van Basic tot Machinetaal

Opbouwwerk voor Commodore 64 gebruikers met Basic Machinetaal, Graphics, Geluid en alle hulptalen.

1

Inhoud

1/1

Voorwoord en goede raad

Het boek dat nu voor u ligt is een produkt van meerdere schrijvers. Dat moet ook wel, want er komt zo veel in dit boek aan de orde dat het onmogelijk zou zijn voor één persoon om al die kennis te bezitten. De Commodore 64 is nu eenmaal een computer met vele, vele mogelijkheden, de 64 is niet voor niets de populairste hobbycomputer in de wereld geworden. Alleen in Nederland staan er al tegen de driehonderdduizend.
En met computers is er sprake van een sneeuwbal-effect, hoe meer er zijn van een bepaald type, hoe meer programma's en andere zaken er voor verschijnen. Waardoor de computer zelf op zijn beurt ook weer aantrekkelijker wordt, enzovoorts enzovoorts. Vandaar dat de 64 zonder meer de computer is waar de meeste programma's voor in de winkels liggen (om over gekraakte programma's maar te zwijgen), de meeste extra's voor zijn te krijgen en, natuurlijk, ook de meeste boeken en tijdschriften voor verschijnen.

Vreemd genoeg was dat laatste juist een van de redenen om VAN BASIC NAAR MACHINETAAL te schrijven. Want er zijn zoveel boeken en bladen, dat de geïnteresseerde hobbyïst door de bomen het bos niet meer ziet. Om een beetje bij te blijven kopen sommige mensen vijf tot tien bladen per maand, plus nog zo nu en dan een boek. In al die tijdschriften en boeken staat wel iets wetenswaardigs, maar het eind van het liedje is toch dat men veel informatie steeds weer opnieuw tegenkomt, telkens in een wat ander jasje gestoken. Bovendien is het terugzoeken altijd weer een kleine ramp, stapels en stapels papier doorwroeten, want het heeft toch ergens ingestaan!

Vandaar dit boek, dat de rol van vraagbaak voor iedere Commodore bezitter zal gaan vervullen. Wat u nu voor u heeft is nog maar het begin. Vier keer per jaar zal er een nieuwe aanvulling uitkomen en u kunt zelf mee bepalen wat daarin komt te staan. Wilt u meer over BASIC programmeren weten? We zijn graag tot uw dienst. Of gaat het u juist om machinetaal? Ook dat kan. Als team van schrijvers zijn wij er vast van overtuigd dat we aan alle wensen kunnen beantwoorden. Ieder heeft zijn specialismes (natuurlijk ook zijn stokpaardjes) en samen denken we alle onderwerpen wat betreft de 64 te beheersen. Stuur maar eens een kaartje met uw wensen, u zult het zien, dit is een boek à la carte.
We hebben trouwens nog een voorsprong op andere boeken en tijdschriften. Elk ander boek (en tijdschriftartikel) moet er van uit gaan dat de lezer bepaalde dingen al weet. Of eigenlijk, juist nog niet weet. Daarom herhalen ze steeds weer allerlei zaken, veel informatie komt iedere keer weer terug. Wij hoeven dat niet te doen.

We beginnen gewoon met alles uit te leggen, zodat ook een beginner het kan begrijpen. Maar we hoeven dat maar één keer te doen. Met een boek als dit is het niet nodig om steeds maar weer dezelfde verhalen te vertellen. Als u iets al weet, dan kunt u dat natuurlijk gewoon overslaan, maar als u iets juist niet weet, dan kunt u het altijd opzoeken.

Als voorproefje kunt u eens het nu volgende programma uitproberen. Het geeft een idee van wat wij een aardig BASIC programma'tje vinden. Spelen met geluid en beeld, dat is nu eenmaal gesneden koek voor de 64. Vandaar dat we daar ons visitekaartje van gemaakt hebben:

```
5 POKE56,160:DIMA(7000):CLR:POKE56,32:CLR:DIMB(7),W(7),B1(7),B2(7)
6 W(7)=128+64:W(6)=32+16:W(5)=8+4:W(4)=2+1:FORT=0TO3:W(T)=W(T+4):NEXT
10 INPUT"WAT IS JE NAAM";A$:SI=54272:POKESI+6,240:POKESI+4,17:POKESI+24,15
15 F=SI+1
20 IFLEN(A$)>20THENPRINT"MINDER DAN 20 LETTERS GRAAG.":GOTO10
30 SP=INT((20-LEN(A$))/2)
40 IFSP>0THENA$=A$+" ":A$=" "+A$:SP=SP-1:GOTO40
50 PRINT"□":POKE53272,24:POKE53265,59
60 FORI=1TOLEN(A$)
70 M$=MID$(A$,I,1):IFM$=" "THEN190
80 S=8192+320*11+(I-1)*16
90 L=ASC(M$):IFL>=64THENL=L-64
95 C=53248+L*8
100 POKE56334,0:POKE1,51:FORJ=0TO7:B(J)=PEEK(C+J):NEXT:POKE1,55:POKE56334,1
110 FORJ=0TO7:B1(J)=0:B2(J)=0
120 FORB=7TO4STEP-1:BW=SGN(B(J)AND(2↑B)):B1(J)=(B1(J)OR(BW*W(B))):NEXT
130 FORB=3TO0STEP-1:BW=SGN(B(J)AND(2↑B)):B2(J)=(B2(J)OR(BW*W(B))):NEXT
140 POKES+J*2-312*(J>3),B1(J):POKES+8+J*2-312*(J>3),B2(J):POKEF,RND(1)*256
150 POKES+1+J*2-312*(J>3),B1(J):POKES+1+8+J*2-312*(J>3),B2(J)
160 NEXT
190 NEXT
200 FORT=1024+11*40TOT+39:POKET+40,INT(RND(1)*15+1)*16
205 POKET,INT(RND(1)*15+1)*16:POKEF,RND(1)*256:NEXT
210 GETA$:IFA$=""THEN200
300 PRINT"□":POKE53272,21:POKE53265,27:POKESI+4,0:POKESI+24,0
```

Behalve allerlei informatie biedt dit boek u ook nog handige hulpmiddelen. Zoals cheatsheets, een soort grote spiekbriefjes die over het toetsenbord heenpassen. Een paar daarvan zijn kant en klaar bedrukt, met allerlei gegevens die toegespitst zijn op het een of andere gebied. Zo is er een die ideaal is voor de BASIC-programmeur, alle afkortingen van de commando's staan erop. En allerlei handige adressen, bijvoorbeeld schermkleur en randkleur. Bovendien ook nog de kleurwaardes, teveel om op te noemen.
Dan is er ook zo'n cheatsheet voor machinetaal. Heel erg handig, het bespaart voortdurend heen en weer bladeren om allerlei zaken op te zoeken.
En, speciaal voor de doe-het-zelver, blanco cheatsheets. Zelf te beschrijven met wat u maar wilt. De commando's van uw tekstverwerker, of de speciale trucs bij een spel, of ....
Bij dit boek hoeft u geen gebruiksaanwijzing te hebben, maar een beetje uitleg over de opbouw van het boek kan geen kwaad.

Zo zal het voor een beginner geen zin hebben om met hoofdstuk 12, programmeerkunde, te beginnen. In dat hoofdstuk wordt een bepaalde kennis van BASIC verondersteld. Maar die valt natuurlijk gemakkelijk op te pikken uit hoofdstuk 7, waar de BASIC helemaal in de doeken gedaan wordt.
Laten we eens hoofdstuk voor hoofdstuk bekijken wat er in staat.

Hoofdstuk 1, dit hoofdstuk, dat zal nu wel duidelijk zijn, is een soort gids door de rest van het boek.

In hoofdstuk 2 vinden we een begrippenlijst - die met iedere aflevering zal worden uitgebreid - waarin u die termen kunt terugzoeken die u elders tegenkomt en niet begrijpt. Meestal worden begrippen meteen uitgelegd, maar het kan natuurlijk zo zijn dat een bepaalde term in een ander hoofdstuk wordt besproken dan het hoofdstuk waarin u het voor het eerst tegenkwam. In zo'n geval biedt de begrippenlijst soelaas.
Bovendien bevat hoofdstuk 2 een register, om het opzoeken van het een en ander wat makkelijker te maken.

Hoofdstuk 3, computers en samenleving, is een geval apart. Hier vindt u geen technische informatie over de Commodore 64, maar allerlei achtergronden en ideeën over wat computers nu eigenlijk voor rol spelen in de maatschappij. Computers zijn niet meer weg te denken uit deze wereld, of het nu grote bedrijfscomputers of kleine hobbycomputers zijn. Natuurlijk heeft dat allerlei invloeden op de samenleving en daar gaat het over in dit hoofdstuk. Dat soort zaken gaat iedereen aan, maar zeker voor computerhobbyïsten is het bijna verplichte leesstof.

Die technische informatie komt in het volgende hoofdstuk, nummer 4, wel aan bod. Hier wordt de opbouw van de 64 behandelt. De hardware, zoals dat in het Engels heet. Hoe zit de machine in elkaar, welke chip doet wat? En waarom? Voorlopig zal dit een mager hoofdstukje blijven althans dat is onze planning. Als u het anders wilt zullen we dat graag horen. Maar tot die tijd denken wij dat bijvoorbeeld hoofdstuk 5 voorrang heeft.

Want daar, in hoofdstuk 5, vindt u een uitleg over de werking van de computer. Het operating system, de Kernal, wordt hier blootgelegd. Adreslijsten, registers, geheugenindelingen, voor zowel BASIC- als machinetaal-programmeurs is dit verplichte kost. Hier leert u alles te doen wat Commodore eigenlijk verboden heeft. Of niet zozeer verboden, maar meer verstopt.

Een kale computer is niet veel waard. Er moet toch op zijn minst een cassetterecorder naast staan om er wat mee te kunnen doen. En, als het even kan een diskdrive en een printer, noem maar op. Hierover gaat hoofdstuk 6: CBM 64, de uitbreidingen. Want hoe al die randapparaten bestuurd moeten worden is voor veel mensen een raadsel. Zo'n gemakkelijk onderwerp is dat dan ook niet, dat kunnen we u verzekeren.
Zelfs oude rotten in het computeren willen hier nog wel eens een steekje laten vallen. Voor iedereen die eens wat meer wil dan alleen maar BASIC programma'tjes laden en saven alweer: verplichte lectuur.

In hoofdstuk 7 komt BASIC zelf aan de beurt. De ingebouwde BASIC wordt van haver tot gort besproken en daarbij beginnen we vanaf het absolute nulpunt. Het aansluiten moet u nog uit het handboekje

halen, daarna nemen wij het over. Alle commando's passeren de revue, hoewel natuurlijk sommige niet zo gedegen behandeld kunnen worden als andere. Maar daar wordt op teruggekomen, dat is een belofte. In eerste instantie voor beginners, straks voor gevorderden.

Behalve BASIC 2.0 zijn er nog meer programmeertalen en daar gaat hoofdstuk 8 over. Allemaal hebben ze hun voor- en nadelen. Wij denken dat SIMONS' BASIC de belangrijkste extra taal is voor 64-bezitters en die komt dan ook als eerste aan bod. Voor mensen die graag al die fraaie beeld- en geluidsmogelijkheden van de 64 willen gebruiken maar het eindeloos PEEKen en POKEn minder leuk vinden is SIMONS' BASIC een uitstekend alternatief.
Maar ook PASCAL, LOGO en COMAL, om er maar een paar te noemen, zullen hier een plaatsje vinden.

En dan nu de hoofdschotel. Met de titel 'VAN BASIC NAAR MACHINETAAL' kunnen we niet anders, maar we zouden ook niet anders willen: hoofdstuk 9 gaat over machinetaal. We starten vanaf nul en zien wel waar we eindigen. Want wie echt de 64 wil kunnen programmeren kan er nu eenmaal niet buiten.
Machinetaal is de taal waarin al die fraaie spellen geschreven worden, alle andere talen zijn te traag. Met machinetaal is het mogelijk om de computer volledig te beheer sen, om alle 64K geheugen te gebruiken zoals men dat zelf wil. Wie wil mag het overslaan, maar doet zichzelf tekort.
Natuurlijk komen de paradepaardjes van de 64 ook aan bod! In hoofdstuk 10 vinden we de graphics, de mogelijkheden van het beeldscherm. Nu nog in BASIC, maar dat zal veranderen.

Hetzelfde gaat op voor hoofdstuk 11, geluid. Eerst wordt de theorie besproken, maar dan brandt het los!

Hoofdstuk 12, programmeerkunde, is andere kost. Niet direct geschikt voor beginners, maar wel razend interressant. De principes van het programmeren worden hier behandeld. Waarom is de ene techniek beter dan de andere? Daarbij worden ook allerlei methodes behandeld om beter te leren programmeren, zoals het tekenen van stroomdiagrammen, waarvoor een schabloon is bijgeleverd.
Maar het fraaiste uit dit hoofdstuk is toch wel dat het een ware schatkist zal worden van handige routines.
Sorteer-routines bijvoorbeeld zullen u nooit meer hoofdbrekens bezorgen, want ze staan er, compleet met uitleg, in.

Hoofdstuk 13 bevat pure informatie. Allerlei adressen van leveranciers, softwarehuizen etcetera. Ideaal voor iemand die een heel speciaal iets zoekt. Als het in de handel is, kunt u hier het adres vinden.

En om te weten wat er allemaal op de markt is, hebben we het laatste hoofdstuk, hoofdstuk 14. Dit bevat informatie over alles wat er maar voor de Commodore te koop is. Vanaf programma's tot en met stofhoezen, vanaf printers tot en met joysticks. Alle zaken die op een 64 kunnen worden aangesloten, er ingeladen of er overheen gelegd vinden hier hun plaatsje. met uitgebreide verhalen over wat het kan, en wat het juist niet kan. De kopersgids bij uitstek.
Dat was het dan, VAN BASIC TOT MACHINETAAL in vogelvlucht. We hopen dat u er veel plezier aan zult hebben. En, nogmaals, mocht u wensen hebben, schrijf ons even.

VOORWOORD BIJ DE PROGRAMMALISTINGS

In dit naslagwerk worden alle programmalistings precies zo gepubliceerd als ze er op het scherm uitzien. Dit betekent dat in de listings bepaalde besturingstekens voor kunnen komen. Deze tekens hebben de volgende betekenis.

De beide sets hebben betrekking op de twee karakter-sets die zich in de Commodore-64 bevinden. Is een listing in hoofdletters geschreven, dan geldt set 1. Is de listing in kleine letters geschreven dan is set 2 van toepassing.

Functie	SET1	SET2
Cursor links		
Cursor rechts		
Cursor laag		
Cursor hoog		
Home		
Clear		
Functietoets 1		
Functietoets 2		
Functietoets 3		
Functietoets 4		
Functietoets 5		
Functietoets 6		
Functietoets 7		
Functietoets 8		
Zwart (CTRL-1)		
Wit (CTRL-2)		
Rood (CTRL-3)		
Cyaan (CTRL-4)		
Paars (CTRL-5)		
Groen (CTRL-6)		
Blauw (CTRL-7)		
Geel (CTRL-8)		
Oranje (Commodore-1)		
Bruin (Commodore-2)		
Lichtrood (Commodore-3)		
Grijs 1 (Commodore-4)		
Grijs 2 (Commodore-5)		
Lichtgroen (Commodore-6)		
Lichtblauw (Commodore-7)		
Grijs 3 (Commodore-8)		
Reverse aan (CTRL-9)		
Reverse uit (CTRL-0)		

1/2

Inhoud

Deel 1: Inhoud

2

Trefwoordenregister

Inhoud

Een boek als dit zou niet compleet zijn zonder zowel een begrippenlexicon als een register. Het begrippenlexicon is noodzakelijk om de lezer de mogelijkheid te bieden in de inhoud te grasduinen; het boek in een zelf gekozen volgorde door te nemen. In zo'n geval zal het regelmatig voorkomen dat die lezer een op een andere pagina omschreven term onverwachts en zonder verdere uitleg in de tekst tegenkomt.
Dan brengt het lexicon uitkomst.
Ook kan de betekenis van een woord even onduidelijk zijn, hoewel de eerste omschrijving al is gelezen. Ook daarom is een begrippenlijst bijna onontbeerlijk.

Een register is zo mogelijk nog noodzakelijker. Hoe kan men anders snel de gezochte informatie opsporen? In een naslagwerk als dit kan men niet zonder.

Een woord van geruststelling: zowel het lexicon als het register zullen in de aanvullingen regelmatig worden aangepast. Terwijl het boek groeit in omvang groeien zowel de begrippenlijst als de index mee. U zit dus niet na een paar aanvullingen met een hopeloos verouderd hoofdstuk 2.

2/1

Begrippenlijst

AC: Alternating current, wisselspanning
ACCES: Toegang
ACCUMULATOR: Intern rekenregister in de microprocessor
ADAPTER: Omvormer, voorziening die de apparatuur de juiste voedingsspanning geeft
ADRES: Getal dat een geheugenplaats aanwijst, bepaalde lokatie in een geheugen
ADRESBUS: Groep printsporen die adresinformatie overbrengt
ADRESLIJNEN: Zie adresbus
ADRESSEREN: Het met behulp van een adres aanwijzen van een geheugenplaats
AGGREGEREN: Een optelbewerking op de records van een bestand
AI: Artificial Intelligence, kunstmatige intelligentie, het vermogen van een programma te 'leren' of menselijk denken te imiteren
ALFANUMERIEK: Gegevens die zowel letters, cijfers als leestekens mogen bevatten, teksten. Ze worden ook wel 'characters' genoemd
ALGORITME: Definitie van een probleem-oplossing in termen van een programma, hoeft geen programma te zijn. Vaste stap voor stap procedure om een bepaald probleem op te lossen
ALLOCEREN: Toewijzen of reserveren van geheugenruimte
ALU: Arithmetic Logical Unit, rekenkundige en logische eenheid, belangrijkste onderdeel in een CPU
ANALOOG: Representatie van de waarden van een variabele door middel van een proportioneel variabele fysieke grootheid
APPLICATIEPROGRAMMA: Specifiek programma dat voor een bepaalde gebruiker en een bepaald doel is ontwikkeld
ARRAY: Lijstvariabele. Variabele die bestaat uit een reeks van afzonderlijk te identificeren elementen. Ook wel geïndiceerde variabele, gedimensioneerde variabele genaamd
ASCII: Gestandaardiseerde codeset voor alfanumeriek informatie. De Commodore wijkt hier in een aantal opzichten vanaf
ASSEMBLER: Programma dat mnemonic code in ML vertaald; Machinetaal programma weergegeven in mnemonic code
ASSEMBLEREN: Opbouwen van ML, meestal met assembler; Samenvoegen van modules tot een werkend programma
ASSIGNEREN: Het toekennen van een waarde aan een variabele
ASYNCHRONE OVERDRACHT: Seriële datatransmissie waarbij de onderlinge afstand tussen de overgezonden bits willekeurig is. Elk teken wordt hierbij voorafgegaan door een startbit en afge-

sloten door een stopbit
ATTACK: Stijgtijd van een toon
AUTO START: Zelfstartend, een soms in een programma ingebouwde eigenschap waardoor dat programma dan onmiddellijk na het laden doorstart
AUTOREPEAT: Eigenschap van toetsen dat het ingedrukte teken zich na enige tijd herhaalt
BACKSPACING: Het terugplaatsen van de print- of schrijfkop
BACKUP: Veiligheidskopie van een bestand op disk of cassette waar op teruggevallen kan worden als het oorspronkelijke bestand verloren is gegaan of is verminkt
BANDBREEDTE: Het verschil tussen de hoogste en de laagste beeldbandfrequentie, maat voor de scherpte van het beeld
BANK: Geheugenblok, meestal in en uit het adresbereik van de microprocessor te schakelen
BANKSWITCHING: Het wijzigen van de geheugenstruktuur door een andere selectie van de banken te maken
BASIC: Beginners All Purpose Symbolic Instruction Code, de programmeertaal van veel home- en personal computers
BASICODE: Computeresperanto waardoor de meeste types homecomputer elkaars programma's kunnen lezen
BATCH: Letterlijk stapelverwerking (Engels). File bestaande uit een reeks commando's die samen aan de computer worden aangeboden alvorens achter elkaar verwerkt te worden. Te onderscheiden van real time verwerking
BAUD: Aantal bits per seconde. Naar de onderzoeker Baudot
BAUDRATE: Overdrachtssnelheid in bits per seconde
BCD: Binary Coded Decimal, notatiemethode voor getallen van meervoudige precisie, waarbij voor de representatie van ieder cijfer 4 bits nodig zijn
BEDRIJFS-SYSTEEM: Een verzameling machinetaal programma's die tezamen de basisfuncties van een computer verzorgen
BENCHMARK: Testprogramma om de prestaties van een computer of een processor te evalueren
BESTAND: Informatie op cassette of diskette, groep bij elkaar behorende gegevens met een naam
BESTURING-SYSTEEM: Zie Operating system
BI-DIRECTIONEEL: Techniek waarbij een afdrukmechanisme de regels om beurten van links naar rechts en omgekeerd print, hetgeen tijdwinst oplevert bij het afdrukken
BINAIR: Tweetallig, vaak gebruikt talstelsel bij microprocessors
BINAIRE CODE: Code waarbij gebruik gemaakt wordt van de tekens 0 en 1
BIT MAPPED: Beeldschermconfiguratie waarbij ieder pixel op het scherm apart kan worden aan- of uitgezet. Ieder pixel komt dan overeen met 1 bit in het geheugen
BIT: Kleinst mogelijke hoeveelheid informatie, 1 of 0, ja of nee, stroom of geen stroom, samentrekking van binary digit
BLINKING: Het knipperen van tekst of gegevens op het scherm
BLOCK: Verzameling aaneengesloten records op disk
BOARD: De kunststof kaart waarop de chips, weerstanden en condensatoren zijn aangebracht en de sporen zijn geprint om deze onderling te verbinden
BOLDFACING: Het dikker afdrukken van een teken door het meerdere keren steeds iets verschoven af te drukken
BOOTSTRAPPING: Het automatisch opstarten van een programma
BUBBLE-MEMORY: Magneetbel-ge-

heugen. Niet vluchtig geheugen waarin de gegevens worden weergegeven door microscopisch kleine magnetische gebieden. Na uitschakelen van de voeding blijven de gegevens bewaard

BUBBLE-SORT: Ietwat trage maar soms goed bruikbare sorteerroutine

BUFFER: Stuk geheugen waar informatie tijdelijk wordt opgeslagen, bijvoorbeeld tot de microprocessor tijd kan vrijmaken om deze verder te verwerken

BUG: Programmafout. Letterlijk insect (Engels)

BURGER-INFORMATICA: Tak van de informatica die zich bezig houdt met de niet-programmatische kant van de automatisering. Zij legt zich toe op het inzicht in de aard en sturktuur van gegevens, de toepassingen van automatische processen en de maatschappelijke problemen daarbij

BUS: Groep printsporen die tezamen een bepaalde soort informatie overbrengen tussen de CPU, het geheugen en randapparatuur. Met de term wordt ook wel interface bedoeld

BYTE: Groep van 8 bits. Grootste eenheid van informatie die de microprocessor in ML opdracht kan hanteren

CAD: Computer Aided Design, het maken van ontwerpen met behulp van de computer

CAL: Computer Aided Learning: toepassing van de computer in het onderwijs, gebaseerd op de dialoog tussen leerling en programma

CAM: Computer Aided Manufacturing, het door de computer sturen van een fabricageproces

CARRIER: Zie draaggolf

CARTRIDGE: In een plastic doosje gevat printplaatje dat op de uitbreidingspoort kan worden aangesloten. Bevat meestal in ROM of EPROM gezet programma

CASSETTE BUFFER: Buffer voor cassette-operaties

CASSETTE POORT: Aansluiting op de computer voor cassetterecorder

CATALOGISEREN: Opslaan op diskette

CBM: Commodore Bussiness Machines

CENTRONICS: Standaard printer aansluiting. Past niet rechtstreeks op de CBM64, heeft interface nodig

CFD: Compact Floppy Disk, het 3.5. inch type in hard plastic behuizing

CHARACTER ROM: Rom dat de character patronen bevat zoals die op het scherm verschijnen

CHARACTER: Teken. Letter, cijfer, leesteken, spatie of grafisch teken

CHIP: Geïntegreerd circuit waarin een groot aantal transistoren, weerstanden en condensatoren zijn ondergebracht

CIA: IC bouwsteen die onder meer de I/O verzorgt

CMOS: Complementary metaloxide semiconductor. Snel uitleesbaar type geheugen, klein stroomverbruik.

COLD RESET: Herstart waarbij reeds in het geheugen aanwezige informatie verloren gaat, ook wel dead start genoemd

COLLATING SEQUENCE: Volgorde die binnen een bepaalde tekenset wordt aangehouden bij sorteren of vergelijken

COMMANDO: Opdracht die meteen wordt uitgevoerd, een opdracht in direct mode

COMPATIBEL: Het onderling uitwisselbaar zijn van programma's en apparatuur

COMPILER: Programma dat een in een hogere taal geschreven programma in zijn geheel vertaalt naar machinecode. Hierdoor ontstaat er van dat programma

een vertaalde versie waar de computer mee kan werken
COMPLEX INTERFACE ADAPTOR: Zie CIA
COMPOSIET VIDEO: Signaal voor een monitor waarbij de basiskleuren gemengd zijn
CONCATENATIESYMBOOL: Schakelteken, teken waarmee twee strings aan elkaar kunnen worden gekoppeld
CONFIGURATIE: Het hele systeem van computer en randapparatuur
CONNECTOR: Aansluiting, plug
CONSTANTE: In een programma opgenomen informatie, kan zowel tekst als getal zijn
CONTROL CHARACTER: Besturingsteken voor een printer in een af te drukken tekst dat als zodanig herkend wordt en zelf niet wordt afgedrukt
CP/M: Licht verouderd bedrijfssysteem voor zakelijke microcomputers. Kan alleen op Z80 microprocessor draaien
CPU: Central Processing Unit. Min of meer synoniem met microprocessor, de eenheid die zorgt voor het ophalen, decoderen en uitvoeren van de programma instructies
CRASHEN: Het vastlopen van een computer als gevolg van een storing of programmafout
CRT: Cathode Ray Tube, kathode straalbuis, het beeldscherm
CURSOR: Positie aanwijzer op het scherm, die aangeeft waar het volgende teken komt te staan of waar een teken overschreven of verwijderd wordt
CVE: Centrale Verwerkings Eenheid, zie CPU
DAISY WHEEL PRINTER: Letterwiel- of margrietwielprinter, afdrukeenheid die door middel van een hamermechanisme tekens afdrukt die aan de uiteinden van de spaken van het letterwiel zijn bevestigd. De tekens worden in een keer afgedrukt
DATA: Gegevens. Meestal wordt hierbij gedoeld op de in- en uitvoergegevens van een programma, niet op de programmaregels zelf
DATABASE: Programma waarmee (relatieve) gegevensbestanden kunnen worden opgebouwd. Het programma biedt vaak functies als zoeken, sorteren en selecteren. Ook wel elektronische kaartenbak genoemd
DATABUS: Groep van printsporen die tezamen een byte informatie kan omvatten. Wordt voor alle gegevenstransport binnen de computer gebruikt
DATAFILE: Gegevensbestand, te onderscheiden van een BASIC programmabestand
DATALIJNEN: Zie Databus
DATARECORDER: Cassetterecorder die geschikt is om computergegevens te registreren
DATASETTE: Speciale Commodore cassetterecorder voor gegevens- en programma-opslag
DC: Direct Current, gelijkspanning
DD: Double Density, dubbele schrijfdichtheid, op een diskette. Dit is geen standaardmaat, maar is per schijfformaat en systeem verschillend
DEAD START: zie Cold reset
DEBUG: Ontdoen van fouten
DECAY: Uitsterftijd van een toon
DECIMAAL: Tientallig, het meest gebruikte talstelsel
DEFAULT: Zie Verstek
DENSITY: Maat voor de schrijfdichtheid op een diskette. Heeft zowel betrekking op het aantal sporen per zijde als het aantal bytes per sector
DEVICE: Eenheid, stuk apparatuur
DIGIT: Teken dat een geheel cijfer, een discrete waarde voorstelt

DIGITAAL: Representatie van waarden door middel van digits
DIP: Dual In Line Package, de behuizing van een chip met twee evenwijdige rijen pennen waarmee deze op de circuitkaart wordt bevestigd
DIRECT ACCESS: De toegangsmethode waarbij door een verwijzing naar hun positie de toegangstijd onafhankelijk is van de plaats van de gegevens in het geheugen
DIRECT MODE: Toestand waarin de computer zich bevindt als er geen programma loopt. In Direct Mode kunnen de meeste BASIC commando's gebruikt worden. Vgl. program mode
DIRECTORY: Index van de files op een diskette
DISASSEMBLER: Programma dat ML code terugvertaalt naar assemblercode
DISK: Magnetisch opslagmedium voor gegevens en programma's
DISKDRIVE: Randapparaat om disks mee te lezen en te schrijven
DISKETTE: Flexibele schijf met een magnetiseerbaar oppervlak, waarop informatie kan worden opgelsagen
DOS: Disk Operating System, besturingssysteem waarbij met disks wordt gewerkt
DOT MATRIX PRINTER: Zie Matrix printer
DS: Double sided, floppy die aan twee kanten beschrijfbaar is
DUMMY PARAMTER: Een door de interpreter of een programma vereiste parameter waar echter verder niets mee gedaan wordt
DUMMY: Loze variabele of parameter die geen duidelijke functie heeft. Wordt gebruikt om een programma in de pas te houden of tijdens de testfase van een programma
DYNAMISCHE RAM: Vorm van RAM zoals die in de 64 gebruikt wordt. De inhoud moet elke paar milliseconden worden ververst, anders gaat deze verloren
EDITOR: Zie Scherm editor
EDITTEN: Het opmaken van tekst op het scherm door middel van de editor
EPROM: Erasable Programmable Rom: wisbaar programmeerbaar geheugen. Cartridge waarin door de gebruiker zelf programma's kunnen worden opgeslagen. Door blootstelling aan ultraviolet licht kunnen deze weer gewist worden
EXECUTEREN: Een programma runnen
EXPANSION PORT: Zie Uitbreidingspoort
EXPRESSIE: Combinatie van rekenkundige bewerkingen
FILE: Zie Bestand
FIRMWARE: De door de fabrikant in de computer als Rom ingebouwde programma's. Kernal en BASIC interpreter
FLAG: Een variabele in een programma die een bepaalde conditie aangeeft waar later in dat programma op gereageerd wordt; een statusbit in een register van de processor dat een bepaalde conditie registreert
FLATBEDPLOTTER: Type plotter waarbij het vel vlak ligt en de pennen over het papier bewegen
FLIP-FLOP: Schakeling in een register die een informatiebit kan opslaan
FLOATING POINT: Zie Drijvende komma
FLOPPY: Diskette. Oneigenlijk gebruikt ook wel disk drive
FLOWCHART: Zie Stroomdiagram
FONT: Lettertype. Een komplete set

letters, cijfers en symbolen in een bepaalde stijl en grootte.
FORMATEREN: Vastleggen van de sporen en sectoren op een diskette
FSK: Frequency Shift Keying, methode van datatransmissie waarbij 0 en 1 ieder een eigen frequentie hebben
FULL DUPLEX: Methode van datatransmissie waarbij er in beide richtingen tegelijk communicatie plaats heeft. Hierbij wordt vaak een echosignaal teruggezonden, waardoor de verzonden gegevens met de oorspronkelijke vergeleken kunnen worden
FUNCTIE: Vaak al in het systeem aanwezige procedure die bij zijn naam wordt aangeroepen en een bepaalde waarde teruggeeft. B.v. de sinusfunctie. De meeste programmeertalen staan het zelfde definiëren van functies toe
GAP: Tussenruimte tussen twee blokken op een diskette
GEBRUIKERS POORT: Aansluiting achter op de computer waar bijvoorbeeld cartridges op kunnen worden aangesloten
GEHEUGEN: Interne opslag voor data en programma opslag. Het geheugen is verdeeld in bytes die ieder een waarde in het bereik 0-255 kunnen opslaan, ieder byte kan apart geadresseerd worden. Zie ook RAM, ROM en EPROM
GENERATIE: De eerste generatie computers was gebaseerd op het gebruik van buizen, de tweede op transistors. In de derde en de vierde generatie zijn geïntegreerde circuits toegepast
GLARE: Hinderlijke weerschijn op het scherm van lichtbronnen in de kamer
GLITCH: Onverklaarbare storing van de apparatuur
GRAPHICS: Iedere toepassing van een computerbeeld. Wordt vaak specifiek gebruikt voor hoog-oplossend vermogen beelden. Zie ook Bit-mapped
GRID: Raster
HACKER: Computerkraker of computerfanaticus
HALF DUPLEX: Transmissiemethode waarbij er slechts in een richting tegelijk gegevens worden verzonden
HANDSHAKE: Protocol bij datacommunicatie. Deze procedure wordt afgehandeld voor en tijdens gegevens worden verstuurd
HANG-UP: Onvoorziene stop van programma waardoor de computer op slot zit
HARD COPY: Afdruk op papier
HARDWARE: De componenten die tezamen een computer vormen. Wordt ook wel gebruikt voor computer en randapparaten samen.
HEADER: Record dat de identificatie van de er op volgende file op cassette bevat
HERTZ: Trillingsgetal, aantal trillingen per seconde. Wordt onder andere gebruikt om toonhoogte van geluid mee aan te geven
HEXADECIMAAL: Zestientallig. Vaak bij assembler en machinetaal toegepast talstelsel
HIGH BYTE: Byte dat bij een dubbelbyte adressering het meest significante deel van het adres bevat
HIRES: High resolution, met groot oplossend vermogen. Dit houdt een fijne detaillering van het beeld in.
HOGERE TAAL: Programmeertaal die dicht bij door mensen gebezigde taal staat. Door gebruik van een compiler of een interpreter is men hierbij minder afhankelijk van het gebruikte computersysteem
Hz: Afkorting van Hertz, trillingen per seconde
I/O: Afkortingen van Input/Output. Ie-

dere uitwisseling van gegevens tussen de computer en de buitenwereld, dus beeldscherm, toetsenbord, printer, cassetterecorder, diskdrive etc.

