X ; VERGELIJK MET CBM80
FDOA DO 03 BNE $FDOF
FDOC CA DEX
FDOD DO F5 BNE $FD04
FDOF 60 RTS
***** ROM-MODUUL IDENTIFICEREN
FD10 C3 C2 CD 38 30 ; CBM80
***** HARDWARE- EN I/O-VECTOREN
; HALEN EN INSTELLEN
FD12 CD 38 30 CMP $3038
FD15 A2 30 LDX m$30
FD17 A0 FD LDY m$FD ; WIJZER OP TABEL $FD30
FD19 18 CLC
FD1A 86 C3 STX $C3
FD1C 84 C4 STY $C4
FD1E A0 1F LDY m$1F
FD20 B9 14 03 LDA $0314,Y
FD23 B0 02 BCS $FD27 ; C=1, DAN VECTOREN HALEN
; C=0 INSTELLEN
FD25 B1 C3 LDA ($C3),Y
FD27 91 C3 STA ($C3),Y
FD29 99 14 03 STA $0314,Y
FD2C 88 DEY
FD2D 10 F1 BPL $FD20
FD2F 60 RTS
***** TABEL VOOR HARDWARE- EN
; EN I/O-VECTOREN
FD30 31 EA 66 FE 47 FE 4A F3
FD38 91 F2 0E F2 50 F2 33 F3
FD40 57 F1 CA F1 ED F6 3E F1
FD48 2F F3 66 FE A5 54 ED F5
***** WERKGEHEUGEN INITIALISEREN
FD50 A9 00 LDA m$00
FD52 A8 TAY
FD53 99 02 00 STA $0002,Y ; ZEROPAGE,
FD56 99 00 02 STA $0200,Y ; PAGE 2 EN
FD59 99 00 03 STA $0300,Y ; PAGE 3 WISSEN
FD5C C8 INY
FD5D DO F4 BNE $FD53
FD5F A2 3C LDX m$3C
FD61 A0 03 LDY m$03
FD63 86 B2 STX $B2 ; CASSETTEBUFFERWIJZER
; OP $033C
FD65 84 B3 STY $B3
FD67 A8 TAY
FD68 A9 03 LDA m$03
FD6A 85 C2 STA $C2 ; RAM VANAF $400?

```

```

FD6C E6 C2      INC $C2
FD6E B1 C1      LDA ($C1),Y
FD70 AA         TAX                      ; WAARDE ONTHOUDEN
FD71 A9 55      LDA m$55                ; %01010101
FD73 91 C1      STA ($C1),Y
FD75 D1 C1      CMP ($C1),Y
FD77 D0 0F      BNE $FD88
FD79 2A         ROL A                    ; %10101010
FD7A 91 C1      STA ($C1),Y
FD7C D1 C1      CMP ($C1),Y
FD7E D0 08      BNE $FD88
FD80 8A         TXA                      ; WAARDE TERUGSCHRIJVEN
FD81 91 C1      STA ($C1),Y
FD83 C8         INY
FD84 D0 E8      BNE $FD6E
FD86 F0 E4      BEQ $FD6C
FD88 98         TYA
FD89 AA         TAX
FD8A A4 C2      LDY $C2
FD8C 18         CLC
FD8D 20 2D FE   JSR $FE2D                ; MEMORY (RAM) TOP INSTELLEN
FD90 A9 08      LDA m$08
FD92 8D 82 02   STA $0282                ; MEMORY (RAM) START OP $800
FD95 A9 04      LDA m$04
FD97 8D 88 02   STA $0288                ; VIDEO-RAM OP $400
FD9A 60         RTS
*****          IRQ-VECTOREN
FD9B 6A FC CD FB 31 EA 2C F9; $FC6A, $FBCD, $EA31, $F92C
*****          INITIALISERING VAN INTERRUPT
FDA3 A9 7F      LDA m$7F                  ; INTERRUPT WISSEN
FDA5 8D 0D DC   STA $DCOD                 ; ICR VAN CIA 1
FDA8 8D 0D DD   STA $DDOD                 ; ICR VAN CIA 2
FDAB 8D 00 DC   STA $DC00                 ; POORT A VAN CIA 1, MATRIXREGEL 0
;                                                  VAN TOETSENBOARD
FDAE A9 08      LDA m$08
FDB0 8D 0E DC   STA $DC0E                 ; CRA: TIMER A VAN CIA 1, ONE SHOT
FDB3 8D 0E DD   STA $DD0E                 ; CRA: TIMER A VAN CIA 2, ONE SHOT
FDB6 8D 0F DC   STA $DC0F                 ; CRB: TIMER B VAN CIA 1, ONE SHOT
FDB9 8D 0F DD   STA $DD0F                 ; CRB: TIMER B VAN CIA 2, ONE SHOT
FDBC A2 00      LDX m$00                 ; INVOERMODOUS
FDBE 8E 03 DC   STX $DC03                 ; DATARICHTINGSREGISTER B VAN CIA 1
FDC1 8E 03 DD   STX $DD03                 ; DATARICHTINGSREGISTER B VAN CIA 2
FDC4 8E 18 D4   STX $D418                 ; VOLUME VOOR SID OP 0
FDC7 CA         DEX                      ; UITVOERMODOUS
FDC8 8E 02 DC   STX $DC02                 ; DATARICHTINGSREGISTER A VAN CIA 1
FDCB A9 07      LDA m$07                 ; VIDEOCONTROLLER OP ONDERSTE 16 K
FDCD 8D 00 DD   STA $DD00                 ; POORT A VAN CIA 2, ATN WISSEN
FDD0 A9 3F      LDA m$3F                 ; BIT 0 TOT 5 OP UITVOER
FDD2 8D 02 DD   STA $DD02                 ; DATARICHTINGSREGISTER A VAN CIA 2

```



```

FDD5 A9 E7 LDA m$E7
FDD7 85 01 STA $01 ; PROCESSORPOORT, GEHEUGENINDELING
FDD9 A9 2F LDA m$2F ; UITGANG: BIT 0-3 EN 5, INGANG:
; BIT 4
FDDB 85 00 STA $00 ; RICHTING VAN GEGEVENS VIA
; PROCESSORPOORT
FDDD AD A6 02 LDA $02A6 ; NTSC-VERSIE?
FDE0 FO 0A BEQ $FDEC ; JA
FDE2 A9 25 LDA m$25
FDE4 8D 04 DC STA $DC04 ; TIMER VOOR PAL-VERSIE INSTELLEN
FDE7 A9 40 LDA m$40 ; $4025 = 16421 CYCLI
FDE9 4C F3 FD JMP $FDF3
FDEC A9 95 LDA m$95
FDEE 8D 04 DC STA $DC04 ; TIMER VOOR NTSC-VERSIE INSTELLEN
FDF1 A9 42 LDA m$42 ; $4295 = 17045 CYCLI
FDF3 8D 05 DC STA $DC05 ; TIMER HIGH
FDF6 4C 6E FF JMP $FF6E ; INTERRUPT DOOR TIMER INSTELLEN
***** PARAMETERS VOOR FILENAMEN
; INSTELLEN
FDF9 85 B7 STA $B7 ; LENGTE
FDFB 86 BB STX $BB ; ADRES LOW
FDFD 84 BC STY $BC ; ADRES HIGH
FDFE 60 RTS
***** PARAMETERS VOOR ACTUELE
; FILE INSTELLEN
FE00 85 B8 STA $B8 ; LOGISCHE FILENUMMER
FE02 86 BA STX $BA ; DEVICE-NUMMER
FE04 84 B9 STY $B9 ; SECUNDAIR ADRES
FE06 60 RTS
***** STATUS HALEN
FE07 A5 BA LDA $BA ; DEVICE-NUMMER
FE09 C9 02 CMP #$02 ; 2?
FE0B D0 0D BNE $FE1A ; NEE
FE0D AD 97 02 LDA $0297 ; STATUS VAN RS-232 HALEN
FE10 48 PHA
FE11 A9 00 LDA #$00 ; STATUS WISSEN
FE13 8D 97 02 STA $0297
FE16 68 PLA
FE17 60 RTS
***** FLAG VOOR MELDING VAN
; BEDRIJFSSYSTEEM ZETTEN
FE18 85 9D STA $9D
FE1A A5 90 LDA $90
***** STATUS INSTELLEN
FE1C 05 90 ORA $90
FE1E 85 90 STA $90
FE20 60 RTS
***** TIME OUT FLAG VOOR IEC ZETTEN
FE21 8D 85 02 STA $0285

```

```

FE24 60      RTS
***** MEMTOP: BOVENGRENS BASIC-RAM
;        HALEN/INSTELLEN
FE25 90 06    BCC $FE2D
FE27 AE 83 02 LDX $0283      ; CARRY GEZET
FE2A AC 84 02 LDY $0284      ; ADRES NAAR X/Y HALEN
FE2D 8E 83 02 STX $0283      ; CARRY GEWIST
FE30 8C 84 02 STY $0284      ; X/Y NAAR ADRES BRENGEN
FE33 60      RTS
***** MEMBOT: ONDERGRENS BASIC-RAM
;        HALEN/INSTELLEN
FE34 90 06    BCC $FE3C
FE36 AE 81 02 LDX $0281      ; ZIE BOVEN
FE39 AC 82 02 LDY $0282
FE3C 8E 81 02 STX $0281
FE3F 8C 82 02 STY $0282
FE42 60      RTS
***** NMI-SPRONG
FE43 78      SEI
FE44 6C 18 03 JMP ($0318)    ; JMP $FE47, NMI-VECTOR
FE47 48      PHA
FE48 8A      TXA
FE49 48      PHA              ; REGISTER WEGZETTEN
FE4A 98      TYA
FE4B 48      PHA
FE4C A9 7F    LDA #$7F
FE4E 8D 0D DD STA $DD0D      ; NMI-MOGELIJKHEDEN WISSEN
FE51 AC 0D DD LDY $DD0D      ; FLAGS LEZEN EN WISSEN
FE54 30 1C    BMI $FE72      ; RS-232 ACTIEF?
FE56 20 02 FD JSR $FD02      ; ROM-MODUUL IN $8000?
FE59 D0 03    BNE $FE5E      ; NEE, VERDER
FE5B 6C 02 80 JMP ($8002)    ; JA, SPRONG NAAR NMI-MODUUL
FE5E 20 BC F6 JSR $F6BC      ; FLAG VOOR STOPTOETS ZETEN
FE61 20 E1 FF JSR $FFE1      ; STOPTOETS INLEZEN
FE64 D0 0C    BNE $FE72      ; NIET INGEDRUKT?
FE66 20 15 FD JSR $FD15      ; VECTOREN VOOR INTERRUPT EN
;        I/O OP DEFAULT
FE69 20 A3 FD JSR $FDA3      ; I/O INITIALISEREN
FE6C 20 18 E5 JSR $E518      ; I/O INITIALISEREN EN BEELD-
;        SCHERM WISSEN
FE6F 6C 02 A0 JMP ($A002)    ; NAAR BASIC WARME START
***** NMI-ROUTINE VOOR RS-232
FE72 98      TYA              ; ICR REGISTER
FE73 2D A1 02 AND $02A1      ; MET NMI-FLAG VOOR RS-232
;        SAMENVOEGEN
FE76 AA      TAX
FE77 29 01    AND #$01        ; AAN HET ZENDEN?
FE79 FO 28    BEQ $FEA3      ; NEE
FE7B AD 00 DD LDA $DD00      ; DATABUS LEZEN

```

```

FE7E 29 FB AND #$FB ; BIT 2 (TXD) WISSEN
FE80 05 B5 ORA $B5 ; TE ZENDEN BIT DOORGEVEN
FE82 8D 00 DD STA $DD00 ; EN WEER IN DATABUS OPSLAAN
FE85 AD A1 02 LDA $02A1 ; NMI-FLAG VOOR RS-232
FE88 8D 0D DD STA $DD0D ; WEER IN ICR SCHRIJVEN
FE8B 8A TXA
FE8C 29 12 AND #$12 ; BIT 1 EN 4 ISOLEREN
FE8E FO 0D BEQ $FE9D
FE90 29 02 AND #$02 ; BIT 1, OPROEP VAN TIMER B
FE92 FO 06 BEQ $FE9A ; NEE, STARTBIT
FE94 20 D6 FE JSR $FED6 ; ONTVANGEN BIT VERWERKEN
FE97 4C 9D FE JMP $FE9D
FE9A 20 07 FF JSR $FF07 ; ONTVANGST VAN VOLGENDE BYTE
; VOORBEREIDEN
FE9D 20 BB EE JSR $EEBB ; TERUGKEER VAN INTERRUPT
FEA0 4C B6 FE JMP $FEB6
FEA3 8A TXA
FEA4 29 02 AND #$02 ; ONTVANGST VAN GEGEVENS?
FEA6 FO 06 BEQ $FEAE ; NEE
FEA8 20 D6 FE JSR $FED6 ; ONTVANGEN BIT VERWERKEN
FEAB 4C B6 FE JMP $FEB6 ; TERUGKEER VAN INTERRUPT
FEAE 8A TXA
FEAF 29 10 AND #$10 ; WACHTEN OP STARTBIT?
FEB1 FO 03 BEQ $FEB6 ; NEE
FEB3 20 07 FF JSR $FF07 ; ONTVANGST VOLGENDE BYTE
; VOORBEREIDEN
FEB6 AD A1 02 LDA $02A1 ; NMI-FLAG VOOR RS-232
FEB9 8D 0D DD STA $DD0D ; WEER IN ICR
FEBB 68 PLA
FEBD A8 TAY
FEBE 68 PLA ; REGISTER TERUGHALEN
FEBF AA TAX
FECO 68 PLA
FEC1 40 RTI
*****
;
FEC2 C1 27 $27C1 = 10177 50 BAUD
FEC4 3E 1A $1A3E = 6718 75 BAUD
FEC6 C5 11 $11C5 = 4549 110 BAUD
FEC8 74 0E $0E74 = 3700 134.5 BAUD
FECA ED 0C $0CED = 3309 150 BAUD
FECC 45 06 $0645 = 1605 300 BAUD
FECE FO 02 $02F0 = 752 600 BAUD
FEDO 46 01 $0146 = 326 1200 BAUD
FED2 B8 00 $00B8 = 184 1800 BAUD
FED4 71 00 $0071 = 113 2400 BAUD
*****
;
FED6 AD 01 DD LDA $DD01 ; POORT VAN REGISTER B
NMI-ROUTINE VOOR INVOER VIA RS-232

```

```

FED9 29 01    AND #$01      ; BIT VOOR RECEIVE DATA ISOLEREN
FEDE 85 A7    STA $A7        ; EN OPSLAAN
FEDD AD 06 DD LDA $DD06      ; TIMER B LOW
FEE0 E9 1C    SBC #$1C      ; -28
FEE2 6D 99 02 ADC $0299
FEE5 8D 06 DD STA $DD06
FEE8 AD 07 DD LDA $DD07      ; TIMERCONSTANTEN VOOR
;                               BAUD RATE (R2S-232)
FEEB 6D 9A 02 ADC $029A
FEEE 8D 07 DD STA $DD07      ; IN TIMER SCHRIJVEN
FEF1 A9 11    LDA #$11      ; TIMER B STARTEN
FEF3 8D 0F DD STA $DD0F      ; CONTROLEREGISTER B
FEF6 AD A1 02 LDA $02A1
FEF9 8D 0D DD STA $DD0D      ; INTERRUPT CONTROLEREGISTER
FEFC A9 FF    LDA #$FF
FEFE 8D 06 DD STA $DD06
FF01 8D 07 DD STA $DD07      ; TIMER INSTELLEN
FF04 4C 59 EF JMP $EF59      ; BIT HALEN
***** NMI-ROUTINE VOOR UITVOER VIA
;                               RS-232
FF07 AD 95 02 LDA $0295
FF0A 8D 06 DD STA $DD06
FF0D AD 96 02 LDA $0296      ; TIMERCONSTANTEN VOOR
;                               BAUD RATE (RS-232)
FF10 8D 07 DD STA $DD07
FF13 A9 11    LDA #$11      ; TIMER B STARTEN
FF15 8D 0F DD STA $DD0F      ; CONTROLEREGISTER B
FF18 A9 12    LDA #$12
FF1A 4D A1 02 EOR $02A1      ; NMI-FLAG VOOR CIA 2
FF1D 8D A1 02 STA $02A1
FF20 A9 FF    LDA #$FF
FF22 8D 06 DD STA $DD06
FF25 8D 07 DD STA $DD07      ; TIMER LADEN
FF28 AE 98 02 LDX $0298      ; AANTAL TE ZENDEN BITS
FF2B 86 A8    STX $A8
FF2D 60       RTS
***** TIMERWAARDE VOOR BAUD RATE
;                               INSTELLEN VOOR ZENDEN
FF2E AA       TAX
FF2F AD 96 02 LDA $0296
FF32 2A       ROL A          ; *2
FF33 A8       TAY
FF34 8A       TXA
FF35 69 C8    ADC #$C8      ; +200
FF37 8D 99 02 STA $0299      ; TIMERWAARDE LOW
FF3A 98       TYA
FF3B 69 00    ADC #$00
FF3D 8D 9A 02 STA $029A      ; TIMERWAARDE HIGH
FF40 60       RTS

```