IC: Geïntegreerd circuit, combinatie van onderling verbonden schakelelementen

INDENTATION: Het laten inspringen van een regel of paragraaf

INDIRECT MODE: Toestand waarin de computer de ingevoerde opdrachten eerst opslaat als programmaregels en pas na het RUN commando uitvoert

INPUT: Zie Invoer

INSTRUCTIE SET: De verzameling ML commando's die een bepaalde microprocessor kent

INTEGER: Geheel getal, dat geen breukwaardes kan aannemen. Vgl. drijvende komma

INTERACTIEF: Programma waarin gebruiker en programma in voortdurende conversatie met elkaar staan

INTERFACE: Electronische schakeling om twee apparaten met elkaar te verbinden. Bijvoorbeeld om Centronics printer op de 64 aan te sluiten

INTERFACEN: Het op de computer aansluiten van allerlei apparatuur

INTERPRETER: ML programma dat de uitvoering van BASIC programma's verzorgt, programma dat instructies per regel naar machinecode vertaalt en uitvoert. De vertaalde code blijft niet bewaard

INTERRUPT ROUTINE: Een zestig maal per seconde uitgevoerde huishoudelijke routine, die onder meer het toetsenbord leest en de jiffy clock bijhoudt. Onderdeel van het bedrijfs systeem

INVOER: Iedere inbreng van gegevens in de computer

JACK: Plug

JIFFY CLOCK: Een automatisch bijgehouden inwendige klok, die 60 keer per seconde door de interrupt-routine wordt opgehoogd

JITTER: Flikkeringen van het schermbeeld

JOYSTICK: Spelpookje voor de besturing van videogames

JUSTEREN: Het verticaal richten van de linker of rechter kantlijn

K: Afkorting van Kilobyte, eenheid van circa eenduizend bytes (in feite 2^10 = 1024 bytes)

KARAKTER: Symbool uit een bepaalde tekenset

KERNAL: Het bedrijfssysteem van de 64. Wordt ook wel gebruikt als naam voor de sprongtabel die de verschillende Commodore computers onderling (min of meer) uitwisselbaar houdt

KETTINGFORMULIEREN: Vellen papier die met een perforatievouw aaneengesloten zijn en via gaatjes in de zijkanten door een afdrukeenheid gevoerd worden

KEY: Toets; sleutel, een gegevensbestanddeel dat de identificatie geeft van een record bij een bepaalde bewerking

KEYBORD: Toetsenbord

KEYWORD: Sleutelwoord. Een gereserveerd woord dat een opdracht in een programmeertaal vertegenwoordigt

KLOKGENERATOR: Eenheid gestuurd door een kwartskristal dat elektrische pulsjes afgeeft en hiermee voor de timing van de CPU zorgt.

KSB: Kathodestraalbuis

LADEN: Het van cassette of diskette naar het geheugen overbrengen van een programma

LCD: Liquid Cristal Display, plat scherm waarop gegevens worden afgebeeld door het laten oplichten van vloeibare kristallen

LED: Licht Emitterende Diode, een halfgeleiderdiode die licht uitzendt als er stroom in de doorlaatrichting doorvloeit
LICHTPEN: Een randapparaat dat reageert op de TV of monitor. Kan o.a. worden gebruikt om op het scherm te tekenen.
LISTING: Een afdruk op papier van de regels van een programma
LOGICAL SEEKING: Techniek waarbij de schrijfkop van een afdrukeenheid steeds de snelste weg naar de plaats van het volgende af te drukken teken zoekt, hetgeen tijdwinst oplevert
LOW BYTE: Byte dat bij een dubbelbyte adressering het minst significante deel van het adres bevat
LSI: Large Scale Intergrated Circuit: chip met duizenden schakelelementen
LUMINANTIE: Helderheid
MACHINECODE: Zie Machinetaal
MACHINETAAL: De codes die door de microprocessor worden uitgevoerd. Iedere groep van 1-3 bytes staat voor 1 opdracht
MACRO: Set instructies met een symbolische naam. Bij de verwerking wordt deze naam vervangen door de set instructies. Het programma wordt hierdoor geëxpandeerd
MAIN FRAME: Zeer snelle computer met grote geheugencapaciteit. Een dergelijk systeem geeft meestal toegang aan vele tientallen gebruikers tegelijk
MANUAL: Handboek
MARGRIETWIEL: Zie Daisy Wheel
MATRIX PRINTER: Printertype dat de af te drukken tekens vormt door een reeks pennetjes tegen een inktlint te slaan
MATRIX: Zie Array
MEGABYTE: Eenheid van circa 1 miljoen bytes (in feite 2^20 = 1048576 bytes)
MEMORY MAPPED I/O: I/O die via adressen in het geheugen verloopt. Als zo'n adres op de adresbus verschijnt zal de informatie op de databus niet van of naar het geheugen gaan, maar naar bijvoorbeeld een register van een CIA. De microprocessor benadert de I/O alsof het gewone geheugenadressen zijn.
MEMORY: Zie Geheugen
MENU: Op het scherm afgebeelde lijst van mogelijke programmawendingen waaruit geselecteerd kan worden
MENUGESTUURD: Programma waar in plaats van rechtstreekse opdrachten menu's gebruikt worden
MFD: Mini Floppy Disk, het veel gebruikte 5.25 inch type; Micro Floppy Disk, 3 inch floppy
MICRO: Een computer waarvan de CPU zich op een enkele chip bevindt
MICROPROCESSOR: Een op een enkel IC ondergebrachte computer rekenen stuureenheid
ML: Zie Machinetaal
ML: Machine Language, machine taal. De code die de computer kan interpreteren. Met de term wordt ook wel assembleertaal bedoeld
MNEMONIC: Afkorting die als geheugensteuntje moet dienen, meestal drieletterig. Vooral assembler maakt hier veel gebruik van
MODEM: Samentrekking van modulator/demodulator. Eenheid voor datacommunicatie die digitale gegevens omzet in een analoog signaal en omgekeerd
MODULE: Bouwsteen, zowel voor soft- als hardware
MODULEREN: Het voor transmissie geschikt maken van de signalen van de computerapparatuur
MONITOR: Hulpprogramma om machinetaal programma te disassembleren alsmede geheugeninhouden te bekijken en te wijzigen. Geavanceerde monitorprogramma's kunnen daarnaast nog

veel meer functies bieden. Beeldbuis met grote bandbreedte; programma waarmee geheugenplaatsen gelezen en direct gewijzigd kunnen worden.

MONOCHROOM: Eenkleurig (met betrekking tot een monitor)

MS-DOS: Microsoft Disk Operating System, besturingssysteem voor pc's

MUIS: Apparaat waarmee uit het menu opties op het scherm kunnen worden gekozen

NANOSECONDE: Een miljardste seconde

NESTEN: Bij programmeren het insluiten van een bepaalde struktuur van hetzelfde type, b.v. een lus binnen een lus

NIBBLE: Aaneengesloten groep van 4 bits

NLQ: Near Letter Quality. Afdruktechniek van een matrixprinter waarbij een regel nogmaals, iets verschoven, geprint wordt. De stipjes sluiten zich hierdoor aaneen zodat een mooiere print onstaat

NUMERIC PAD: Eiland op toetsenbord met extra numerieke toetsen

NUMERIEK: Een variabele of constante die alleen maar een getal kan bevatten

OBJECT-CODE: Binaire kode, door compiler of assembler vertaald programma dat door de computer kan worden verwerkt

OCTAAL: In het achttallig stelsel

OEM: Original Equipment Manufacturers, fabrikanten die merkloze apparatuur leveren die de afnemer onder eigen naam verkoopt
Vaak in het Nederlands vertaald als: Onder Eigen Merk

OFF-LINE: Niet in verbinding staand met een centrale computer, niet gekoppeld aan een netwerk

ON-LINE: In rechtstreekse verbinding met een centrale computer of een databank, b.v. Viditel

OPERAND: Grootheid waarop een bewerking wordt uitgevoerd

OPERATIE: Bewerking, programmastap

OPERATING SYSTEM: Zie Bedrijfssysteem

OPERATOR: Bewerkingsteken. Symbool dat aangeeft welke wiskundige bewerking verricht moet worden

OUTPUT: Zie Uitvoer

OVERFLOW: Het overschrijven van de geheugencapaciteit

OVERLAY: Deel van een programma dat op een zeker moment bijgeladen wordt en daarbij de plaats inneemt van een ander deel dat dan niet meer nodig is

PAL: Phase Alternating Line. Standaard kleursysteem dat voor een normale KTV wordt gebruikt

PARALLEL: Het gelijktijdig verwerken van 8 bits

PARAMETER: Een bij een commando of functie meegegeven waarde. Kan zowel numeriek als alfanumeriek moeten zijn

PARITEIT: Foutenopsporings-techniek waarbij in een aparte bit aangegeven wordt of het aantal enen in een woord even of oneven is

PARSE: Het syntactisch ontleden van een opdracht

PARSER: Het programma dat voor de parsing zorg draagt

PC: Personal computer

PERIFERIE: Zie Randapparatuur

PERIPHERALS: Zie Randapparatuur

PITCH: De printdichtheid, het aantal te printen characters per inch

PIXEL: Een enkel beeldpuntje op het

scherm
PLATEN: (Engels) de schrijfrol op een printer
PLOTTER: Een door een computer bestuurde tekenatomaat
POINTER: Adreswijzer naar de lokatie van een gegeven
POLLING: Navraag doen. Het beurtelings navragen door de CPU van eenheden om na te gaan of deze gegevens willen sturen. Te onderscheiden van een interrupt
POORT: Een aansluitmogelijkheid op een computer
PRECISIE: Mate van nauwkeurigheid waarmee een variabele of konstante verwerkt wordt. Aantal bytes dat een grootheid in het geheugen krijgt toegewezen
PRINTER: Een apparaat dat uitvoer op papier produceert
PROCESSOR: De eenheid die de programma instructies leest, interpreteert en uitvoert
PROGRAM COUNTER: Zie Programma teller
PROGRAMMA TELLER: Intern register van de microprocessor, waarin het adres van de uit te voeren ML instructie wordt bijgehouden
PROGRAMMA MODE: Toestand waarin de computer zich bevindt tijdens het uitvoeren van een programma. Vgl. direct mode
PROMPT: Teken van een programma dat de vorige instruktie is verwerkt en dat de gebruiker opnieuw iets mag invoeren
PROPORTIONEEL SCHRIFT: Afdruktechniek waarbij elk teken zijn eigen breedte heeft, in tegenstelling tot de gewone schrijfmachine waar elk teken dezelfde breedte heeft
PROTOCOL: De regels waaronder gegevens tussen twee systeemcomponenten worden uitgewisseld
PUBLIC DOMAIN: Publiekelijk bezit, programma waarvan de copyrights zijn vrijgegeven
QUEUE: Wachtrij
QUICK DISK: Eenheid voor opslag van gegevens. Werkt met diskettes waarop gegevens sequentieel worden opgeslagen
RAM: Random Access Memory. Computergeheugen waarin tijdelijke programma's en gegevens kunnen worden opgeslagen. Het geheugen dat gelezen of beschreven kan worden
RAMPACK: Geheugenuitbreidingskaart, extra RAM
RAMSCHIJF: Stuk werkgeheugen dat wordt gebruikt alsof het een disk is
RANDAPPARATUUR: Alle aan de computer gekoppelde apparaten, zoals diskdrive, printer en cassetterecorder
RANDOM ACCESS: Toegangsmethode waarbij elke geheugenlokatie via een adreswijziging direct toegankelijk is
RANDOM FILE: Datastruktuur waarbij de records zich in willekeurige volgorde bevinden. De records worden via de direct access methode opgehaald of opgeslagen
RASTER: Het coördinatenstelsel van adresseerbare beeldschermpuntjes
REAL TIME: De werkwijze waarbij gegevens worden verwerkt zodra ze zijn ingevoerd. De interactieve methode
RECORD: Een bij elkaar behorend blok informatie, bijvoorbeeld de gegevens van 1 klant in een klantenbestand; Engels voor opnemen, toets op de cassetterecorder
REËEL GEHEUGEN: In fysieke zin van het hoofdgeheugen
REFERENCE MANUAL: Uitgebreid naslagerk waarin alle aspecten van een bepaald systeem belicht worden

REGISTERS: Plaatsen in IC bouwstenen waar allerlei informatie kan worden opgeslagen. Sommige registers kunnen alleen gelezen worden, andere alleen beschreven, weer andere zowel gelezen en geschreven. Zie ook memory mapped I/O
REKENMATRIX: Spreadsheet
RELATIVE FILE: Random file
RELEASE: Uitsterftijd van een toon
REMOTE CONTROL: Op afstand bestuurbaar
RESET: Herstart. Kan zowel softwareals hardwarematig teweeg worden gebracht
RESOLUTIE: Oplossend vermogen of beeldpuntdichtheid, maat voor de scherpte van het beeld
RETRIEVE: Het opzoeken van informatie op een informatiedrager
RETURN: Het afsluiten van een programmaregel of opdracht (ook Enter); het retourneren van een waarde door een functie of andere routine
REVERSE VIDEO: Omgekeerd weergeven van voor- en achtergrondkleur
RF: Radio Frequency
RGB: Rood Groen Blauw, het gescheiden overbrengen van de 3 basiskleuren, waardoor een grotere beeldscherpte en betere kleurverzadiging bereikt wordt
RIGHT JUSTIFICATION: Het creëren van een rechter kantlijn, b.v. door het uitvullen van de regels
ROM: Read Only Memory. Geheugen dat alleen gelezen kan worden en zijn inhoud behoudt na het uitzetten van de computer
ROMPACK: Insteek module, ROM-cartridge
RS 232: Gestandaardiseerde verbinding tussen computer en randapparatuur. Vereist interface
SAVEN: Het wegschrijven van een programma uit het geheugen naar cassette of diskette
SCANNING: Het aftasten van een bepaalde eenheid b.v. het toetsenbord
SCHADUWBEELD: Sprite
SCHERM EDITOR: Deelprogramma in bedrijfssysteem dat de opmaak-functie op het beeldscherm verzorgt
SCHERM GEHEUGEN: Geheugenblok dat de gegevens zoals die op het scherm te zien zijn bevat. Adres en grootte kunnen veranderen bij de diverse scherm-modi
SCHERM-MODUS: Instelling van het beeldscherm. Behalve standaardtekst kent de 64 o.a. bit-mapped
SCHOOTCOMPUTER: Draagbare, volwaardige computer met toetsenbord en LCD scherm
SCREEN EDITOR: Zie Scherm editor
SCREENDUMP: Het afdrukken van de beeldscherm-inhoud op een printer
SCREENING: Het checken van gegevens op mogelijke tegenstrijdigheden
SCROLLING: Het over het scherm heen bewegen van tekst of graphics. Bijvoorbeeld het van onder naar boven rollen van een programmalisting
SD: Single Density. Maat voor de schrijfdichtheid op een diskette
SECONDAIR ADRES: Extra parameter die bij een opdracht naar een randapparaat kan worden meegegeven
SECTOR: Gedeelte van een spoor op diskette dat een gegevensblok kan bevatten
SEQUENTIAL FILE: Bestand waarin de volgorde van de records overeenkomt met de volgorde waarin deze zijn opgeslagen. De toegangstijd tot de gezochte gegevens is afhankelijk van de positie in de file
SEQUENTIEEL: Het in een bepaalde

volgorde plaatsvinden van een bewerking
SERIEEL: Bit voor bit overdracht van gegevens
SHADOWPRINTING: Bij een afdrukeenheid het nogmaals, een fractie ernaast, afdrukken van een teken, zodat een donkerder print ontstaat
SHEETFEEDER: Hulpstuk bij een afdrukeenheid dat zorgt voor het toevoeren van losse vellen papier uit een invoermagazijn
SHELL-SORT: Redelijk snel sorteeralgoritme
SID: Geluids bouwsteen voor de 64. In feite primitief synthesizer-IC
SKIP: Het overslaan van instrukties
SMOOTH SCROLLING: Scrolling effect waarbij de beeldinformatie één beeldpuntje per keer verschuift. Geeft een glijdend effect, illusie van beweging
SOFTWARE: Verzamelnaam voor programmatuur. Alles dat niet hard in de computer is ingebouwd
SOUND INTERFACE DEVICE: Zie SID
SOURCE CODE: Broncode, de instructies waarin de programmeur een programma schrijft
SPREADSHEET: Calculatieprogramma, waarin matrices doorberekend worden. Wijzigingen van de inhoud van een cel leidt steeds tot aanpassing van randtotalen, percentages enz
SPRITE: Figuurtje waaraan zelf vorm kan worden gegeven door een puntjespatroon te definiëren. De sprites kunnen met een bepaalde onderlinge voorrang over het scherm bewogen worden
STACK POINTER: Intern register van de microprocessor dat de eerstvolgende vrije positie op de stack aangeeft
STACK: Geheugengedeelte dat gereserveerd is voor tijdelijke gegevensopslag, bijvoorbeeld returnadressen na subroutine aanroep
STATEMENT: Programma opdracht in indirect mode
STATUS REGISTER: Intern register van de microprocessor dat per bit allerlei informatie over de laatste ML opdracht aangeeft
STATUSBALK: Kader op het scherm waarin bepaalde condities van een programma zijn af te lezen
STRING: Tekst. Iedere vorm van alfanumerieke informatie in een programma
STRING FLOPPY: Snel spoelend, eindeloos cassettebandje
STROBE: Bij datacommunicatie een kloksignaal dat voor de timing van de signalen tussen computer en randapparaat zorgt
STROOMDIAGRAM: Een tekening die de logica en de samenhang van een programma of -systeem weergeeft
SUBROUTINE: Min of meer op zichzelf staand deel van een programma dat vanuit verschillende punten in het hoofdprogramma aangeroepen kan worden
SUBSCRIPT: De indicering van een array-variabele; het een halve regelpositie omlaag afdrukken van een teken als b.v. in H_2O
SUPERSCRIPT: Het een halve regelpositie omhoog afdrukken van een teken als b.v. in x^2
SUSTAIN: Toon-niveau aanhouden
SWITCH: Schakelaar
SYMBOLISCHE CODE: Assembleertaal
SYNCHROON: Communicatiemethode waarbij gegevens worden verzonden met vaste frequentie en faseverhouding
SYNTAX: De grammaticale regels van een programmeertaal
SYSTEEM VARIABELEN: Door het

bedrijfssysteem en de interpreter gebruikte in RAM staande gegevens
TEKEN-GENERATOR: Zie Character Rom
TEKEN-ROM: Zie Character Rom
TEKST-MODE: Scherm mode waarin het beeldscherm primair voor tekstbedrijf ingericht is
TEKSTVERWERKER: Programma om tekst op te stellen. De tekst wordt eerst in het geheugen opgeslagen, zodat zij verder kan worden bewerkt alvorens naar een afdrukeenheid te worden gestuurd of permanent te worden opgeslagen
THERMISCH PAPIER: Papiersoort waarop een thermische printer de tekens kan inbranden
THERMISCHE PRINTER: Printer die door plaatselijke verhitting tekens afdrukt op speciaal geprepareerd papier
TIME-SHARING: Methode waarbij meerdere gebruikers tegelijkertijd van een computersysteem gebruik maken
TOEGANGSTIJD: De tijd die de computer nodig heeft om na een leesopdracht de gevraagde gegevens voor verdere verwerking beschikbaar te hebben
TOGGLE SWITCH: Toets waarmee tussen twee stabiele toestanden gewisseld kan worden
TOKEN: Inwendige 1 byte code die voor een BASIC commando of functie staat
TOKENIZED FILE: Door gebruik van tokens gecomprimeerde file
TOONGENARATOR: Geluids-chip
TOP-DOWN PROGRAMMEREN: Gestruktureerd programmeren waarbij een zekere hierarchie van subroutines wordt aangehouden
TOUCH SCREEN: Beeldscherm waarop door aanraking met de vinger menu's kunnen worden geselecteerd
TRACK: Cirkelvormig spoor op een diskette. Op een quick disk schijfje een spiraalvormig spoor
TRACTOR: Hulpstuk bij een afdrukeenheid dat voor de doorvoer van kettingformulieren zorgt
TRANSMISSIE: Het verzenden van een signaal of data
UHF: Ultra High Frequency, frequentie van 300 tot 3000 Mhz, de TV kanalen 14 tot en met 83
UITBREIDINGSPOORT: Aansluitmogelijkheid achter op de computer waarop bijvoorbeeld RS 232 interfaces kunnen worden aangesloten
UITVOER: Elke vorm van informatieoverdracht van de computer naar de buitenwereld, b.v. beeldscherm, printer
UNFORMATTED: Nog niet geformatteerd
UPDATE: Opschonen, bijwerken van gegevens bestanden
USER-PORT: Zie Gebruikerspoort
UTILITY: Hulpprogramma, handige routine die het werken met andere programma's vergemakkelijkt
VARIABELE: Een grootheid die een bepaalde waarde kan aannemen. Bij het programmeren een symbolische naam die naar een waarde verwijst
VDP: Video Display Processor: chip die voor de beeldschermopbouw zorgt
VERIFY: Het controleren van een bewerking. Het na een schrijfoperatie nogmaals lezen van de gegevens, waarbij deze vergeleken worden met die in het geheugen
VERSTEK: Een door de computer ingevulde waarde voor een parameter die wordt gebruikt als er geen waarde is opgegeven
IC-II: Video IC zoals die in de 64 gebruikt wordt. Alle informatie op het

scherm wordt door dit IC verzorgt
VIRTUEEL GEHEUGEN: Schijngeheugen. Extern geheugen dat zich aan de gebruiker als werkgeheugen voordoet, d.w.z. adresseerbaar is
VLSI: Very Large Scale Integration circuit: circuit met zeer hoge integratie (meer dan een miljoen schakelelementen op een vierkante centimeter)
VLUCHTIG GEHEUGEN: Geheugen waarvan de inhoud verloren gaat als de voeding wordt onderbroken, RAM
WAARDE: Een grootheid die aan een constante of variabele wordt toegekend
WAFER: Siliciumschijf, prefabricaat van een chip
WARM RESET: Een herstart waarbij eerder in het geheugen ingevoerde gegevens niet verloren gaan. Ook wel warme start
WILD CARD: Jokerteken. Symbool dat de plaats van een willekeurig ander teken of reeks tekens inneemt
WOORD: Reeks van bits waarmee door de computer als eenheid wordt gewerkt en in een geheugenplaats kan worden opgeslagen
WORD-PROCESSOR: Tekstverwerker
WRAP-AROUND: Het weer in beeld laten verschijnen van de cursor, tekst of sprite aan de zijde tegenover die waar zij eerder uit het beeld verdwenen
WRITE PROTECT: De mogelijkheid een diskette te beschermen tegen een schrijfopdracht
X-REGISTER: Intern register van de microprocessor
Y-REGISTER: Intern register van de microprocessor
ZERO-PAGE: Ram geheugen met de adressen 0-255. Wordt bijna geheel gebruikt voor diverse systeem-variabelen, daar het sneller te adresseren valt dan de rest van het geheugen
1541: Commodore diskdrive, speciaal bedoeld voor gebruik met de 64
6510: Typenummer van de in de 64 toegepaste microprocessor. Deze is qua instructieset geheel gelijk aan de 6502

VOORWOORD BIJ DE PROGRAMMALISTINGS

In dit naslagwerk worden alle programmalistings precies zo gepubliceerd als ze er op het scherm uitzien. Dit betekent dat in de listings bepaalde besturingstekens voor kunnen komen. Deze tekens hebben de volgende betekenis.

De beide sets hebben betrekking op de twee karakter-sets die zich in de Commodore-64 bevinden. Is een listing in hoofdletters geschreven, dan geldt set 1. Is de listing in kleine letters geschreven dan is set 2 van toepassing.

Functie	SET1	SET2
Cursor links		
Cursor rechts		
Cursor laag		
Cursor hoog		
Home		
Clear		
Functietoets 1		
Functietoets 2		
Functietoets 3		
Functietoets 4		
Functietoets 5		
Functietoets 6		
Functietoets 7		
Functietoets 8		
Zwart (CTRL-1)		
Wit (CTRL-2)		
Rood (CTRL-3)		
Cyaan (CTRL-4)		
Paars (CTRL-5)		
Groen (CTRL-6)		
Blauw (CTRL-7)		
Geel (CTRL-8)		
Oranje (Commodore-1)		
Bruin (Commodore-2)		
Lichtrood (Commodore-3)		
Grijs 1 (Commodore-4)		
Grijs 2 (Commodore-5)		
Lichtgroen (Commodore-6)		
Lichtblauw (Commodore-7)		
Grijs 3 (Commodore-8)		
Reverse aan (CTRL-9)		
Reverse uit (CTRL-0)		

2/2

Trefwoordenregister

Dit trefwoordenregister is een aanvulling op de inhoudsopgave. Het stelt u in staat snel plaatsen te vinden, waar iets wordt gezegd over bepaalde onderwerpen.

De cijfercombinaties achter de woorden geven deel, hoofdstuk en bladzijde aan, waar het bepaalde woord of onderwerp ter sprake komt. Bijv: 11/1.2-2 betekent deel 11 hoofdstuk 1.2 pagina 2.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

3

Computers en samenleving

Inhoud

[1]) Dit hoofdstuk heeft een eigen inhoudsopgave.

Moderne microcomputers zijn meer dan alleen maar technisch speelgoed. Natuurlijk is de homecomputer inderdaad het speeltuig bij uitstek, niet voor niets worden er zoveel spelprogramma's op de markt gebracht. Maar met de opkomst van de homecomputer voltrekt zich een revolutie, de informatie-revolutie. Computers, groot en klein, worden steeds belangrijker in de wereld. Op steeds meer gebieden worden ze ingezet, tot in wasmachines en televisietoestellen aan toe. Dit, gecombineerd met de opkomst van de gegevensnetwerken, zal in de nabije toekomst steeds duidelijker zijn stempel op de maatschappij gaan drukken. Daarover zal het gaan in dit hoofdstuk. Wat zijn de gevolgen van de introductie van de homecomputer, wat voor effect heeft het dat er stees meer computers bij de mensen thuis staan.

Ooit – en dat is nog niet eens zo lang geleden – dacht men in de Verenigde Staten dat er misschien ooit wel twee computers nodig zouden zijn in dat land. Eentje aan de oostkust en eentje aan de westkust, zodat ieder deel van Amerika zijn eigen computer zou hebben. De computers die men toen in gedachten had bezaten minder mogelijkheden dan de Commodore 64, waarvan er intussen alleen al in Nederland meer dan tweehonderdduizend staan.

Zo'n ontwikkeling blijft niet zonder gevolgen, dat valt tegenwoordig regelmatig ook in de krant te lezen. Inbraken in computers komen regelmatig in het nieuws, want de daarbij gestolen informatie kan kapitalen waard zijn. Bedrijfsgeheimen, of allerlei gevoelige gegevens over mensen zoals die in overheidscomputers opgeslagen zijn hebben een ontzaglijke waarde. Van dat soort computermisdaden komt overigens maar een klein gedeelte aan het licht, waarschijnlijk is de omvang van dat soort zaken veel en veel groter dan wie dan ook weet. Om over andere mogelijkheden, zoals het inbreken in defensie-computers maar te zwijgen. Een film als 'Wargames', waar een computerkraker per ongeluk bijna de derde wereldoorlog doet losbranden, is natuurlijke pure fictie. Hopelijk althans.

3/1

Kennis is macht

Inhoud

Een uitspraak die al zo lang als de mens bestaat waar is, kennis is macht. Maar gedurende de laatste tientallen jaren worden de mogelijkheden van de mens om kennis, informatie, te hanteren steeds groter. Computers stellen ons in staat om ontzaglijk grote hoeveelheden informatie op een zinnige manier te gebruiken.
Daar zitten zowel voor- als nadelen aan, zoals we zullen zien.

3/1.1

De grote automatisering

Iedere stad in de wereld heeft wel een afdeling Ruimtelijke Ordening, waar ambtenaren proberen de groei van de stad in goede banen te leiden. Ze zoeken uit waar er huizen gebouwd kunnen worden en hoe die woningen bereikbaar gemaakt kunnen worden. Want een woonwijk zonder goede verbindingen met andere delen van de stad is onleefbaar, mensen moeten naar hun werk, boodschappen doen en op bezoek. Zo'n wijk moet op de een of andere manier dan ook weer aansluiten op de rest van het wegen- en stratennet in en rond de stad. Dergelijke besluiten kunnen op twee manieren genomen worden, gevoelsmatig of op grond van feiten, informatie. Meestal zal zo'n besluit uiteindelijk op een mix van die twee genomen worden, waarbij ook allerlei politieke belangen meespelen.
De soort informatie waar het om gaat is velerlei. Zo is bijvoorbeeld de capaciteit van het bestaande wegennet van belang, door een buurtstraatje kunnen nu eenmaal minder auto's per uur dan over een vierbaansweg. Maar met alleen die capaciteit is men er nog lang niet, de huidige belasting is ook van belang. Hoeveel kunnen er nog bij, dat is eigenlijk de vraag. Bovendien moet men ook nog proberen in te schatten hoe vaak en waarheen de bewoners van de nieuw te bouwen buurt zich zullen gaan verplaatsen. Als de verbindingen met het stadscentrum goed zijn is dat nog geen verzekering dat ook de verbindingen met het industrieterrein in orde zijn. Als de mensen tenminste daar werken, misschien werkt het merendeel juist op kantoor. En hoe ziet het er over tien jaar uit, is er dan ook nog genoeg wegcapaciteit voor al die bewoners? Vragen genoeg, dat wel. En ze zijn ook zeker allemaal te beantwoorden, door allerlei onderzoeksmethodes zijn de gegevens wel te verzamelen. Het verkeer kan geteld worden, de wegcapaciteiten berekend. Iedere grote stad houdt zo nu en dan enquêtes, waarmee juist vragen zoals 'hoeveel verplaatsingen en waarheen' beantwoord kunnen worden. Maar het verwerken van al die informatie is de flessehals. Om zo'n soort vraagstelling met de hand uit te werken is een werkelijk onvoorstelbaar karwei.
Een enquête omvat al gauw tientallen vragen, nog afgezien van alle andere informatie die uit die vragen weer kan worden afgeleid. En dan zijn er geen tien of honderd personen geënquêteerd, maar vaak duizenden.
Het tellen van verkeer is ook al een hele klus. Tot voor enige jaren gebeurde dat met de hand - vooral in universiteitssteden een bekende bijverdienste voor studenten - of met mechanische tellers. Al die telgegevens moesten dan weer worden uitge-

werkt, alweer handmatig.
Het inschatten hoe de situatie zich over een aantal jaren zou ontwikkelen was een vorm van geïnspireerd gokwerk. Eens diep nadenken, met wat collega's overleggen en, misschien, een paar statistische berekeningen op de achterkant van een envelop.

Op een dergelijke manier werden allerlei stadsuitbreidingsplannen ontwikkeld, waarvan sommige heden ten dage de nachtmerrie van de planologen zijn geworden. Verkeerde inschattingen en te weinig feitelijke informatie, omdat juist het vergaren en verwerken van die gegevens met de toenmalige methodes niet te doen was.

De komst van de computer heeft daar grote veranderingen in gebracht. Het verwerken van allerlei enquête-gegevens is gesneden koek voor de computer, er bestaan gespecialiseerde programmapakketten voor. Ook verkeerstellingen kunnen door computers worden verricht. Door het leggen van telslangen, die met luchtdruk werken, of inductielussen, die het metaal van een auto kunnen 'voelen', kan volledig geautomatiseerde meetapparatuur de drukte op een weg vastleggen. Alleen fietsers en voetgangers moeten nog met de hand geteld worden. Deze gegevens worden dan langs elektronische weg in de grote computers gebracht, waar de verdere verwerking snel en foutloos plaatsvindt.

Daarnaast is er ook een geheel nieuw instrument ontwikkeld, de computersimulatie. In een programma wordt een model opgebouwd van de hoofdwegen in en rond een stad. Van die wegstukken liggen allerlei zaken vast, zoals capaciteit, gemiddelde rijsnelheid etcetera. Dit model is onderverdeeld in een groot aantal kleine gebiedjes, die meestal overeenkomen met stadswijken. En van al die gebiedjes is bekend, op grond van tellingen en enquetes, hoeveel auto's of fietsers er in het spitsuur vertrekken en naar welk ander gebiedje ze gaan. Kortom, in de computer is een zo volledig mogelijk beeld opgebouwd van de stad en zijn verkeersstro en althans, verkeersstromen gezien in termen van herkomst en bestemming. Want dat is de grote truc: de computer besluit welke routes al die auto's gaan volgen. En wel op dezelfde grond als de automobilist dat zelf ook doet, er wordt gekeken naar de kortste rijtijd. Op grond van die berekeningen wordt de verkeersdruk op de wegen in het model bepaald. Desgewenst wordt daar dan nog een kaart van getekend ook, op een plotter.

Een dergelijk instrument is de planologenspeeltuin bij uitstek. Want nu kan het effect van allerlei veranderingen in de stad eerst door het model worden berekend. Wat gebeurt er precies als er een brug wegens renovatie een half jaar afgesloten moet worden? Waarheen wijken de verkeerstromen uit? Een simpele vraag, de verandering wordt in het model doorgevoerd waarna het nieuwe verkeersbeeld berekend kan worden. De knelpunten komen al aan het licht nog voor de eerste drilboor in het wegdek gezet wordt. Allerlei toekomstprojecties zijn zo ook te maken. De model-gegevens wat betreft het aantal ritten en hun herkomst en bestemming zijn deels berekend op grond van allerlei sociale en economische tendensen. Door die tendensen naar de toekomst door te trekken kan zo'n belastingstabel ook worden opgesteld voor zeg over tien jaar. Alweer, de knelpunten die dan

zullen ontstaan zijn nu al door het model te voorspellen.

Maar alles heeft zijn prijs. Ten eerste financieel, het volledig doorrekenen van het model voor bijvoorbeeld een stad als Amsterdam kost alleen al aan computergebruik vele duizenden guldens. De specialisten die zo'n model bedienen zijn ook niet goedkoop en een grote modelwijziging kost vele uren werk. Modellen kunnen niet overal voor ingezet worden, daar zijn ze te duur voor. Nog ernstiger is het risico van fouten. Veel mensen denken dat computerberekeningen feilloos zijn. Op zich is dat waar, maar de uitkomsten kunnen nooit betrouwbaarder zijn dan wat er in gestopt is. Een foutje in de definitie van het model en..... Temeer daar de politici die de uiteindelijke beslissingen nemen ook denken dat computers geen fouten maken. De uitkomsten worden voor zoete koek aangenomen. Maar het ergste is toch wel dat hier inderdaad blijkt dat kennis, informatie, macht is. Inspraak van betrokkenen wordt overstemd door met computeruitdraaien wapperende ambtenaren. Het vroegere beslissen op grond van inzicht en ervaring is vervangen door kille feiten, die voor andere opinies geen ruimte laten. De computer geeft het besluit al aan, voor discussie en inbreng is geen oor. De informatie is alles bepalend, wie daar toegang toe heeft bezit inderdaad macht. En zo ontstaat er een soort informatie-elite.

3/1.2

Homecomputer en databank

Gelukkig snijdt het mes aan twee kanten. Kennis is macht, maar hoe kan kennis worden verworven? Om informatie te krijgen, maar ook om informatie te verspreiden, zijn niet zoveel manieren. Van mond tot mond heeft een beperkte reikwijdte en meestal vertelt de derde persoon al een heel ander verhaal dan de eerste. De drukkunst heeft vele eeuwen lang verreweg de belangrijkste rol gespeeld in de informatieverspreiding. Het feit dat iedere regering vroeger of later censuur toepast spreekt boekdelen wat dat betreft. Tegenwoordig spelen radio, televisie en andere elektronische media ook een grote rol. Het nadeel van deze media is echter dat de informatie vluchtig is, tenzij iemand het toevallig opneemt. Bovendien is men afhankelijk van wat er maar net wordt uitgezonden. Wat dat betreft blijft het gedrukte woord verreweg het belangrijkst, men kan het nog altijd eens terugzoeken en nalezen.

Maar het in eerste instantie vinden van een bepaald feit is een probleem met boeken en ander drukwerk. Er is zo veel gedrukt materiaal, dat het netjes opbergen daarvan tegenwoordig al een beroep op zich is. Wie zich ooit eens in een werkelijk grote bibliotheek gewaagd heeft om daar een bepaald feit op te diepen weet er alles van. Het vinden van dergelijke kennis is ook al een specialistenklus geworden.

Dat is dan ook precies de rol van de computer in dit verhaal. Al vele jaren wordt allerlei informatie in speciale computersystemen opgeslagen, de zogenaamde databanken. Het raadplegen van een databank is vrij simpel, vaak zijn er alleen wat trefwoorden nodig om het gewenste feit op het beeldscherm te krijgen. Soms vergezeld van een gerichte verwijzing naar een of enkele boeken.

Wie met zo'n databank wilde werken moest in het verleden er zelf naartoe, de beeldschermen stonden in hetzelfde gebouw of misschien een straat verderop. Daarna brak er een tijd aan dat men met speciale, dure apparaten de databank kon gebruiken via de telefoonlijn, maar het bleef voorbehouden aan beroepsmensen. De kosten waren hoog, zowel van de apparatuur als van het eigenlijke raadplegen van een databank.
Daar begint nu verandering in te komen. In plaats van een dure terminal kan een simpele 64 ook worden ingezet om met een databank te communiceren, mits er een modem en een communicatie-programma voorhanden zijn.

Ook aan de andere kant wordt daar op ingespeeld. Databanken worden voornamelijk overdag gebruikt, tijdens kantooruren, 's Avonds was er tot voor kort weinig belangstelling voor, die tijd was feite-

1.2 Homecomputer en databank

lijk leegloop. En dát is nu precies de tijd van de dag dat de meeste hobbyïsten van een databank gebruik willen maken. De computer-hobbyïst is dus ook commercieel hoogst interessant voor de bedrijven die databanken exploiteren. Zo interessant zelfs, dat er in de Verenigde Staten al meerdere databanken zijn opgericht die zich alleen maar op die hobby-gebruiker richten.

Met de mogelijkheid om zo'n databank te gebruiken, gewoon via de telefoon, kan iedereen de werkelijk formidabele hoeveelheid informatie in de databanken gaan gebruiken. Het opzoeken van de gewenste informatie is een fluitje van een cent, een kind kan de was doen. De kennis die zo voor iedereen toegankelijk wordt is echter alles behalve kinderspel en: kennis is inderdaad macht.

In de toekomst zullen deze ontwikkelingen nog veel verder gaan. Tot nog toe was het gebruik van de databank (relatief) duur, omdat de markt klein was. Hoe meer mensen in staat zijn om de netwerken te gebruiken, hoe goedkoper zowel gebruik als verbinding zullen worden. Want ook daar staat de techniek niet stil. Datacommunicatie kan weliswaar over het gewone telefoonnet, maar er zijn betere oplossingen denkbaar. Speciale netwerken, waarover vele tientallen computerverbindingen tegelijk lopen, met aparte schakel- en verbindingscomputers die er voor zorgen dat de verzonden informatie op de juiste plek aankomt.

Toekomstmuziek? Nee, alles behalve. Multinationals als IBM hebben al jaren dergelijke netten wereldwijd opgezet voor eigen gebruik. Allerlei telefoondiensten zetten ze tegenwoordig eveneens op, ook in Nederland. Maar, alweer, tot het aantal gebruikers stijgt is het niet echt goedkoop. In de VS is men ons al ettelijke stappen voor, daar hebben de databanken, ook de op hobbyïsten gespecialiseerde, in allerlei steden een lokaal telefoonnummer. Opbellen naar de databank kost de gebruiker dus geen scheppen vol geld, het is een lokaal gesprek. Achter dat telefoonnummer schuilt dan wat apparatuur, die alle verbindingen die via dat nummer gelegd worden concentreert en dan via een gespecialiseerd computernet naar de centrale computers sturen, die duizenden kilometers verderop kunnen staan. Al met al is dit veel goedkoper en betrouwbaarder dan rechtstreeks opbellen.

Deze ontwikkelingen zullen hier ook komen. Wellicht sneller dan de meeste mensen verwachten zal de maatschappij een heel nieuw soort democratisering mee gaan maken. Kennis, informatie, voor iedereen beschikbaar die met een computer kan omgaan. Tegen die tijd zal dat overigens al heel eenvoudig zijn, met een simpel vraag en antwoordspel kan de gebruiker zijn of haar wensen opgeven. Via een en hetzelfde apparaat kan men zowel het telefoonboek raadplegen – dat dan ook niet meer gedrukt zal worden, in Frankrijk wordt hier al mee geëxperimenteerd – als het banksaldo opvragen. Maar desgewenst ook een volledige universiteits-bibliotheek raadplegen, waarbij de automatische bibliothecaris u persoonlijk helpt.

In de nabije toekomst zal er voor encyclopedieën-verkopers geen droog brood meer te verdienen zijn. Wie wil er nog geld uitgeven aan een serie boeken die al beginnen te verouderen voor ze op de plank staan? In de elektronische encyclopedie

1.2 Homecomputer en databank

kan alle informatie dagelijks worden bijgewerkt. Desgewenst met afbeeldingen op de kleurenterminal. Informatie, snel, makkelijk, up-to-date en voor iedereen beschikbaar.

3/1.3

Prikborden

Steeds meer mensen hebben de mogelijkheid om met computers te communiceren. De ouwe trouwe Commodore 64 – of welke home-computer dan ook – gecombineerd met een modem en communicatie-programmatuur opent vele mogelijkheden om aan informatie te komen. Bijvoorbeeld de grote databank, zoals in hoofdstuk 3/1.2 beschreven is, maar sommige van de andere nieuwe ontwikkelingen zijn zeker even interessant.

Een daarvan is het bulletin board, het prikbord in goed Nederlands. Alweer, ook dit is een gecomputeriseerde variant van een vertrouwd verschijnsel. Op allerlei plekken waar veel mensen komen – scholen, bedrijven, buurthuizen, supermarkten etc. – vinden we gewone prikborden, waarop allerlei informatie kan hangen. Vanaf een mededeling van het bestuur of de directie tot en met kleine advertenties. Zo'n prikbord is voor veel mensen een uitstekend communicatiemiddel. Het stelt ze in staat om "informatie" aan te bieden aan wie het maar wil lezen. Ook het lezen van al die boodschappen heeft iets spannends, want wie weet, misschien zit er wel iets heel interessants tussen.

Alleen, het bereik van het traditionele prikbord is beperkt. Uitsluitend mensen uit de eigen wijk of uit het eigen bedrijf kunnen de boodschappen lezen.

Met de opkomst van computer is er, natuurlijk, ook een elektronische variant ontstaan. In eerste instantie als een mededelingenbord met huishoudelijke zaken, niet zozeer als prikbord voor iedereen. Wie "inlogt" op een van de grote computers – die soms door honderden mensen gebruikt worden – krijgt meestal wat min of meer dringende mededelingen van de systeembeheerder te lezen. Om een voorbeeld te geven, zo'n "login bulletin" vermeldt wanneer de computer wegens onderhoud niet bereikbaar zal zijn en welke nieuwe procedures beschikbaar zijn.

Al gauw breidde dit zich verder uit, er kwamen speciale programma's om allerlei informatie over de computer en de beschikbare programmatuur op te vragen. Handig, nietwaar. De gebruikers van zo'n systeem zijn zo meestal in staat zelf het antwoord op hun vragen te vinden, in feite wordt er een gespecialiseerde database geraadpleegd. Alleen voor bijzondere vragen moet men nog een voorlichter opbellen.

De volgende stap ligt ook voor de hand, namelijk de mogelijkheid om ook als gebruiker allerlei meldingen en vragen in te voeren. Maar al te vaak moeten computergebruikers het wiel opnieuw uitvinden, veel werk wordt niet slechts dubbel,

1.3 Prikborden

maar werkelijk honderden keren gedaan. Als iemand een simpel programma om een bepaalde berekening uit te voeren nodig heeft is vaak de enige oplossing het zelf maar te schrijven, terwijl er misschien wel tien versies van bestaan in dezelfde computer. Het is echter onmogelijk om die andere al geschreven gebruikers-programma's te localiseren, zonder een manier om zo'n vraag aan die honderden mensen voor te leggen. Vandaar het echte prikbord: naast de algemene meldingen en de ingebouwde gebruiksaanwijzingen kwamen er mogelijkheden om een soort vraag- en aanbod rubriek te gebruiken. Daar waren behalve de computer-gerichte onderwerpen snel ook allerlei andere zaken te vinden, net zoals op het schoolprikbord behalve officiële mededelingen ook privé-boodschappen verschijnen.

Voor sommige mensen vormen dergelijke prikborden een ware attractie, iets wat de prikborden overigens gemeen hebben met bijna iedere vorm van communicatie. Vandaar dat men in de vrije tijd dergelijke prikbord-programma's thuis op de micro-computer ging nabouwen. Met groot succes, vooral in Amerika zijn dergelijke bulletin boards heel snel aangeslagen.
In een tijdsbestek van slechts een paar jaar zijn aan de andere kant van de oceaan de hobbyisten-bulletin boards als paddestoelen uit de grond geschoten. Er zijn er vele honderden, als hun aantal nog niet in de duizenden geteld moet worden. Een simpele home-computer, een modem met auto-answer mogelijkheid (inderdaad, de computer neemt zelf de telefoon op) en een bulletin board programma is alles wat men nodig heeft om sysop te worden. De sysop, een afkorting van system operator oftewel systeem beheerder, maakt het telefoonnummer waarop zijn of haar nieuwe bulletin board te bereiken valt bekend – u raadt het al, door het in andere bulletin boards te vermelden, bijna allemaal hebben ze daar een speciaal hoekje voor – en het feest kan beginnen.

Mensen bellen op, lezen wat er aan informatie te vinden is en zetten zelf ook weer allerlei wetenswaardigheden in het bulletin board system. Zo ontstaat er binnen de kortste keren een soort van ruilmarkt, waar allerlei gegevens te verkrijgen zijn. Veel van de professionele programmeurs gebruiken de bulletin boards ook, soms blijkt iemands vraag over een bepaald programma door de programmeur zelf beantwoord te worden. Wat ook wel gebeurt is dat mensen die in dienst zijn bij de grote bedrijven zoals Commodore de bulletin boards gebruiken om geheime informatie te laten uitlekken, bijvoorbeeld over de verschillen tussen de verscheidene versies van de Commodore 64. Hoogst interessant, wat er allemaal op de bulletin boards verschijnt.

De absolute anonimiteit bij sommige boards zal daar zeker aan bijdragen, want hoewel de meeste sysops er op staan om hun gebruikers te registreren zodat althans de sysop weet wie "bitbyter" is, staan andere open voor ieder telefoontje. Vandaar dat men juist op die boards allerlei "boeiende" informatie kan aantreffen, tot en met gebruiksaanwijzingen aan toe hoe men bepaalde computers kan kraken. De "X-rated" secties zijn ook heel populair, er staan volledige pornografische verhalen in.
Zelfs zijn er al sysops veroordeeld tot hoge boetes, omdat hun bulletin board

1.3 Prikborden

ronduit tot criminaliteit uitnodigde. Zo doet het verhaal de ronde van de bankemployee die op een bulletin board haarfijn uit de doeken deed hoe men telefonische overboekingen kon laten registreren in de bankcomputer!

Voor een goede sysop is een bulletin board trouwens een tijdrovende hobby. Een dikke twintig uur per week schijnt eerder regel dan uitzondering te zijn, om alle informatie te schiften en up-to-date te houden.
Ook in ons land beginnen bulletin boards steeds meer opgang te krijgen. Momenteel zijn er een twintigtal in bedrijf, waarvan sommige 24 uur per dag bereikbaar zijn. Als u een modem aanschaft kan de handelaar u hopelijk wel een telefoonnummer bij u in de buurt geven, zodat u eens kunt proberen of zo'n bulletin board u wat lijkt.

Voor de meeste mensen is het bestaan van de bulletin boards niets meer dan een hele leuke uitbreiding van hun computer-hobby. Maar het eigenlijke belang van bulletin boards gaat veel verder, het is een nieuw communicatiekanaal. Allerlei soorten informatie kunnen zich via de bulletin boards razendsnel verspreiden, zonder daarbij echt aan grenzen gebonden zijn. In de nabije toekomst wordt het zelfs mogelijk om via een bulletin boardachtig systeem, het FIDO-net, met Amerika te communiceren tegen aanvaardbare kosten.

Door het op dit moment groeiende netwerk van bulletin boards ontstaat een heel nieuwe manier kennis te delen, wat ons terugvoert op het centrale thema van dit hoofdstuk.
Bulletin boards zijn veel meer dan een vorm van computer-speelgoed, het is de zoveelste uiting van de nieuw informatiemaatschappij. Gelukkig is dit er een die aan iedereen ten goede kan komen, een voorbeeld van computergebruik die kennis – en dus macht – kan verspreiden.

3/2

Veranderende methodes

Inhoud

Het gebruik van de moderne microcomputer, zeker nu deze steeds meer ingeburgerd raken, heeft op bepaalde gebieden onverwachte maar verstrekkende gevolgen. Allerlei nieuwe technieken raken steeds wijder verspreid, de computer is een echt werkinstrument aan het worden; en dan niet alleen voor wetenschappers.

Steeds meer schrijvers bijvoorbeeld stappen van de traditionele manier van werken over op de tekstverwerker. Die tekstverwerker kan best een Commodore 64 zijn, de capaciteit van onze home-computer is meer dan groot genoeg.

Nog een voorbeeld: ook kleinere bedrijfjes gaan over op een geautomatiseerde boekhouding. Wat tot voor kort voorbehouden was aan werkelijk grote ondernemingen is nu ook binnen het bereik van de buurtwinkelier. Sterker nog, die buurtwinkelier zal wel moeten overstappen op de computer als hij concurrerend wil blijven.

Zo zijn er legio plekken in de samenleving te vinden waar de computer in doordringt. Allerlei manieren van werken staan op de helling.
In dit hoofdstuk zullen we eens gaan kijken wat dat voor gevolgen kan hebben.

3/2.1

Het verdwenen manuscript

Schrijven is iets wat we allemaal wel doen. De een wat meer, de ander wat minder.
Boodschappenlijstjes, liefdesbrieven, ansichtkaarten, zakelijke correspondentie, uitnodigingen voor feesten, rapportages, opstellen, boeken, tijdschriften, er wordt heel wat afgeschreven. Er zijn heel wat mensen die in het kader van hun werk of hun opleiding lijvige stukken moeten leveren, maar ook veel mensen schrijven voor hun eigen plezier.

Nog steeds wordt verreweg het meeste schrijfwerk op de traditionele wijze gedaan: met een balpen, vulpen of andere schrijfstift. Zelfs als het uiteindelijke resultaat netjes getypt moet zijn zullen bijna alle mensen eerst in "klad" schrijven, om het daarna over te (laten) typen. Daar is echter op het moment verandering in aan het komen, want elke computer en zeker de kleine home- en personalcomputer is in principe ook een tekstverwerker. Bovendien worden veel mensen steeds vertrouwder met het gebruik van toetsenborden, omdat computers op allerlei plekken voor de meest uiteenlopende doeleinden gebruikt worden.

Die combinatie, de steeds bredere verspreiding van potentiële tekstverwerkers enerzijds en de steeds grotere gewenning aan toetsenborden anderzijds begint gevolgen te krijgen. Niet voor niets is er een ware overvloed aan tekstverwerkende programma's voor de 64 op de markt gebracht.
Natuurlijk zullen we de boodschappenlijstjes met de hand blijven schrijven, net zoals allerlei andere geschreven mededelingen. Het is nu eenmaal niet zo'n best idee om een uitgetypte – of liever gezegd: geprinte – liefdesbrief te verzenden. En wie zijn of haar dagboek in de tekstverwerker zet mist ook een hoop, want bij een dagboek zijn de doorhalingen bijna belangrijker dan wat er nog wel staat. Al die andere soorten van schrijfwerk echter lenen zich meer dan uitstekend voor de tekstverwerking, iets dat iedereen die ook maar enige ervaring met een goede tekstverwerker heeft opgedaan zal beamen. Door het gebruik van een tekstverwerker wordt het schrijfwerk een stuk beter van kwaliteit terwijl het minder inspanning vergt, want de klad-fase verdwijnt. Niet dat een tekstverwerker iemand die geen aanleg heeft kan leren schrijven, maar de mogelijkheid om zonder al te diep te hoeven nadenken de gedachten op papier – of beeldscherm – te kunnen zetten is heel prettig. De fouten worden er dan wel in tweede instantie uitgehaald, want er is nog niets definitief. Of desgewenst kan zelfs de volgorde nog helemaal omgegooid worden, de mogelijkheden zijn zo ongeveer onbeperkt.

2.1 Het verdwenenen manuscript

Het gebruik van tekstverwerkers heeft echter wel wat neven-effecten waar de meeste mensen niet zo gauw stil bij zullen staan. Tegenwoordig stappen meer en meer professionele schrijvers over op zo'n apparaat en dan niet alleen de mensen – zoals wij – die technische onderwerpen beschrijven. Ook literaire schrijvers, of ze nu "erkend" zijn of niet, maken de overstap.

Dat heeft verregaande gevolgen voor de wetenschap der literatuur-geschiedenis, die de ontwikkeling van schrijvers en het schrijven bestudeerd.
Voor deze wetenschappers zijn natuurlijk boeken een bron van informatie waaruit ze zich een beeld kunnen vormen over de lijnen waarlangs een bepaalde schrijver of een bepaald genre zich heeft ontwikkeld. Maar de originele manuscripten zijn zeker even belangrijk, want daar staan ook de doorhalingen en verbeteringen in. Zo'n manuscript geeft een inzicht in hoe een bepaald boek tot stand gekomen is, het bevat veel meer informatie dan het uiteindelijke gedrukte produkt. Soms heeft men zelfs de beschikking over meer dan één manuscript, vaak genoeg werden – en worden, in principe is daar geen verandering in opgetreden – stukken meerdere malen overgeschreven voor ze hun uiteindelijke vorm gekregen hadden.
Zoals de kunst-historicus zijn of haar onderzoek niet zonder allerlei schetsen en voorstudies kan bedrijven, zo is de literatuur-historicus afhankelijk van het bestaan van de originele manuscripten.

En daar zit hem nu net de kneep. Die manuscripten worden met een tekstverwerker niet meer geproduceerd. Of beter gezegd, er worden wel degelijk nog steeds manuscripten geproduceerd, maar ze worden niet meer bewaard.
Waar in het verleden de schrijver een stuk enkele malen opnieuw schreef, met de hand of met de schrijfmachine, daar wordt tegenwoordig de vorige versie opnieuw geladen en aangepast, waarna in de meeste gevallen die vorige versie op de diskette domweg overgeschreven wordt met de aangepaste. De print-out met de verbeteringen erop verdwijnt vervolgens in de prullenmand of wordt nog eens als kladpapier gebruikt, en weg is het materiaal waarmee de toekomstige literatuur-wetenschappers moeten werken. Een onverwacht bijverschijnsel van de steeds verder voortschrijdende gebruik van de computer als tekstverwerker.