```

FF41 EA      NOP
FF42 EA      NOP
***** SPRONG VANUIT BANDROUTINE
FF43 08      PHP
FF44 68      PLA
FF45 29 EF    AND #$EF      ; BREAK FLAG WISSEN
FF47 48      PHA
***** SPRONG NAAR IRQ
FF48 48      PHA
FF49 8A      TXA
FF4A 48      PHA      ; REGISTER WEGZETTEN
FF4B 98      TYA
FF4C 48      PHA
FF4D BA      TSX
FF4E BD 04 01 LDA $0104,X  ; BREAK FLAG VAN STACK HALEN
FF51 29 10    AND #$10     ; EN TESTEN
FF53 F0 03    BEQ $FF58    ; NIET GEZET
FF55 6C 16 03 JMP ($0316)  ; BREAK-ROUTINE
FF58 6C 14 03 JMP ($0314)  ; INTERRUPT-ROUTINE
***** VIDEO RESET
FF5B 20 18 E5 JSR $E518    ; VIDEO CONTROLLER INITIALISEREN
FF5E AD 12 D0 LDA $D012    ; RASTERREGEL
FF61 D0 FB    BNE $FF5E    ; WACHT OP EINDE VIDEO-REGEL
FF63 AD 19 D0 LDA $D019    ; INTERRUPT DOOR RASTERREGEL?
FF66 29 01    AND #$01
FF68 8D A6 02 STA $02A6    ; PAL/NTSC-VERSIE ONTHOUDEN
FF6B 4C DD FD JMP $FDDD    ; INTERRUPTROUTINE INSTELLEN
***** TIMER VOOR INTERRUPT INSTELLEN
FF6E A9 81    LDA #$81     ; TIMER A ONDERLOOP
FF70 8D 0D DC STA $DCOD    ; INTERRUPT CONTROLEREGISTER
FF73 AD 0E DC LDA $DCOE    ; CONTROLEREGISTER A
FF76 29 80    AND #$80     ; BIT 7 REDDEN, TRIGGER VOOR KLOK
;                               (50/60 HZ)
FF78 09 11    ORA #$11     ; TIMER A STARTEN
FF7A 8D 0E DC STA $DCOE    ; CONTROLEREGISTER A
FF7D 4C 8E EE JMP $EE8E    ; SERIELE FREQUENTIE UIT
FF80 00      BRK
***** SPRONGTABEL: ROUTINES VOOR
;                               BEDRIJFSSYSTEEM
FF81 4C 5B FF JMP $FF5B    ; VIDEO RESET

FF84 4C A3 FD JMP $FDA3    ; CIA'S INITIALISEREN
FF87 4C 50 FD JMP $FD50    ; RAM WISSEN/TESTEN
FF8A 4C 15 FD JMP $FD15    ; I/O INITIALISEREN
FF8D 4C 1A FD JMP $FD1A    ; I/O-VECTOREN INITIALISEREN

```

FF90	4C 18 FE JMP \$FE18	; STATUS INSTELLEN
FF93	4C B9 ED JMP \$EDB9	; SECUNDAIR ADRES NA LISTEN ZENDEN
FF96	4C C7 ED JMP \$EDC7	; SECUNDAIR ADRES NA TALK ZENDEN
FF99	4C 25 FE JMP \$FE25	; RAM-EINDE INSTELLEN/HALEN
FF9C	4C 34 FE JMP \$FE34	; RAM-BEGIN INSTELLEN/HALEN
FF9F	4C 87 EA JMP \$EA87	; TOETSENBORD UITLEZEN
FFA2	4C 21 FE JMP \$FE21	; TIME OUT FLAG VOOR IEC-BUS INSTELLEN
FFA5	4C 13 EE JMP \$EE13	; INVOER VAN IEC-BUS
FFA8	4C DD ED JMP \$EDDD	; UITVOER VAN IEC-BUS
FFAB	4C EF ED JMP \$EDEF	; UNTALK ZENDEN
FFAE	4C FE ED JMP \$EDFE	; UNLISTEN ZENDEN
FFB1	4C 0C ED JMP \$EDOC	; LISTEN ZENDEN
FFB4	4C 09 ED JMP \$ED09	; TALK ZENDEN
FFB7	4C 07 FE JMP \$FE07	; STATUS HALEN
FFBA	4C 00 FE JMP \$FE00	; FILEPARAMETERS INSTELLEN
FFBD	4C F9 FD JMP \$FDF9	; FILENAAMPARAMETERS INSTELLEN
FFC0	6C 1A 03 JMP (\$031A)	; \$F34A OPEN
FFC3	6C 1C 03 JMP (\$031C)	; \$F291 CLOSE
FFC6	6C 1E 03 JMP (\$031E)	; \$F20E CHKIN: INVOERAPPARAAT
	;	INSTELLEN
FFC9	6C 20 03 JMP (\$0320)	; \$250 CKOUT: UITVOERAPPARAAT
	;	INSTELLEN
FFCC	6C 22 03 JMP (\$0322)	; \$F333 CLRCH: IN/UITVOER TERUGZETTEN
FFCF	6C 24 03 JMP (\$0324)	; \$F157 BASIN: INVOER VAN EEN TEKEN
FFD2	6C 26 03 JMP (\$0326)	; \$F1CA BSOUT: UITVOER VAN EEN TEKEN
FFD5	4C 9E F4 JMP \$F49E	; LOAD

```

FFD8 4C DD F5 JMP $F5DD      ; SAVE
FFDB 4C E4 F6 JMP $F6E4      ; TIME ZETTEN
FFDE 4C DD F6 JMP $F6DD      ; TIME HALEN
FFE1 6C 28 03 JMP ($0328)    ; $F6ED  STOPTOETS INLEZEN
FFE4 6C 2A 03 JMP ($032A)    ; $F13E  GET
FFE7 6C 2C 03 JMP ($032C)    ; $F32F  CLALL
FFEA 4C 9B F6 JMP $F69B      ; TIME VERHOGEN
FFED 4C 05 E5 JMP $E505      ; SCREEN: AANTAL REGELS/KOLOMMEN HALEN
FFF0 4C 0A E5 JMP $E50A      ; CURSOR INSTELLEN/CURSORPOSITIE HALEN
FFF3 4C 00 E5 JMP $E500      ; BEGINADRES VAN I/O-BEREIK HALEN
FFF6 52 52 42 59

***** HARDWAREVECTOREN
FFFA 43 FE      $FE43        ; NMI-VECTOR
FFFC E2 FC      $FCE2        ; RESET-VECTOR
FFFE 48 FF      $FF48        ; IRQ-VECTOR

```

BEDRIJFSSYSTEEM VAN DE SX-64

In de volgende tabel staan de afwijkingen van het bedrijfssysteem van de SX-64 ten opzichte van dat van de Commodore 64.

```

***** INSCHAKELMELDING
E479 20 2A 2A 2A 2A 2A 20 20; *****
E481 53 58 2D 36 34 20 42 41; SX-64 BA
E489 53 49 43 20 56 32 2E 30; SIC V2.0
E491 20 20 2A 2A 2A 2A 2A 0D; *****
E4AC B3
***** RS-232-ROUTINES
E4D3 85 A9      STA $A9
E4D5 A9 01      LDA #$01
E4D7 85 AB      STA $AB
E4D9 60         RTS
***** ACHTERGRONDKLEUR BEPALEN
E4DA AD 86 02   LDA $0286      ; ACTUELE TEKSTKLEUR
E534 A9 06      LDA #$06      ; KLEUR BIJ INSCHAKELLEN (BLAUW)

```

***** CURSORPOSITIE BEREKENEN

```

E57C 20 FO E9 JSR $E9FO
E57F A9 27 LDA #$27 ; 39
E581 E8 INX
E582 B4 D9 LDY $D9,Y ; MSB VAN BEGINADRES VAN REGEL
E584 30 06 BMI $E58C
E586 18 CLC
E587 69 28 ADC #$28 ; +40
E589 E8 INX
E58A 10 F6 BPL $E582
E58C 85 D5 STA $D5
E58E 4C 24 EA JMP $EA24 ; WIJZER OP KLEUR-RAM BEREKENEN
E591 E4 C9 CPX $C9
E593 FO 03 BEQ $E598
E595 4C ED E6 JMP $E6ED ; MSB'S VOOR REGEL BEREKENEN
E598 60 RTS
E599 EA NOP

```

***** TEKST NA INDRUKKEN VAN

```

; SHIFT-RUN/STOP
E5EE A2 OF LDX #$0F ; 15 TEKENS
E5FO 78 SEI
E5F1 86 C6 STX $C6 ; AANTAL TEKENS IN
; IN TOETSENBORDBUFFER
E5F3 BD D7 FO LDA $FOD7,X ; TEKST: LOAD" :* ",8 <CR> RUN <CR>
E5F6 9D 76 02 STA $0276,X ; IN TOETSENBORDBUFFER SCHRIJVEN
*****

```

E621 20 91 E5 JSR \$E591 ; CURSORPOSITIE BEELDSCHERM

***** BEELDSCHERMREGEL WISSEN