3/2.2

De onkreukbare boekhouder

Een van de meest afschuwelijke karweitjes voor iedereen die ermee geconfronteerd wordt is de administratie. Of het nu een eenvoudige privé- of een ingewikkelder bedrijfsboekhouding is, vrijwel niemand doet het nu echt uit aardigheid. Dom sleurwerk dat echter wel heel zorgvuldig en nauwkeurig moet gebeuren en dat tot overmaat van ramp bijna altijd zwaar achterloopt. De bekende schoenendoos met bonnen, die eens per kwartaal zuchtend moet worden uitgezocht, gesorteerd, opgeteld en in het kasboek opgeschreven. Waarna men nog een of twee bonnen vindt die per ongeluk verkeerd waren terechtgekomen en weer opnieuw kan beginnen.

Geen wonder dat de computer ook hier een hoge vlucht heeft genomen, die machine is nu een maal nauwkeuriger, zeker met dergelijk sleurwerk, dan een mens. Bovendien ook flexibeler, als de programmatuur maar goed in elkaar zit kunnen die "vergeten" posten heel simpel alsnog tussengevoegd worden. Werkelijk, met een goed boekhoud-pakket kan de administratie "opzitten en pootjes geven".

Volgens de inspecteur der belastingen kan zo'n pakket zelfs wat al te flexibel worden, "opzitten en pootjes geven" zou volgens de belastingdienst wel eens heel nadelig voor de bedragen op de aanslag kunnen uitpakken. Althans, gezien vanuit het standpunt van die belasting-inner. Vandaar dat u niet zo maar uw eigen administratie-programmatuur mag schrijven. Hoewel, het schrijven kunnen ze u natuurlijk niet verbieden, maar het gebruik ervan wel.

Om een geautomatiseerde administratie te mogen voeren moet er gebruik gemaakt worden van een programma-pakket dat de inspectie heeft goedgekeurd. Allen in dat geval kunt u computer-uitdraaien presenteren in plaats van het gebruikelijke kasboek.
Zo'n goedgekeurd programma moet aan een hele waslijst van eisen voldoen, waarvan de belangrijkste allemaal te maken hebben met controleerbaarheid. Van iedere aangebrachte verandering in de boekhouding moet een uitdraai bestaan, die laat zien wat er hoe gemuteerd is. Het waarom hoeft er niet bij te staan, die vraag komt onvermijdelijk boven tafel tijdens de periodieke contrôle.
Bovendien moet onomstotelijk vaststaan in welke volgorde de uitdraaien zijn gemaakt doordat het volgnummer van de vorige altijd vermeldt wordt. Mocht er bij een contrôle een uitdraai verdwenen blijken, dan is Leiden in last. Dan gaat de controleur echt spitten, teneinde de vermeende fraude vast te stellen.

2.2 De onkreukbare boekhouder

Allemaal niet zo onbegrijpelijk overigens, ook als een administratie op de ouderwetse manier in een kasboek wordt bijgehouden wordt zo'n controlerend ambtenaar heel erg achterdochtig en lastig als een bladzijde uit het boek verdwenen is.
Zonder die eis van bewijsbare continuïteit wordt het namelijk wel heel gemakkelijk om de boekhouding wat op te sieren, wat nog niet eens frauduleus hoeft te zijn. Het simpele doorschuiven van een boekingspost naar een volgende periode kan soms het verschil tussen het ene belastingtarief en het andere uitmaken.

Om kort te gaan, de computer kan een uitstekende hulp zijn bij de boekhouding, maar jammer genoeg zonder dat daar "extra" voordelen uit voortvloeien. Het zou, gezien vanuit het standpunt van de "slimme" belastingbetaler weliswaar heel prettig zijn om eens wat verschillende opstellingen door te kunnen rekenen, maar die weg is vakkundig afgesneden door de eisen die aan zo'n goedgekeurd pakket gesteld worden. Denkt de inspectie der belastingen tenminste.
Want aan de andere kant van de medaille is het ook nog een keer zo dat belastingbetalers behoorlijk inventief zijn. In theorie moet er een verslag op papier bestaan van iedere wijziging. Als er geen printer is aangesloten, of als deze printer niet aanstaat, dan zal het goedgekeurde programma dienst wiegeren.

Hoe controleert dat programma echter of er aan die voorwaarde voldaan wordt? In de praktijk komt het er op neer dat als het "printer ready" signaal ontbreekt de zaak blokkeert; en dit signaal is alleen maar een bepaalde spanning op een bepaalde pen van de plug. Er bestaat dan ook een bloeiende markt in loze printerpluggen, die alleen maar voorzien zijn van een soldeerverbinding waardoor de computer "denkt" dat er een werkende printer aanwezig is. Hoezo, een papieren verslag van iedere actie?

Het kan nog bonter worden. Want ook een programma kan "aangepast" worden, zodat de situatie ontstaat dat er weliswaar een goedkeuring bestaat voor het een of andere boekhoud-programma maar er tegelijkertijd versies van in omloop zijn waarin die beveiligingen weer keurig zijn uitgeschakeld. Gezien het betrekkelijke gemak waarmee de meest complexe kopieer-beveiligingen worden doorbroken, waar soms letterlijk maanden aan gewerkt is door hoog-gekwalificeerde specialisten, zal de beveiliging in een boekhoudpakket niet zo'n probleem zijn.

Wie denkt dat computers goede boekhouders zijn heeft gelijk, wie denkt dat ze ook nog "onkreukbaar" zijn vergist zich. En hoe!

3/3

Toekomstmuziek

Inhoud

3/2.3

Computer-analfabetisme

Onze maatschappij is voor een groot gedeelte op kennis gebaseerd. Voor ieder gebied bestaan specialisten en het lijkt wel of veel van die specialisten niets anders doen dan zich nog eens extra specialiseren op nog kleinere deelgebiedjes. Zodat ook dat weer aparte specialismen worden, etcetera.

Erg verwonderlijk is dat ook niet, als we ons beseffen dat de menselijke kennis iedere tien jaar verdubbelt in omvang. Elke tien jaar weten we met zijn allen twee keer zoveel als daarvoor.

Vandaar dat het gewoon nodig is om al die kennis in deelgebieden op te splitsen en die deelgebieden steeds kleiner te maken. Een mens, al is het nog zo'n groot genie, kan nu eenmaal maar een beperkte hoeveelheid kennis overzien. Let wel, overzien, niet omvatten. Omvatten zou inhouden dat al die kennis parate kennis is, dat een specialist alles wat er maar te weten valt op haar of zijn gebied inderdaad in het hoofd meedraagt.

Gelukkig is dat niet noodzakelijk. Het zou zelfs onmogelijk zijn. Als men maar in grote lijnen een overzicht heeft over wat er bekend is en weet waar die kennis te vinden is kan men iets met die wetenschap uitrichten.

En daar komen onze goede vrienden de computers weer om de hoek kijken. Want computers zijn het gereedschap bij uitstek om die berg van kennis mee te kunnen hanteren. Zonder allerlei handige zoeksystemen, die een algemeen geformuleerde vraag kunnen oppikken en de revelante feiten – met verwijzingen naar verdere bronnen zoals boeken – is de hedendaagse specialist machteloos. Voor sommige vakgebieden kan men wel stellen dat het kunnen hanteren van een computer net zo belangrijk is als het kunnen lezen en schrijven.

Nu is deze ontwikkeling nog niet zo oud. In de wetenschappelijke wereld doen allerlei kennisbanken pas de laatste jaren opgang. Voor die tijd was de hardware te duur – en onder ons gezegd ook te beperkt – om een kennisbank te bouwen die voor niet-specialisten toegankelijk was. Computerspecialisten dan wel te verstaan.

Desondanks blijkt ook onder wetenschappers het toch wel even omschakelen te zijn, voor men aan de nieuwe technieken gewend is. Op veel laboratoria lopen een soort bemiddelaars rond, mensen die én van het vakgebied op dat lab én van computers het een en ander afweten. Die mensen zijn nu nog hard nodig, de oude wetenschappelijke garde is over

2.3 Computer-analfabetisme

het algemeen niet zo ingeschoten op computersgebruik. Ergens lijkt het wel alsof er een bepaalde angst bestaat voor die electronica.

Mensen die echter net afgestudeerd zijn zullen over het algemeen wel met computers om weten te gaan. Of het nu theologen, biologen, natuurkundigen of, om maar een dwarsstraat te noemen, sinologen (chinakenners) zijn, naar alle waarschijnlijheid hebben ze tijdens hun opleiding wel met computers te maken gekregen.

Op scholen zien we hetzelfde effect, veel leerlingen worden vroeger of later wel achter een computer gezet, meestal door een enthousiaste leraar. Ze leren daarbij dan de grondbeginselen van het computergebruik, en vaak ook nog een beetje programmeren. Misschien dat ze op een later tijdstip, in het kader van hun verdere opleiding of hun werkkring, nog meer ervaring opdoen, misschien ook niet. Feit blijft dat ze altans in principe met een computer kunnen omgaan, dat ze er niet bang voor zijn.

Want dat is vaak een reden waarom mensen niet met computers willen werken, ze zijn er een beetje bang voor. Computerfabrikanten beseffen zich dat terdege en proberen hun produkten dan ook zo min mogelijk op een computer te laten lijken. Zo denkt men bijvoorbeeld dat het hogere kader in het bedrijfsleven – over het algemeen al wat oudere mensen – aan een zogenaamde 'toetsenboord-angst' lijdt. Vandaar dat veel machines die als Personal Computer verkocht worden en dan ook bij die managers op het bureau zouden moeten komen te staan het toetsenbord proberen te omzeilen.

Dat leidt dan tot oplossingen zoals de 'muis', een apparaatje waarmee men in feite de cursor bestuurt. Als die cursor – vaak een vriendelijk handje op het scherm of wat dan ook, als het maar niet op een computer lijkt – eenmaal de juiste functie aanwijst is er nog maar een druk op de knop knop nodig om die functie op te starten.

Een andere 'vermomming' is het aanraak-scherm, een beeldscherm waarbij de computer kan bepalen naar welk punt op het scherm de gebruiker met zijn vinger wijst. Op grond daarvan kan de computer, zonder toetsenbordgebruik, de gewenste functie uitvoeren.

Al met al probeert men blijkbaar voor deze managers de scherpe computerkantjes er van af te halen. Men vreest dat deze invloederijke groep anders het computergebruik links zou laten liggen.

Toch wordt het steeds duidelijker dat computers niet meer weg te denken zijn uit onze samenleving. Waar men het management nog in de watten legt en voor wetenschappers een buffer-functie creëert, daar worden in allerlei kantoren de mensen zonder pardon achter het toetsenbord gezet. Niets geen muizen of aanraakschermen, naakte toetsenborden. Soms geeft dat grote problemen, hele afdelingen worden lamgelegd door de invoering van de automatisering, maar dat wordt nog wel eens in een andere Computers en Samenleving bijdrage belicht.

Steeds vaker gaan werkgevers er van

2.3 Computer-analfabetisme

uit dat men met computers moet kunnen leren werken, en eigenlijk zouden ze liefst zien dat sollicitaten al de nodige toetsenbord-ervaring hebben. Want ook in die hoek van de maatschappij groeit de complexiteit, net als dat in de wetenschappelijke wereld het geval is.

Zonder de inzet van computers is het langzaam maar zeker onmogelijk aan het worden om de zee van feiten te blijven overzien.

De volgende stap ligt voor de hand, lezers en lezeressen van dit boek hebben hem al gedaan. De computer wordt namelijk in de huidelijke kring ook steeds breder verspreid. Sommigen programmeren erop uit hobby, andere willen juist kant-en-klare toepassingen zoals tekstverwerking of databases gebruiken. Om nog maar niet te spreken over de wereld van mogelijkheden die opengaat bij het gebruik van computercommunicatie. Viditel en Girotel – het systeem waarbij men elektronisch kan thuisbankieren – zijn daar nog maar de voorbodes van.

Over enkele jaren zal het kunnen omgaan met computers inderdaad net zo belangrijk zijn als bijvoorbeeld lezen en schrijven. Al was het maar omdat een groot gedeelte van de post via elektronische wel zal verlopen. Wie zich op dit moment afsluit voor het verschijnsel computer kan binnenkort wel eens lelijk op zijn of haar neus kijken.

In de maatschappij bestaat er, naast alle specialismen, een tendens om van iedereen een bepaalde basis-kennis te verwachten, en die basis-kennis wordt al heel langs steeds omvangrijker. Konden in de vorige eeuw de meeste mensen niet rekenen, tegenwoordig kan men niet zonder enige reken-kennis. Dat hetzelfde effect optreedt bij een nieuwe ontwikkeling als de computer is niet verwonderlijk.

Alleen, het beangstigende is de snelheid waarmee het gebeurt. Immers, computers zijn pas zo'n tien jaar geleden begonnen aan hun huidige opmars. De vraag rijst dan ook wat we tien jaar na nu absoluut moeten kunnen, welke ontwikkeling die nu nog in de kinderschoenen staat dan 'verplichte'kost zal wezen. Hopelijk houden we, met zijn allen, die snelle ontwikkelingen bij.

3/3.1

Computer-home

Als we het over onze 64 hebben, dan praten we vaak in termen zoals 'home-computer', of – in goed Nederlands – 'huis-computer'. Die naam is gekozen om onze machines te onderscheiden van de grote computers, zoals die in de bedrijven gebruikt worden. Hoewel, het is blijkbaar een wet van Meden en Perzen dat een model, dat vandaag nog als uiterst professioneel wordt beschouwd, morgen al zo beperkt gevonden wordt dat het tot home-computer 'degradeert'.

Blijkbaar staat de techniek niet stil, als we ons eventjes bedenken dat de Commodore 64 meer mogelijkheden aan boord heeft dan bijvoorbeeld een universiteits-computer van twintig jaar terug.
De capaciteit van zo'n 'simpele' 64 wordt door de meeste gebruikers eigenlijk maar voor een heel klein stukje benut. De gemiddelde computer staat zo'n 90 procent van zijn tijd te verdoen met domweg wachten tot die trage menselijke bediener weer eens een toets heeft ingedrukt.

Dat gegeven heeft al allerlei mensen op hele leuke ideeën gebracht. Want het is natuurlijk best mogelijk om een computer allerlei klussen en klusjes te laten uitvoeren in die anders toch maar verspilde wachttijd. 'Multi-tasking' heet zoiets met een mooi woord en op bijvoorbeeld de Personal Computers is het momenteel een soort modebegrip. Daar worden allerlei programma's gebruikt die 'in de achtergrond' werken, zoals dat heet.
De gebruiker merkt er niets van, maar terwijl hij of zij de tekstverwerker gebruikt kan de computer ondertussen rustig de telefoon opnemen – middels een 'auto-answer' modem, een modem dus dat automatische data-verbindingen mogelijk maakt – en gegevens van een andere computer ontvangen.

Nu is het aardige dat dit 'werken in de achtergrond' niet zozeer mogelijk is door de snelheid van die PC, maar dat het veeleer de programmering is die zulke zaken mogelijk maakt. Min of meer op dezelfde manier waarop onze Commodores 60 keer per seconde de interrupt-routine afhandelt zonder dat we dat eigenlijk merken kan een gehaaide programmeur ook een communicatie-programma zo laten werken.
Weliswaar zal zo'n programma wel steeds kleine beetjes tijd afsnoepen van het hoofd-programma, maar dat stond toch bijna altijd alleen maar op ons te wachten. In feite draaien er op dat moment twee programma's onafhankelijk van elkaar op één en de zelfde computer.

Dat stelt natuurlijk wel de nodige eisen aan zowel de machine als de programmeur. Zo zal er genoeg geheugen aan

boord moeten zijn om beide programma's – en hun gegevens – tegelijkertijd te kunnen laden. Het is ook niet echt handig als beide programma's bijvoorbeeld tegelijkertijd via hetzelfde kanaal de diskdrive zouden willen gebruiken, dat leidt onherroepelijk tot moeilijkheden.

Kortom, die twee – of meer! – programma's moeten wel wat rekening met elkaar houden.

En daar komt de programmeur om de hoek kijken. Want die moet daar zorg voor dragen. Om nog maar eens een voorbeeld te noemen, die beide programma's moeten natuurlijk niet hetzelfde geheugen willen gebruiken om hun gegevens in op te slaan. Tegelijkertijd het scherm gebruiken kan ook misverstanden geven, het programma dat als tweede komt zal er zorg voor moeten dragen dat het oorspronkelijke scherm weer hersteld wordt, als het zijn melding gedaan heeft. Anders wordt het een puinhoop!

Maar het kan allemaal wel, dat is op de PC zo langzaam maar zeker wel bewezen. Het is prima te doen om één en dezelfde computer tegelijkertijd meerdere taken te laten verrichten. Zelfs op onze ouwe trouwe 64 is het mogelijk, hoewel zelfs de beste programmeur daar al snel tegen een aantal beperkingen oploopt.
Zo is de klok-snelheid van de processor wat laag, terwijl ook het beschikbare geheugen eigenlijk te klein is. Ideaal gesproken moet men namelijk die gelijktijdig geladen programma's ieder een fors brok RAM geven, om te voorkomen dat ze 'door elkaar' gaan lopen.

Maar nu even terug naar de stelling uit het begin van het verhaal. Dat werd namelijk gezegd dat de professionele machine van vandaag de home-computer van morgen is. En hoe goed de Commodore 64 ook is, ook onze lieveling zal vroeger of later door de voortschrijdende techniek worden ingehaald. Momenteel geldt nog dat die echte PC's allerlei zaken ontberen die onze 64 nu juist wel heeft. Zo zijn geluid en grafiek op de meeste PC's eigenlijk maar heel primitief, zakelijke gebruikers stellen dergelijke zaken niet zo op prijs. Het kan allemaal wel, maar het is erg kostbaar. Maar ook daar zal wel eens verandering in gaan komen, gezien de nog steeds dalende prijzen op de computermarkt. Vroeger of later zullen er PC's verschijnen die de geheugen-grootte en rekenkracht van de huidige generatie koppelen aan goed geluid en prima kleurengrafiek.

Op dat moment staat er dan een machine in de huizen die werkelijk het grootste gedeelte van zijn tijd met zijn duimen staat te draaien. Want ook de snelste typist zal de capaciteit van zo'n toekomstige home-computer maar voor een heel stukje nodig hebben met een tekstverwerker. Of, om iets anders te noemen, zelfs het snelste spel zal inwendig moeten worden vertraagd, omdat het anders domweg te snel – en dus onspeelbaar – wordt.

Die extra capaciteit nu, die deze machines zullen hebben, biedt allerlei hele leuke mogelijkheden om achtergrondprogramma's te laten draaien. En daarmee krijgt de term, home-computer een hele nieuwe inhoud.
Als we die twee woorden, 'huis' en

3.1 Computer-home

'computer', nu eens omdraaien dan staat er opeens 'computer-huis'. En dat is precies wat we kunnen verwachten in de niet al te verre toekomst.

Al die anders toch niet benutte capaciteit van de volgende generatie home-computers zal voor programmeurs werkelijk onweerstaanbaar zijn. Als vliegen rond de strooppot zullen ze er opaf komen, want er zijn meer dan genoeg leuke dingen te bedenken die onze computer kan doen in 'extra' tijd. Het al genoemde opnemen van de telefoon bijvoorbeeld, om dan via die telefoonlijn de 'elektronische post' te ontvangen. Of zelf opbellen – dat kan ook – om bijvoorbeeld het weerbericht voor ons op te halen.

Data-communicatie zal een hoge vlucht gaan nemen, ook bij de mensen thuis. En dan niet het tegenwoordige ook al steeds vaker voorkomende wat rondneuzen in bulletin-boards en dergelijke. Die toekomstige datacommunicatie zal grotendeels automatisch verlopen, zonder dat er een mens bij betrokken is. Daarbij moet u zich het gericht en zonder menselijke ingreep ophalen van nuttige – en ongetwijfeld onnuttige – informatie voorstellen. Dergelijke mogelijkheden zullen ook allerlei verschuivingen in de manier warop we met informatie omgaan met zich meebrengen, waarbij vele zaken wel eens ingrijpend zouden kunnen veranderen.

Wat bijvoorbeeld te denken van dag- en weekbladen. Een van de eerste 'slachtoffers' zouden wel eens de programmabladen kunnen zijn. Want wat men in principe van zo'n televisiegids verwacht – een overzicht van wat er op welke zender wordt uitgezonden, liefst met accurate tijden – kan natuurlijk heel gemakkelijk door zo'n super home-computer worden afgehandeld. Iedere dag op een van tevoren bepaalde tijd belt onze huisvriend het telefoonnummer van de 'ACOV', de 'Algemene Computer Omroep Vereniging', en haalt de actuele informatie op. Geen problemen met op het laatste moment omgegooide uitzend-schema's meer, alle gegevens kunnen in de centrale computer desgewenst per minuut bijgewerkt worden.

Die computer van ons heeft trouwens zoveel capaciteit dat het heel wel mogelijk is om er ook allerlei andere aparaten aan te knopen. Zo zou een computer-gestuurde video-recorder wel eens heel handig kunnen blijken.
Niet alleen kan men het programmeren van die recorder een stuk eenvoudiger maken – met de huidige modellen is dat soms een vreselijk gemodder – maar als de computer immers al de programmagegevens ophaalt, dan moeten we toch ook in staat zijn om te programmeren dat we alle afleveringen van Koot en Bie willen opnemen.
Op de dag van de uitzending 'kijkt' de computer eventjes of er wel een lege cassette in de recorder zit, om ons vervolgens keurig te waarschuwen als dat niet het geval mocht zijn. Het valt alleen nog maar te hopen dat men tegen de tijd dat dit zich allemaal gaat afspelen eindelijk een 'start'-signaal geeft via de zender, bij het begin van een programma, zodat je niet steeds het begin mist of juist een stukje van een vorige programma mee opneemt.

Toekomst-muziek? Ja, zeer zeker

Momenteel zijn dergelijke toepassingen van de home-computer nog lang niet in zicht. Maar dat het kan, dat staat als een paal boven water.

En als er nu eenmaal iets mogelijk is met computers, dan zal het vroeger of later ook gerealiseerd worden. Meestal is dat eerder vroeger dan later, overigens. Dat heeft de ervaring van de laatste paar decades ons wel geleerd.

4

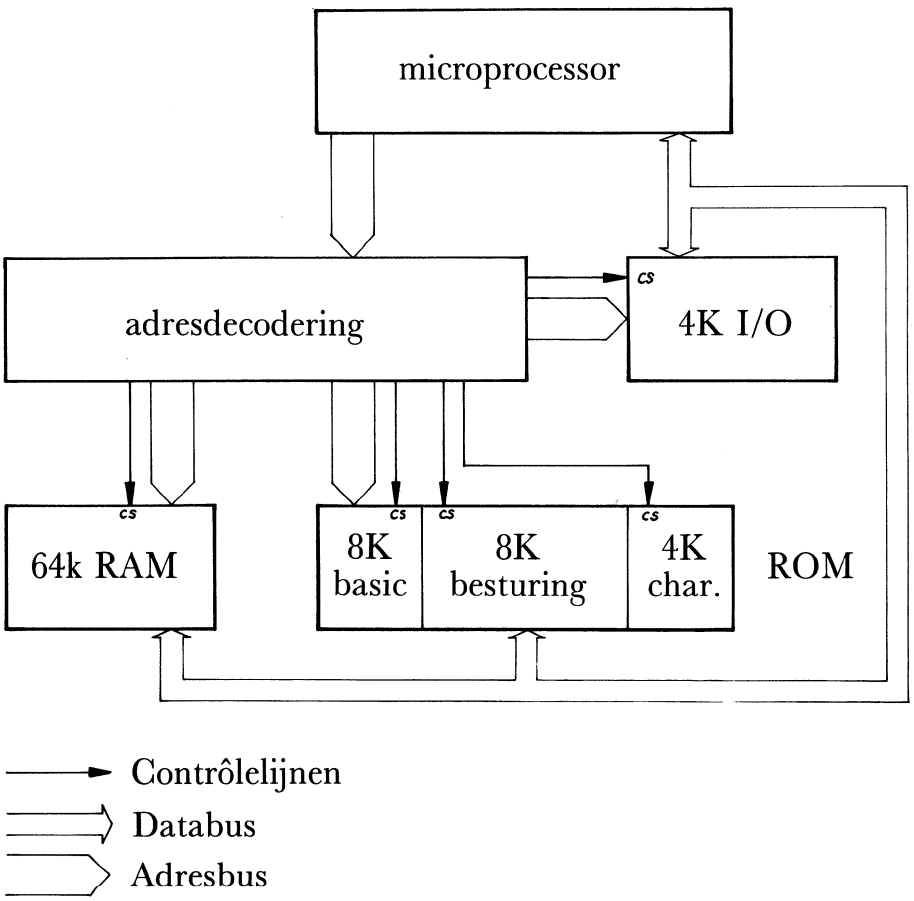
De opbouw

Inhoud

[1)] Dit hoofdstuk heeft een eigen inhoudsopgave.

Elke computer is óp te splitsen in 3 functionele gedeeltes: Het geheugen, de microprocessor of central processing unit (C.P.U.) en het invoer/uitvoer gedeelte (kortweg I/O, van het Engelse Input/Output). De Commodore 64 is ook op deze manier opgebouwd. De machine bevat 64 kB aan lees/schrijf geheugen (Random Access Memory of RAM) en 20 kB onveranderlijk geheugen (Read-Only Memory of ROM). Het RAM is voor het grootste gedeelte gereserveerd voor gebruikersprogramma's. Een klein gedeelte hiervan wordt door de computer zelf in beslag genomen, bijvoorbeeld voor de beeldscherminformatie. In het ROM staat onder andere het complete besturingssysteem (Operating System) van de Commodore 64 en de BASIC-vertaler. Maar meer over de besturing in Hoofdstuk 5.

In figuur 4-1 zijn de 3 belangrijkste deelgebieden gemakkelijk terug te vinden, aan de hand van deze figuur zullen we nu wat nader op de karakteristieken van de hardware ingaan.



Figuur 4-1: Architectuur C64

4/1

Het geheugen

Zoals gezegd heeft de Commodore 64 evenveel kilobytes aan RAM als zijn naam doet vermoeden. Daarnaast is er nog eens 20 kB ROM aanwezig, 8 kB voor het besturingssysteem, 8 kB voor de BASIC-vertaler en 4 kB voor de karakterset. Ja, ook de lettertjes die op het beeldscherm verschijnen moeten ergens gedefinieerd worden, dat gebeurt in het karakter-ROM. Aangezien er twee maal 256 karakters in de Commodore 64 aanwezig zijn (de hoofdletters/grafische tekens en de kleine letters/hoofdletters), die elk uit 8 bytes zijn opgebouwd, komen we op 2*256*8=4096 bytes, oftewel 4 kB.

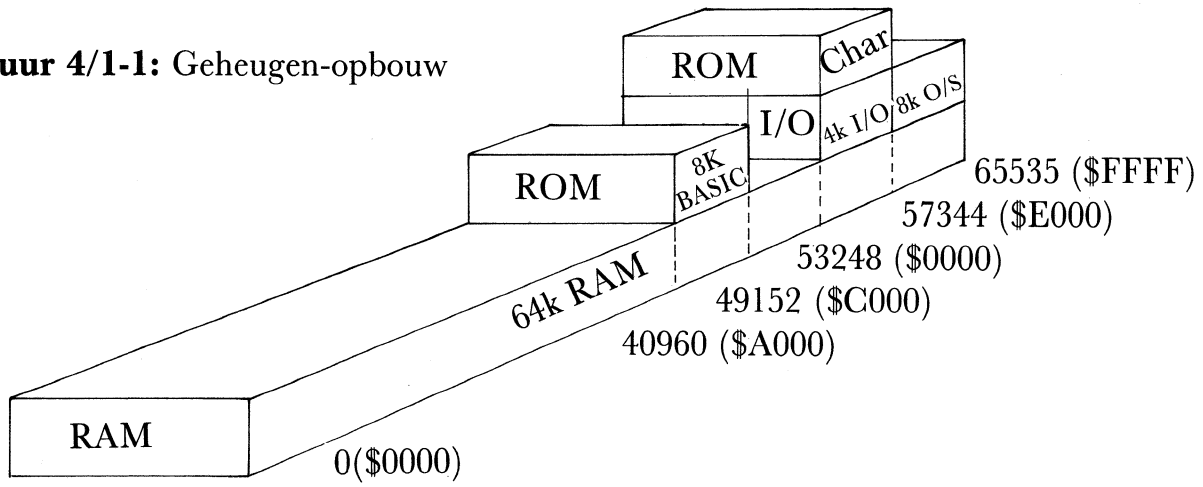
Nu heeft de verantwoordelijke microprocessor nogal een vervelende eigenschap: hij kan maar 64 kB geheugen tegelijk 'zien' (addresseren). Hiervan is 16 kB in gebruik voor ROM, de overige 4 kB ROM bevatten de karakters, die worden niet door de microprocessor gestuurd, maar door de grafische chip, de VIC-II. De VIC-II is namelijk de enige die verantwoordelijk is voor de opbouw van het beeldscherm. Hij leest zelf de juiste karakterinformatie uit het geheugen en vertaalt deze in voor ons leesbare letters. De 'plaatjes' van de letters staan in het karakter-ROM, dat alleen met een bepaalde truc voor de microprocessor 'zichtbaar' gemaakt kan worden. Details hierover zijn in hoofdtuk 10 te vinden. Daarbij is ook nog een gedeelte ter grootte van 4 kB in gebruik voor de in- en uitvoer, daar de I/O-chips via het geheugen geadresseerd worden. Dit heet memory-mapped I/O (zie ook hoofdstuk 4/4). Zelf heeft de computer ook nog eens 2 kB nodig (onder andere voor het beeldscherm en de cassettebuffer). Tel daar nog eens het feit bij op dat Commodore 4 kB geheugen heeft vrijgelaten voor machinetaalroutines zoals tape-snelladers en BASIC-uitbreidingen (een geheugengebied dat dus niet door BASIC programma's gebruikt wordt) dan kom je al gauw tot de conclusie dat van de royale 64 kB RAM slechts 64-16-4-2-4=38 kB overblijft: de 38911 BASIC bytes free die zich melden bij het aanschakelen van de computer!