```

EA07 20 DA E4 JSR $E4DA ; CURSORKLEUR INSTELLEN
EA0A A9 20 LDA #$20 ; SPATIE
EA0C 91 D1 STA $(D1),Y ; IN VIDEO-RAM SCHRIJVEN
EA0E 88 DEY
EA0F 10 F6 BPL $EA07
EA11 60 RTS
EA12 EA NOP

```

ECD9 03 01

EF94 4C D3 E4 JMP \$E4D3 ; NAAR GEWIJZIGDE RS-232-ROUTINES

***** TEKST NA SHIFT-RUN/STOP

FOD8 4C 4F 41 44 22 3A 2A 22; LOAD" :* "

FOEO 2C 38 OD 52 55 4E OD ; ,8 RUN

***** DEVICE-NUMMER 1 NEGEREN (DATASETTE)

F386 DO 08 BNE \$F390

F4F6 90 85 BCC \$F47D

F5F8 90 F7 BCC \$F5F1

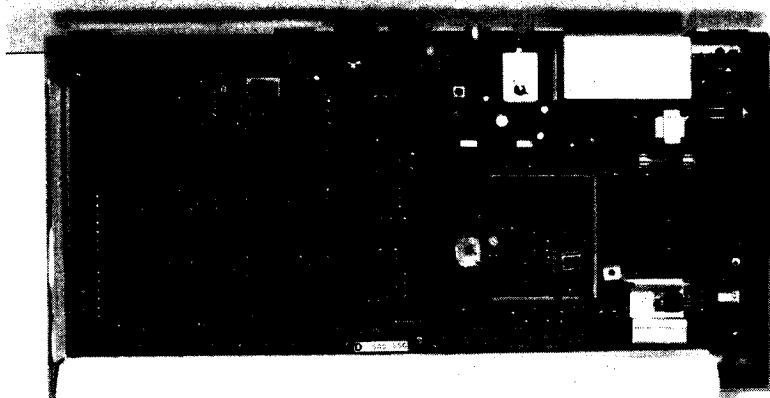
FF80 43

9 DOCUMENTATIE BIJ HET SCHEMA VAN DE COMMODORE 64

9.1 Opmerking vooraf

In dit hoofdstuk wordt elementaire kennis van de digitaal- en computertechniek bekend verondersteld. U moet bijvoorbeeld een AND- van een OR-gate kunnen onderscheiden en hexadecimale getallen kunnen gebruiken. Heeft u die kennis, maar weet u niets van de hardware van microcomputers, laat u zich dan niet van de wijs brengen door het aantal lijnen en IC's in het schema. Voor iedereen geldt, dat een goed begrip van alle functies een grondige studie vereist.

Voordat u de Commodore 64 opent, trekt u de stekker uit het stopcontact en ontkoppelt de aangesloten apparatuur. Draai met een kruiskop schroevendraaier de schroeven aan de onderkant los en verwijder de onderste helft van het computerhuis. U zult dan ongeveer hetzelfde zien als op de foto. De lay-out van de printplaat is verschillende keren gewijzigd. Op de foto staat een computer uit 1984.



9.2 De stroomvoorziening

Het lichtnet voedt de computer via de trafo. Samen met een gelijkrichter is hij via een 7-polige DIN-stekker aangesloten op bus CN7.

De trafo produceert een wisselspanning van 9 Volt op de pins 6 en 7 van CN7. Via een tweede trafowikkeling levert de gelijkrichterschakeling een gestabiliseerde gelijkspanning van 5 Volt (pin 5 van CN7). De massa ligt aan de pins 1, 2 en 3. Om storingen van het lichtnet te vermijden worden de spanningen van de contacten gefilterd door de spoelen L2, L4 en de condensatoren C20, C21, C98, C99 en C100. SW1 is de tweepolige aan/uit schakelaar aan de rechterkant. Zekering F1 (1 Ampere) beveiligd de 9-Volts wisselspanning aan de contacten 11 en 12 van de userport. Deze gelijkgerichte en gefilterde spanning kunt u voor externe apparatuur gebruiken. De maximale belasting is 100 mA.

Is de zekering kapot, dan brandt de LED op uw 64 wel. Ook een aangesloten floppy wordt gereset, als u de computer aanzet. Maar het beeldscherm blijft leeg. Kijk eerst of de televisie op het goede kanaal staat en of de HF-kabel is aangesloten. Is de zekering echt doorgebrand, vervang hem dan door een van 1.25 Ampere. Als die ook de geest geeft, is er waarschijnlijk een defect.

De gelijkrichter na de zekering levert een gestabiliseerde spanning van 5 en 12 Volt en een ongestabiliseerde spanning van 9 Volt. De gelijkrichterschakeling bestaat uit de bruggelijkrichter CR4 en de dioden CN5 en CN6. De geïntegreerde spanningsregelaar VR2 stabiliseert de 9 Volt achter de bruggelijkrichter. Daarna bedraagt de spanning 5 Volt. De dioden CN5 en CN6 richten de wisselspanning gelijk. Het resultaat is een ongestabiliseerde spanning van ongeveer 16 Volt. Spanningsregelaar VR1 maakt daar een gestabiliseerde 12 Volt van.

De 5 Volt van de trafo is al gestabiliseerd door een spanningsregelaar in het trafohuis. De warmte, die daarbij vrijkomt, heeft dus geen invloed op de computer. Die wordt uit zichzelf al warm genoeg. De meeste IC's van de C64 worden met deze spanning gevoed. Ten behoeve van kleine projecten ligt ze aan pin 2 van userport CN2. De maximale belasting is 100 mA, ruim voldoende voor een paar IC's.

De 5 Volt, die binnenin de Commodore 64 wordt geproduceerd (CAN+5) voedt de videocontroller (VIC), de video-uitgangstrap en alle IC's

voor de produktie van de klokfrequentie. De VIC krijgt deze spanning direkt, de video-uitgangstrap via spoel L1 en de condensatoren C61, C63 en C64. De 5 Volt voor de klok loopt via L2, C65, C66 en C67. De datasette heeft geen netaansluiting. De computer levert de 5 Volt voor de electronica aan de contacten 2 en B van cassetteport-stekker CN3 en via de transistoren Q1, Q2 en Q3 de 6 Volt voor de motor (contacten 3 en C). Als de processor portbit 5 high maakt, wordt transistor Q2 geleidend. Zenerdiode CR2 is daardoor kort gesloten: transistor Q1 krijgt geen basisvoorspanning. Q1 en Q3 blokkeren en de motor stopt. Is het portbit low, dan is Q2 geblokkeerd. Aan de basis van Q1 ligt de zenerspanning van 7.5 Volt, die Q1 en Q3 stuurt. Aan de emitter van Q3 ligt een gereduceerde zenerspanning van ongeveer 6 Volt. De basis-emitterspanningen van de transistoren (ongeveer 1.5 Volt) zijn van de 7.5 Volt afgetrokken. Het resultaat van deze stabilisatie van de motorschakeltrap is een constant toerental.

De 12 Volt is voor de VIC, de SID (sound interface device) en de audiouitgangstrap met transistor Q8.

De kleine schakeling bij gate U27 maakt niet direkt deel uit van de stroomvoorziening, maar krijgt haar signalen wel via het lichtnet. Het gaat om een AND-verbinding. Aan ingangspin 13 ligt 5 Volt, aan ingangspin 12 de 9 Volt wisselspanning via weerstand R5. Deze spanning zou dus veranderen met een frequentie van 50 Hz. Een TTL-ingang is slecht bestand tegen 9 Volt en een negatieve spanning van -9 Volt zou het IC zeker kapot maken. Zenerdiode CR1 beperkt deingangsspanning tot +2.7 Volt (logisch high-sigitaal) en -0.7 Volt (logisch low-sigitaal). De spanning aan pin 12 van U27 wisselt dus tussen low en high met het ritme van het lichtnet. Dezelfde frequentie geldt voor de uitgang.

Weerstand R37 versnelt de attack- en decaytijden om keurige blokpulsen voor verder gebruik te kunnen geven. Deze 50 Hz pulsen gaan naar de IC's U1 en U2, de twee CIA's, waarin realtimeclocks zijn ondergebracht. De 50 Hz van het lichtnet is als constante frequentie het gemakkelijkst te verkrijgen en daarom uiterst geschikt voor tijdmetingen.

9.3 De produktie van de klokfrequentie

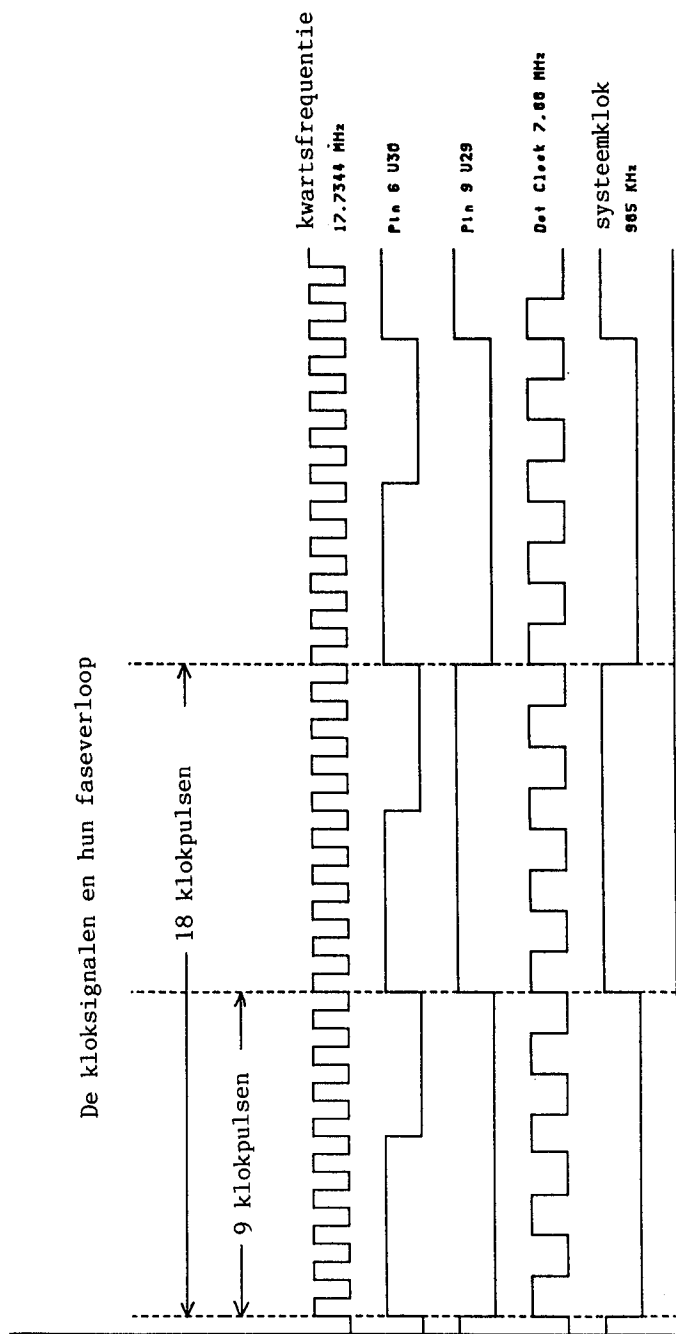
Constance, stabiele kloksignalen zijn voor het functioneren van een computer net zo belangrijk als een stabiele, storingsvrije

stroomvoorziening. De IC's voor de produktie van de klokfrequentie zitten samen met de VIC in het blikken kastje midden op de printplaat, niet te verwarren met het kastje voor de aansluiting van de televisie (UHF-modulator). Dit metalen kastje schermt de hoogfrequente stoorstraling af, die bij de produktie van de klokfrequentie ontstaat. Een onvoldoende afgeschermd computer stoort op radio en televisie. Het zou praktisch onmogelijk zijn de televisie als beeldscherm te gebruiken. De volgende gegevens zijn van toepassing op voor de Europese markt geproduceerde apparaten met PAL-uitgang.

Kwarts kristal Y1 (17,734472 MHz) produceert de klokfrequentie. Via C70 is hij aangesloten op IC U31. Dit TTL-IC 74LS629 bevat twee onafhankelijke, spanningsgestuurde oscillatoren (VCO's). Door de gelijkspanning aan de sturingang kan de frequentie binnen bepaalde grenzen worden veranderd. Pin 1 is de sturingang van VCO1. Potentiometer R27 laat een kleine verandering in de uitgangsfrequentie toe, zodat de gewenste frequentie precies kan worden ingesteld: ook kwarts kristallen hebben een bepaalde tolerantie. Pin 10 is de uitgang van VCO1. De frequentie aan deze pin gaat als signaal OCOLOR direkt naar de VIC. Tegelijkertijd komt het signaal aan bij IC U30 (74LS193), dat als frequentiedeler is geschakeld. Deze deler heeft een instelbare delerverhouding. Afhankelijk van het signaal aan pin 1, 9, 10 en 15 kan elke delerverhouding tussen 1:1 en 15:1 worden ingesteld. In dit geval is de verhouding 9:1, dus 17,734 MHz : 9. Aan pin 6 ligt daarom 1.9704 MHz. Deze frequentie gaat naar pin 11 van IC U29, dat twee flip-flops bevat. Bij elke positieve flank van het kloksignaal (pin 11) wordt de informatie aan de ingang van de flip-flop (pin 12) op uitgang Q gezet (pin 9). De uitgang -Q (pin 8) heeft dezelfde informatie als de ingang, maar dan geïnverteerd. De frequentie van het kwarts kristal gedeeld door 9 levert het kloksignaal voor FF1. De ingang is verbonden met de uitgang -Q. Als die laatste high is, wordt bij de volgende positieve flank aan pin 11 het high-signaal op de Q-uitgang gezet. Tegelijkertijd wordt de uitgang -Q low. Bij de volgende positieve flank wordt low aan Q gelegd, -Q is nu weer high, enzovoort. In de volgende tekening staan frequenties en faseverloop van dit proces schematisch weergegeven.

Bij elke tweede klokpuls verandert de toestand van de uitgangen, overeenkomstig een frequentiedeling door factor 2. Aan de uitgang verschijnt 985,248 MHz, de klokfrequentie van de processor. Dit signaal wordt niet direkt als klok gebruikt; de zaak ligt iets gecompliceerder. Het signaal dot clock (7,88198 MHz) is door frequentiedeling niet uit de kwartsfrequentie af te leiden. Een PLL-schakeling (phase locked loop: fasegestuurde lus) is nodig voor een frequentiesynthese. In de Commodore 64 komt de PLL tot stand door de IC's U31, U32 en de VIC.

De belangrijkste component van een PLL is een fasencomparator met twee ingangen. Deze levert aan zijn uitgang een gelijkspanning, evenredig aan het faseverloop van beide signalen. IC U32 en transistor Q7 dragen deze functie. Aan de ingang van U32 (pin 1) ligt een frequentie van 985 KHz, geleverd door de uitgang van flipflop U29. Aan de tweede ingang van de PLL (pin 3) ligt het signaal 0o, het van de VIC afkomstige kloksignaal voor de processor met een nog onbepaalde frequentie. Dit signaal is het door 8 gedeelde uitgangssignaal van VC02 in U32. Deze frequentiedeling vindt plaats in de VIC. De condensator C86, niet een kwartskristal, bepaalt de frequentie van VC02. De uitgang van fasencomparator U32 produceert de stuurspanning van VC02. Als die ongeveer 3 Volt bedraagt, is de frequentie van VC02 7,88198 MHz.



Aangenomen, dat de frequentie van flipflop U29 hoger is dan frequentie Oo (VCO2 trilt bijvoorbeeld met slechts 7.7 MHz), dan levert de uitgang van de fasencomparator (pin 8) een spanning kleiner dan 3 Volt. De VCO trilt daardoor met een hogere frequentie. De frequentie aan pin 3 van de fasencomparator wordt dan ook hoger en benadert de referentiefrequentie aan pin 1. De stuurspanning komt in de buurt van 3 Volt. Zijn de frequenties aan pin 1 en 3 gelijk, dan wordt de VCO geregeld totdat niet alleen de frequentie, maar ook de spanning van de signalen gelijk is. hetzelfde proces vindt plaats, wanneer de VCO met een te hoge frequentie trilt. De stuurspanning wordt dan groter dan 3 Volt. De VCO trilt langzamer en de stuurspanning neemt af totdat frequenties en fasen van de signalen gelijk zijn. Dan is het signaal dot clock goed. Deze regelprocessen duren kort. Na 100 msec. zijn alle frequenties weer vrij. Tot slot de functie van de FF2 en de gang van zaken in een 64 met NTSC-kleuruitgang. In deze voor de Amerikaanse markt geproduceerde apparaten zit een kwartskristal van 14,31818 MHz. De VIC-chip van de NTSC-versie is een 6567, bij de PAL-versie is dit een 6569. Ten derde is de draadbrug tussen E1 en E2 of E3 anders gelegd. Bij PAL-apparatuur is die tussen E1 en E2 geschakeld en liggen pin 1 en 10 van deler U30 dus aan +5 Volt. Ook ligt pin 4 van IC U29 aan high. Deze pin is de presetingang van FF2. Klok-, data- en clearingang van deze FF zijn verbonden met massa. Een low-sigitaal aan de clearingang brengt de flipflop in een gedefinieerde toestand. Onafhankelijk van de andere ingangssignalen wordt uitgang Q low, -Q high. Om dit alles te bereiken moet de presetingang high zijn. Bij de PAL-versie is aan deze voorwaarde via de draadbrug voldaan. De ingangen 1, 9, 10 en 15 van teller U30 bepalen in binaire code de beginwaarde van de teller. Omdat de teller tot 16 telt, kan met deze waarde de delerverhouding worden ingesteld. De waarde begint dan bij het geprogrammeerde getal. Ingang A is het LSB, ingang D het MSB. Aan deze ingangen ligt in decimale termen een 7. De teller gaat tot 16 en begint dan weer bij 7. Daarvoor zijn 9 getalimpulsen nodig, de teller deelt dus door 9. De FF2 is niets anders dan een inverter: is de uitgang high, dan is de ingang low en omgekeerd. Bij de NTSC-versie is de presetingang van U29 low. Volgens de datasheet zijn de uitgangen Q en -Q dan high. Dat is ongewoon, maar het IC blijft intact. De delerverhouding van U30 bedraagt 7:1, met de flipflop, die volgt 14:1. De klokfrequentie van de processor is dan 1,0227 MHz, iets hoger dan de werkfrequentie van een PAL-apparaat.

9.4 De processor

Het grootste verschil tussen de 6510 en de 6502 is de poort, die in de chip is opgenomen. Deze poort heeft 6 programmeerbare I/O-lijnen. Dat is voor een 8-bits processor een beetje vreemd, maar er waren geen lijnen meer vrij om een volledige 8-bits poort te realiseren. De configuratie van de 40 pins van de 6510 wordt in paragraaf 1.3 uitvoerig behandeld. Ook de 6510 heeft een databus van 8 en een adresbus van 16 bits. De processor kan dus een geheugenbereik van 64K direct adresseren.

De signalen OIN en O2 zijn de kloksignalen van het systeem. De VIC produceert het signaal OIN (ongeveer 985 KHZ), de basis voor signaal O2, dat de processor levert. Dit signaal is als referentiefrequentie voor alle operaties van de processor uiterst belangrijk voor het samenspel tussen processor en periferie. Met het signaal -RES worden de processor en andere IC's in een gedefinieerde toestand gebracht. De reset vindt plaats als de computer wordt aangezet. Het IC U20, een NE555 met twee identieke timers, levert het -RES-signaal. Door een eenvoudige externe schakeling kan men met deze timers het IC als oscillator of pulsgever schakelen. Dat laatste is hier het geval. De bedrijfsspanning laadt condensator C105 op via weerstand R50 en C24 via R34. De eigenlijke puls wordt gestart als na ongeveer 10 msec. de spanning aan C105 boven 1.6 Volt komt (1/3 van de bedrijfsspanning). Via pin 13 wordt condensator C24 in een keer ontladen. De uitgang van de timer (pin 9) wordt tegelijkertijd aan 5 Volt gelegd. Via R34 wordt C24 vervolgens weer opgeladen. Ingangspin 12 houdt nu de spanning in de gaten. Als de spanning groter wordt dan 3.3 Volt (2/3 van de bedrijfsspanning), wordt de uitgang weer low. Die toestand treedt na ongeveer 0.5 sec. in. De inverter aan de uitgang van de timer (pin 9) verandert die positieve puls in een negatieve, het eigenlijke -RES-signaal. Op het moment dat de wisseling van high naar low plaatsvindt, begint de processor te werken. Als eerste haalt hij op de adressen \$FFFC en \$FFFFD (reset-vectoren) het adres van de volgende instructie. Op dat adres begint het eigenlijke bedrijfssysteem.

Pin R/-W geeft aan of de processor een lees- of schrijfactie onderneemt. Is deze lijn high, dan leest de processor gegevens uit RAM, ROM of interfacechips. Bij low (schrijfactie) slaat hij gegevens op in een geadresseerde bouwsteen. In de ROM kan niet geschreven worden, omdat die gegevens al bij de fabricage onveranderlijk zijn vastgelegd. Pin -NMI (non maskable interrupt) maakt de onderbreking van een lopend programma mogelijk. Niet maskeerbaar wil zeggen, dat de interrupt altijd is toegestaan.

Door software kan de interrupt niet ongedaan worden gemaakt. Wordt de pin aan massa gelegd, dan verlaat men het lopende programma door beëindiging van de laatste machinetaalinstructie. De processor haalt op \$FFFA en \$FFFB (NMI=vectoren) het adres van de NMI-vector en vertakt daar. -NMI kan drie oorzaken hebben.

(1) Als RESTORE wordt ingedrukt, produceert de tweede timer van U20 een geschikte puls. Het indrukken ontladent condensator C38 in een keer. Via weerstand R35 wordt U20 weer opgeladen, ook al is RESTORE nog ingedrukt. Is de spanning aan pin 6 van U20 groter dan 1.6 Volt, dan wordt de NMI-puls gestart. De uitgang van de timer (pin 5) wordt high en de uitgang van inverter U6 low. C23 wordt ontladen via pin 1 van U20 en daarna opgeladen via R33. Na ongeveer 18 microseconden is C23 op 2/3 van de bedrijfsspanning. Uitgangspin 5 wordt weer low, de -NMI-ingang van de processor is weer high.

(2) CIA U2 zorgt voor de interrupt. In bepaalde gevallen kan pin 21 van dit IC low worden. Meer hierover in paragraaf 9.9.

(3) Het kortsluiten van pin D van de cartridge expansion. Externe bouwstenen kunnen dan een interrupt veroorzaken.

-IRQ (interrupt request) is aan -NMI verwant. De interruptvectoren van -IRQ liggen op \$FFFE en \$FFFF. In tegenstelling tot -NMI kan deze interrupt door software worden uitgeschakeld. Als in het statusregister van de processor de I-flag (bit 2) is gezet, worden alle interrupts genegeerd. -IRQ is niet flankgestuurd en moet dus wachten totdat de processor de pin afcheckt. Deze interrupt kan drie oorzaken hebben. (1) Als bepaalde programmeerbare toestanden worden bereikt, wordt pin 21 van CIA U1 low, wat bij de processor een -IRQ tot gevolg heeft. (2) De VIC veroorzaakt de interrupt doordat pin 8 low wordt als gevolg van een geprogrammeerd proces. (3) Pin 4 van de cartridge expansion plug (CN6) wordt kortgesloten. Externe schakelingen kunnen dan voor een -IRQ zorgen.

Pin RDY informeert de processor over de geldigheid van de informatie op de databus. Is de pin laag, dan kan de processor de gegevens nog niet overnemen. Hij gaat dan in een wachtcyclus en controleert bij elke klokpuls of de RDY-pin al hoog is. In oudere processorsystemen werd deze mogelijkheid gebruikt om langzame geheugen- en perifere bouwstenen op de processor aan te sluiten. Gewoonlijk spreekt de VIC de RAM alleen aan, als de processor niet actief is (O2=low). Voor bepaalde operaties heeft de VIC meer tijd nodig, bijvoorbeeld de afbeelding van sprites. De VIC maakt pin BA (bus available) dan low. Dat signaal wordt via AND-gate U27 op de RDY-ingang van de processor gezet, waarna deze de bus zolang als nodig vrijgeeft voor de VIC.

De VIC levert ook het signaal AEC. Als de VIC de bus gebruikt, wordt deze pin 0. Dit low-signaal wordt aan pin AEC van de processor gelegd, waardoor deze zijn lijnen hoogohmig maakt (tri-state). Het lijkt alsof de processor helemaal niet in een IC-voetje zit. Zolang AEC low is blijft deze toestand bestaan en kunnen andere IC'S, bijvoorbeeld een externe processor of de VIC, de systeembus gebruiken.

De in de processor geïntegreerde poort ligt aan de pins 24-29. Portbit 0 (-LORAM) schakelt in het adresbereik \$A000-\$BFFF tussen RAM (low) en ROM. Portbit 1 (-HIRAM) doet hetzelfde bij \$E000-\$FFFF. Portbit 2 (-CHAREN) selecteert bij een low-signaal de karakter-ROM. Karakter-ROM en I/O-bereik gebruiken hetzelfde adresbereik (\$D000-\$DFFF). Via -CHAREN wordt bepaald of de karakter-ROM geselecteerd is, het I/O-bereik of de VIC, SID of CIA's.

De drie resterende bits zijn gereserveerd voor de datasette. Portbit 3 levert de schrijfgegevens voor de datasette. Deze processorpin is verbonden met pin E en 5 van de cassettepoort. Portbit 4 (cass sense) controleert of de PLAY-toets van de datasette is ingedrukt. Dit bit ligt aan pin F en 6 van de cassettepoort. Bit 5 stuurt de motor van de recorder (zie paragraaf 9.2).

9.5 Decodering van adressen

De 6510 kan een adresbereik van 64K adresseren. Maar omdat de RAM al die omvang heeft, zorgt adress-manager U17 (een FPLA-IC (field programmable logic array)) voor het beheer van de geheugenbereiken, die elkaar gedeeltelijk overlappen. Zijn speciale logische eigenschappen dankt het aan zijn programmering. Dit IC vervangt een groot aantal verschillende gates. De pinconfiguratie is als volgt:

pin naam functie

1	FE	niet gebruikt
2	I7	ingang, A13: pin 20 van de 6510
3	I6	ingang, A14: pin 22 van de 6510
4	I5	ingang, A15: pin 23 van de 6510
5	I4	ingang, -VA14: bit 0, pin 2 van CIA 2 poort A
6	I3	ingang, -CHAREN: bit 2, pin 27 van de 6510-poort

7	I2	ingang, -HIRAM: bit 1, pin 28 van de 6510-poort
8	I1	ingang, -LORAM: bit 0, pin 29 van de 6510-poort
9	I0	ingang, -CAS: pin 19 van de VIC
10	F7	uitgang, -ROMH: pin B naar het expansieslot
11	F6	uitgang, -ROML: pin 11 naar het expansieslot
12	F5	uitgang, -I/O: pin 1 naar decoder U15
13	F4	uitgang, GR/-W: pin 10 naar kleur-RAM U6
14	GND	massa van de bedrijfsspanning
15	F3	uitgang, -CHAROM: pin 20 naar karakter-ROM U5
16	F2	uitgang, -KERNAL: pin 20 naar KERNAL-ROM U4
17	F1	uitgang, -BASIC: pin 20 naar BASIC-ROM U3
18	F0	uitgang, -CASRAM: pin 15 naar de RAM's
19	-OE	ingang, output enable aan massa
20	I15	ingang, -VA12: pin 28 van de VIC
21	I14	ingang, -VA13: pin 29 van de VIC
22	I13	ingang, -GAME: pin 8 van het expansieslot
23	I12	ingang, -EXROM: pin 9 van het expansieslot
24	I11	ingang, R/-W: pin 38 van de 6510
25	I10	ingang, -AEC: pin 16 van de VIC
26	I9	ingang, BA: pin 12 van de VIC
27	I8	ingang, A12: pin 19 van de 6510
28	VCC	bedrijfsspanning +5 Volt

De vraag is nu wat het resultaat is van de ingangssignalen op de uitgangen van de AM. Bij 16 ingangslijnen zijn er 65536 mogelijke combinaties. Uit het feit, dat de AM 8 uitgangen heeft, volgt dat verschillende ingangscombinaties dezelfde uitgangscombinatie hebben. Maar een aantal van de 256 uitgangscombinaties zijn werkelijk zinvol voor de computer. Een overzicht hiervan vormen de geheugenkaarten aan het einde van paragraaf 1.4.

9.6 De videocontroller 6569

De eenheden voor invoer en uitvoer zijn de belangrijkste perifere apparaten van een computer. De uitvoer van de Commodore 64 komt meestal op het beeldscherm, dat op signalen van de VIC reageert, te weten sync- en helderheidsimpulsen en kleurwaarden. Daarnaast zorgt de VIC voor de frequentie van de CPU, de refresh van de dynamische RAM's en de stuursignalen voor het gebruik daarvan. De pinconfiguratie van de VIC is als volgt:

pin	naam	functie
1-7	D6-DO	databus van processor
8	-IRQ	uitgang, interrupt request
9	-LP	ingang, light pen
10	-CS	ingang, chip select
11	R/-W	read/-write
12	BA	bus available
13	VDD	bedrijfsspanning +12 Volt
14	COLOR	uitgang, kleurinformatie
15	SYNC	uitgang, regel- en beeldsynchronisatie-impulsen
16	AEC	uitgang, adress enable control
17	OOUT	uitgang, systeemklok
18	-RAS	uitgang, row adress select
19	-CAS	uitgang, column adress select
20	GND	massa van bedrijfsspanning
21	OCOLOR	ingang, kleurfrequentie
22	OIN	ingang, dotfrequentie
23	A11	adresbus van processor
24-29	AO/A8-A5/A13	gemultiplexte (video)RAM-adresbus
30	A6	(video-)RAM-adresbus
31	A7	(video-)RAM-adresbus
32-34	A8-A10	adresbus van processor
35-38	D11-D8	databus van kleur-RAM
39	D7	databus van processor
40	VCC	bedrijfsspanning +5 Volt

In vergelijking met de processor zijn de signalen -CS, -RAS, -CAS en de lijnen D8-D11 nieuw. Dat geldt ook voor de gemultiplexte adresbus, omdat bij de processor alle adressignalen aan afzonderlijke pins liggen.

De dot clock (7,85 MHz) bepaalt het hele tijdsverloop in de computer. Een bepaalde trap in de VIC deelt die frequentie door 8 (=980 KHz). De nieuwe frequentie ligt aan pin 17 als systeemklok OOUT. De dot clock levert ook de signalen voor de synchronisatie van het beeld op het scherm. De klok bepaalt zelf wanneer de punten van de tekens op het scherm verschijnen. De frequentie van OCOLOR (17,734472 MHz: kwarts kristal Y1) is nodig voor de informatie over de kleuren. De genoemde frequenties zijn van toepassing op een computer, die geschikt is voor een televisie met een PAL-sigitaal.

Als de processor de registers van de VIC wil aanspreken, moet die worden geadresseerd. Pas als lijn -CS low is, kan de processor via het adres op de adresbus het gewenste register aanspreken. De VIC gebruikt in het I/O-bereik (\$D000-\$DFFF) de adressen \$D000-\$D3FF. Daarom geeft de AM bij een actie op dit bereik een low-sigitaal aan pin 12 (-I/O), dat naar pin 1 van decoder U15

wordt gestuurd. Die wordt dan vrijgegeven en afhankelijk van de adreslijnen A10 en A11 (pin 2 en 3) wordt de uitgang van de decoder low. De binaire weergave van het begin- en eindadres van de VIC levert het volgende bitpatroon op:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0=\$D000
1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1=\$D3FF

De adresbits A10 en A11 blijven low en dus is ook uitgang Y0 van de decoder low. De VIC geadresseerd. Pas bij het volgende adres (\$D400) wordt A10 high en Y0 ook, Y1 wordt low. De SID is geadresseerd.

Omdat de VIC alleen de adresbits A0-A13 heeft, kan hij maar een adresbereik van 16K adresseren. Bovendien liggen de adreslijnen niet als bij de processor afzonderlijk aan de pins, maar zijn gemultiplext. Pin 24 is bijvoorbeeld zowel adresbit 0 als 8. Eerst is de pin het ene, dan het andere adresbit. Met de hulpsignalen -CAS en -RAS kan worden uitgemaakt om welk bit het gaat. Ze dienen ook voor de sturing van de dynamische RAM's, die ook een gemultiplexte adresbus hebben. Het aanspreken van het geheugen verloopt als volgt. -CAS en -RAS zijn high. Eerst wordt het LSB op de bus gezet: -RAS wordt low. De adresbyte wordt daardoor in de RAM's opgeslagen. Dan verandert de informatie op de bus. A0 wordt A8, A1 A9 enzovoort: -CAS wordt low. Deze dalende flank wordt aan AM gegeven en resulteert in een iets vertraagde dalende flank aan de uitgang -CASRAM. Daarmee wordt het MSB in de RAM's opgeslagen. Nu hebben we het volledige adres en verschijnen de gegevens op de databus. Het hele proces is in het timing-diagram op de volgende bladzijde nog eens schematisch weergegeven.

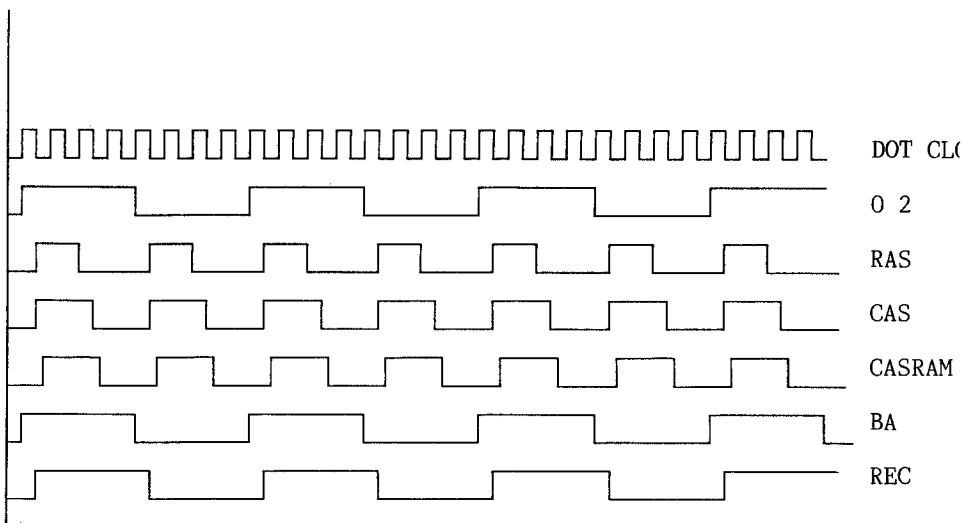
9.6.1 De aansluiting tussen RAM en VIC

Omdat de VIC alleen de adresbits A0-A13 heeft, levert poort A van CIA 2 de bits, die voor de adressering van de 64K RAM ontbreken. De portbits 0 en 1 zijn de adresbits 14 en 15. Om hun functie in het multiplex-proces te kunnen vervullen zijn ze geschakeld via IC U14, dat vier 2:1 multiplexers bevat met een inverterende werking. Een multiplexer is een soort wisselschakelaar, die een van de twee ingangen aan een uitgang legt. Het signaal aan ingang

S schakelt de multiplexers. Is S low, dan zijn de A-ingangen met de uitgang verbonden en als S high is, de B-ingangen. De adresbits A6 en A7 van de VIC liggen aan de multiplexer: A6 aan de ingangen 13 en 14, A7 aan de ingangen 10 en 11. Als met het S-sigitaal tussen de ingangen heen en weer geschakeld wordt, verandert er aan de uitgangen niets, omdat de adresbits met de beide ingangen zijn verbonden. De inverse werking van de multiplexer verwisselt alleen de polariteit van de signalen aan de uitgangen. Die geïnverteerde adressignalen worden aan de B-ingangen van de beide andere multiplexers gelegd: -A6 aan pin 3 en -A7 aan pin 6. De A-ingangen zijn verbonden met de portbits van CIA 2: bit 0 (-VA14) met pin 2, bit 1 (-VA15) met pin 5. Omdat -CAS de S-ingang stuurt, ligt de nog een keer geïnverteerde adresbit A6 aan uitgang pin 4, als -CAS high is. Is -CAS low, dan ligt A14 aan de uitgang. Aan uitgang pin 7 wordt tussen -A7 en -VA15 geschakeld. Door de inverse werking van de multiplexer is dit signaal A7 of A15.

Pin 15 van U14 is als -OE (output enable) met AEC verbonden. Is AEC high, dan worden de uitgangen van U14 uitgeschakeld of in de tri state gebracht. Bij AEC high zet de processor zijn adressen op de bus via multiplexer U13 en U25. Alleen als AEC low is, zijn de uitgangen van U14 vrij en kan de VIC de bus gebruiken.

fasenverloop van de stuursignalen van de bus



9.6.2 Het kleur-RAM: 16 kleuren met 4 bits

Met de bits 35-38 van de VIC kunnen alle 512 tekens in 16 verschillende kleuren worden afgebeeld. Aan deze pins liggen de datalijnen van het kleur-RAM U6, een statische RAM met 4096 geheugenplaatsen. Op elke plaats past een bit en een adres spreekt vier plaatsen aan. Als -CS van U16 low is, wordt het RAM geselecteerd en verlaten de datalijnen de tri state. AND-gate U27 produceert het signaal -CS op twee manieren. In de schakeling functioneert U27 als OR-gate. Een AND-gate maakt de uitgang high, als alle ingangen high zijn. Daaruit volgt, dat de uitgang low is, als de ene of (OR) de andere ingang low is. De Commodore 64 maakt van deze laatste optie gebruik. Het kleur-RAM bezet de adressen \$D000-\$DBFF. Is AEC high, dan gebruikt de processor de bus. Daardoor is de ene ingang van de AND-gate high. Als de processor het kleur-RAM niet aanspreekt, is uitgang -COLOR van decoder U15 ook high, evenals ingang -CS van het kleur-RAM: het kleur-RAM is niet geselecteerd. Als de processor het kleur-RAM wil aanspreken, zet hij het geheugenadres op de databus. De decoding verloopt net als bij de VIC, alleen is bit All met zekerheid geset. Uitgang -COLOR van decoder U15 wordt daardoor low. Een ingang van de AND-gate is dan low en dus de uitgang ook: het kleur-RAM is geselecteerd. AEC is op dit moment high, waardoor de vier analoogschakelaars in U16 gesloten zijn. De datalijnen van het kleur-RAM zijn verbonden met de vier minst significante lijnen van de processor: het kleur-RAM is vrij voor lees- en schrijfacties. Wordt AEC low, dan neemt de VIC de bus over en worden de analoogschakelaars geopend. Tegelijkertijd wordt de uitgang (pin 8) van AND-gate U27 low: de VIC heeft het kleur-RAM geselecteerd. Omdat de VIC alleen via de lijnen A8-A11 met de adresbus is verbonden, levert het TTL-IC U26 (74LS373) de adresbits A0-A7. Dit IC bevat 8 tussengeheugens (latches). De ingangen zijn met de gemultiplexte adresbus verbonden. De gegevens worden opgeslagen als signaal -RAS low wordt. De LSB van het adres ligt dan op de bus. De uitgangen van U16 zijn verbonden met de minst significante adresbytes van de processorbus, die de adresinformaties leveren als de processor zich in de tri state bevindt. Op die manier kan de VIC het kleur-RAM adresseren. Ook het tussengeheugen is met signaal AEC verbonden. Via pin 1 van U16 zorgt AEC high ervoor, dat de uitgangen hoogohmig worden, zodat de processor niet wordt gestoord. Behalve de gemultiplexte adresbus (A0-A13) heeft de VIC vier adreslijnen (pin 23, 32, 33, 34) nodig om tegelijk met een adres van het beeldschermgeheugen (\$0400-\$07FF) de plaats in het kleurgeheugen (\$D800-\$DBFF) aan te kunnen spreken. De vier adresbits vormen dus de noodzakelijke tweede bus, die onafhankelijk van de normale adresbus functioneert.

9.7 De karaktergenerator

Wis het scherm door SHIFT en CLR/HOME tegelijk in te drukken en toets een A in. Het puntraster ziet er zo uit:

12345678