Desalniettemin een aardige hoeveelheid. We zullen al gauw tot de ontdekking komen dat die 4 kB I/O zéér veel goed maken en dat de 4 kB extra RAM ook zeer nuttig zijn, temeer omdat de BASIC van de Commodore 64 hier en daar nogal eens te wensen overlaat. Om een en ander nog even samen te vatten is in figuur 4/1-1 een beknopte memory map ('geheugenkaart') geschetst, die de plaats van de diverse onderdelen in het geheugen even laat zien. Let daar bij op dat diverse soorten geheugen elkaar overlappen (bijvoorbeeld RAM en operating system, I/O en karakterset) en dat in één bereik zelfs 3 soorten

geheugen elkaar overlappen (van geheugenplaats (adres) 53248 tot 57344 bevinden zich RAM, het karakter-ROM èn het I/O-gebied).

Commodore 64 memory-map:
64k RAM van $0000-$FFFF
8k ROM van $A000-$BFFF (BASIC-interpreter)
4k ROM van $D000-$DFFF (Character sets 1&2)
8k ROM van $E000-$FFFF (Besturingssysteem)
4k I/O van $D000-$DFFF

Figuur 4/1-1: Geheugen-opbouw



4/2

De microprocessor

In de vorige paragraaf is hij al enige malen ter sprake gekomen: de microprocessor, ook wel het 'brein' van de computer genoemd. Het woord brein staat expres tussen aanhalingstekens, daar dit begrip meestal aan intelligentie gekoppeld wordt en dat is iets wat de MCS 6510 microprocessor van MOS Technology (lees: Commodore Semiconductor Group) helaas niet heeft. Anders zouden we nooit van een 'Syntax Error' gehoord hebben! Wat kan of doet een microprocessor nou eigenlijk?

Een microprocessor (en ook de 6510 uit de Commodore 64) voert constant simpele instructies uit. Instructies die varieren van het lezen of schrijven van het geheugen tot het optellen of aftrekken van twee getallen. De enige die verantwoordelijk is voor de machtige spellen en fabelachtige muziek die uit de Commodore 64 kan komen is de programmeur, die deze simpele instructies op de juiste manier achter elkaar zet. Toegegeven, de hardware die in de volgende paragraaf besproken wordt maakt een en ander wel een heel stuk eenvoudiger!

De mogelijkheden en snelheid van de microprocessor worden bepaald door zijn interne bouw, ook wel architectuur genoemd. De opbouw van de 6510 is vrij eenvoudig; een beperkt aantal instructies (56, met in totaal 151 varianten), een drietal interne geheugenplaatsen, ook wel registers genoemd en een matige klokfrequentie. De klokfrequentie bepaalt de snelheid waarmee de instructies uitgevoerd worden en is voor te stellen als de hartslag van de microprocessor. Deze hartslag komt voor de 6510 overeen met 1 miljoen pulsjes per seconde. Daar een instructie van de 6510 gemiddeld 4 pulsjes (klokslagen) duurt worden er per seconde zo'n 250.000 instructies uitgevoerd. Dat lijkt veel, maar dat is het eigenlijk niet. Ter vergelijking, een populaire collega van de 6510, de Z80a, draait op 4 miljoen pulsjes per seconde (4 MHz), heeft 24 interne registers en meer dan 100 instructies in meer dan 300 varianten. Maar goed, we klagen niet. Het is niet alleen de microprocessor die de computer maakt, het zijn voornamelijk de I/O chips die de mogelijkheden van de computer bepalen. Een punt waarop de Commodore 64 duidelijk met kop en schouders boven alle concurrenten uitsteekt.

Overigens, de 6510 kan ook op 2 MHz draaien, maar daar is de Commodore 64 niet geschikt voor. Wel is het misschien het vermelden waard dat de 6510 een verbeterde versie is van de zeer bekende 6502 (Apple, BBC, Vic-20). De verbetering zit in het feit dat de ongebruikte aansluitpennetjes van de 6502 zijn ver-

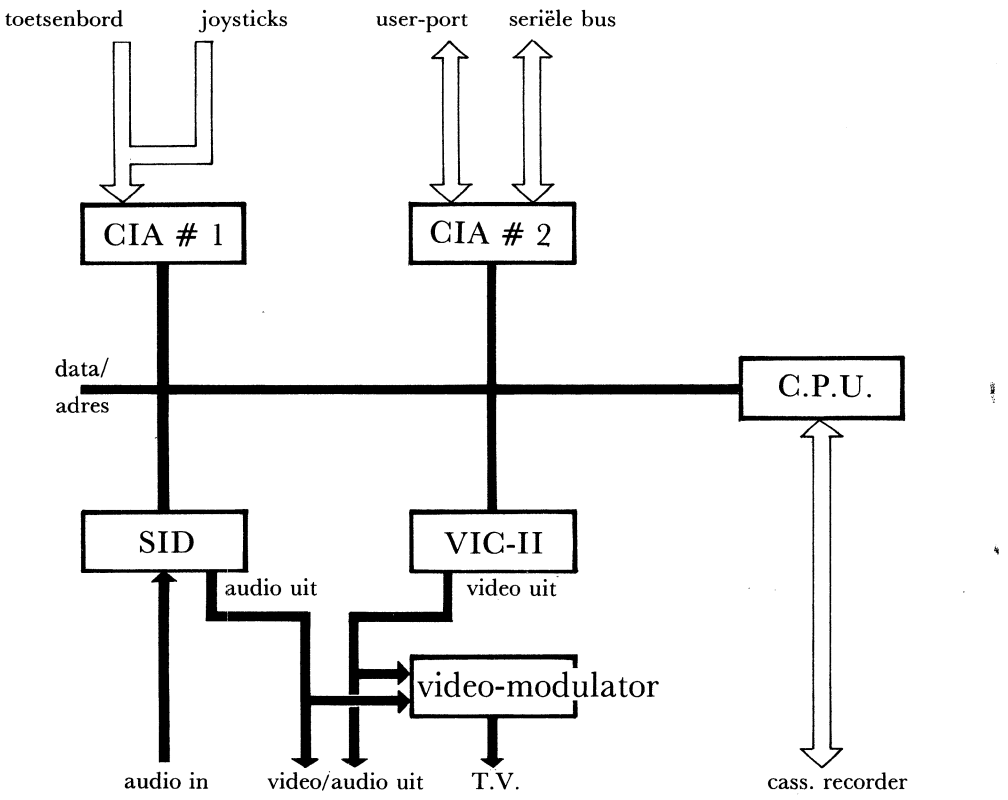
vangen door 6 extra I/O lijnen, die het mogelijk maken in- en uitvoer te bedrijven zonder dat hier speciale I/O-chips voor nodig zijn. Het voordeel hiervan zit hem in het feit dat de 6510 op deze manier zeer eenvoudig meer dan 64 kB geheugen kan adresseren, door via deze I/O-lijnen blokken (ook wel banken genaamd) geheugen in- en uit te schakelen. De Commodore 64 maakt hier ook gebruik van. Hoe, dat wordt in hoofdstuk 5 besproken.

4/3

De invoer/uitvoer

Inhoud

Voor het bespreken van de in- en uitvoer van de Commodore 64 is het belangrijk dat we eerst naar figuur 4/3-1 kijken, waar duidelijk de diverse componenten aangegeven zijn. Het spreekt voor zich dat de voornaamste communicatiemiddelen van de Commodore 64 het toetsenbord (voor invoer) en het beeldscherm (voor uitvoer) zijn. Maar daar houdt het verhaal niet mee op. Denk maar aan de cassette-recorder, de disk-drive, de printer, het modem voor communicatie via de telefoon, de versterker (ja, het is mogelijk om het hi-fi geluid dat uit de Commodore 64 komt via een versterker door de huiskamer te laten galmen), de joysticks, paddles, lichtpennen en ga zo maar door. Om bij het begin te beginnen zullen we eerst het toetsenbord onder de loep nemen.



Figuur 4/3-1: I/O-schema

4/3.1

Het toetsenbord

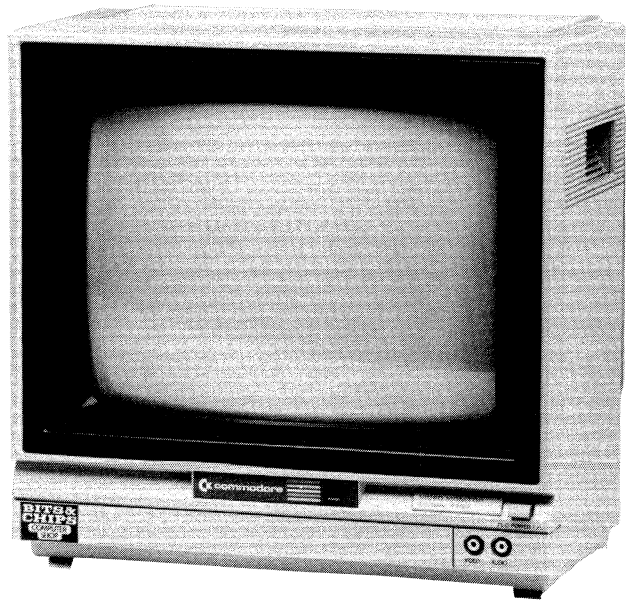
Het toetsenbord van de Commodore 64 is een schrijfmachine-toetsenbord. Dat maakt het invoeren van programma's redelijk eenvoudig (vroeger ging dat met ponskaarten!). Het besturingssysteem ziet het toetsenbord als een matrix (vlechtwerk) van 8 bij 8 draden waarvan de knooppunten met schakelaars (de toetsen) verbonden zijn. De RESTORE-toets heeft een aparte functie en de linker SHIFT- en de SHIFT/LOCK-toets zijn met elkaar verbonden. Deze matrix wordt 60 keer per seconde op contacten gecontroleerd. Zodra er een contact tussen 2 lijnen geconstateerd wordt en de ingedrukte toets dus bekend is wordt deze toetscode vertaald naar een nieuwe code, die in de Commodore tekenset voor een letter, cijfer, grafisch symbool of functie (bijv. CLR/HOME) staat. Indien dat de bedoeling is wordt dit teken dan op het scherm geprint. De chip waar deze toetsenbord-matrix op aangesloten is is Complex Interface Adapter (CIA) nr. 1. Op deze zelfde CIA worden ook de beide joysticks alsmede de lichtpen aangesloten. De RESTORE-toets is via een aparte tijdschakeling direct op de non-maskable interrupt (NMI)-lijn van de 6510 aangesloten en genereert dan ook een non-maskable interrupt zodra deze ingedrukt wordt. In hoofdtuk 9 wordt uit de doeken gedaan wat de NMI precies inhoudt.

4/3.2

Het beeldscherm

Het beeldscherm wordt getekend (gegenereerd) door de VIC-II chip. Deze chip heeft wel zeer veel mogelijkheden en wordt dan ook compleet ontleed in hoofdstuk 10, dat geheel aan de grafische mogelijkheden van de Commodore 64 gewijd is. Een van de extra zaken die de VIC-II regelt is de zogenaamde memory-refresh. Het type geheugen dat in de Commodore 64 gebruikt wordt is namelijk van het type 'dynamisch RAM'. Dit is goedkoop en verbruikt weinig stroom, maar een nadeel is echter dat de informatie die er in opgeslagen staat slechts een beperkte tijd intact blijft, te weten ongeveer 1 milliseconde (een duizendste seconde). De memory refresh nu zorgt er voor dat alle geheugenplaatsen tijdig worden uitgelezen en ook weer worden teruggeschreven. Een geheugenplaats wordt dus beschreven met zijn eigen inhoud, waardoor de informatie niet verloren gaat.

De beeldscherminformatie die de VIC-II genereert kan op twee manieren zichtbaar gemaakt worden: via een normale TV, hiervoor bevindt zich achter op de computer een TV-uitgang, of via een monitor, waarvoor ook een speciale uitgang achter op de computer is. Een monitor geeft een rustiger en scherper beeld dan een TV. Hoe de mooiste resultaten op TV en monitor verkregen kunnen worden wordt haarfijn uit de doeken gedaan in hoofdstuk 10.

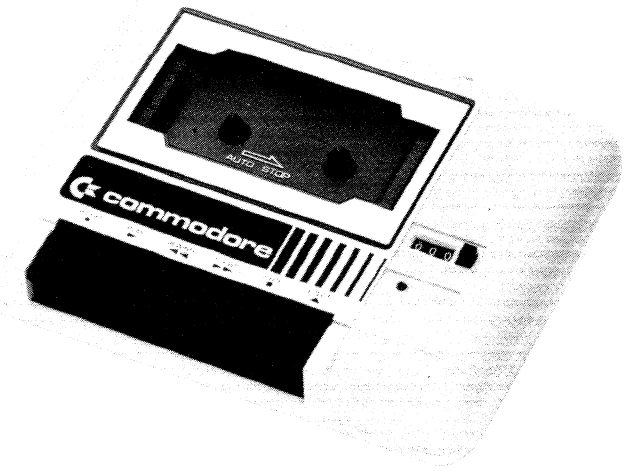


4/3.3

De cassette-recorder

Het eerste externe geheugen waar 95% van de Commodore-gebruikers mee werkt is de cassette-recorder (Datasette). De datasette heeft als voordeel dat hij relatief goedkoop is. Ook het opslagmedium, de muziek- of datacassette, is zeer goedkoop. Een groot nadeel echter is de snelheid, of moeten we zeggen het gebrek aan snelheid van de data-overdracht. Wachttijden van 5 tot 10 minuten voordat een programma geladen is zouden heden ten dage toch niet meer nodig moeten zijn.

Het inlezen van de bits gaat via de I/O-lijnen van de 6510, die in paragraaf 4/2 al even genoemd werden. Deze lijnen worden ook gebruikt om te kijken of een van de toetsen op de datasette is ingedrukt en om de motor van de cassetterecorder te starten. Het is dus theoretisch mogelijk om de cassetterecorder onder programmabesturing te bedienen, alhoewel dit in de praktijk tegenvalt. Het gebruik en de mogelijkheden van de datasette worden in hoofdstuk 6 besproken. De datasette wordt aangesloten op de speciale poort die zich hiervoor achter op de Commodore 64 bevindt.



4/3.4

De disk-drive

Ook de werking en het gebruik van de disk-drive wordt uitvoerig in hoofdstuk 6 besproken, maar wat betreft aansluiting op de Commodore 64 gaan we er hier even iets dieper op in.

De 1541 Single Drive Floppy Disk wordt op de seriële bus (zie paragraaf 4/3.6) achter in de computer aangesloten. Het voordeel van de disk-drive ten opzichte van de datasette is de snelheid van de data-overdracht. Alhoewel deze in verhouding met professionele standaarden nog zeer langzaam is, is het toch al een zeer grote verbetering; de drive is ongeveer een factor 9 sneller. Andere voordelen van de disk-drive zijn: hij is in staat grote bestanden -groter dan de geheugencapaciteit van de computer toelaat- te verwerken alsmede programma's snel en makkelijk op te zoeken. Alleen de naam hoeft te worden opgegeven, de disk-drive zoekt het programma zelf op. De datalijnen naar de disk-drive komen uit CIA nr. 2, die de seriële bus voor zijn rekening neemt.

4/3.5

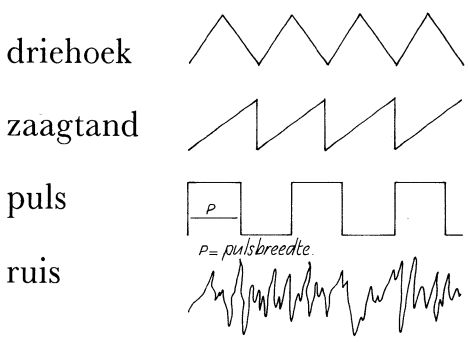
Geluid

Ook geluid vormt weer een hoofdstuk apart op de Commodore 64. Voor ons een reden om ook aan de Sound Interface Device, kortweg de SID (ook al van MOS, net zoals de VIC-II) een geheel hoofdstuk te wijden, namelijk hoofdstuk 11. Daar wordt volledig aandacht besteed aan het programmeren van de SID, maar hier zullen we even stil staan bij de hardware-mogelijkheden van deze ook al weer magnifieke chip.

Alhoewel men in eerste instantie zou verwachten dat de SID enkel en alleen een output-chip was, blijkt bij nadere bestudering dat de SID ook 3 ingangen heeft: 2 analoge ingangen voor game-paddles (met andere woorden, aangesloten paddles staan direct met de SID in verbinding) en 1 ingang voor een extern audio-signaal dat gebruik kan maken van SID's filtermogelijkheden. Deze externe ingang kan gebruikt worden om een tweede SID op aan te sluiten, waarop evt. nog meer SID's aangesloten zijn, zodat een oneindige hoeveelheid aan geluidsmogelijkheden voor handen zijn.

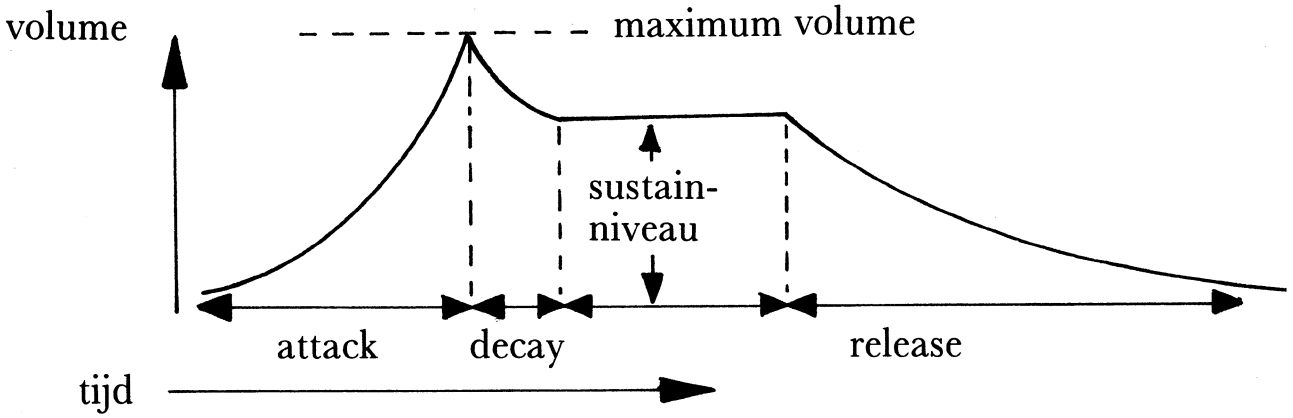
Waar bestaat de SID nu uit

De SID herbergt 3 geluidgeneratoren, meestal stemmen genoemd, die elk een zaagtand, een driehoek, een (variabele) puls of een ruisvorm kunnen produceren. Hoe deze golven er uit zien is in figuur 4/3.5-1 te zien.



Figuur 4/3.5-1: Golfvormen

Elk van deze geluidsgeneratoren wordt door een omhullende-generator gevolgd, dat is een elektronische schakeling die de diverse tijdsduren in de hoorbare golf definieert, te weten de aanzet (Attack), de terugval (Decay), het houd-niveau (Sustain) en de uitsterftijd (Release). Deze kreten bepalen de amplitude (geluidssterkte) van een toon naar gelang de tijd verstrijkt. In figuur 4/3.5-2 wordt een en ander nog even verduidelijkt.
Verder kan uiteraard de frequentie, dat is de hoogte van de toon, per stem geregeld worden van 0 tot 4 kHz en kan per stem bepaald worden of deze gefilterd moet worden. Daarvoor is er keuze uit een laagdoorlaatfilter, een hoog-doorlaatfilter en



Figuur 4/3.5-2: ASDR-verloop

een band-doorlaatfilter, of een combinatie hiervan. Het is ook mogelijk om de frequentie van stem 1 of stem 2 met de frequentie van stem 3 te moduleren, wat goed gebruikt kan worden voor het crëeren van sirenes.

Diverse synthesisertrucs zoals ringmodulatie en synchronisatie zijn met de SID ook allemaal mogelijk. Het moet duidelijk zijn: hoofdstuk 11 mag niet overgeslagen worden, al is het alleen maar voor de uitleg van de hier genoemde termen.

De uitgang van de SID wordt via de monitor-uitgang naar buiten gevoerd, welke direct op de AUX- of Tape-ingang van een HIFI-versterker aangesloten kan worden. Het heeft geen zin om uit te leggen hoe dat klinkt, dat moet gewoon gehoord worden om geloofd te kunnen worden, zo verbluffend is het resultaat!

4/3.6

Overige in- en uitvoer

Tot nu toe hebben we nog gezwegen over de printers. Commodore brengt diverse printers uit die zo op de seriële bus aangesloten kunnen worden.

Het probleem is dat er veel professionele printers op de markt zijn die geen Commodore-seriële ingang hebben, maar een Centronics-ingang. Hier zijn weer zogenaamde interfaces voor in de handel, een schakeling die een koppeling tussen 2 of meer apparaten verzorgt die niet zonder meer op elkaar aangesloten kunnen worden. Een nadeel van interfaces is dat ze nogal prijzig kunnen zijn. De user-port brengt hier uitkomst. De user-port is een directe 8-lijns in/uitgang (programmeerbaar per lijn!), afkomstig van CIA nr. 2. Deze is door de gebruiker zelf te programmeren, er zijn dan ook diverse programma's in de handel die printers aangesloten op de user-port perfect besturen. Het nadeel is dat er wel eens programma's kunnen zijn waarmee deze software-interface (in tegenstelling tot een hardware-interface) niet 100% functioneert. Dat is jammer, maar er is helaas niets aan te doen. Een andere vorm van printer-ingang is de RS-232C-ingang. Dat is een seriële ingang die vroeger veel gebruikt werd en nog steeds redelijk populair is. Ook modems hebben zo'n ingang. Ook voor deze aansluiting is CIA nr. 2 verantwoordelijk. De RS-232C-lijnen bevinden zich ook in de user-port, zodat het niet mogelijk is tegelijkertijd een Centronics-printer en een RS-232C-printer te sturen. Een voordeel van het RS-232C-protocol is dat hier geen extra software voor nodig is, enkel een kabel en een spanningsomvormertje van ca. *f* 70,–

Ook joysticks kunnen op de Commodore 64 aangesloten worden. Joysticks worden voornamelijk bij spelletjes gebruikt en behoeven geen nadere uitleg. De twee joystickpoorten bevinden zich aan de rechterkant van de computer. Op deze poorten kunnen ook paddles (draai-joysticks) en een lichtpen aangesloten worden.

Nu is er nog één onbesproken poort over op de Commodore 64, en dat is de expansion-port. De expansion-port is een zeer bijzondere poort, want die heeft directe toegang tot de adres-bus en de data-bus (over deze bussen vindt al het data-transport binnen de Commodore 64 plaats). We zullen er hier niet te diep op in gaan, maar mogelijkheden van de expansion-port zijn o.a. BASIC-uitbreidingen (Simons' BASIC), disk- en tapesnelladers en andere microprocessoren (b.v. de in paragraaf 4/2 genoemde Z80a, zodat ook het CP/M operating-system op de Commodore 64 kan draaien).

3.6 Overige in- en uitvoer

5

De besturing

Inhoud

[1]) Dit hoofdstuk heeft een eigen inhoudsopgave.

In dit hoofdstuk wordt wat dieper op de besturing van de Commodore 64 ingegaan. Uitgelegd wordt onder andere wat de Kernal is en doet en hoe de BASIC-vertaler te werk gaat. In hoofdstuk 4 is al even ter sprake gekomen dat in de Commodore 64 16 kB ROM aanwezig is wat voor BASIC en het besturingssysteem gereserveerd is. Wat is nu eigenlijk een besturingssysteem?

5/1

Het besturingssysteem

Inhoud

Wanneer de Commodore 64 aangezet wordt duurt het altijd even voordat de bekende startboodschap op het scherm staat. Het besturingssysteem is dan namelijk bezig de computer klaar te maken voor het gebruik. De volgende taken worden stuk voor stuk afgehandeld.

De RESET-routine wordt aangeroepen. Deze RESET-routine doet het volgende:

- Allereerst wordt er gekeken of er een insteekmoduul (cartridge) met autostart aanwezig is. Zo ja dan wordt de rest van het opstart-gebeuren overgeslagen en wordt de cartridge opgestart.
- Vervolgens worden de I/O-chips geïnitialiseerd. Dit is belangrijk om een goede communicatie (bijvoorbeeld via het toetsenbord, de cassette-recorder) te garanderen.
- Dan wordt het RAM geïnitialiseerd, dit houdt in dat de eerste 256 bytes van het geheugen, alsmede de cassettebuffer, met nullen gevuld worden, waarna de rest van het RAM getest wordt. Tijdens dit testen gaat de inhoud van het RAM niet verloren. Dit houdt in dat programma's na een RESET (zonder de computer uit te zetten natuurlijk, want dan gaat de inhoud van het geheugen wel verloren) nog steeds in het geheugen staan. Hoe we hier gebruik van kunnen maken wordt achterin dit hoofdtuk uit de doeken gedaan. Nadat het RAM getest is wordt de 'bovenkant' (het hoogste adres) van het RAM bepaald, dit verschilt, afhankelijk van de aanwezigheid van een cartridge.
- Hierna worden de diverse tabellen waar de computer gebruik van maakt opgezet. Deze tabellen staan uiteraard al in ROM, maar ze worden naar RAM gecopieerd zodat ze door de gebruiker aangepast kunnen worden. Dit maakt onder andere het realiseren van BASIC-uitbreidingen mogelijk.
- Tenslotte worden de screen-editor en de video-chip (VIC-II) geinitialiseerd, dit houdt onder andere in dat de kleuren ingesteld worden, waarna de welkomstboodschap op het scherm verschijnt. De screen-editor is het programma in ROM dat U in staat stelt BASIC-regels in te toetsen, te LISTen en te verbeteren (editen).

Dit alles is een van de vele taken die het besturingssyteem op zich neemt. Het besturingssysteem maakt ook de communicatie tussen gebruiker en computer mogelijk. Het houdt bij waar de diverse zaken op het scherm geprint moeten worden, hoeveel en welke files open zijn, welk lettertje welke kleur heeft, dat het toetsenbord wordt uitgelezen en nog veel meer. In de volgende paragraaf zullen we een interessant mechanisme uit het besturingssysteem, het interrupt-mechanisme uitleggen

5/1.1

De interrupt-routine

Misschien is het wel eens opgevallen dat het op de Commodore 64 mogelijk is commando's in te typen, terwijl een programma uitgevoerd wordt, of terwijl een listing over het scherm rolt. Of terwijl er een programma van disk geladen wordt. De ingetypte letters worden dan bewaard en verschijnen op het scherm onder de READY-prompt. Hoe is het nu mogelijk dat de computer, terwijl een bepaalde taak wordt uitgevoerd, toch de ingetypte letters kan inlezen en onthouden? Hier is nu de interrupt voor verantwoordelijk.

Interrupt is het Engelse woord voor onderbreking en dat is precies wat er gebeurt. Het is mogelijk om de microprocessor op elk gewenst moment te interumperen door een IRQ (Interrupt ReQuest, verzoek om een onderbreking) te geven. Op dat moment staakt de microprocessor al zijn activiteiten, haalt het adres van de interrupt-routine op (voor de 6502/6510 staat dit altijd op de zelfde plaats, te weten in de geheugenplaatsen FFFE en FFFF, dat is 65534 en 65535) en gaat daar dan ook mee verder. Omdat de computer het adres van zijn eigenlijke taak eerst veilig heeft opgeborgen is het mogelijk om na afloop van de interrupt-routine gewoon verder te gaan met hetgeen hij bezig was. Welnu, de interrupt-routine wordt door een speciaal tellertje precies 60 keer per seconde aangeroepen. De zaken die dan uitgevoerd worden zijn:

- Het nagaan of de stoptoets is ingedrukt. Zo ja, dan wordt dit gemeld aan het besturingssysteem, zodat bijvoorbeeld een lopend BASIC-programma onderbroken kan worden.
- Het ophogen van de jiffy-clock. De jiffy-clock is een teller die tot 5.184.000 telt. Daarna wordt weer met 0 verder gegaan. Aangezien hij 60 keer per seconde opgehoogd wordt komt dit overeen met 5.184.000/60=86.400 seconden. Dit komt overeen met 1440 minuten oftewel 24 uur.
  De variabelen TI en TI$ (zie hoofdstuk 7) worden van deze teller afgeleid. TI bevat altijd de actuele waarde van de jiffy-teller, TI$ bevat de 24-uurs representatie hiervan.
  Voorbeeld: als TI=300.000 dan komt dit overeen met 300.000/60=5.000 seconden. Dit is dan weer gelijk aan 83 minuten en 20 seconden. TI$ zal er dan als „012320" uitzien: 1 uur, 23 minuten en 20 seconden. De jiffy-teller is te beinvloeden door aan TI$ een waarde toe te kennen. Bijvoorbeeld: TI$="120000".
- Het doen knipperen van de cursor. Door de interrupt-routine meer dan 60 keer per seconde aan te roepen is het mogelijk om de cursor sneller te laten knipperen. Zie ook achterin dit boek.
- Het aan en uitzetten van motor van de

1.1 De interrupt-routine

cassette-recorder, afhankelijk van de stand van de toetsen.
- Het uitlezen van het toetsenbord en de codes van de ingedrukte toetsen opslaan in de 'keyboard-buffer'. Deze keyboard-buffer heeft een lengte van 10 plaatsen, dus alleen de eerste tien toetsen die ingedrukt worden kunnen bewaard worden.

Tijdens het laden of saven naar de cassette-recorder wordt de interrupt-routine uitgeschakeld. Het spreekt voor zich dat TI$ dan niet meer gelijk loopt. Bij gebruik van TI$ dient hier rekening mee gehouden te worden.

Het leuke van de interrupt-routine is dat deze 'omgeleid' kan worden. Dat wil zeggen dat het mogelijk is om een eigen interrupt-routine te maken en daarna eventueel de originele routine ook nog aan te roepen. Dit maakt allerlei interessante toepassingen mogelijk, hiervoor wordt verwezen naar Hoofdstuk 9 (Machinetaal).
Door de interrupt-routine om te leiden wordt het eenvoudig gemaakt om de voorzieningen tot het professioneel programmeren tot het uiterste uit te buiten. Maar er is meer mogelijk. Ook het besturingssysteem zelf is voor de (machinetaal)programmeur zeer eenvoudig te gebruiken. Dit wordt mogelijk gemaakt door de Kernal

5/1.2

De Kernal en zijn routines

De Kernal is de benaming voor de tabel met sprongadressen achter in het ROM van het besturingssysteem. Op de adressen FF81 t/m FFF5 (65409 t/m 65526) staan de adressen van alle in- en uitvoer routines die aanwezig zijn om communicatie tussen de Commodore 64 en zijn periferie (randapparatuur) zoals toetsenbord, disk-drive, printer en beeldscherm te bewerkstelligen. De Kernalroutines worden in hoofdstuk 9 uitgebreid besproken. Hier geven we een kleine uitleg bij de opbouw van de Kernal. In de nu volgende tabel staan de eerste 9 bytes van de Kernal. Allereerst het adres, dan de machinetaal instructie die op dat adres staat. Daarna de symbolische naam van de routine, gevolgd door een kleine uitleg:

FF81 JMP $FF5B ;CINT
Scherm-editor initialiseren
FF84 JMP $FDA3 ;IOINIT
CIA's initialiseren
FF87 JMP $FD50 ;RAMTAS
RAM schoonmaken/testen

De JMP-instructie betekent: Spring naar (vergelijkbaar met het BASIC-commando GOTO). De routine om bijvoorbeeld de CIA's te initialiseren begint op geheugenplaats FDA3 (64931). Het is dus mogelijk om vanuit een eigengeschreven programma deze routine met JSR $FDA3 (JSR betekent Jump to Sub routine en komt overeen met het BASIC-commando GOSUB) aan te roepen. Als de Commodore-programmeurs nu een fout in het besturingssysteem vinden en deze in de nieuwe ROM-versies willen verwijderen of verbeteren, kan het zijn dat de IOINIT-routine verplaatst moet worden, bijvoorbeeld door een uitbreiding van een voorgaande routine. Het start-adres van de IOINIT-routine kan dan op bijvoorbeeld $FDA4 komen te liggen, waardoor het eigengeschreven programma niet meer zou werken, omdat dat naar $FDA3 springt.

Hier komt nu het nut van de Kernal naar voren. Wordt het besturingssysteem namelijk aangepast dan wordt de sprongtabel ook meteen aangepast. Deze zal er dan zo komen uit te zien:

FF81 JMP $FF5B ;CINT Scherm-editor initialiseren
FF84 JMP $FDA4 ;IOINIT CIA's initialiseren
FF87 JMP $FD50 ;RAMTAS RAM schoonmaken/testen

Het eigenlijke aanroepadres van IOINIT blijft daarbij gelijk, namelijk $FF84. Door nu, als programmeur, een besturingssysteemroutine altijd via de Kernal aan te roepen wordt gegarandeerd dat een programma ook op een verbeterde ROM-versie blijft werken.

1.2 De Kernal en zijn routines

Een ander voordeel van deze Kernal-routines is dat veel van deze routines via een tabel in RAM aangeroepen worden. Dit is dezelfde tabel die door de reset-routine aangemaakt wordt. Door de adressen van de diverse routines in RAM nu te veranderen is het mogelijk om het besturingssysteem uit te breiden of aan te passen. Voorbeelden hiervan vinden we onder andere in tape-snelladers (die de save-routine verleggen naar een eigen, snellere save-routine) en Centronics-interfaces op softwarebasis. Dit zijn programma's die het mogelijk maken een niet-Commodore printer op de user-port aan te sluiten. Wat deze programma's doen is de PRINT-routine (ook een Kernal-routine) omleiden en bij elke letter kijken of deze naar de printer gestuurd moet worden. Zo ja, dan wordt de eigen PRINT-routine gebruikt, zo nee dan wordt de normale PRINT-routine aangeroepen.
Aangezien ook de BASIC-vertaler een aantal belangrijke adressen in RAM heeft is het ook mogelijk BASIC te veranderen of aan te passen. SIMON'S BASIC is hier een goed voorbeeld van.

Een tabel met adressen wordt ook wel een vector-tabel genoemd, daar deze adressen naar belangrijke geheugenplaatsen wijzen (gericht zijn).

5/1.3

De scherm-editor

De scherm-editor is het programmadeel in het besturingssysteem dat het mogelijk maakt regels in te typen en te verbeteren.

Wellicht staat U er niet altijd bij stil, maar tot voor kort, en in sommige gevallen nog steeds, was een scherm-editor een zeer grote luxe op home- en personal computers. Meestal werden programma's ingetypt met een zogenaamde line-editor, een besturingsprogramma-deel dat het mogelijk maakt een regel in te typen, waarna verbetering van deze regel niet of slechts met veel kunstgrepen mogelijk was. Een line-editor onthoudt namelijk alleen de karakters die zijn ingetypt, of waar de cursor overheen gegaan is. Dit betekende vaak dat indien in een regel een fout was opgetreden deze of opnieuw ingetypt moest worden, of verbeterd moest worden met het EDIT-commando. Doorgaans heksentoeren die het programmeren niet eenvoudiger maakten. Commodore was een van de eersten die een zeer uitgebreide scherm-editor in zijn PET stopte. Het is dezelfde editor die ook furore in de VIC 20 en de Commodore 64 maakte, een verbeterde versie is te vinden in de Commodore 128.

Het algemene principe van een scherm-editor is dat teksten op willekeurige plaatsen op het scherm veranderd kunnen worden, waarna de computer deze veranderingen verwerkt. Ook bij de 64 is dit zo, maar een verbeterde regel wordt echter pas uitgevoerd of in het geheugen opgeslagen nadat er op RETURN gedrukt wordt. Dat is echter gebruikelijk.

Het kopiëren van BASIC-regels wordt op deze manier ook heel makkelijk gemaakt. Nadat een regel is ingetypt en op RETURN is gedrukt hoeft enkel met de cursor-toets naar boven gegaan te worden, het regelnummer veranderd te worden en weer op RETURN gedrukt te worden. Klaar is Kees!

Een ander voordeel van de scherm-editor is dat de cursor-besturingstoetsen binnen de BASIC-opdracht 'PRINT' gebruikt kunnen worden. Dit maakt een gevarieerde opmaak van het scherm uiterst eenvoudig.

Het is ook mogelijk om de diverse machinetaalroutines uit deze scherm-editor vanuit BASIC (met een SYS-commando) aan te roepen, om bijvoorbeeld een regel op het scherm te wissen of de cursor naar een specifieke schermlocatie te sturen. Uitgebreide informatie hierover is te vinden achterin dit boek.

5/1.4

De zero-page

De zero-page neemt een heel belangrijke plaats in binnen de besturing van de Commodore 64. Zero-page is de benaming voor de eerste 256 bytes van het geheugen. Voor de 65XX microprocessor is dit geheugengebied zeer belangrijk, omdat er een speciale manier van adresseren (geheugenplaatsen aanwijzen) op deze microprocessor mogelijk is, het zero-page adresseren. Dit heeft twee voordelen, te weten een besparing aan geheugenruimte en een winst in snelheid. Heel in het kort komt het er op neer dat er maar 1 byte nodig is om een adres aan te wijzen, omdat dit adres in de zero-page ligt. De precieze details zijn in hoofdstuk 9 te vinden.

Om nu het besturingssysteem van de Commodore 64 zo efficient mogelijk te laten werken wordt er van de zero-page zeer intensief gebruik gemaakt, zowel door de BASIC-vertaler als door het besturingssysteem. Het veranderen van deze geheugenplaatsen, mits gericht gedaan, kan leuke effecten opleveren. De gehele zero-page wordt aan het eind van dit hoofdstuk besproken, hoe deze tot ons voordeel beïnvloed kan worden wordt uit de doeken gedaan in hoofdstuk 9.

Naast de zero-page wordt er nog een belangrijk geheugengedeelte door BASIC en het besturingssysteem in beslag genomen, te weten de geheugenplaatsen 512 t/m 1019. Van 512 t/m 600 vinden we de BASIC input-buffer. Een buffer is een geheugengedeelte wat voor tijdelijke verwerking van bepaalde gegevens gebruikt wordt. Buffers die de Commodore 64 gebruikt zijn onder andere de cassettebuffer, de toetsenbord-buffer en de inputbuffer. Elke regel of commando dat ingetypt wordt komt hier terecht om vertaald te worden. Van 601 t/m 630 worden de geopende files met hun devicenummers en secondary addresses bijgehouden. De keyboard-buffer, die bijhoudt welke toetsen er ingetypt zijn of worden, bevindt zich van 631 t/m 640 en is dus 10 plaatsen groot. Dan volgen nog wat losse systeemvariabelen, die niet meer in de zero-page pasten. Op geheugenplaats 768 begint dan de eerder genoemde vectortabel naar de diverse routines van de BASIC-vertaler. De vectoren naar de diverse Kernalroutines beginnen op 788 en lopen door tot 819. Een uitgebreide bespreking is aan het eind van dit hoofdstuk te vinden.
De cassette-buffer, het geheugengebied dat gebruikt wordt tijdens LOADen en SAVEn met de cassetterecorder, bevindt zich op de geheugenlocaties 828 t/m 1019.

5/2

De BASIC-vertaler

INHOUD

De BASIC-vertaler is het programma in ROM dat ons in staat stelt in BASIC te programmeren. De microprocessor in de Commodore 64 begrijpt geen BASIC. Hij kent alleen maar machinetaal, reeksen codes tussen de 0 en de 255. De BASIC-vertaler (of BASIC-interpreter) is een programma, geschreven in machinetaal, dat de BASIC-commando's omzet van BASIC naar machinetaal. Ook de BASIC-vertaler is een belangrijk onderdeel van de Commodore 64, zonder deze zou de computer onze commando's niet kunnen begrijpen of uitvoeren.

5/2.1

BASIC

Dit hoofdstuk vertelt niets over de taal BASIC, alleen hoe de Commodore 64 met BASIC omgaat. Voor BASIC zelf wordt naar hoofdstuk 7 verwezen.

Wanneer een commando (of programma-regel) ingetypt wordt komt dit commando in de input-buffer terecht. Dan gaat de tokenizer aan de slag om de regel te tokeniseren. Tokeniseren betekent dat BASIC-woorden (GOTO, PRINT etc.) vertaald worden naar tokens, dit zijn codes van 1 byte lengte. Tokens hebben een waarde tussen de 128 en 204. Als dit gebeurd is wordt de regel opgeslagen indien er een regelnummer voor stond, of uitgevoerd indien er geen regelnummer voor stond. Dit laatste is de zogenaamde direct mode.

Het uitvoeren gebeurt nu door de eigenlijke interpreter-routine die bij de tokens die in de regel staan de startadressen van de overeenkomstige (machinetaal)routines zoekt, die de bepaalde commando's dan uitvoeren. Het zijn deze machinetaal-routines die de nodige foutmeldingen genereren indien dingen gebeuren waar ze geen raad mee weten.

Deze manier van werken (tokeniseren) is zeer efficiënt: elk BASIC-commando neemt slechts 1 byte in beslag in plaats van bijvoorbeeld de 5 die het woord PRINT zou innemen. Dit is uiteraard ook gunstig bij het opslaan op cassette of disk. Daarnaast maakt het het uitvoeren van een programma ook snel, daar de start-adressen van de machinetaalroutines, uitgaande van de tokens, zeer snel gevonden kunnen worden.
Het feit dat er in een programma-listing geen gehele BASIC-commando's staan heeft natuurlijk wel consequenties voor de LIST-routine. Deze moet, ten tijde van een LIST-commando, de tokens door de juiste BASIC-commando's vervangen. De LIST-routine doet dus precies het tegenovergestelde als de tokenizer. Hierdoor boet het LIST-commando natuurlijk wel aan snelheid in, alhoewel dat niet echt te merken is!

5/2.1.1

De tokenizer

Zoals gezegd is de tokenizer het gedeelte van de BASIC-vertaler dat de ingetypte commando's naar tokens vertaalt. Tokens zijn codes van 1 byte lang die een BASIC-commando voorstellen. uitgaande van de waarde van de token kan het adres van de bijbehorende machinetaalroutine zeer snel gevonden worden.

De token-waardes in de 64 lopen van 128 t/m 204. Theoretisch kunnen hier 52 aan toegevoegd worden. Daar is echter onder normale omstandigheden geen rekening mee gehouden, de BASIC-interpreter zal deze tokens niet herkennen en dus ook niet kunnen uitvoeren. Daarnaast weet de tokenizer ook niet hoe tokens boven de 204 gegenereerd moeten worden, dus wat dat betreft heffen de tokenizer en de interpreter elkaars gebreken netjes op.

Toch is het mogelijk BASIC uit te breiden, getuige programma's zoals Simons' BASIC en Super BASIC. Daartoe is in eerste instantie echter kennis van de werking van de tokenizer een noodzaak. Deze paragraaf legt een en ander uit, in deel 9 zal een daadwerkelijke uitbreiding gepresenteerd worden.

```
0   A579   6C0403     JMP   ($0304)
1   A57C   A67A       LDX   $7A
2   A57E   A004       LDY   #$04
3   A580   840F       STY   $0F
4   A582   BD0002     LDA   $0200,X
5   A585   1007       BPL   $A58E
6   A587   C9FF       CMP   #$FF
7   A589   F03E       BEQ   $A5C9
8   A58B   E8         INX
9   A58C   D0F4       BNE   $A582
10  A58E   C920       CMP   #$20
11  A590   F037       BEQ   $A5C9
12  A592   8508       STA   $08
13  A594   C922       CMP   #$22
14  A596   F056       BEQ   $A5EE
15  A598   240F       BIT   $0F
16  A59A   702D       BVS   $A5C9
17  A59C   C93F       CMP   #$3F
18  A59E   D004       BNE   $A5A4
19  A5A0   A999       LDA   #$99
20  A5A2   D025       BNE   $A5C9
21  A5A4   C930       CMP   #$30
22  A5A6   9004       BCC   $A5AC
23  A5A8   C93C       CMP   #$3C
24  A5AA   901D       BCC   $A5C9
25  A5AC   8471       STY   $71
26  A5AE   A000       LDY   #$00
27  A5B0   840B       STY   $0B
28  A5B2   88         DEY
29  A5B3   867A       STX   $7A
30  A5B5   CA         DEX
31  A5B6   C8         INY
32  A5B7   E8         INX
33  A5B8   BD0002     LDA   $0200,X
34  A5BB   38         SEC
35  A5BC   F99EA0     SBC   $A09E,Y
36  A5BF   F0F5       BEQ   $A5B6
37  A5C1   C980       CMP   #$80
38  A5C3   D030       BNE   $A5F5
39  A5C5   050B       ORA   $0B
40  A5C7   A471       LDY   $71
41  A5C9   E8         INX
42  A5CA   C8         INY
43  A5CB   99FB01     STA   $01FB,Y
44  A5CE   B9FB01     LDA   $01FB,Y
```

```
45  A5D1  F036      BEQ  $A609
46  A5D3  38        SEC
47  A5D4  E93A      SBC  #$3A
48  A5D6  F004      BEQ  $A5DC
49  A5D8  C949      CMP  #$49
50  A5DA  D002      BNE  $A5DE
51  A5DC  850F      STA  $0F
52  A5DE  38        SEC
53  A5DF  E955      SBC  #$55
54  A5E1  D09F      BNE  $A582
55  A5E3  8508      STA  $08
56  A5E5  BD0002    LDA  $0200,X
57  A5E8  F0DF      BEQ  $A5C9
58  A5EA  C508      CMP  $08
59  A5EC  F0DB      BEQ  $A5C9
60  A5EE  C8        INY
61  A5EF  99FB01    STA  $01FB,Y
62  A5F2  E8        INX
63  A5F3  D0F0      BNE  $A5E5
64  A5F5  A67A      LDX  $7A
65  A5F7  E60B      INC  $0B
66  A5F9  C8        INY
67  A5FA  B99DA0    LDA  $A09D,Y
68  A5FD  10FA      BPL  $A5F9
69  A5FF  B99EA0    LDA  $A09E,Y
70  A602  D0B4      BNE  $A5B8
71  A604  BD0002    LDA  $0200,X
72  A607  10BE      BPL  $A5C7
73  A609  99FD01    STA  $01FD,Y
74  A60C  C67B      DEC  $7B
75  A60E  A9FF      LDA  #$FF
76  A610  857A      STA  $7A
77  A612  60        RTS

READY.
```

Uitleg

Het allereerste wat opvalt is de indirecte sprong: JMP ($0304). Dit commando bekijkt de geheugenplaatsen $0304 en $0305 en gebruikt de inhoud hiervan als adres waar naartoe gesprongen wordt. Wanneer de 64 aangezet wordt is dit adres gelijk aan $A57C, wat dus niets opzienbarends inhoudt. Geheugenplaats $7A wijst naar de ingelezen tekst zodat deze teken voor teken geadresseerd kan worden. Tekens met een waarde boven de 127 worden overgeslagen, behalve het PI-teken ($FF, regel 6). Spaties worden uiteraard gewoon overgenomen, ze worden gedetecteerd in regel 10.

Wanneer tekens binnen quotes (") staan worden deze letterlijk overgenomen en in het geheugen opgeslagen. De routine in de regels 56 t/m 63 is hier verantwoordelijk voor. De quote staat namelijk in geheugenplaats 8 opgeslagen (regel 12) en regel 58 zoekt naar de volgende. Staat er slechts één quote in een regel dan wordt dit door 56 (einde regel=0) opgevangen.

Dat geheugenplaats $0F voor meer doeleinden gebruikt wordt blijkt uit de regels 5 en 16. Hier wordt gecheckt of bit 6 van zowel het ingelezen teken als van $0F gezet zijn (de waardes lopen dan van 64 t/m 127). Zo ja, dan wordt het teken gewoon opgeslagen en wordt er niet verder gezocht of het een BASIC-commando betreft. Dit heeft te maken met het DATA-commando. Tekst achter DATA wordt namelijk niet getokeniseerd. Verderop in de routine wordt bit 6 van $0F gezet wanneer het DATA-commando gevonden is (regels 49 t/m 51).
Is van voornoemde situatie echter geen sprake dan wordt allereerst gekeken of het ingelezen teken een vraagteken is. Zo ja dan wordt dit door de PRINT-token vervangen (17 t/m 29). Waardes tussen de $30 (48, waarde voor '0') en $3B (59, waardoor voor ';') worden ook letterlijk overgenomen (21 t/m 24).

Wanneer echter geen van de voornoemde gevallen van toepassing is dan kan het eigenlijke vergelijken beginnen. Geheugenplaats $0B gaat het commando-nummer bijhouden, vandaar dat deze op 0 gezet wordt (26 en 27). Verder wordt

2.1. BASIC

Y (pointer op de outputbuffer, oftwel de getokeniseerde tekst) in $71 opgeslagen en wordt X (pointer op de inputbuffer, oftwel de te tokeniseren tekst) in $7A opgeslagen. Het vergelijken begint in 33 en loopt door tot 38. Voordat dit uitgelegd kan worden dient echter iets over de aard van de commandotabel gezegd te zijn. Deze tabel begint op $A09E en bevat, een voor een, alle BASIC-commando's, inclusief de mathematische operatoren (+, *, / etcetera). Om nu de commando's onderling te kunnen onderscheiden is van elk woord de laatste letter gevlagd door middel van het achtste bit dat in deze letter gezet is. Op die manier is niet alleen het ene commando van het andere te onderscheiden, echter op die manier is het ook mogelijk commando's af te korten. Dat dit zeer ingenieus gevonden is blijkt uit de zoekroutine zelf.

Van het teken dat eventueel deel van een commando uit moet maken wordt de waarde van de eerste letter uit de tabel afgetrokken (regel 35). Is de uitkomst niet gelijk aan nul dan wordt gekeken of het resultaat 128 is, is dit ook niet het geval dan wordt via $A5F5 het volgende commando uit de tabel opgezocht. Is de uitkomst echter wel gelijk aan 128 dan kan dit twee oorzaken hebben, namelijk het commando is in zijn geheel goed ingetypt, of het commando is afgekort met behulp van een SHIFT-teken. In beide gevallen levert het verschil (teken min laatste teken of SHIFT-teken min teken) een resultaat van 128 op en in beide gevallen kan de bijbehorende token ingevuld worden.
Het is ook nog mogelijk dat de allereerste vergelijking 0 oplevert (regel 36), zo ja dan wordt het volgende teken uit de tabel met het volgende teken van het eventuele commando vergeleken (31 t/m 35).

Wanneer een BASIC-commando gevonden is dan staat in $0B op welke plaats dat commando in de tabel staat, $0B wordt namelijk verhoogd wanneer er met het volgende woord vergeleken gaat worden. Door deze waarde met 128 te OR-en (regel 39, de accu heeft daar namelijk een waarde van $80) wordt automatisch de tokenwaarde gevonden. Deze kan dan in de outputbuffer opgeborgen worden (regel 41 t/m 44, tevens het sprongadres, na de meeste vergelijkingen zoals bijvoorbeeld in 11). Let er op dat de outputbuffer op dezelfde plaats ($01FB,Y) als de inputbuffer ligt. Dit is mogelijk omdat de getokeniseerde tekst altijd even lang of korter dan de originele tekst is. Overigens is de initiële waarde van Y gelijk aan 4 (regel 2) zodat na de eerste INY-instructie de outputbuffer inderdaad op $01FB+5=$0200 begint.

Na het tokeniseren wordt in regel 45 gekeken of het einde van de regel bereikt is (bytewaarde 0), zo ja dan wordt via $A609 de routine verlaten. Is de regel nog niet ten einde dan wordt gekeken of de laatste token van het DATA-commando is (regel 49). Op die manier kan bit 6 van $0F gevlagd worden zodat na een DATA-commando de rest niet getokeniseerd wordt. Ook na een REM-commando wordt niet getokeniseerd, de routine in de regels 56 t/m 63 neemt de tekst na een REM-commando letterlijk over nadat de REM-token eerst in 53 gesignaleerd is.

Het enige wat nog overblijft uit de tokenizer is de routine die het volgende commando opzoekt. Wanneer op een gegeven moment blijkt dat een ingetypte tekst

niet met een commando overeenkomt dan kan de rest van de vergelijking overgeslagen worden. Het vergelijken moet dan wel bij het volgende woord beginnen, vandaar dat er een teken gezocht worden waarvan bit 8 gezet is. Na dit teken begint namelijk het volgende woord, als die er nog zijn. Het moet namelijk ook mogelijk zijn om het einde van de tabel te detecteren, dat wil zeggen wanneer het ingetypte commando nergens mee overeen komt dan dient er uiteraard ook niet getokeniseerd te worden. Regels 64 t/m 70 houden rekening met al deze eisen. In 66 t/m 68 wordt een teken met een gezet achtste bit gezocht, regel 69 checkt of het eerstvolgende teken een waarde 0 heeft. Zo nee dan kan de vergelijking met volgende woorden via $A5B8 verder gaan, is het einde van de tabel wel bereikt dan dienen de letters en/of tekens gewoon overgenomen te worden. Dit gebeurt dan ook in $A5C7 via regel 72.

5/2.1.2

De LIST-routine

In de vorige paragraaf wordt de werking van de tokenizer uitgelegd. Eerder is al genoemd dat het tokeniseren consequenties voor de LIST-routine heeft: tokens zijn niet echt leesbaar. Vandaar dat bij de te LIST-en tokens de bijbehorende commando's weer opgezocht moeten worden. De routine in de 64 die hier voor verantwoordelijk is, is betrekkelijk klein en ziet er zo uit:

```
 0  A717  6C0603     JMP  ($0306)
 1  A71A  10D7       BPL  $A6F3
 2  A71C  C9FF       CMP  #$FF
 3  A71E  F0D3       BEQ  $A6F3
 4  A720  240F       BIT  $0F
 5  A722  30CF       BMI  $A6F3
 6  A724  38         SEC
 7  A725  E97F       SBC  #$7F
 8  A727  AA         TAX
 9  A728  8449       STY  $49
10  A72A  A0FF       LDY  #$FF
11  A72C  CA         DEX
12  A72D  F008       BEQ  $A737
13  A72F  C8         INY
14  A730  B99EA0     LDA  $A09E,Y
15  A733  10FA       BPL  $A72F
16  A735  30F5       BMI  $A72C
17  A737  C8         INY
18  A738  B99EA0     LDA  $A09E,Y
19  A73B  30B2       BMI  $A6EF
20  A73D  2047AB     JSR  $AB47
21  A740  D0F5       BNE  $A737

READY.
```

Ook de LIST-routine wordt via een vector in RAM aangeroepen. Deze vector staat in de geheugenplaatsen $0306 en $0307 en wijst naar $A71A. Zolang deze niet veranderd wordt is er ook niets vreemds aan de hand. Anti-LIST POKE's hebben tot doel deze vector ergens anders naar toe te laten wijzen zodat programma's inderdaad niet meer te listen zijn.

Het te LIST-en teken/token staat in de accu. Is het achtste bit niet gezet (het gaat dan om een normaal teken), is de code gelijk aan 255 (PI) of is de quotevlag gezet (tekst binnen strings) dan wordt de routine meteen verlaten, er valt dan namelijk niets te de-tokeniseren. In alle andere gevallen wordt de token met 127 verlaagd (regel 7) en dan net zolang verlaagd totdat de overeenkomstige gevonden is. Ondertussen wordt ook elke keer het volgende commandowoord gezocht (regel 13 t/m 15), is dit gevonden dan kan het afdrukken in regel 18 beginnen. Dit gaat tot de op één na laatste letter door, de laatste wordt dan via regel 21 afgedrukt, op het daar aangewezen adres volgt namelijk een AND #$7F-instructie zodat het achtste bit gewist wordt.

5/2.1.3

De interpreter

Wanneer een BASIC-programma uitgevoerd wordt dan worden de door de interpreter opgepikte tokens naar machinetaaladressen vertaald. Deze staan in een aparte tabel welke op $A00C begint. Dit klinkt op zich eenvoudig en dat is het ook wel. De enige catch zit in de manier waarop de aldus gevonden machinetaalroutine door de 64 aangeroepen wordt:

```
 0  A7E1  6C0803    JMP   ($0308)
 1  A7E4  207300    JSR   $0073
 2  A7E7  20EDA7    JSR   $A7ED
 3  A7EA  4CAEA7    JMP   $A7AE
 4  A7ED  F03C      BEQ   $A82B
 5  A7EF  E980      SBC   #$80
 6  A7F1  9011      BCC   $A804
 7  A7F3  C923      CMP   #$23
 8  A7F5  B017      BCS   $A80E
 9  A7F7  0A        ASL   A
10  A7F8  A8        TAY
11  A7F9  B90DA0    LDA   $A00D,Y
12  A7FC  48        PHA
13  A7FD  B90CA0    LDA   $A00C,Y
14  A800  48        PHA
15  A801  4C7300    JMP   $0073

READY.
```

De interpreter begint ook met de inmiddels bekende indirecte sprong. CHRGET ($0073) haalt het eerstvolgende teken uit de tekst. Regel 2 roept de eigenlijke uitvoerroutine aan waarop terug in de interpreter-hoofdlus (regel 3) gesprongen wordt.

De uitvoerroutine kijkt eerst of het einde van een regel bereikt is. Zo ja dan wordt de routine via $A82B (RTS) verlaten. Is het teken gelijk aan een token dan wordt gekeken of het een BASIC-commando-token is (het kan namelijk ook een functie- of bewerkings-token zijn). De commando-tokens zijn de eerste 33 uit de lijst, vandaar dat er met #$23 vergeleken wordt (regel 7). De aldus gevonden waarde (tussen de 0 en 32) wordt met twee vermenigvuldigd waarop het bijbehorende adres uit de tabel gelezen kan worden. De grap is nu dat dit adres op de stack gePUSHd wordt. Op die manier wordt de routine via CHRGET ($0073, regel 15) aangeroepen, deze eindigt namelijk op een RTS. Op die manier staat het eerstvolgende teken al in de accu, dit is een vereiste in de 64-interpreter. Het gevolg is wel dat de adressen in de tabel op $A00C niet de echte startadressen zijn, maar met één verlaagd zijn. Na een RTS wordt namelijk het terugkeeradres van de stack gehaald en met één verhoogd. Het resultaat is dan uiteraard het bedoelde adres.