```
A  ...**...
B  .*****.
C  .**..**
D  .*****.
E  .**..**
F  .**..**
G  .**..**
H  .......
```

Elke rij bestaat uit acht bits, een byte. * staat voor een high-, . voor een low-signaal. Acht rijen leveren dus acht bytes op, die in het karakter-ROM zijn opgeslagen. Naar die bytes verwijst de code in het beeldschermgeheugen. Ze is dus het basisadres van het teken en niet het teken zelf. Om de acht bytes te kunnen adresseren zijn drie extra lijnen nodig, A8-A10 van de adresbus. De vrije lijn A11 heeft een bijzondere functie. het beeldschermgeheugen kan een 8-bits byte opslaan. Daardoor kunnen 256 verschillende tekens in het karakter-ROM worden geadresseerd. Maar de 64 kan 512 tekens afbeelden. De computer beschikt over twee stel mogelijke tekens. Het eerste stel bestaat uit 128 hoofdletters en grafische tekens. Ze kunnen ook reverse worden afgebeeld. Daarmee komen we op 256 mogelijke tekens. Het tweede stel schakelt u in door C= en SHIFT tegelijk in te drukken. Het zijn kleine letters en hoofdletters, die ook reverse kunnen worden afgebeeld. In totaal zijn er dus 512 verschillende tekens. De vrije adresbit A11 regelt het omschakelen van het ene stel tekens naar het andere.

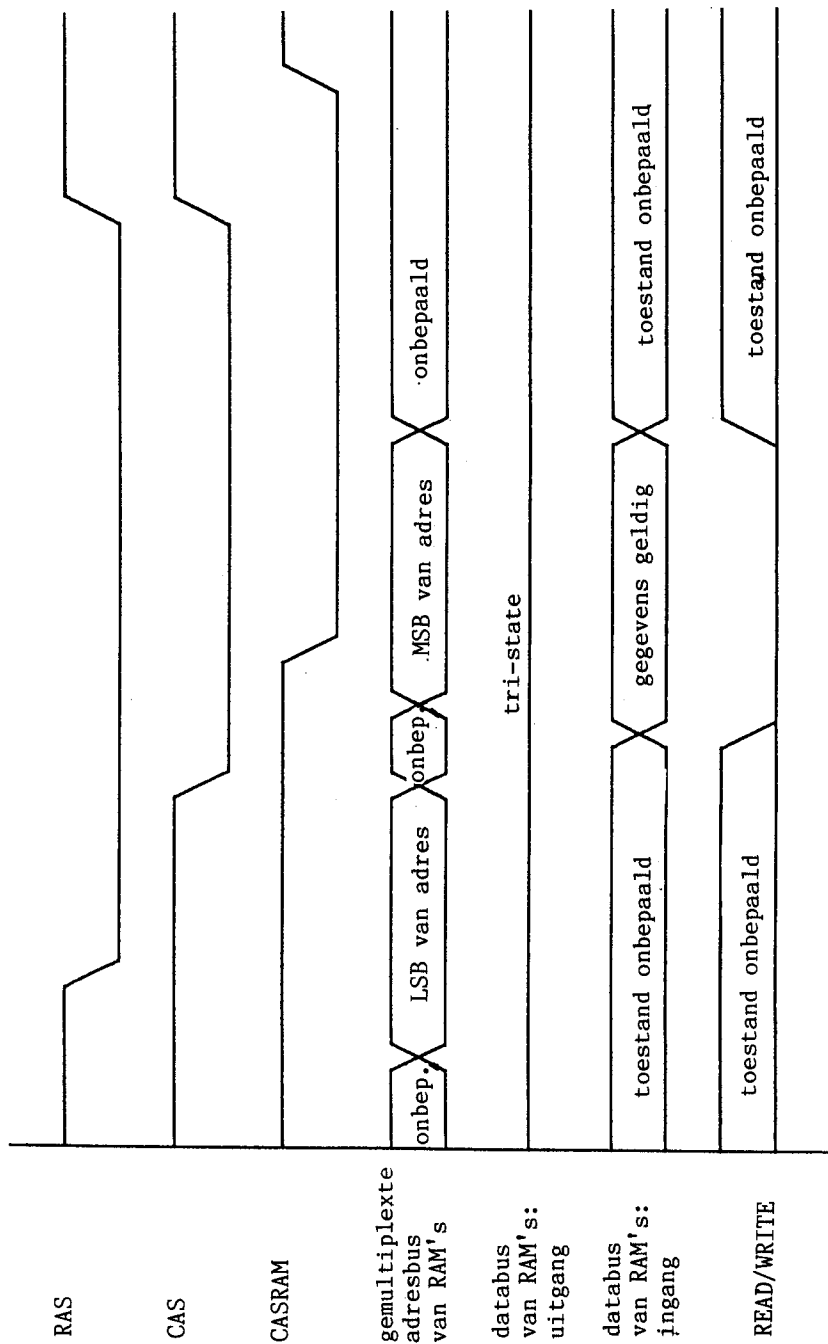
9.7.1 De processor en het RAM

De VIC en de processor lezen, als signaal R/-W high is. Het RAM heeft een gemultiplexte adresbus nodig. Daar zorgen twee multiplexers voor (U13 en U25 (74LS257)), omdat de processor dat

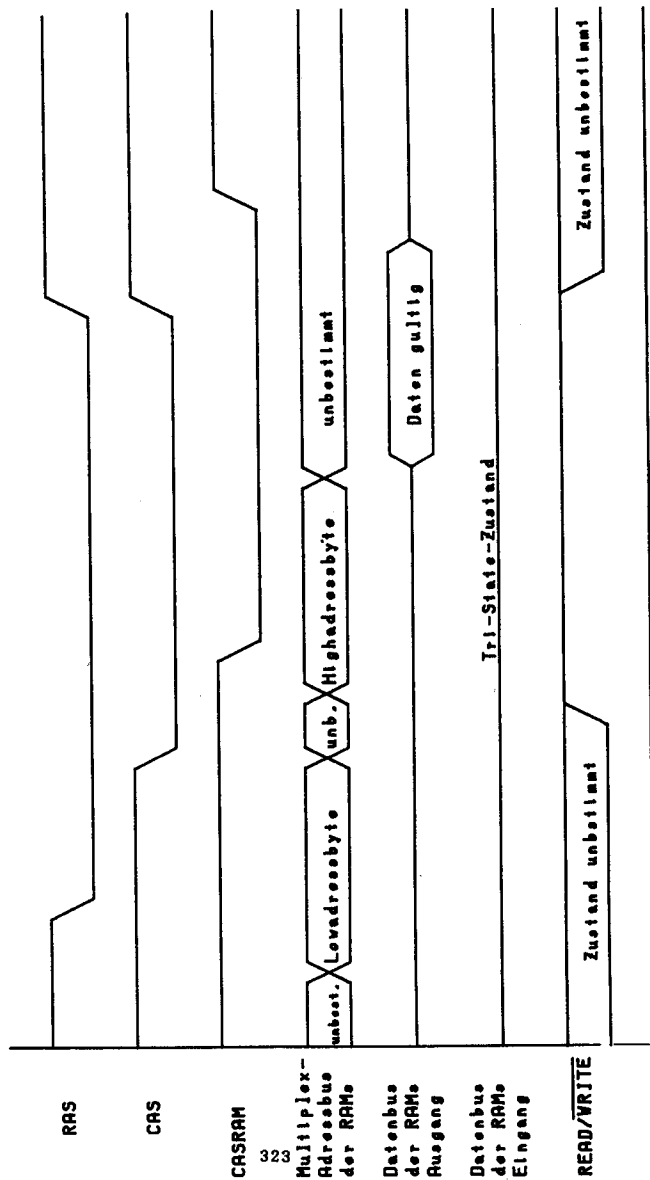
niet kan. Ze werken als U14 (paragraaf 9.6.1), maar inverteren de uitgangssignalen niet. Aan de ingangen van deze IC's ligt de adresbus van de processor (A0-A15). SELECT zorgt ervoor, dat er van A0 naar A8, A1 naar A9 enzovoort wordt omgeschakeld. De adressering van de RAM's verloopt als volgt. De SELECT-ingang van de multiplexer is high, waardoor het LSB van het adres op de RAM's is geschakeld. Bij een dalende flank van het -RAS-sigitaal wordt dit byte in de RAM's overgenomen. Even later wordt -CAS low, waardoor de multiplexers omschakelen. De tweede ingang van de multiplexer wordt op de juiste uitgangen geschakeld en het MSB van het adres ligt aan de RAM's. Signaal -CAS wordt via de AM weer iets vertraagd. Uitgang -CASRAM neemt ook hier de functie van -CAS over. Bij een dalende flank aan -CASRAM wordt het MSB van het adres opgeslagen in de RAM's. Vervolgens wordt de geadresseerde geheugenplaats in de RAM's aangesproken en verschijnen de gegevens op de databus.

Bij een schrijfactie wordt processorpin R/-W low, nadat de processor het adres van de geheugenplaats op de adresbus heeft gezet. Daardoor weet het RAM, dat de byte op de databus in die geheugenplaats moet worden opgeslagen. de RAM's verlangen van R/-W, dat het pas low wordt, nadat -RAS low is geworden. Maar -CASRAM moet nog high zijn. R/-W moet dus low worden tussen de dalende flanken van -RAS en -CASRAM. Het tijdsverloop van de signalen -RAS, -CAS en -CASRAM is hetzelfde als bij de leesacties. Op de volgende bladzijden zijn beide acties schematisch weergegeven.

schrijfactie op het dynamische RAM



Lesezugriff auf das dyn. RAM



9.8 De SID 6581

De pinconfiguratie is behandeld in paragraaf 2.1.1. Op de eerste vier pins (CAP1A-CAP2B) zijn de condensatoren C10 en C11 aangesloten. Ze worden gebruikt voor de frequentie-filters, die in de SID (chip U18) zijn geïntegreerd. Een elektronisch frequentie-filter laat bepaalde frequenties door en andere niet. Er zijn in totaal vier soorten frequentie-filters mogelijk: LOWPASS-, HIGHPASS-, BANDPASS- en SPER-filter. Een LOWPASS-filter laat alleen bepaalde lage frequenties door. Men denke aan de bass-knop van de toonregeling op een versterker. Daarmee wordt de hoogste frequentie (grensfrequentie) ingesteld, die moet worden doorgelaten. Een HIGHPASS-filter doet het omgekeerde, denk aan treble. Ook de BANDPASS- en het SPER-filter doen precies het tegenovergestelde. Een BANDPASS-filter is een HIGH- en LOWPASS-filter tegelijk. Vanaf een bepaalde frequentie worden hogere frequenties doorgelaten tot een bepaald maximum. Nog hogere frequenties worden geblokkeerd. Een SPER-filter blokkeert alle frequenties uit een bepaald gebied. Een voorbeeld daarvan is het brom-filter in goede stereoapparatuur. Maar dat blokkeert alleen de frequentie van het lichtnet (50 Hz). Al deze-filters kunnen in de SID geprogrammeerd worden.

Pin 5, de RESET-ingang van U18, brengt het IC in een gedefinieerde toestand. Nadat de computer is aangezet, is het signaal aan deze pin 0.5 seconde low, waardoor alle registers in de 6581 worden gewist. Als dat niet gebeurde, zouden de waarden van de registers net zo toevallig zijn als het signaal aan de audio-uitgang.

De toonfrequenties van de SID worden door deling vervaardigd uit de frequentie van signaal O2. Zoals bij alle andere perifere bouwstenen is O2 ook de referentiefrequentie voor de schrijf- en leesacties van de processor. Zoals bekend bepaalt R/-W of er in de registers van de SID wordt gelezen of geschreven.

De 6581 gebruikt de adressen \$D400-\$D7FF. Net als bij de VIC decoderen AM en de decoder in U15 (74LS139) dit bereik. Als de processor een van deze adressen op de bus zet en -CHAREN high is, wordt uitgangspin 5 van de U15 low en daardoor ingang -CS van de SID ook. Via de adreslijnen A0-A5 worden de registers in de SID geadresseerd. Zijn deze adresbits low en is de SID met -CS geselecteerd, dan kan in register 0 worden geschreven of gelezen. Bij A0 high is register 1 geselecteerd enzovoort. Zo kunnen alle 29 registers worden aangesproken.

De datalijnen DO-D7 (pin 15-22) zijn verbonden met de databus van de processor. Als -CS high is, bevinden de datalijnen van de SID zich in de tri state. Bij -CS low bepaalt R/-W of de datalijnen als ingang (schrijfactie) of uitgang (leesactie) fungeren.

POTX en POTY zijn de ingangen van de A/D convertor. Een digitale waarde is high of low. In de Commodore 64 is +5 Volt high en 0 Volt low. Een analoog signaal ligt niet zo vast, maar kan elke waarde daartussen, -boven of -onder aannemen. Het is vaak wenselijk, dat een analoge waarde kan worden ingevoerd en verwerkt. Dat kan bij de Commodore 64. De A/D convertor wordt hoofdzakelijk bij de paddles gebruikt. Een paddle is een potentiometer, een variabele weerstand met een minimale weerstand van 100 Ohm en een maximale weerstand van ongeveer 500 KOhm. De A/D convertor verandert de analoge weerstandswaarde in een digitaal, in dit geval 8-bits, signaal. Deze byte kan uit een van de SID-registers worden gelezen. De verandering wordt uitgevoerd door de ingestelde weerstandswaarde en de condensatoren C48 en C93. Via de potentiometers worden de condensatoren voor 0.25 milliseconden opgeladen. Wordt de spanning aan de condensatoren groter dan de referentiespanning van de SID, dan stopt er in de SID een teller. De tellerstand is de maatstaf voor de ingestelde weerstand. Hoe groter de weerstand van de potentiometer, des te langzamer wordt de condensator opgeladen. De spanning aan de condensator bereikt de referentiespanning later, waardoor de teller langer kan lopen en de waarde groter wordt. Is de weerstandswaarde te groot (ongeveer 200 KOhm), dan bereikt de spanning aan de ingang van de convertor de referentiespanning niet binnen de meettijd (0.25 milliseconden). De teller loopt dan tot aan zijn eindwaarde. In het A/D-register staat 255. Als de weerstand te klein wordt (ongeveer 2000ohm), is de condensator zo snel opgeladen, dat de teller direkt stopt. In het register komt een 0 te staan. Na de meettijd worden de condensatoren via de A/D-ingang in een keer ontladen. De teller wordt op nul gezet en na 0.25 milliseconden begint een nieuwe meetcyclus. Een volledige cyclus duurt dus 0.5 milliseconden. De actuele weerstandswaarden worden per seconde 2000 maal gemeten. Ze staan even vaak ter beschikking. De weerstand mag niet kleiner zijn dan 100 Ohm. Anders kan de ontlaadtrap aan de ingang kapot gaan, omdat de spanningen bij het ontladen van de condensatoren te groot worden. De ingangen POTX en POTY liggen niet direkt aan een van de bussen van de C64, maar aan pin 2, 3, 9 en 10 van U28. Dit IC (CMOS 4066) heeft vier analoogschakelaars en is nodig om twee paar paddles te kunnen aansluiten. Een analoge switch werkt als een relais. Als er een spanning aan de sturingang ligt, wordt de analoge ingang op de uitgang doorgeschakeld: de schakelaar is dicht. Ligt de sturingang aan massa, dan is de verbinding tussen uit- en ingang geblokkeerd: de schakelaar is open. De analoge ingangen zijn verbonden met de controllports CN8 en CN9. Contact 5 en 9 zijn

zijn bestemd voor de aansluiting van de paddles. Pin 5, 6, 12 en 13 zijn sturingangen. Pin 13 controleert schakelaar 1 tussen de aansluitingen 1 en 2, pin 5 schakelaar 2 tussen 4 en 3, pin 6 3 tussen 8 en 9 en pin 12 4 tussen 11 en 10. Deze ingangen zijn paarsgewijs met elkaar verbonden: pin 13 met 5, pin 6 met 12. Ze liggen aan pin 8 en 9 van CIA U1. Via deze lijnen kan worden bepaald welke potentiometers aan de ingangen van de A/D-converter liggen. Bij een low-sigitaal aan pin 8 en 9 van CIA U1 is geen potentiometer op de converter aangesloten. Is pin 8 high, dan zijn de analoogschakelaars 3 en 4 gesloten. De paddles aan controllport 1 worden dan verbonden met POTX en POTY. Een high-sigitaal aan pin 9 betekent, dat de analoogschakelaars 1 en 2 gesloten zijn. De paddles aan CN8 liggen dan aan de ingangen van de converter.

Via de FN-uitgang AUDIO OUT kunt u beschikken over het geluid, dat de synthesizer produceert. Bij een maximaal volume is het audiosigitaal 2Vss. Transistor Q8 is als emittervolger op de uitgang geschakeld. Omdat het sigitaal aan de emitter via weerstand R38 wordt afgenomen, heeft Q8 geen spanningsversterking. Het sigitaal aan uitgangspin 3 van de 8-polige video-audio-bus CN5 is dus ook 2Vss. Op deze uitgang kan een kleine luidspreker van 8 Ohm worden aangesloten, maar het beste resultaat krijgt u, wanneer het sigitaal via stereoapparatuur hoorbaar wordt gemaakt. U kunt ook de televisie beeld en geluid laten verzorgen. Via EXT IN kunnen externe signalen aan de synthesizer worden geleverd, bijvoorbeeld het enigzins versterkte sigitaal van microfoon, gitaar of orgel, maar ook een tweede SID, een andere Commodore 64 dus. Denk erom, dat het ingangssigitaal niet groter mag zijn dan 3Vss. Via condensator C12 is de ingang met contact 5 verbonden van de 8-polige video-audio-bus CN5.

9.9 De CIA's 6526

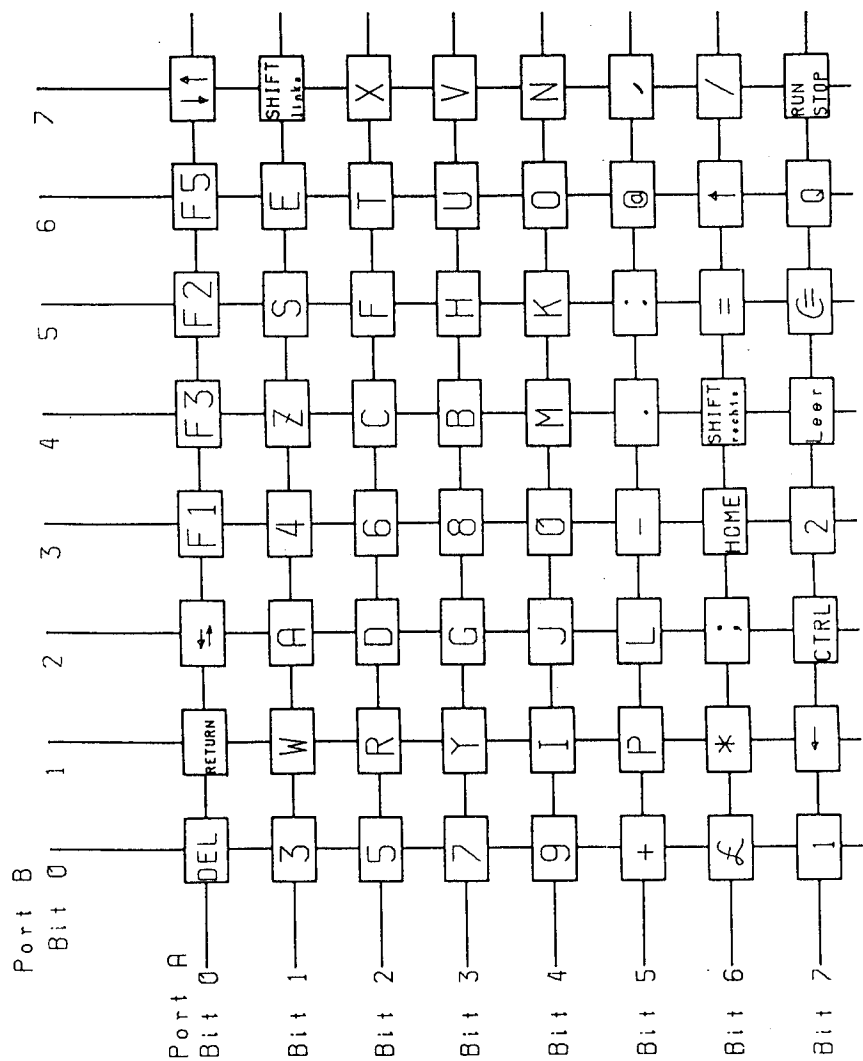
In de Commodore 64 zorgen de CIA's (U1 en U2) voor het uitlezen van toetsenbord en joysticks, de seriële databus voor floppy en printer, een optionele RS-232 aansluiting, de in- en omschakeling van de analoge ingangen, een deel van de besturing van de datasette en de hulpadresbits A14 en A15 voor de VIC. Een CIA (complex interface adapter) heeft 16 programmeerbare in- en uitvoerlijnen, twee intervaltimers, een realtimeclock met instelbare alarmtijd en een 8-bits schuifregister voor seriële in- en uitvoer. De pinconfiguratie werd in paragraaf 4.1 uit de doeken gedaan.

De verbindingen PA0-PA7 zijn bi-direktioneel en kunnen als in- en uitvoer worden geprogrammeerd. Daardoor kunnen 8- of 16-bits gegevens worden gelezen of uitgevoerd. CIA 1 zet de 16 I/O-lijnen op de stekker, waarmee het toetsenbord, een matrix van 8x8 lijnen is aangesloten. Het toetsenbord van de Commodore 64 heeft 66 toetsen, RESTORE en SHIFT/LOCK zijn de extra twee. De RESTORE-toets is niet in de matrix opgenomen. Deze toets legt de ingang van U20 aan massa en produceert een NMI. SHIFT/LOCK is met een SHIFT-toets parallel geschakeld en dus in de matrix overbodig. In de volgende tekening staan de plaatsen van de toetsen in de matrix en de aansluiting op de CIA schematisch weergegeven. Als bij de CIA's een portbit als ingang is geschakeld, maar niet wordt gebruikt, maakt de CIA die aansluiting high.

PA0-PA7 zijn uitgangen, PB0-PB7 ingangen. Als het bedrijfssysteem het toetsenbord wil uitlezen, worden de aansluitingen van port A low. Drukt u bijvoorbeeld op H, dan wordt bit 5 van port B low. De computer weet, dat F3, S, F, H, K, :, = of C= is ingetoetst, nog niet welke van de acht. Een druk op een toets maakt de uitgangen van port A een voor een low en elke keer wordt nagegaan of een ingang low is. Bij een H wordt dat geconstateerd als bit 3 van port A low is: zijn plaats in de matrix is duidelijk.

CIA 1 zorgt ook voor het uitlezen van de joysticks. Een joystick bestaat uit 5 schakelaars. Vier daarvan zijn voor de richtingen, de vijfde is de rode knop. Deze schakelaars bevinden zich niet in een matrix, maar leggen elke keer een portbit aan massa. Joystick 1 regelt de portbits 0-4 van poort B, nummer 2 0-4 van poort A. Bovendien zorgen bits 6 en 7 van poort A van CIA 1 voor de omschakeling van de paddles (paragraaf 9.8).

Ingangspin 19 (TOD: time of day) levert de frequentiesignalen voor de realtimeclock in de CIA's. Op deze ingangen wordt de blokpuls van 50 Hz gezet, die gate U27 levert. De CIA's delen deze frequentie door 5. 10 Hz is 10 impulsen per seconde; 1/10 seconde



is de kleinste tijdseenheid, die de CIA's kunnen meten. -IRQ (pin 21) zorgt voor interrupts. Bij CIA 1 is dit signaal met de -IRQ-ingang van de processor verbonden. De aansluiting is low, als bepaalde programmeerbare gebeurtenissen in de CIA optreden. R/-W (pin 22) regelt de manier, waarop gegevens worden overgedragen. Moeten de registers van CIA 1 en 2 worden gelezen, dan is R/-W high, bij een schrijfactie low.

Alle perifere bouwstenen in de Commodore 64 zijn memory mapped, u kunt ze dus aanspreken net als geheugenplaatsen in het RAM. Bij elke activiteit van de processor in adresbereik \$DC00-\$DFFF wordt uitgangspin 7 van decoder U15 low, als -CHAREN high is. Daardoor wordt de ENABLE-ingang van de tweede decoder in U15 low: decoder 2 is vrijgegeven. Aan de ingangen (pin 13 en 14) van deze decoder liggen adresbit A8 en A9, waarmee het door decoder 1 gedecodeerde adresbereik wordt verdeeld in vier stukken van 256 bytes. Het eerste bereik (\$DC00-\$DCFF (pin 12 van U15)) selecteert CIA 1. Het tweede (\$DD00-\$DDFF (pin 11)) selecteert CIA 2. De twee vrije gedecodeerde bereiken (\$DE00-\$DEFF en \$DFO0-\$DFFF) liggen als IO1 en IO2 aan pin 7 en 10 van de cartridge expansion. IO1 is nodig om de CP/M-cartridge in te schakelen, IO2 is niet voor een CBM-uitbreiding gereserveerd en dus vrij voor eigen projecten. Is een CIA geselecteerd, dan bepalen RSO-RS3 welk intern register moet worden aangesproken. Deze ingangen zijn daarom verbonden met de LSB's van de adressen van de processorbus. De datalijnen DO-D7 (pin 33-26 van de CIA's) zijn verbonden met de databus van de processor. Via deze lijnen worden in de registers gegevens gelezen of geschreven. Ingang -FLAG van CIA 1 heeft een dubbele functie. Aangesloten op pin D en 4 van cassetteport CN3 worden via deze lijn de weergavesignalen ingelezen, die de datasette levert. Ten tweede is -FLAG verbonden met pin 1 (-SRQIN) van de seriële bus (CN4). Via deze service request ingang kan perifere apparatuur laten weten of ze gegevens voor de computer hebben. Geen enkele Commodore computer gebruikt die functie. Deze -FLAG-ingang is flankgetriggerd. Bij een negatieve puls worden de gegevens aan port B tussentijds in de CIA opgeslagen. Bovendien merkt de CIA elke wisseling van high naar low, waardoor een interrupt aan pin 21 mogelijk is. De functie van uitgang -PC komt ongeveer overeen met die van de -FLAG-ingang. Als register 1 van de CIA wordt aangesproken, wordt aan -PC een negatieve puls van een microseconde geproduceerd. Al naar gelang de programmering is SP (serial port: pin 39) in- of uitgang van het schuifregister in de CIA. Met een schuifregister kan zonder veel moeite een seriële aansluiting worden gerealiseerd. SP van CIA 1 ligt aan pin 5 van userport CN2. CNT van CIA 1 ligt aan pin 4 van die poort en wordt gebruikt om de frequentie van het schuifregister uit of in te voeren. Bovendien kan CNT als frequentie-ingang geprogrammeerd worden voor de tellers van de CIA.

Poort A van CIA 2 vervult verschillende functies. Bit 0, VA14, is het hulpadresbit voor de video, bit 1 VA15. PA2 aan pin M is het enige bit van poort A, die geen functie heeft. Bit 3-7 zijn voor de seriële IEC-bus. Bit 3, de uitgang, die het -ATN-sigitaal levert, is verbonden met pin 1 van U8. U8 bevat 6 buffers, signaalversterkers, die het signaal niet inverteren. Ze zijn voorzien van open collectoruitgangen en hebben aan de uitgang een weerstand nodig naar +5 V. Uitgangspin 2 (-ATN) van de buffer is verbonden met pin 3 van de seriële bus en pin 9 van de userport. Het portbit kan ook gebruikt worden voor uw schakelingen aan de userport, maar alleen als uitgang. Portbit 4 (CLK OUT) en 5 (DATA OUT) worden ook via U8 gebufferd. De uitgangen van beide buffers zijn met pin 4 (CLK) en 5 (DATA) van de seriële bus verbonden. In de rusttoestand zijn deze ingangen high. Tegelijkertijd liggen portbit 6 en 7 aan de uitgangen van de buffer, omdat CLK en DATA bidirectioneel zijn. Ze worden niet alleen in de computer aangemaakt, maar ook in een aangesloten floppy of printer. Bit 6 en 7 zijn dus als ingangen geprogrammeerd en een extern apparaat kan ze aan massa leggen. Alle 8 lijnen van poort B liggen nu aan de userport, zodat de gebruiker over deze in- en uitvoermogelijkheden kan beschikken. Gedetailleerde informatie over programmering en gebruik van de userport vindt u in paragraaf 1.6. Pin 8 van de userport is met -PC (pin 18) van CIA 2 verbonden. Met een negatieve puls geeft de CIA via deze uitgang aan, dat gegevens van poort B in een register werden gelezen of geschreven. -FLAG (pin 24) van CIA 2 ligt aan pin B van de userport. Is poort B als ingangspoort geprogrammeerd, dan kunnen de gegevens aan die poort bij een negatieve flank tussentijds in de CIA worden opgeslagen. Met de -IRQ-uitgang van CIA 2 kan een interrupt worden gepleegd. Een interrupt van CIA 2 veroorzaakt aan de processor een -NMI, die door software niet is uit te schakelen. De decoding van adresbereiken gebeurt net als bij CIA 1 door de decoders in U15. CIA 2 gebruikt de adressen \$DD00-\$DDFF. Het -CS-sigitaal, dat aan pin 23 nodig is, wordt aan pin 11 van de decoder geproduceerd. Aan pin 34 van de beide CIA's ligt het RESET-sigitaal voor het bedrijfssysteem. Daarmee worden in deze perifere bouwstenen alle registers na het inschakelen van de computer in een gedefinieerde begintoestand gebracht. De overige signalen aan U2 hebben dezelfde functie als aan CIA 1.

9.9.1 De modulator

In tegenstelling tot de VIC 20 zit de modulator van de Commodore 64 in het apparaat. Er is helaas geen schema van de modulator. De modulator bestaat voornamelijk uit een oscillator met een frequentie in het UHF-gebied. Op dit signaal worden de signalen SYNC+LUM en COLOR van de VIC 6569 en het audio-signaal van de SID 6581 gesuperponeerd. Het resultaat ligt aan de cinch-bus en kan met een coax-kabel op de antenne-ingang van een televisie worden gezet. De afregelcomponenten, die door de openingen in de modulator te zien zijn, mag u nooit verstellen. De modulator is in de fabriek gegarandeerd optimaal afgeregeld.

9.10 Een defecte Commodore 64

Als uw Commodore 64 niet goed functioneert en de garantietermijn is nog niet verstreken, ga dan zo snel mogelijk terug naar de winkel. Maar als u de computer al een half jaar in huis hebt, kunt u proberen of u hem zelf weer aan de praat kunt krijgen. In deze paragraaf staan enkele tips en een testprogramma voor de geheugenbouwstenen, waarmee u het mankement misschien kunt opsporen en verhelpen. De reparatie van een computer is iets anders dan de programmering. Maar als u met een soldeerbout om kunt gaan en het volgende gereedschap bij de hand hebt, kunt u het rustig eens proberen.

soldeerbout met een fijne punt, liefst 30 Watt
desoldeerapparaat
strip/kniptang
universeelmeter: gelijk- en wisselspanning
kruiskop schroevendraaier
64 INTERN

Bovendien is het handig om een oscilloscoop in huis te hebben (een een-kanaals is voldoende) en iemand, die zijn Commodore 64 meeneemt, zodat de IC's kunnen worden verwisseld. Eerst moeten we er achter zien te komen of de computer volledig is uitgevallen of dat alleen bepaalde functies niet goed werken.

9.10.1 Geen beeld

Als de computer aan staat, maar er geen beeld op het scherm verschijnt, is de computer volledig uitgevallen. Controleer eerst de stroomtoevoer. Heel vaak is de zekering van 1 of 1.25 Ampere om onverklaarbare redenen kapot. Dan brandt de LED op de computer wel, maar u heeft geen beeld. U moet geen sterkere zekering gebruiken dan een van 1.25 Ampere. Aluminiumfolie als zekering heeft al heel wat schade aangericht die had kunnen worden voorkomen. Is de zekering in orde, meet dan de spanningen door, die de voeding levert aan de contacten van de DIN-bus. Vaak ontbreekt de 5 Volt gelijkspanning, die de stabilisatieschakeling in de voeding levert. Maar ook met de 9 Volt wisselspanning is er soms wat mis. Als er helemaal geen spanning aan de contacten ligt, controleer dan de zekering bij de trafo. Met een grote kruiskop schroevendraaier kunt u de huls van de zekering losschroeven. Gebruik alleen zekeringen met dezelfde waarde. Is de zekering in orde, maar liggen er geen spanningen aan de contacten, haal dan een nieuwe voeding. Bij de nieuwere versie van de Commodore 64 is het huis van de voeding aan elkaar gelijmd, zodat er niet zo makkelijk kan worden ingegrepen. Wees erg voorzichtig als u de voeding opent: op de primaire kant van de trafo staat 220 Volt wisselspanning. Meet de gelijkrichterschakeling door als een echte elektricien: met een hand in uw broekzak. Werkt de trafo, controleer dan de stroomtoevoer in de computer aan de hand van het schema. Aan de TTL-IC's moet een spanning van 5 Volt liggen. De 12-volts gelijkspanning kunt u bijvoorbeeld aan pin 28 van de SID (U18) meten. Is de stroomtoevoer in orde, dan kan het defect aan de modulator of de aansluitkabel liggen. De computer werkt wel, maar het is niet te zien. Door een floppy of datasette aan te sluiten kunt u nagaan of dit het geval is. Is een datasette aangesloten, druk dan de toetsen SHIFT/RUN en STOP in en daarna de play-toets van de recorder. Begint vervolgens de as in de recorder te draaien, controleer dan eerst de kabel naar monitor of televisie. Is daar niets mee aan de hand, dan is waarschijnlijk de modulator defect. In dat geval moet de computer naar de werkplaats. Bij een floppy kunt u blind LOAD"\$",8 intoetsen. Werkt de floppy, dan geldt hetzelfde als bij de recorder.

Vaker dan dat er helemaal geen beeld is, gebeurt het, dat de computer niet reageert, nadat kader en achtergrond op het scherm verschenen zijn. Meet eerst de -IRQ-lijn door. Gewoonlijk is er aan deze aansluiting (pin 3 van de processor) elke 16 milliseconden 200 microseconden lang een negatieve puls. Een universeelmeter meet alleen een gelijkspanning van circa 4.5 Volt.

Met een LED en een weerstand van 220 Ohm kunt u waarnemen of de interrupts komen. Verbind de weerstand met 5 Volt en gebruik hem als voorweerstand voor de LED. Houd de kathode van de LED aan processorpin 3. De LED moet gaan branden en een beetje knipperen. Beter is natuurlijk een logic analyzer of een oscilloscoop. Zijn er geen interrupts en wordt het signaal IRQ, dat in het begin high is, na korte tijd low, dan ligt de oorzaak van de fout bij de processor, het ROM van het bedrijfssysteem of het RAM. Dan is het een geval voor een goed geoutilleerde werkplaats, tenzij u de mogelijkheid heeft om de IC's uit een andere computer te proberen. Een defecte CIA 6526 is makkelijk op te sporen, omdat er twee CIA's in de Commodore 64 zitten. U kunt ze verwisselen, want de computer werkt ook met een CIA. Helaas is Commodore er steeds meer toe overgegaan om de IC's direkt op te printplaat te solderen. In maar weinig exemplaren zitten processor, CIA's en ROM's in voetjes. Als u tijdens een reparatiepoging een IC uit de plaat hebt gesoldeerd, loont het de moeite om wat geld in voetjes te investeren. Dat geldt zeker voor de CIA's, omdat die dan makkelijk kunnen worden verwisseld. Levert het verwisselen van de CIA's niets op, controleer dan of de processorport defect is. Meet -LORAM, -HIRAM en -CHAREN aan pin 27, 28 en 29 van de processor door. Als u de computer aanzet, moeten ze high zijn. In het andere geval werkt de computer niet goed, omdat niet de ROM's zijn ingeschakeld, maar het RAM is aangesproken, dat eronder ligt. Omdat de computer, als hij is aangezet het ROM nodig heeft, kan worden nagegaan of de processorport de oorzaak van het euvel is. Soldeer hem uit de printplaat en plaats daar een 40-polige voet. Buig pin 27, 28 en 29 naar buiten, zodat ze de printplaat niet raken als de processor in zijn voet wordt geplaatst. Daardoor denkt AM U17, dat de signalen in orde zijn, omdat een open ingang aan TTL-IC's als high wordt geïnterpreteerd. Werkt de processor na deze procedure, dan moet hij worden vervangen. Hebt u na alle pogingen het defect niet kunnen localiseren, dan moet de computer de deur uit. Het mankement zit ergens in de hardware en kan met amateuristische hulpmiddelen praktisch niet worden opgespoord.

9.10.2 Defecten van andere aard

Bij mankementen, die na een bepaalde tijd of sporadisch optreden, komt het voor, dat de computer in de werkplaats dagen achtereen perfect functioneert, hoewel de klacht was, dat het apparaat er na een half uur mee ophoudt. Vaak gaat het om een thermisch defect

in een bepaalde component. Heeft u dergelijke klachten, schaf dan in de elektronica-vakhandel een flinke bus koelspray aan. U kunt de componenten daarmee tot -40 graden Celsius afkoelen. Nadat het toetsenbord is verwijderd, wordt de printplaat gelijkmatig verwarmd met een föhn. Maakt de computer een fout, koel dan systematisch alle onderdelen af en schakel na elke component de computer uit en weer in. Functioneert de machine op een gegeven moment, verwarm dan de IC's, die u het laatst heeft bespoten, nog een keer. Zo controleert u of ze het defect ook werkelijk hebben veroorzaakt. Als dat zo is, moeten ze worden verwisseld. Ook de printplaat kan de oorzaak van het euvel zijn. Door verwarming zet het materiaal uit, waardoor haarscheurtjes ontstaan, die alleen met veel geluk te vinden zijn. In de Commodore 64 komt dit mankement weinig voor, omdat de printplaat van goede kwaliteit is. Ook defecte RAM's kunnen sporadische fouten opleveren. Het volgende programma test het hele 64K RAM, dus ook het RAM onder BASIC-ROM, karakter-ROM en bedrijfssysteem. Zodra een fout in een geheugenplaats wordt opgemerkt stopt het programma en wijst het RAM-IC aan, dat vermoedelijk defect is. Maar als de fout in het kleur-RAM zit, loopt het programma verder. U herkent deze fouten aan het feit, dat de posities op het scherm niet van kleur veranderen. Lees het programma in en test het zolang de computer goed werkt, zodat u het kunt laden en starten als het nodig is.