5/2.2

Variabelen

BASIC kent 3 soorten variabelen:

Floating point variabelen, dit zijn getallen tussen de −1.70141183E+38 en 1.70141183E+38. Getallen die hieronder of hierboven komen geven een ?OVERFLOW ERROR. Het kleinste getal (het getal dat het dichtst bij 0 komt) is 2.93873588E-39, getallen die dichter bij 0 komen worden als een 0 gezien, zonder dat er een foutmelding verschijnt. Floating point variabelen worden aangeduid met de normale variabele-notatie, zonder achtervoegsel. Voorbeelden van floating point-variabelen zijn:
A=12.5
PI=3.1415926
X=-7

Integer-variabelen zijn variabelen die alleen gehele getallen tussen de -32768 en +32767 kunnen voorstellen. Integer variabelen zijn te herkennen aan het achtervoegsel '%'. Voorbeelden van integervariabelen zijn:
A%=12
X1%=3.7 (X1% krijgt dan de waarde 3)
Y2%=-4.2 (Y2% krijgt dan de waarde -5)

String-variabelen tenslotte zijn variabelen die uit letters, cijfers of grafische symbolen bestaan, welke tussen aanhalingstekens staan. De minimale lengte van een string is 0, de maximale lengte van een string is 255. String-variabelen zijn te herkennen aan het achtervoegsel '$'.
Voorbeelden van string-variabelen zijn:
A$="Hallo allemaal"
ZZ$="Dit is een goed gedefinieerde string"

Hoe slaat de Commodore 64 deze variabelen nu op? Tenslotte, als we een variabele definieren, willen we deze ook weer terug kunnen krijgen. De computer moet de variabelen dus ergens in zijn geheugen bewaren en ook weer terug kunnen vinden. Ook hier is de BASIC-vertaler verantwoordelijk voor. Het gaat als volgt in zijn werk:
Variabelen hebben altijd een lengte van 7 bytes, of ze nu van het type integer, floating point of string zijn. Dit is misschien niet erg efficient qua geheugenruimte, maar het zoeken naar een bepaalde variabele gaat wel snel. Floating point variabelen bestaan uit 2 bytes voor de naam van de variabele, de overige 5 bytes bepalen de waarde: 1 byte voor de exponent en 4 bytes voor de mantisse. De mantisse stelt een 31-bits getal tussen de 0.5 en 1 voor. Het 32ste bit houdt het teken (+ of -) van het getal bij. De waarde van de mantisse wordt als volgt berekend:
Mantisse:
01110101000000000000000000000000

Het meest linkse bit heeft de waarde 2 tot de macht -1, oftewel 1/2. Het bit daar-

2.2 Variabelen

naast heeft de waarde 2 tot de macht -2, oftewel 1/4. De bits daarnaast hebben de waardes 1/8, 1/16, 1/32 t/m 1/4.294.967.296 (2 tot de macht -32). Het eerste bit wordt verondersteld een 1 te zijn. Door deze aanname is dit bit vrij te gebruiken en wordt dan ook als teken-bit gebruikt. De waarde van de mantisse is dan ook altijd groter dan 1/2. Daarnaast worden de waardes van alle bits opgeteld, tenminste die waardes waarvan het bit op 1 staat. Bovenstaande mantisse heeft dan ook de volgende waarde:

bit 1 : 1 * 1/2^1 = 0.5 (bit 1 was per definitie 1!)
bit 2 : 1 * 1/2^2 = 0.25
bit 3 : 1 * 1/2^3 = 0.125
bit 4 : 1 * 1/2^4 = 0.0625
bit 5 : 0 * 1/2^5 = 0
bit 6 : 1 * 1/2^6 = 0.015625
bit 7 : 0 * 1/2^7 = 0
bit 8 : 1 * 1/2^8 = 0.00390625
bit 9 : 0 * 1/2^9 = 0

Daar alle volgende bits ook 0 zijn kunnen we de bovenstaande uitkomsten bij elkaar optellen. De uitkomst is 0.95703125. De exponent is een 1-bytes getal met waardes tussen de -128 en +127. De exponent is de waarde waartoe 2 tot de macht verheven wordt. Stel dat de exponent gelijk is aan +4, dan is 2 tot de macht 4 gelijk aan 16. Deze waarde wordt met de mantisse vermenigvuldigd om de uiteindelijke waarde van het floating point getal te vinden. In ons voorbeeld zou dat dus 16 * 0.95703125 = 15.3125 zijn.

Deze manier van getallen opslaan verklaart ook de reken-onnauwkeurigheid van de 64, het laatste bit heeft een waarde van 1/2^32. Dit komt overeen met 0.000000000023283... of 2 * 10^-10. Een grotere nauwkeurigheid dan 9 cijfers achter de komma is dan niet mogelijk. Bij integer-variabelen zijn ook de eerste 2 bytes gereserveerd voor de naam. In deze bytes staat bit 7 op 1, om de variabele van het floating point type te kunnen onderscheiden. Bit 7 wordt in bytes die letters voorstellen niet gebruikt, vandaar dat dit ongestraft kan. Van de overige 5 bytes worden er slechts 2 gebruikt om de waarde aan te duiden. Dit 16-bits getal vertegenwoordigt direct een waarde tussen de 0 en 65535, daar met 16 bits 2^16 (=65536) verschillende combinaties gemaakt kunnen worden (zie hoofdstukken 7, 9 en 10). Waardes boven de 32767 worden met 65536 verminderd. Op deze mannier zijn getallen tussen de -32768 en 32767 mogelijk.

Bij string variabelen duiden ook de eerste 2 bytes de naam aan, hier is alleen in het tweede byte bit 7 op 1 gezet. Van de volgende 5 bytes worden er slechts 3 gebruikt: 1 voor de lengte van de string en 2 voor het eigenlijke adres waar deze string staat. Deze adressen lopen vanaf de bovenkant van het vrije RAM-geheugen naar beneden toe. Een nadeel van deze dynamische string-allocatie is dat indien een string opnieuw gedefinieerd wordt deze onder de voorgaande string geplakt wordt. De oude versie van de string gaat dus verloren, maar blijft wel geheugen innemen. Dit gaat net zolang door totdat de onderste string tegen de bovenste variabele aankomt, daar de variabelen vanaf de bovenkant van het programma naar de bovenkant van het geheugen toe opgestapeld worden. Op dat moment moeten alle oude ongebruikte strings opgeruimd worden. Dit heet dan ook zeer toepasselijk 'garbage collection', wat vuilnis ophalen betekent. Dit garbage collecting proces kan echter enige minuten duren. Vooral

2.2 Variabelen

tijdens het sorteren van een array van strings komt het vaak voor dat de computer een tijd schijnbaar niets doet, ook de RUN/STOP-toets helpt dan niet meer. Op die momenten is de computer druk bezig de oude strings op te ruimen.

Over arrays gesproken, deze worden iets anders opgeslagen. Achter de variabelen staan de eventuele arrays in het geheugen. In floating point arrays hebben de getallen een lengte van 5 bytes, in integer arrays een lengte van 2 bytes. Alleen bij het gebruik van arrays strekt het tot aanbeveling integer variabelen te gebruiken, aangezien deze een aanzienlijke hoeveelheid geheugen sparen. De verwerking van integers door de BASIC-vertaler is wel langzamer dan de verwerking van floating point getallen, daar elke integer eerst naar floating point formaat geconverteerd moet worden voordat er wat mee gedaan kan worden. Na de berekening dient er dan ook weer teruggeconverteerd te worden.

5/2.3

Bruikbare routines

Het spreekt voor zich dat alle (machinetaal)routines in de BASIC-vertaler ook vanuit onze eigen machinetaalroutines aan te roepen zijn. Routines die beschikbaar zijn onder andere zijn conversieroutines, inlees-routines en rekenkundige routines zoals vermenigvuldiging en deling. Hoe deze routines door ons gebruikt kunnen worden en waar ze zich precies in de BASIC-ROM bevinden wordt in hoofdtuk 9 uit de doeken gedaan.

5/3

De systeemvariabelen

Inhoud

5.3 De systeemvariabelen

Systeemvariabelen zijn waardes die van belang voor het besturingssysteem of de BASIC-vertaler zijn. Bijvoorbeeld de cursorpositie. Deze wordt door het besturingssysteem in 2 geheugenplaatsen in de zero-page bewaard, een voor de X-positie en een voor de Y-positie. Welke gegevens nu op welke plaats bewaard worden is vooral voor machinetaalprogrammeurs, maar soms ook voor BASIC-programmeurs onmisbare informatie.

Hieronder volgt een uitgebreide memory map van de eerste 1019 bytes van de Commodore 64. Ze zijn van groot belang voor de besturing van de Commodore 64, vandaar dat ze in dit hoofdstuk opgenomen zijn.

5/3.1

De inhoud van de zero-page

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
D6510	0000	0	6510 I/O data direction register
R6510	0001	1	6510 I/O data register
	0002	2	Ongebruikt
ADRAY1	0003-0004	3-4	Vector: conversie floating-integer
ADRAY2	0005-0006	5-6	Vector: conversie integer-floating
CHARAC	0007	7	Zoekteken
ENDCHR	0008	8	Vlag: Quote-mode
TRMPOS	0009	9	Schermkolom voor laatste TAB
VERCK	000A	10	Vlag: 0=LOAD, 1=VERIFY
COUNT	000B	11	Wijzer in inputbuffer/ Aantal dimensies
DIMFLG	000C	12	Vlag: DIM-opdracht
VALTYP	000D	13	Data type: $FF=string, $00=numeriek
INTFLG	000E	14	Data type: $80=integer, $00=floating
GARBFL	000F	15	Vlag: Data/LIST/Garbage collection
SUBFLG	0010	16	Vlag: FN/dimensies
INPFLG	0011	17	Vlag: $00=INPUT, $40=GET, $98=READ
TANSGN	0012	18	Vlag: teken van TAN of SIN Resultaat van vergelijking
CHANNL	0013	19	Vlag: prompt bij INPUT
LINNUM	0014-0015	20-21	Integer-waarde regelnummer bij LIST, GOTO, ON, GOSUB
TEMPPT	0016	22	Wijzer naar tijdelijke string-stack
LASTPT	0017-0018	23-24	Laatste tijdelijke stringadres
TEMPST	0019-0021	25-33	Stack voor tijdelijke strings
INDEX	0022-0025	34-37	Diverse wijzers
RESHO	0026-002A	38-42	Product bij FP-vermenigvuldiging

3.1 De inhoud van de zero-page

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
TXTTAB	002B-002C	43-44	Wijzer naar start van BASIC
VARTAB	002D-002E	45-46	Wijzer naar start van variabelen
ARYTAB	002F-0030	47-48	Wijzer naar start van arrays
STREND	0031-0032	49-50	Wijzer naar einde van arrays+1
FRETOP	0033-0034	51-52	Wijzer naar onderkant van strings
FRESPC	0035-0036	53-54	Hulpwijzer voor strings
MEMSIZ	0037-0038	55-56	Wijzer naar hoogste BASIC-adres
CURLIN	0039-003A	57-58	Huidig BASIC-regelnummer
OLDLIN	003B-003C	59-60	Vorig BASIC-regelnummer
OLDTXT	003D-003E	61-62	Wijzer naar BASIC-commando bij CONT
DATLIN	003F-0040	63-64	Huidige DATA-regel
DATPTR	0041-0042	65-66	Wijzer naar huidig DATA-element
INPPTR	0043-0044	67-68	Vector: INPUT-routine
VARNAM	0045-0046	69-70	Huidige variabele-naam
VARPNT	0047-0048	71-72	Huidig variabele-adres
FORPNT	0049-004A	73-74	Wijzer naar FOR/NEXT-variabele
OPPTR	004B-004C	75-76	Tijdelijke wijzer
OPMASK	004D	77	Masker bij vergelijkingen
DEFPN	004E-004F	78-79	Wijzer voor FN
DSCPN	0050-0053	80-82	Stringdescriptors
FOUR6	0053	83	Constante voor GARBAG collection
JMPER	0054-0056	84-86	Constante $4C (JMP) + Vector voor functies
	0057-005B	87-91	Rekenaccumulator #3, FAC3
	005C-0060	92-96	Rekenaccumulator #4, FAC4
FACEXP	0061	97	Floating point accu #1, exponent
FACHO	0062-0065	98-101	FAC #1, mantisse
FACSGN	0066	102	FAC #1, teken
SGNFLG	0067	103	Teller voor polynoomsberekening
BITS	0068	104	Afrondingsbyte voor FAC #1
ARGEXP	0069	105	Floating point accu #2, exponent
ARGHO	006A-006D	106-109	FAC #2, mantisse
ARGSGN	006E	110	FAC #2, teken
ARISGN	006F	111	Vergelijking teken van FAC #1 en #2
FACOV	0070	112	Afrondingsbyte voor FAC #1
FBUFPT	0071-0072	113-114	Wijzer voor formule evaluatie
CHRGET	0073-008A	115-138	Subroutine: Haal volgende BASIC-teken
CHRGOT	0079-008A	121-138	Subroutine: Haal hetzelfde BASIC-teken
TXTPTR	007A-007B	122-123	Wijzer naar byte in BASIC-tekst
RNDX	008B-008F	139-143	Floating point RND-getal
STATUS	0090	144	Kernal I/O statusbyte (ST)

3.1 De inhoud van de zero-page

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
STKEY	0091	145	Vlag: STOP-toets/RVS-toets
SVXT	0092	146	Tijdconstante voor band
VERCK	0093	147	Vlag: 0=LOAD, 1=VERIFY
C3PO	0094	148	Vlag: Karakter op seriële bus
BSOUR	0095	149	Output-buffer voor seriële bus
SYNO	0096	150	SYNC-nummer van cassette
XSAV	0097	151	Tijdelijke data
LDTND	0098	152	Aantal open files / Index naar file-tabel
DFLTN	0099	153	Verstek input-device (0=toetsenbord)
DFLTO	009A	154	Verstek output-device (3=scherm)
PRTY	009B	155	Pariteit voor band
DPSW	009C	156	Vlag: Byte van band ontvangen
MSGFLG	009D	157	Vlag: $80=Direct-mode, $00=Programma
PTR1	009E	158	Band pass 1 checksum
PTR2	009F	159	Band pass 2 foutcorrectie
TIME	00A0-00A2	160-162	Tijd voor TI en TI$
	00A3	163	Bit-teller bij seriële uitvoer
	00A4	164	Teller voor band
CNTDN	00A5	165	Teller voor SYNCs op band
BUFPNT	00A6	166	Wijzer voor casettebuffer
INBIT	00A7	167	RS-232 input-bits/Band data
BITCI	00A8	168	RS-232 input-bit teller/Band data
RINONE	00A9	169	RS-232 vlag: Check op startbit
RIDATA	00AA	170	RS-232 input-byte buffer/Band data
RIPTRY	00AB	171	RS-232 input pariteit/Band teller
SAL	00AC-00AD	172-173	Wijzer voor cassetebuffer / Wijzer voor scrollen van scherm
EAL	00AE-00AF	174-175	Wijzer naar file-end bij LOAD/SAVE
CMP0	00B0-00B1	176-177	Tijdconstanten voor band
TAPE1	00B2-00B3	178-179	Wijzer naar start van cassettebuffer
BITTS	00B4	180	RS-232 output-bit teller/Band data
NXTBIT	00B5	181	RS-232 volgend bit/band EOT-vlag
RODATA	00B6	182	RS-232 output-byte buffer
FNLEN	00B7	183	Lengte van huidige file-naam
LA	00B8	184	Huidig logische file-nummer
SA	00B9	185	Huidige secondary address
FA	00BA	186	Huidige device-nummer
FNADR	00BB-00BC	187-188	Wijzer naar huidige file-naam
ROPRTY	00BD	189	RS-232 output pariteit/Band data
FSBLK	00BE	190	Blokteller voor band
MYCH	00BF	191	Buffer voor seriële uitvoer

3.1 De inhoud van de zero-page

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
CAS1	00C0	192	Vlag: Bandmotor
STAL	00C1-00C2	193-194	I/O startadres
MEMUSS	00C3-00C4	195-196	Tijdelijke wijzer bij band-LOAD
LSTX	00C5	197	Laatst ingedrukte toets (64=geen)
NDX	00C6	198	Aantal karakters in keyboard-buffer
RVS	00C7	199	Vlag: $01=RVS ON, $00=RVS OFF
INDX	00C8	200	Wijzer naar eind van INPUT-tekst
LXSP	00C9-00CA	201-202	Cursorpositie bij INPUT
SFDX	00CB	203	Laatst ingedrukte toets
BLNSW	00CC	204	Vlag: $00=knipperende cursor
BLNCT	00CD	205	Teller voor knipperende cursor
GDBLN	00CE	206	Karakter onder cursor
BLNON	00CF	207	Vlag: Laatste cursorstand
CRSW	00D0	208	Vlag: INPUT of GET van toetsenbord
PNT	00D1-00D2	209-210	Wijzer naar adres van scherm-regel
PNTR	00D3	211	Kolom waar de cursor zich bevindt
QTSW	00D4	212	Vlag: Quote-mode ($00=niet)
LNMX	00D5	213	Max. lengte van de scherm-regel
TBLX	00D6	214	Regel waar de cursor zich bevindt
	00D7	215	Tijdelijke data
INSRT	00D8	216	Aantal inserts
LDTB1	00D9-00F2	217-242	Adressen van scherm-regels
USER	00F3-00F4	243-244	Wijzer naar kleuren-RAM
KEYTAB	00F5-00F6	245-246	Wijzer naar toetsen-decodeer-tabel
RIBUF	00F7-00F8	247-248	Wijzer naar RS-232 inputbuffer
ROBUF	00F9-00FA	249-250	Wijzer naar RS-232 outputbuffer
FREKZP	00FB-00FE	251-254	Ongebruikt
BASZPT	00FF	255	Tijdelijke BASIC-data

5/3.2

De geheugenplaatsen 256 en verder

De 6510 microprocessor heeft zijn stack op de geheugenplaatsen 256 t/m 511. Voor het begrip stack wordt naar hoofdtuk 9 verwezen. Een klein gedeelte van de stack wordt ook door het besturingssysteem in gebruik genomen. Een compleet overzicht tot en met geheugenplaats 2023 volgt hieronder:

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
	0100-01FF	256-511	Microprocessor stack
	0100-010A	256-266	Conversiebuffer floating naar ASCII
BAD	0100-013E	256-318	Buffer voor bandcorrectie
BUF	0200-0258	512-600	Inputbuffer
LAT	0259-0262	601-610	Actieve file-nummers
FAT	0263-026C	611-620	Actieve device-nummers
SAT	026D-0276	621-630	Actieve secondary addresses
KEYD	0277-0280	631-640	Toetsenbord-buffer
MEMSTR	0281-0282	641-642	Wijzer naar RAM-begin
MEMSIZ	0283-0284	643-644	Wijzer naar RAM-einde
TIMOUT	0285	645	Vlag: Seriële timeout
COLOR	0286	646	Kleurencode van tekst
GDCOL	0287	647	Kleur onder cursor
HIBASE	0288	648	Wijzer naar beeldscherm (highbyte)
XMAX	0289	649	Lengte van de toetsenbord-buffer
RPTFLG	028A	650	Vlag: $80=repeterende toetsen
KOUNT	028B	651	Teller voor repeteersnelheid
DELAY	028C	652	Teller voor repeteervertraging
SHFLAG	028D	653	Vlag: SHIFT/CTRL/Commodore
LSTSHF	028E	654	Laatste SHIFT-stand
KEYLOG	028F-0290	655-656	Wijzer naar toetsenbord decodeer routine
MODE	0291	657	Vlag: $80=sta SHIFT/Commodore toe, $00=sta SHIFT/Commodore niet toe
AUTODN	0292	658	Vlag: $00=auto-scroll aan

3.2 De geheugenplaatsen 256 en verder

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
M51CTR	0293	659	RS-232 control register
M51CDR	0294	660	RS-232 command register
M51AJB	0295-0296	661-662	RS-232 bit-tijd
RSSTAT	0297	663	RS-232 status register
BITNUM	0298	664	RS-232 aantal bits nog te zenden
BAUDOF	0299-029A	665-666	RS-232 baud-rate. Bit-tijd in us
RIDBE	029B	667	RS-232 index naar eind van inputbuffer
RIDBS	029C	668	RS-232 index naar start van inputbuffer (HB)
RODBS	029D	669	RS-232 index naar start van outputbuffer (HB)
RODBE	029E	670	RS-232 index naar eind van outputbuffer
IRQTMP	029F-02A0	671-672	Opslag van IRQ-vector tijdens band I/O
ENABL	02A1	673	CIA #2 NMI-vlag
	02A2	674	CIA #1 timer A
	02A3	675	CIA #1 interruptvlag
	02A4	676	CIA #1 vlag voor timer A
	02A5	677	Index voor beeldschermregel
	02A6	678	Vlag: $00=NTSC, $01=PAL
	02A7-02FF	679-767	Ongebruikt
IERROR	0300-0301	768-769	Vector naar foutmelding-routine
IMAIN	0302-0303	770-771	Vector naar BASIC warmstart
ICRNCH	0304-0305	772-773	Vector naar BASIC tokenizer
IQPLOP	0306-0307	774-775	Vector naar LIST-routine
IGONE	0308-0309	776-777	Vector naar BASIC interpreter
IEVAL	030A-030B	778-779	Vector naar BASIC evaluatie-routine
SAREG	030C	780	Accumulatorinhoud bij SYS-commando
SXREG	030D	781	X-registerinhoud bij SYS-commando
SYREG	030E	782	Y-registerinhoud bij SYS-commando
SPREG	030F	783	PS-registerinhoud bij SYS-commando
USRPOK	0310	784	Constante $4C (JMP) voor USR-functie
USRADD	0311-0312	785-786	Vector naar USR-functie ($B248)
	0313	787	Ongebruikt
CINV	0314-0315	788-789	Vector naar IRQ-routine ($EA31)
CBINV	0316-0317	790-791	Vector naar BRK-routine ($FE66)
NMINV	0318-0319	792-793	Vector naar NMI-routine ($FE47)
IOPEN	031A-031B	794-795	Vector naar OPEN-routine ($F34A)
ICLOSE	031C-031D	796-797	Vector naar CLOSE-routine ($F291)
ICHKIN	031E-031F	798-799	Vector naar CHKIN-routine ($F20E)
ICKOUT	0320-0321	800-801	Vector naar CHKOUT-routine ($F250)
ICLRCH	0322-0323	802-803	Vector naar CLRCHN-routine ($F333)
IBASIN	0324-0325	804-805	Vector naar CHRIN-routine ($F157)

3.2 De geheugenplaatsen 256 en verder

NAAM	HEX ADRES	DECIMAAL ADRES	BESCHRIJVING
IBSOUT	0326-0327	806-807	Vector naar CHROUT-routine ($F1CA)
ISTOP	0328-0329	808-809	Vector naar STOP-routine ($F6ED)
IGETIN	032A-032B	810-811	Vector naar GETIN-routine ($F13E)
ICLALL	032C-032D	812-813	Vector naar CLALL-routine ($F32F)
USRCMD	032E-032F	814-815	Vector naar eigen routine ($FE66)
ILOAD	0330-0331	816-817	Vector naar LOAD-routine ($F4A5)
ISAVE	0332-0333	818-819	Vector naar SAVE-routine ($F5ED)
	0334-033B	820-827	Ongebruikt
TBUFFR	033C-03FB	828-1019	Cassettebuffer
	03FC-03FF	1020-1023	Ongebruikt

Niet alle hierboven genoemde adressen zijn tot ons voordeel te beinvloeden. In hoofdstuk 9 wordt het nut van de meeste adressen gedemonstreerd aan de hand van voorbeelden en nuttige routines.