```

100 FOR I=32768 TO 33319
110 READ X:POKEI,X:S=S+X:NEXT
120 DATA 120,162,255,154,160, 0,169, 1,153, 0,216,153
130 DATA 0,217,153, 0,218,153, 0,219,169, 32,153, 0
140 DATA 4,153, 0, 5,153, 0, 6,153, 0, 7,200,208
150 DATA 225,160, 13,185,199,129, 41, 63,153, 92, 4,136
160 DATA 16,245,160, 15,185,213,129, 41, 63,153,171, 4
170 DATA 136, 16,245,160, 8,185,229,129, 41, 63,153,148
180 DATA 5,136, 16,245,160, 2,162, 0,169, 85,153, 0
190 DATA 0,232,208,253,217, 0, 0,208, 21,140, 3, 5
200 DATA 42,153, 0, 0,232,208,253,217, 0, 0,208, 6
210 DATA 140, 4, 5,200,208,222,240, 3, 76,153,129,160
220 DATA 1,185, 5,130, 41, 63,153,174, 5,136, 16,245
230 DATA 160, 4,185,238,129, 41, 63,153,228, 5,136, 16
240 DATA 245,160, 0,169, 85,153, 0, 1,232,208,253,217
250 DATA 0, 1,208, 21,140, 43, 5, 42,153, 0, 1,232
260 DATA 208,253,217, 0, 1,208, 6,140, 44, 5,200,208
270 DATA 224,240, 3, 76,153,129,160, 1,185, 5,130, 41
280 DATA 63,153,254, 5,136, 16,245,160, 7,185,243,129
290 DATA 41, 63,153, 52, 6,136, 16,245,160, 0,132, 1
300 DATA 169,250,133, 16,169,128,133, 17,132, 18,132, 20
310 DATA 200,132, 19,200,132, 21,160,100,177, 16,145, 18
320 DATA 136, 16,249,162, 0,160, 0, 76, 0, 1,177, 20
330 DATA 72,169, 85,145, 20,209, 20,208, 57, 42,145, 20
340 DATA 209, 20,208, 50,104,145, 20,200,208,232,230, 21

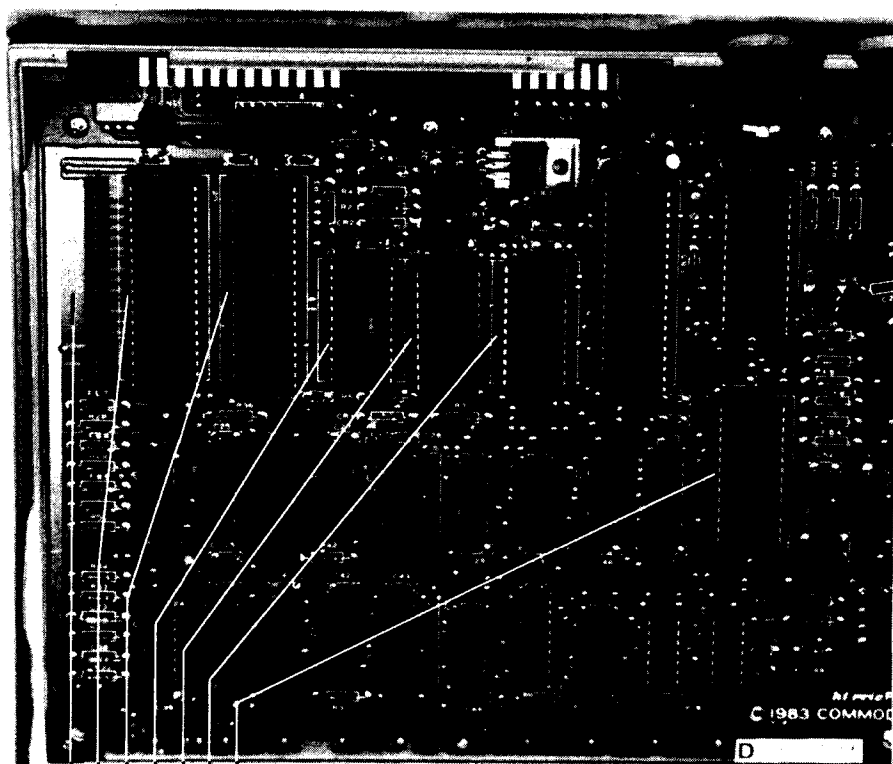
```

```

350 DATA 173, 3, 5, 16, 16, 41,127,141, 3, 5,141, 4
360 DATA 5,141, 43, 5,141, 44, 5,208, 14, 9,128,141
370 DATA 3, 5,141, 4, 5,141, 43, 5,141, 44, 5,165
380 DATA 21,208,191, 76, 65,129, 76,153,129,160, 1,185
390 DATA 5,130, 41, 63,153, 78, 6,136, 16,245,160, 9
400 DATA 185,251,129, 41, 63,153,132, 6,136, 16,245, 32
410 DATA 188,129, 32,188,129, 32,188,129,169, 55,133, 1
420 DATA 162, 4,134, 19,177, 18, 9,128,145, 18,200,208
430 DATA 247,230, 19,202,208,242,169, 15,153, 0,216,153
440 DATA 0,217,153, 0,218,153, 0,219,200,208,241, 32
450 DATA 188,129,170,202,138,208,233, 32,188,129, 76, 4
460 DATA 128, 24,162,255,232,106,144,252,134,255, 24,138
470 DATA 101,255, 32, 32, 32, 50, 53, 53, 32, 50, 53, 53
480 DATA 32, 32, 32, 49, 48, 49, 32, 49, 48, 49, 32, 32
490 DATA 32, 49, 51, 56, 32, 49, 51, 56, 32, 32, 32, 50
500 DATA 52, 32, 50, 52, 32, 32, 32, 50, 53, 53, 32, 50
510 DATA 53, 53, 32, 32, 32, 49, 51, 52, 32, 49, 51, 52
520 DATA 32, 32, 32, 50, 53, 50, 32, 50, 53, 50, 32, 32
530 DATA 32, 49, 52, 52, 32, 49, 52, 52, 32, 32, 32, 49
540 DATA 48, 54, 32, 49, 48, 54, 32, 32, 32, 50, 51, 50
550 DATA 32, 50, 51, 50, 32, 32, 32, 50, 53, 53, 32, 50
560 DATA 53, 53, 32, 32, 32, 49, 54, 50, 32, 49, 54, 50
570 DATA 32, 32, 32, 50, 52, 32, 50, 52, 32, 32, 32, 49
580 IF S<> 51991 THEN PRINT "FOUT IN DATA": STOP
590 PRINT "OK": SYS 32768

```

9.11 het binnenwerk



SID 6581

Karakter-ROM

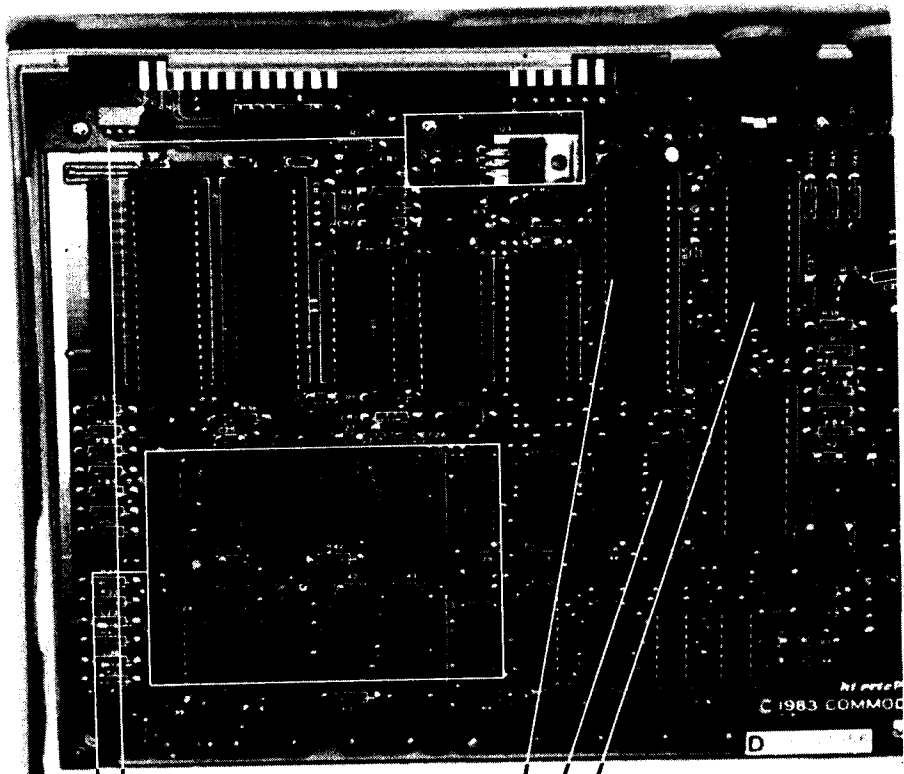
Kernal-ROM

BASIC-ROM

CIA U2, userport, optionele RS-232, gedeeltelijk
seriële IEC-bus, video hulpadressen,
cassette-READ

CIA U1, toetsenbord uitlezen, controlports

aansluiting voor toetsenbord



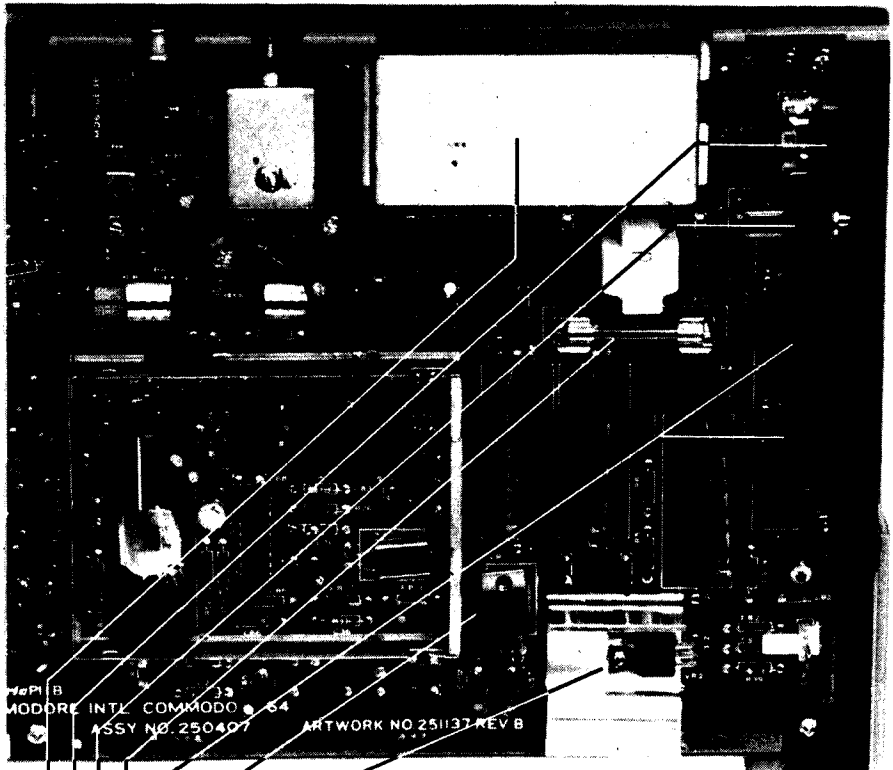
memory manager

kleur-RAM

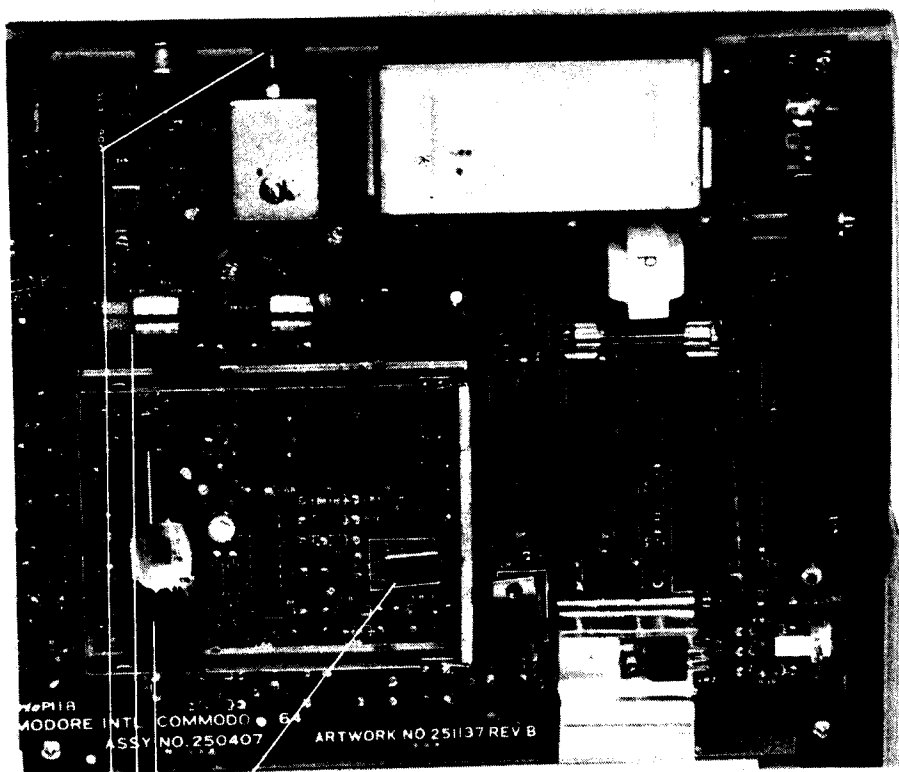
processor 6510

transistortrap: stuurt motor van datasette

64K RAM en multiplexer



- spanningsregelaar: 5 Volt gelijkspanning
- spanningsregelaar: 12 Volt gelijkspanning
- controlports
- zekering
- aan/uit-schakelaar
- expansiepoort



kwartskristal

VIC 6569

geopende modulator

instelbare schakeling voor kanaalfrequentie van modulator. Kan versteld worden als een sterke televisiezender op hetzelfde kanaal zit.

9.12 Lijst van halfgeleiders

naam	type	gefabriceerd door
U1	6526 CIA	Commodore MOS
U2	6526 CIA	Commodore MOS
U3	2364A BASIC	Commodore MOS
U4	2364A KERNAL	Commodore MOS
U5	2332A CHARACTER	Commodore MOS
U6	2114L-3 COLOR RAM	verschillende fabrikanten
	bijvoorbeeld:	
	MSM 2114L-3	OKI
	2114L-3	Fairchild
	HM2114L-3	Hitachi
	MPS2114L-30	MOS
	MCM2114L-30	Motorola
	uPD2114L-1	NEC
U7	6510 MPU	Commodore MOS
U8	7406	verschillende fabrikanten
U9)	4164 RAM	verschillende fabrikanten
U10)	bijvoorbeeld:	
U11)	uPD4146-2	NEC
U12)	MK4146-10	MOSTEK
U21)		
U22)		
U23)		
U24)		
U13	SN74LS257	verschillende fabrikanten
U14	SN74LS278	verschillende fabrikanten
U15	SN74LS139	verschillende fabrikanten
U16	MC4066	verschillende fabrikanten
U17	825100	Signetics, geprogrammeerd door Commodore
U18	6581 SID	Commodore MOS
U19	6589 VIC	Commodore MOS
U20	556	verschillende fabrikanten
U25	SN74LS257	verschillende fabrikanten
U26	SN74LS373	verschillende fabrikanten
U27	SN74LS08	verschillende fabrikanten
U28	MC4066	verschillende fabrikanten
U29	SN74LS74	verschillende fabrikanten

U30	SN74LS193	verschillende fabrikanten
U31	SN74LS629	verschillende fabrikanten
U32	MC4044	Motorola
spanningsregelaars:		
VR1	7812 12V	verschillende fabrikanten
VR2	7805 5V	verschillende fabrikanten

9.13 De draagbare SX-64

De draagbare versie van de Commodore 64 meet 30 bij 14 bij 42 centimeter en is voorzien van een handgreep. Hij bestaat uit de volgende componenten:

- de aangepaste printplaat van de Commodore 64 in twee delen
- de aangepaste printplaat van floppy VC-1541
- een luidspreker van 8 centimeter doorsnee
- een monitor, monochroom of in kleur
- een of twee diskdrives
- een heel dun toetsenbord (middelste hoogte 3 cm), dat de voorkant van het apparaat afsluit

Andere bedieningselementen zijn:

- aan en uit schakelaar aan de achterkant
- aan de voorkant achter een klepje knoppen voor:
volume, contrast, helderheid, kleurverzadiging en rood-groen balans

De aansluitingen zijn voor externe apparatuur zijn:

- twee aansluitingen voor joy sticks
- de seriële IEC-bus
- monitoraansluiting (niet gemoduleerd)
- userport
- expansie poort met veerkleppen aan bovenkant

N.B: Er is geen aansluiting voor een cassetterecorder.

De printplaten staan rechtop en zijn met een stekker verbonden. De onderdelen zijn dicht op elkaar gepakt, maar luchtgaten zorgen ervoor, dat de overvloedige warmte kan ontsnappen. Thermische problemen treden daarom niet op. Het beeldscherm van 5 inch (circa 13 centimeter) produceert een scherp beeld met heldere

kleuren. Dat is te danken aan het feit, dat het videosignaal niet via een modulator loopt. Bij gebruik van een televisie was dat wel nodig geweest. Wie een groter beeld wil, kan dus niet zomaar de televisie aansluiten. De SX-64 is een handzaam en ruimtebesparend apparaat.

CHIP : VIC (6567) Video-Controller

+0 #0000
+1 #0001
+2 #0002
+3 #0003
+4 #0004
+5 #0005
+6 #0006
+7 #0007
+8 #0008
+9 #0009
+10 #000A
+11 #000B
+12 #000C
+13 #000D
+14 #000E
+15 #000F
+16 #0010
+17 #0011
+18 #0012
+19 #0013
+20 #0014
+21 #0015
+22 #0016
+23 #0017
+24 #0018
+25 #0019
+26 #001A
+27 #001B
+28 #001C

Sprite 0							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 1							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 2							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 3							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 4							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 5							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 6							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite 7							
X-Koordinate (0-255)							
Y-Koordinate							
Sprite X-Koordinate +256							
<div> <div>Sprite 7</div> <div>Sprite 6</div> <div>Sprite 5</div> <div>Sprite 4</div> <div>Sprite 3</div> <div>Sprite 2</div> <div>Sprite 1</div> <div>Sprite 0</div> </div>							
Raster	Mintorg.	Mi-Rac	Blid	25/24	Position		
RC0	Mehrfarb	Grafik	ana	Zellen	YSCL2	YSCL1	YSCL0
Raster							
RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
Light - Pen							
X-Koordinate							
Light - Pen							
Y-Koordinate							
Sprite an/aus							
SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0
Nicht benutzt			Mehrfarb	40/30	Position		
#	#	#	Modus	Spalten	XSCL2	XSCL1	XSCL0
Sprite vergrössern Y-Koordinate							
SEXY7	SEXY6	SEXY5	SEXY4	SEXY3	SEXY2	SEXY1	SEXY0
Bildschirmspeicher Adr. (01024)				Zeichengr. Adr. (02048)		Nicht	
CB13	CB12	CB11	CB10	YS13	YS12	YS11	benutzt
IR0	Nicht benutzt			Light-P.	Sp.-Sp.	Sp.-Hm.	Raster
Flag	#	#	#	,-Flag	Kell.	Kell.	,-Flag
Nicht benutzt							
#	#	#	#	Mask	Mask	Mask	Mask
				HLPI	HLSC	HLSC	HLIR0
Sprite-Mintorggrund Prioritaet							
OSP7	OSP6	OSP5	OSP4	OSP3	OSP2	OSP1	OSP0
Mehrfarbiges Sprite							
SHC7	SHC6	SHC5	SHC4	SHC3	SHC2	SHC1	SHC0

53248
53249
53250
53251
53252
53253
53254
53255
53256
53257
53258
53259
53260
53261
53262
53263
53264
53265
53266
53267
53268
53269
53270
53271
53272
53273
53274
53275
53276

		Spalte vorgezeichnet X-Koordinate								
+29	0001D	9EXX7	9EXX8	9EXX5	9EXX4	9EXX3	9EXX2	9EXX1	9EXX0	53277
		Spalte-Spalte Nullstellen								
+30	0001E	99C7	99C8	99C5	99C4	99C3	99C2	99C1	99C0	53278
		Spalte-Hintergrund Nullstellen								
+31	0001F	98C7	98C8	98C5	98C4	98C3	98C2	98C1	98C0	53279
		Randfarbe (0-15)								
		EXT1								53280
		Hintergrundfarbe Nr. 0 (0-15)								
+33	00021	BKGD0								53281
		Hintergrundfarbe Nr. 1 (0-15)								
+34	00022	BKGD1								53282
		Hintergrundfarbe Nr. 2 (0-15)								
+35	00023	BKGD2								53283
		Hintergrundfarbe Nr. 3 (0-15)								
+36	00024	BKGD3								53284
		Gemeinsame Spalte-Farbe Nr. 0 in Mehrfarben-Modus (0-15)								
+37	00025	SHCP								53285
		Gemeinsame Spalte-Farbe Nr. 1 in Mehrfarben-Modus (0-15)								
+38	00026	SHC1								53286
		Farbe Spalte Nr. 0 (0-15)								
+39	00027	SPCOL								53287
		Farbe Spalte Nr. 1 (0-15)								
+40	00028	S1COL								53288
		Farbe Spalte Nr. 2 (0-15)								
+41	00029	S2COL								53289
		Farbe Spalte Nr. 3 (0-15)								
+42	0002A	S3COL								53290
		Farbe Spalte Nr. 4 (0-15)								
+43	0002B	S4COL								53291
		Farbe Spalte Nr. 5 (0-15)								
+44	0002C	S5COL								53292
		Farbe Spalte Nr. 6 (0-15)								
+45	0002D	S6COL								53293
		Farbe Spalte Nr. 7 (0-15)								
+46	0002E	S7COL								53294

CHIP : SID (6581) Sound Synthesizer

#D400	OSZILLATOR 1								54272	
Frequenz, low Byte										
#D401	Frequenz, high Byte								54273	
#D402	Pulsebreite, low byte								54274	
#D403	0	0	0	0	high Byte				54275	
#D404	Wellenform								54276	
RAUSCH RECHTECK SAEGER.				DREIECK TEST RING SYNC KEY						
Anstiegszeit				Abfallzeit						
#D405									54277	
SUSTAIN				Ausklingszeit						
#D406									54278	
OSZILLATOR 2										
#D407	Frequenz, low Byte								54279	
#D408	Frequenz, high Byte								54280	
#D409	Pulsebreite, low byte								54281	
#D40A	0	0	0	0	high Byte				54282	
#D40B	Wellenform								54283	
RAUSCH RECHTECK SAEGER.				DREIECK TEST RING SYNC KEY						
Anstiegszeit				Abfallzeit						
#D40C									54284	
SUSTAIN				Ausklingszeit						
#D40D									54285	
* OSZILLATOR 3										
#D40E	Frequenz, low Byte								54286	
#D40F	Frequenz, high Byte								54287	
#D410	Pulsebreite, low Byte								54288	
#D411	0	0	0	0	high Byte				54289	
#D412	Wellenform								54290	
RAUSCH RECHTECK SAEGER.				DREIECK TEST RING SYNC KEY						
Anstiegszeit				Abfallzeit						
#D413									54291	
SUSTAIN				Ausklingszeit						
#D414									54292	
#D415	0	0	0	0	0	low Byte			54293	
Filterfrequenz										
#D416	high Byte								54294	
#D417	Resonanz				EXT	OS3	OS2	OS1	54295	
#D418	Filterart				Lautstaerke					54296
03 AUS	HP	BP	TP							
#D419	PADDLE 1								54297	
#D41A	PADDLE 2								54298	
#D41B	RAUSCHEN 3 (Zufallszahl)								54299	
#D41C	Huellkurve 3								54300	

CHIP : CIA (6526)

+0 #DC00	PAR	56320
+1 #DC01	PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0	56321
+2 #DC02	P07 P06 P05 P04 P03 P02 P01 P00	56322
+3 #DC03	DP07 DP06 DP05 DP04 DP03 DP02 DP01 DP00	56323
+4 #DC04	TAL7 TAL6 TAL5 TAL4 TAL3 TAL2 TAL1 TAL0	56324
+5 #DC05	TAM7 TAM6 TAM5 TAM4 TAM3 TAM2 TAM1 TAM0	56325
+6 #DC06	T0L7 T0L6 T0L5 T0L4 T0L3 T0L2 T0L1 T0L0	56326
+7 #DC07	T0M7 T0M6 T0M5 T0M4 T0M3 T0M2 T0M1 T0M0	56327
+8 #DC08	T0D10THS	56328
+9 #DC09	T0D SEC	56329
+10 #DC0A	T0D MIN	56330
+11 #DC0B	T0D HR	56331
+12 #DC0C	SDR	56332
+13 #DC0D	ICR	56333
+14 #DC0E	CRA	56334
+15 #DC0F	CRO	56335

SPRITE - ENTWURFSBLATT