5/4

ROM disassembly

Dit hoofdstuk bevat de complete ROM-disassembly listing van de Commodore 64. Voorzien van commentaar is dit een waardevolle naslag-rubriek.

```
a000 94 e3                        startvector $E394
a002 7b e3                        NMI-vector $E37B
a004 43 43 4d 42 41 53 49 43  "cbmbasic"

-------------------------------------------------------------
-- Vectortabel van BASIC-adressen min 1. De adressen zijn met 1 --
-- verlaagd omdat ze op de stack worden geplaatst waarna ze met --
-- RTS aangeroepen worden (via CHRGET)                       --
-------------------------------------------------------------

                          Token     Adres      Commando
-------------------------------------------------------------
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
-------------------------------------------------------------------
-- Vectortabel van BASIC-functies. De adressen worden naar       --
-- $55/$56 gekopieerd en met een JSR aangeroepen.                --
-------------------------------------------------------------------

                        Token   Adres     Commando
-------------------------------------------------------------------
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 oe 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     $b737     MID$
```

```
-------------------------------------------------------------------
--    Vectortabel van BASIC-operatoren met hun prioriteiten      --
-------------------------------------------------------------------

                          Token     Adres      Operator Hierarchiecode
-------------------------------------------------------------------
a080 79 69 b8              $aa      $b86a         +             $79
a083 79 52 b8              $ab      $b853         -             $79
a086 7b 2a ba              $ac      $ba2b         *             $7b
a089 7b 11 bb              $ad      $bb12         /             $7b
a08c 7f 7a bf              $ae      $bf7b         ^             $7f
a08f 50 e8 af              $af      $af9e       AND             $50
a092 46 e5 af              $b0      $afe6        OR             $46
a095 7d b3 bf                       $bfb4     teken '-'         $7d
a098 5a d3 ae              $a8      $aed4       NOT             $5a
a09b 64 15 b0       $b1,$b2,$b3     $b016      >,=,<            $64

-------------------------------------------------------------------
-- Tabel van BASIC commando's. De laatste letter van elk com-    --
-- mando is geSHIFT.                                             --
-------------------------------------------------------------------
a09e 45 4e
a0a0 c4 46 4f d2 4e 45 58 d4
a0a8 44 41 54 c1 49 4e 50 55
a0b0 54 a3 49 4e 50 55 d4 44
a0b8 49 cd 52 45 41 c4 4c 45
a0c0 d4 47 4f 54 cf 52 55 ce
a0c8 49 c6 52 45 53 54 4f 52
a0d0 c5 47 4f 53 55 c2 52 45
a0d8 54 55 52 ce 52 45 cd 53
a0e0 54 4f d0 4f ce 57 41 49
a0e8 d4 4c 4f 41 c4 53 41 56
a0f0 c5 56 45 52 49 46 d9 44
a0f8 45 c6 50 4f 4b c5 50 52
a100 49 4e 54 a3 50 52 49 4e
a108 d4 43 4f 4e d4 4c 49 53
a110 d4 43 4c d2 43 4d c4 53
a118 59 d3 4f 50 45 ce 43 4c
a120 4f 53 c5 47 45 d4 4e 45
a128 d7 54 41 42 a8 54 cf 46
a130 ce 53 50 43 a8 54 48 45
a138 ce 4e 4f d4 53 54 45 d0
a140 ab ad aa af de 41 4e c4
a148 4f d2 be bd bc 53 47 ce
a150 49 4e d4 41 42 d3 55 53
a158 d2 46 52 c5 50 4f d3 53
a160 51 d2 52 4e c4 4c 4f c7
a168 45 58 d0 43 4f d3 53 49
a170 ce 54 41 ce 41 54 ce 50
a178 45 45 cb 4c 45 ce 53 54
a180 52 a4 56 41 cc 41 53 c3
a188 43 48 52 a4 4c 45 46 54
a190 a4 52 49 47 48 54 a4 4d
a198 49 44 a4 47 cf 00
```

```
-------------------------------------------------------------------
-- Tabel van BASIC foutmeldingen. De laatste letter van elke    --
-- melding is geSHIFT.                                          --
-------------------------------------------------------------------
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
a1d0 44 45 56 49 43 45 20 4e        5. device not present
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        8. missing file name
a200 49 53 53 49 4e 47 20 46
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. devision 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 0e a3 1e a3
a360 24 a3 83 a3

-----------------------------------------------------------------
--              Meldingen van de BASIC-interpreter             --
-----------------------------------------------------------------
a364 0d 4f 4b 0d                  ok
a368 00 20 20 45 52 52 4f 52      error
a370 00 20 49 4e 20 00 0d 0a      in
a378 52 45 41 44 59 2e 0d 0a      ready
a380 00 0d 0a 42 52 45 41 4b      break
a388 00 a0

-----------------------------------------------------------
--    Stack-zoek routine voor FOR, NEXT en RETURN        --
-----------------------------------------------------------
a38a  ba        tsx           ;SP->X
a38b  e8        inx           ;4 bytes terugkeeradres
a38c  e8        inx           ;overslaan
a38d  e8        inx
a38e  e8        inx
a38f  bd0101    lda  $0101,x  ;stack testen op
a392  c981      cmp  #$81     ;FOR-code
a394  d021      bne  $a3b7    ;nee, dan klaar
a396  a54a      lda  $4a      ;FOR-NEXT pointer
a398  d00a      bne  $a3a4    ;=0?
a39a  bd0201    lda  $0102,x  ;ja, dan variabelenpointer
a39d  8549      sta  $49      ;van de stack naar $49/$4a
a39f  bd0301    lda  $0103,x  ;brengen
a3a2  854a      sta  $4a
a3a4  dd0301    cmp  $0103,x  ;vergelijken met stack
a3a7  d007      bne  $a3b0    ;ongelijk, dan volgende
a3a9  a549      lda  $49      ;zoeken
a3ab  dd0201    cmp  $0102,x
a3ae  f007      beq  $a3b7    ;gelijk, dan gevonden
a3b0  8a        txa           ;zoekpointer+18
a3b1  18        clc
a3b2  6912      adc  #$12
a3b4  aa        tax
a3b5  d0d8      bne  $a38f    ;en verder zoeken
a3b7  60        rts
```

```
-------------------------------------------------------------
-- Blok-verschuifroutine. Verschuift een blok met start-    --
-- adres in $5F/$60 en eindadres+1 in $5A/$5B naar gebied   --
-- gebied met eindadres+1 in $58/$59. Het nieuwe eind-      --
-- adres moet hoger dan het oude eindadres zijn!            --
-------------------------------------------------------------
a3b8  2008a4     jsr  $a408    ;checkt vrij geheugen
a3bb  8531       sta  $31      ;A/Y als pointer naar het
a3bd  8432       sty  $32      ;vrije geheugen laten wijzen
a3bf  38         sec            ;eindadres min startadres
a3c0  a55a       lda  $5a      ;van het blok geeft de blok-
a3c2  e55f       sbc  $5f      ;lengte in X/Y
a3c4  8522       sta  $22
a3c6  a8         tay
a3c7  a55b       lda  $5b
a3c9  e560       sbc  $60
a3cb  aa         tax            ;X houdt de 'pages' bij
a3cc  e8         inx
a3cd  98         tya            ;restbyte
a3ce  f023       beq  $a3f3    ;0, dan hele pages verschuiven
a3d0  a55a       lda  $5a      ;eindadres bron min restbyte
a3d2  38         sec            ;geeft adres van restbereik
a3d3  e522       sbc  $22
a3d5  855a       sta  $5a
a3d7  b003       bcs  $a3dc
a3d9  c65b       dec  $5b
a3db  38         sec            ;nieuw eindadres min restbyte
a3dc  a558       lda  $58      ;geeft het nieuwe adres van
a3de  e522       sbc  $22      ;het restbereik
a3e0  8558       sta  $58
a3e2  b008       bcs  $a3ec
a3e4  c659       dec  $59
a3e6  9004       bcc  $a3ec    ;onconditioneel
a3e8  b15a       lda  ($5a),y ;transfer-lus voor restbyte
a3ea  9158       sta  ($58),y
a3ec  88         dey
a3ed  d0f9       bne  $a3e8
a3ef  b15a       lda  ($5a),y ;transfer-lus voor hele
a3f1  9158       sta  ($58),y ;pagina
a3f3  c65b       dec  $5b
a3f5  c659       dec  $59
a3f7  ca         dex            ;aantal pages min 1
a3f8  d0f2       bne  $a3ec
a3fa  60         rts

-------------------------------------------------------------
--          Test op genoeg plaats in de stack               --
-------------------------------------------------------------
a3fb  0a         asl  a        ;accu maal 2
a3fc  693e       adc  #$3e     ;+62
a3fe  b035       bcs  $a435    ;accu>96, dan fout
a400  8522       sta  $22      ;SP < 2*accu+62
a402  ba         tsx
a403  e422       cpx  $22
a405  902e       bcc  $a435    ;dan fout
a407  60         rts
```

```
-------------------------------------------------------------
--          Test op genoeg plaats in geheugen              --
-------------------------------------------------------------
a408  c434      cpy  $34      ;Accu/Y is nieuwe eindadres
a40a  9028      bcc  $a434
a40c  d004      bne  $a412
a40e  c533      cmp  $33      ;adres<stringstart?
a410  9022      bcc  $a434    ;ja, dan OK
a412  48        pha           ;Accu..,
a413  a209      ldx  #$09
a415  98        tya           ;Y...
a416  48        pha
a417  b557      lda  $57,x    ;en $57..$60 bewaren
a419  ca        dex
a41a  10fa      bpl  $a416
a41c  2026b5    jsr  $b526    ;garbage collect
a41f  a2f7      ldx  #$f7     ;gegevens terughalen
a421  68        pla
a422  9561      sta  $61,x
a424  e8        inx
a425  30fa      bmi  $a421
a427  68        pla
a428  a8        tay
a429  68        pla
a42a  c434      cpy  $34      ;opnieuw vergelijken
a42c  9006      bcc  $a434
a42e  d005      bne  $a435    ;fout
a430  c533      cmp  $33
a432  b001      bcs  $a435    ;fout
a434  60        rts

-------------------------------------------------------------
--              OUT OF MEMORY afdrukken                    --
-------------------------------------------------------------
a435  a210      ldx  #$10     ;foutnr in X, 16=OUT OF MEMORY

-------------------------------------------------------------
--              Foutmeldingen afdrukken                    --
-------------------------------------------------------------
a437  6c0003    jmp  ($0300) ;$A43A
a43a  8a        txa          ;foutnr maal 2
a43b  0a        asl  a
a43c  aa        tax
a43d  bd26a3    lda  $a326,x ;foutadres in $22/$23
a440  8522      sta  $22
a442  bd27a3    lda  $a327,x
a445  8523      sta  $23
a447  20ccff    jsr  $ffcc    ;CLRCH, alle kanalen sluiten
a44a  a900      lda  #$00     ;input=toetsenbord
a44c  8513      sta  $13
a44e  20d7aa    jsr  $aad7    ;carriage return afdrukken
a451  2045ab    jsr  $ab45    ;'?' afdrukken
```

```
a454  a000      ldy  #$00     ;foutmelding teken voor teken
a456  b122      lda  ($22),y ;afdrukken
a458  48        pha
a459  297f      and  #$7f     ;bit 7 wissen
a45b  2047ab    jsr  $ab47    ;teken afdrukken
a45e  c8        iny
a45f  68        pla          ;oude waarde terughalen
a460  10f4      bpl  $a456    ;bit 7=0 dan opnieuw
a462  207aa6    jsr  $a67a    ;CONT uitschakelen
a465  a969      lda  #$69     ;Accu/Y naar 'ERROR' laten
a467  a0a3      ldy  #$a3     ;wijzen
a469  201eab    jsr  $ab1e    ;afdrukken
a46c  a43a      ldy  $3a      ;highbyte reelnummer=255?
a46e  c8        iny
a46f  f003      beq  $a474    ;ja, dan direct mode
a471  20c2bd    jsr  $bdc2    ;anders 'IN xxxxx' afdrukken

--------------------------------------------------------------
--                      READY. afdrukken                    --
--------------------------------------------------------------
a474  a976      lda  #$76     ;Accu/Y naar 'READY.' laten
a476  a0a3      ldy  #$a3     ;wijzen
a478  201eab    jsr  $ab1e    ;afdrukken
a47b  a980      lda  #$80     ;direct mode inschakelen
a47d  2090ff    jsr  $ff90
a480  6c0203    jmp  ($0302) ;$A483
a483  2060a5    jsr  $a560    ;input-wachtlus
a486  867a      stx  $7a      ;CHRGET-pointer naar input-
a488  847b      sty  $7b      ;buffer laten wijzen
a48a  207300    jsr  $0073    ;CHRGET, lees een teken
a48d  aa        tax           ;0?
a48e  f0f0      beq  $a480    ;ja, dan verder wachten
a490  a2ff      ldx  #$ff     ;direct mode zetten
a492  863a      stx  $3a
a494  9006      bcc  $a49c    ;eerste teken een cijfer?
a496  2079a5    jsr  $a579    ;nee, dan tokeniseren
a499  4ce1a7    jmp  $a7e1    ;en uitvoeren

--------------------------------------------------------------
--  Wissen, invoegen en vervangen van programmaregels       --
--------------------------------------------------------------
a49c  206ba9    jsr  $a96b    ;regelnummer naar $14/$15
a49f  2079a5    jsr  $a579    ;regel tokeniseren
a4a2  840b      sty  $0b      ;lengte van de regel
a4a4  2013a6    jsr  $a613    ;startadres naar $5F/$60
a4a7  9044      bcc  $a4ed    ;regel niet gevonden
a4a9  a001      ldy  #$01
a4ab  b15f      lda  ($5f),y ;startadres high van volgende
a4ad  8523      sta  $23      ;regel naar $23
a4af  a52d      lda  $2d      ;var-pointer low naar $22
a4b1  8522      sta  $22
a4b3  a560      lda  $60      ;startadres high van te wissen
```

```
a4b5  8525      sta  $25      ;regel naar $25
a4b7  a55f      lda  $5f      ;startadres low van te wissen
a4b9  88        dey           ;regel min startadres low van
a4ba  f15f      sbc  ($5f),y  ;volgende regel
a4bc  18        clc
a4bd  652d      adc  $2d      ;plus var-pointer geeft nieuwe
a4bf  852d      sta  $2d      ;var-pointer
a4c1  8524      sta  $24
a4c3  a52e      lda  $2e
a4c5  69ff      adc  #$ff
a4c7  852e      sta  $2e
a4c9  e560      sbc  $60      ;min startadres high van te
a4cb  aa        tax           ;wissen regel geeft # paginas's
a4cc  38        sec           ;startadres low van te wissen
a4cd  a55f      lda  $5f      ;regel min var-pointer geeft
a4cf  e52d      sbc  $2d      ;restbyte
a4d1  a8        tay
a4d2  b003      bcs  $a4d7
a4d4  e8        inx
a4d5  c625      dec  $25
a4d7  18        clc
a4d8  6522      adc  $22
a4da  9003      bcc  $a4df
a4dc  c623      dec  $23
a4de  18        clc
a4df  b122      lda  ($22),y ;transfer-lus
a4e1  9124      sta  ($24),y
a4e3  c8        iny
a4e4  d0f9      bne  $a4df
a4e6  e623      inc  $23
a4e8  e625      inc  $25
a4ea  ca        dex           ;aantal paginas -1
a4eb  d0f2      bne  $a4df
a4ed  2059a6    jsr  $a659    ;CHRGET-pointer terugzetten+CLR
a4f0  2033a5    jsr  $a533    ;linkpointer van regel zetten
a4f3  ad0002    lda  $0200    ;eerste teken in buffer was 0
a4f6  f088      beq  $a480    ;ja, dan wissen; klaar

a4f8  18        clc           ;regel tussenvoegen
a4f9  a52d      lda  $2d      ;var-pointer is eindadres oude
a4fb  855a      sta  $5a      ;blok, var-pointer+regellengte
a4fd  650b      adc  $0b      ;is eindadres nieuwe blok
a4ff  8558      sta  $58
a501  a42e      ldy  $2e
a503  845b      sty  $5b
a505  9001      bcc  $a508
a507  c8        iny
a508  8459      sty  $59
a50a  20b8a3    jsr  $a3b8    ;blok verschuiven
a50d  a514      lda  $14      ;regelnummer voor inputbuffer
a50f  a415      ldy  $15      ;zetten
```

```
a511  8dfe01     sta  $01fe
a514  8cff01     sty  $01ff
a517  a531       lda  $31      ;nieuwe var-pointer zetten
a519  a432       ldy  $32
a51b  852d       sta  $2d
a51d  842e       sty  $2e
a51f  a40b       ldy  $0b      ;Y bevat lengte regel
a521  88         dey
a522  b9fc01     lda  $01fc,y ;buffer naar tekst kopieren
a525  915f       sta  ($5f),y
a527  88         dey
a528  10f8       bpl  $a522
a52a  2059a6     jsr  $a659    ;CHRGET-pointer plus CLR
a52d  2033a5     jsr  $a533    ;linkpointer zetten
a530  4c80a4     jmp  $a480    ;en naar inputlus

-------------------------------------------------------------
--           Linkpointer regels opnieuw berekenen          --
-------------------------------------------------------------
a533  a52b       lda  $2b      ;programmastart naar $22/$23
a535  a42c       ldy  $2c
a537  8522       sta  $22
a539  8423       sty  $23
a53b  18         clc           ;linkadres high
a53c  a001       ldy  #$01
a53e  b122       lda  ($22),y
a540  f01d       beq  $a55f    ;0, dan programmaeinde
a542  a004       ldy  #$04     ;Y naar eerste teken v/d regel
a544  c8         iny           ;regeleind zoeken
a545  b122       lda  ($22),y
a547  d0fb       bne  $a544
a549  c8         iny           ;pointer $22/$23 + regellengte
a54a  98         tya           ;wijst naar volgende regel,
a54b  6522       adc  $22      ;lowbyte in X
a54d  aa         tax
a54e  a000       ldy  #$00     ;opbergen als linkpointer low
a550  9122       sta  ($22),y
a552  a523       lda  $23      ;pointer high berekenen
a554  6900       adc  #$00
a556  c8         iny           ;opbergen als linkpointer high
a557  9122       sta  ($22),y
a559  8622       stx  $22      ;nieuwe zoekpointer in $22/$23
a55b  8523       sta  $23
a55d  90dd       bcc  $a53c    ;onconditioneel
a55f  60         rts

-------------------------------------------------------------
--          Inpunt-wachtlus, wacht op RETURN               --
-------------------------------------------------------------
a560  a200       ldx  #$00     ;pointer in inputbuffer
a562  2012e1     jsr  $e112    ;teken lezen
a565  c90d       cmp  #$0d     ;RETURN?
```

```
a567  f00d      beq  $a576    ;ja, dan klaar
a569  9d0002    sta  $0200,x  ;teken opbergen
a56c  e8        inx           ;en pointer verhogen
a56d  e059      cpx  #$59     ;89e teken? (VIC 20!)
a56f  90f1      bcc  $a562    ;nee, dan verder
a571  a217      ldx  #$17     ;ja, dan STRING TOO LONG
a573  4c37a4    jmp  $a437    ;afdrukken
a576  4ccaaa    jmp  $aaca    ;buffer met 0 afsluiten en
                              ;pointer zetten
--------------------------------------------------------
--          Tekst in de inputbuffer tokeniseren         --
--------------------------------------------------------
a579  6c0403    jmp  ($0304)  ;$A57C
a57c  a67a      ldx  $7a      ;$7A/$7B wijst naar regel
a57e  a004      ldy  #$04     ;Y wijst naar getok. regel
a580  840f      sty  $0f      ;vlag wissen
a582  bd0002    lda  $0200,x  ;teken lezen
a585  1007      bpl  $a58e    ;niet geSHIFT
a587  c9ff      cmp  #$ff     ;PI?
a589  f03e      beq  $a5c9    ;ja, dan opbergen
a58b  e8        inx           ;anders teken overslaan
a58c  d0f4      bne  $a582
a58e  c920      cmp  #$20     ;spatie?
a590  f037      beq  $a5c9    ;ja, opbergen
a592  8508      sta  $08      ;anders teken bewaren
a594  c922      cmp  #$22     ;quote (")?
a596  f056      beq  $a5ee    ;ja
a598  240f      bit  $0f      ;V-vlag testen
a59a  702d      bvs  $a5c9    ;gezet, dan DATA-mode
a59c  c93f      cmp  #$3f     ;vraagteken (PRINT)?
a59e  d004      bne  $a5a4    ;nee
a5a0  a999      lda  #$99     ;ja, dan code voor PRINT
a5a2  d025      bne  $a5c9
a5a4  c930      cmp  #$30     ;teken<'0'?
a5a6  9004      bcc  $a5ac    ;ja
a5a8  c93c      cmp  #$3c     ;teken groter dan '<'
a5aa  901d      bcc  $a5c9    ;nee
a5ac  8471      sty  $71      ;pointer opslaan
a5ae  a000      ldy  #$00     ;tokenteller op 0
a5b0  840b      sty  $0b
a5b2  88        dey
a5b3  867a      stx  $7a      ;wijzer naar regel opslaan
a5b5  ca        dex
a5b6  c8        iny           ;zoeken...
a5b7  e8        inx
a5b8  bd0002    lda  $0200,x  ;teken uit buffer
a5bb  38        sec           ;min teken uit commandotabel
a5bc  f99ea0    sbc  $a09e,y
a5bf  f0f5      beq  $a5b6    ;gevonden?
a5c1  c980      cmp  #$80     ;of gelijk op bit 7 na?
a5c3  d030      bne  $a5f5    ;nee
```

```
a5c5  050b      ora  $0b      ;bit 7 in tokenregel zetten
a5c7  a471      ldy  $71      ;pointer terughalen
a5c9  e8        inx           ;en verhogen
a5ca  c8        iny
a5cb  99fb01    sta  $01fb,y  ;code opslaan en testen
a5ce  b9fb01    lda  $01fb,y
a5d1  f036      beq  $a609    ;code was 0, dan klaar
a5d3  38        sec           ;dubbele punt?
a5d4  e93a      sbc  #$3a
a5d6  f004      beq  $a5dc    ;ja
a5d8  c949      cmp  #$49     ;code voor DATA?
a5da  d002      bne  $a5de    ;nee
a5dc  850f      sta  $0f      ;ja, dan DATA-mode zetten
a5de  38        sec           ;REM?
a5df  e955      sbc  #$55
a5e1  d09f      bne  $a582    ;nee
a5e3  8508      sta  $08      ;ja, dan vlag zetten
a5e5  bd0002    lda  $0200,x  ;teken uit buffer lezen
a5e8  f0df      beq  $a5c9    ;0, dan klaar
a5ea  c508      cmp  $08      ;als ASCII opbergen?
a5ec  f0db      beq  $a5c9    ;nee
a5ee  c8        iny           ;opslaan
a5ef  99fb01    sta  $01fb,y
a5f2  e8        inx           ;volgende teken
a5f3  d0f0      bne  $a5e5    ;opbergen
a5f5  a67a      ldx  $7a      ;wijzer naar buffer terughalen
a5f7  e60b      inc  $0b      ;tokenteller+1
a5f9  c8        iny           ;Y wijst naar volgende entry
a5fa  b99da0    lda  $a09d,y  ;in de tabel
a5fd  10fa      bpl  $a5f9
a5ff  b99ea0    lda  $a09e,y  ;hele tabel gehad?
a602  d0b4      bne  $a5b8    ;nee
a604  bd0002    lda  $0200,x  ;volgend teken
a607  10be      bpl  $a5c7    ;<128
a609  99fd01    sta  $01fd,y  ;anders opslaan
a60c  c67b      dec  $7b      ;CHRGET-pointer terugzetten
a60e  a9ff      lda  #$ff
a610  857a      sta  $7a
a612  60        rts

--------------------------------------------------------------
--       Adres van een programmaregel berekenen             --
--------------------------------------------------------------
a613  a52b      lda  $2b      ;Accu/X bevat wijzer naar
a615  a62c      ldx  $2c      ;programmastart
a617  a001      ldy  #$01
a619  855f      sta  $5f      ;opbergen in $5F/$60
a61b  8660      stx  $60
a61d  b15f      lda  ($5f),y  ;linkpointer high testen
a61f  f01f      beq  $a640    ;0, dan einde programma, C=0
a621  c8        iny
a622  c8        iny
```

```
a623  a515      lda  $15      ;regelnummer high
a625  d15f      cmp  ($5f),y  ;vergelijken
a627  9018      bcc  $a641    ;kleiner, niet gevonden; C=0
a629  f003      beq  $a62e    ;gelijk, dan low testen
a62b  88        dey
a62c  d009      bne  $a637    ;onconditioneel
a62e  a514      lda  $14      ;regelnummer low
a630  88        dey           ;vergelijken
a631  d15f      cmp  ($5f),y
a633  900c      bcc  $a641    ;kleiner, niet gevonden; C=0
a635  f00a      beq  $a641    ;gevonden, C=1
a637  88        dey           ;linkpointer high in X
a638  b15f      lda  ($5f),y
a63a  aa        tax
a63b  88        dey           ;linkpointer low in accu
a63c  b15f      lda  ($5f),y
a63e  b0d7      bcs  $a617    ;verder zoeken
a640  18        clc           ;niet gevonden, C=0
a641  60        rts

-------------------------------------------------------------
--                  BASIC-commando NEW                     --
-------------------------------------------------------------
a642  d0fd      bne  $a641    ;nog tekens? dan SYNTAX ERROR
a644  a900      lda  #$00     ;0 in de eerste twee bytes van
a646  a8        tay           ;de programmatekst schrijven
a647  912b      sta  ($2b),y
a649  c8        iny
a64a  912b      sta  ($2b),y
a64c  a52b      lda  $2b      ;programmastart+2 geeft
a64e  18        clc           ;variabelen-start
a64f  6902      adc  #$02
a651  852d      sta  $2d
a653  a52c      lda  $2c
a655  6900      adc  #$00
a657  852e      sta  $2e
a659  208ea6    jsr  $a68e    ;CHRGET-pointer zetten

-------------------------------------------------------------
--                  BASIC-commando: CLR                    --
-------------------------------------------------------------
a65e  d02d      bne  $a68d    ;nog tekens, dan SYNTAX ERROR
a660  20e7ff    jsr  $ffe7    ;alle kanalen sluiten
a663  a537      lda  $37      ;stringpointer := RAM-top
a665  a438      ldy  $38
a667  8533      sta  $33
a669  8434      sty  $34
a66b  a52d      lda  $2d      ;array-pointer := var-pointer
a66d  a42e      ldy  $2e
a66f  852f      sta  $2f
```

```
a671  8430      sty  $30
a673  8531      sta  $31      ;var-einde := var-start
a675  8432      sty  $32
a677  201da8    jsr  $a81d    ;RESTORE
a67a  a219      ldx  #$19     ;pointer in stringstack
a67c  8616      stx  $16
a67e  68        pla           ;terugkeeradres van stack
a67f  a8        tay           ;halen (vies!)
a680  68        pla
a681  a2fa      ldx  #$fa     ;stackpointer terugzetten
a683  9a        txs
a684  48        pha           ;en terugkeeradres weer op
a685  98        tya           ;stack zetten
a686  48        pha
a687  a900      lda  #$00     ;CONT uitschakelen
a689  853e      sta  $3e
a68b  8510      sta  $10      ;functie-vlag wissen
a68d  60        rts

-------------------------------------------------------------
--                  CHRGET-pointer zetten                  --
-------------------------------------------------------------
a68e  18        clc           ;programmawijzer min 1 geeft
a68f  a52b      lda  $2b      ;CHRGET-pointer
a691  69ff      adc  #$ff
a693  857a      sta  $7a
a695  a52c      lda  $2c
a697  69ff      adc  #$ff
a699  857b      sta  $7b
a69b  60        rts

-------------------------------------------------------------
--                  BASIC-commando: LIST                   --
-------------------------------------------------------------
a69c  9006      bcc  $a6a4    ;volgt er een getal?
a69e  f004      beq  $a6a4    ;of niets?
a6a0  c9ab      cmp  #$ab     ;of een '-'
a6a2  d0e9      bne  $a68d    ;nee, dan SYNTAX ERROR
a6a4  206ba9    jsr  $a96b    ;regelnummer naar $14/$15
a6a7  2013a6    jsr  $a613    ;startadres zoeken; in $5F/$60
a6aa  207900    jsr  $0079    ;laatste teken opnieuw lezen
a6ad  f00c      beq  $a6bb    ;niets meer
a6af  c9ab      cmp  #$ab     ;anders moet het een '-' zijn
a6b1  d08e      bne  $a641    ;ook niet, dan SYNTAX ERROR
a6b3  207300    jsr  $0073    ;CHRGET
a6b6  206ba9    jsr  $a96b    ;en volgend regelnummer halen
a6b9  d086      bne  $a641    ;indien nog tekens dan SYNTAX
a6bb  68        pla           ;terugkeeradres weghalen (bah)
a6bc  68        pla
a6bd  a514      lda  $14      ;tweede regelnr = 0?
a6bf  0515      ora  $15
a6c1  d006      bne  $a6c9    ;nee
```

```
a6c3  a9ff      lda  #$ff     ;ja, dan 2e nummer:= 65535
a6c5  8514      sta  $14
a6c7  8515      sta  $15
a6c9  a001      ldy  #$01
a6cb  840f      sty  $0f      ;quote-mode uitschakelen
a6cd  b15f      lda  ($5f),y  ;linkadres high=0?
a6cf  f043      beq  $a714    ;ja, dan warm start (jammer)
a6d1  202ca8    jsr  $a82c    ;RUN/STOP-toets checken
a6d4  20d7aa    jsr  $aad7    ;carriage return
a6d7  c8        iny           ;regelnr naar A/X
a6d8  b15f      lda  ($5f),y
a6da  aa        tax
a6db  c8        iny
a6dc  b15f      lda  ($5f),y
a6de  c515      cmp  $15      ;laatste regel bereikt?
a6e0  d004      bne  $a6e6
a6e2  e414      cpx  $14
a6e4  f002      beq  $a6e8
a6e6  b02c      bcs  $a714    ;ja, dan klaar
a6e8  8449      sty  $49      ;Y bewaren
a6ea  20cdbd    jsr  $bdcd    ;regelnr afdrukken
a6ed  a920      lda  #$20     ;spatie naar accu
a6ef  a449      ldy  $49      ;Y terughalen
a6f1  297f      and  #$7f     ;bit 7 wissen
a6f3  2047ab    jsr  $ab47    ;accu afdrukken
a6f6  c922      cmp  #$22     ;quote?
a6f8  d006      bne  $a700    ;nee
a6fa  a50f      lda  $0f      ;ja, dan quote-mode
a6fc  49ff      eor  #$ff     ;omschakelen
a6fe  850f      sta  $0f
a700  c8        iny           ;pointer+1
a701  f011      beq  $a714    ;stoppen na 255 tekens
a703  b15f      lda  ($5f),y  ;volgende teken halen
a705  d010      bne  $a717    ;einde?
a707  a8        tay           ;Y:=0
a708  b15f      lda  ($5f),y  ;startadres van volgende regel
a70a  aa        tax           ;naar $5F/$60
a70b  c8        iny
a70c  b15f      lda  ($5f),y
a70e  865f      stx  $5f
a710  8560      sta  $60
a712  d0b5      bne  $a6c9    ;onconditioneel
a714  4c86e3    jmp  $e386    ;READY afdrukken (warm start)

a717  6c0603    jmp  ($0306)  ;$A71A
a71a  10d7      bpl  $a6f3    ;code<128, dan afdrukken
a71c  c9ff      cmp  #$ff     ;PI
a71e  f0d3      beq  $a6f3    ;ja, dan afdrukken
a720  240f      bit  $0f      ;quote-mode aan?
a722  30cf      bmi  $a6f3    ;ja, dan ook afdrukken
a724  38        sec           ;tokennummer naar X
a725  e97f      sbc  #$7f
```

```
a727  aa        tax
a728  8449      sty  $49      ;Y opslaan
a72a  a0ff      ldy  #$ff     ;pointer in commando-tabel
a72c  ca        dex           ;tokennummer-1 (!)
a72d  f008      beq  $a737    ;0, dan gevonden
a72f  c8        iny           ;anders volgend woord in tabel
a730  b99ea0    lda  $a09e,y  ;zoeken
a733  10fa      bpl  $a72f
a735  30f5      bmi  $a72c    ;en tokennummer opnieuw testen
a737  c8        iny           ;teken uit tabel lezen
a738  b99ea0    lda  $a09e,y
a73b  30b2      bmi  $a6ef    ;bit 7 gezet?
a73d  2047ab    jsr  $ab47    ;nee, dan teken afdrukken
a740  d0f5      bne  $a737    ;volgende teken

--------------------------------------------------------------
--                  BASIC-commando: FOR                     --
--------------------------------------------------------------
a742  a980      lda  #$80     ;vlag zetten om integer-
a744  8510      sta  $10      ;variabelen te verhinderen
a746  20a5a9    jsr  $a9a5    ;LET: variabele toekennen
a749  208aa3    jsr  $a38a    ;FOR met zelfde var. zoeken
a74c  d005      bne  $a753    ;niet gevonden
a74e  8a        txa           ;stackpointer op oude lus
a74f  690f      adc  #$0f
a751  aa        tax
a752  9a        txs
a753  68        pla           ;terugkeer wissen
a754  68        pla
a755  a909      lda  #$09     ;check plaats op stack
a757  20fba3    jsr  $a3fb
a75a  2006a9    jsr  $a906    ;':' zoeken, offset naar Y
a75d  18        clc           ;startadres van FOR-loop
a75e  98        tya           ;berekenen en op stack zetten
a75f  657a      adc  $7a
a761  48        pha
a762  a57b      lda  $7b
a764  6900      adc  #$00
a766  48        pha
a767  a53a      lda  $3a      ;regelnummer op stack
a769  48        pha
a76a  a539      lda  $39`
a76c  48        pha
a76d  a9a4      lda  #$a4     ;check op 'TO'
a76f  20ffae    jsr  $aeff
a772  208dad    jsr  $ad8d    ;numerieke waarde testen
a775  208aad    jsr  $ad8a    ;en berekenen (eindwaarde)
a778  a566      lda  $66      ;teken naar bit 7
a77a  097f      ora  #$7f     ;andere bits zetten
a77c  2562      and  $62      ;bits 0-6 naar accu
a77e  8562      sta  $62      ;en met teken opbergen
a780  a98b      lda  #$8b     ;terugkeeradres $A78B in A/Y
```

```
a782  a0a7      ldy  #$a7
a784  8522      sta  $22      ;opslaan
a786  8423      sty  $23
a788  4c43ae    jmp  $ae43    ;'TO' naar stack en naar $A78B
a78b  a9bc      lda  #$bc     ;A/Y op $B9BC (adres van con-
a78d  a0b9      ldy  #$b9     ;stante 1)
a78f  20a2bb    jsr  $bba2    ;als STEP-waarde naar FAC
a792  207900    jsr  $0079    ;volgend teke lezen
a795  c9a9      cmp  #$a9     ;'STEP'-code?
a797  d006      bne  $a79f    ;nee, dan waarde 1 behouden
a799  207300    jsr  $0073    ;anders num. waarde testen
a79c  208aad    jsr  $ad8a    ;en inlezen
a79f  202bbc    jsr  $bc2b    ;teken halen
a7a2  2038ae    jsr  $ae38    ;teken+waarde naar stack
a7a5  a54a      lda  $4a      ;var-pointer naar stack
a7a7  48        pha
a7a8  a549      lda  $49
a7aa  48        pha
a7ab  a981      lda  #$81     ;FOR-code op stack
a7ad  48        pha

-------------------------------------------------------------
--                      Interpreter-lus                    --
-------------------------------------------------------------
a7ae  202ca8    jsr  $a82c    ;STOP-toets checken
a7b1  a57a      lda  $7a      ;CHRGET-pointer naar A/Y
a7b3  a47b      ldy  $7b
a7b5  c002      cpy  #$02     ;direct mode?
a7b7  ea        nop
a7b8  f004      beq  $a7be    ;ja
a7ba  853d      sta  $3d      ;CONT-pointer zetten
a7bc  843e      sty  $3e
a7be  a000      ldy  #$00     ;huidig teken lezen
a7c0  b17a      lda  ($7a),y
a7c2  d043      bne  $a807    ;geen regeleind
a7c4  a002      ldy  #$02     ;link-pointer overslaan
a7c6  b17a      lda  ($7a),y ;nieuw teken
a7c8  18        clc
a7c9  d003      bne  $a7ce    ;<>0
a7cb  4c4ba8    jmp  $a84b    ;anders programmaeinde
a7ce  c8        iny           ;regelnr. naar $39/$3A
a7cf  b17a      lda  ($7a),y
a7d1  8539      sta  $39
a7d3  c8        iny
a7d4  b17a      lda  ($7a),y
a7d6  853a      sta  $3a
a7d8  98        tya           ;CHRGET-pointer naar tekst-
a7d9  657a      adc  $7a      ;start van regel
a7db  857a      sta  $7a
a7dd  9002      bcc  $a7e1
a7df  e67b      inc  $7b
a7e1  6c0803    jmp  ($0308) ;$A7E4
```

```
a7e4  207300     jsr  $0073    ;teken lezen
a7e7  20eda7     jsr  $a7ed    ;regel verwerken
a7ea  4caea7     jmp  $a7ae    ;en terug naar begin

-------------------------------------------------------------
--                      Regel verwerken                    --
-------------------------------------------------------------
a7ed  f03c       beq  $a82b    ;niets meer? dan klaar
a7ef  e980       sbc  #$80     ;teken min 128
a7f1  9011       bcc  $a804    ;<0, dan geen token, dus LET
a7f3  c923       cmp  #$23     ;token>162?
a7f5  b017       bcs  $a80e    ;ja
a7f7  0a         asl  a        ;(teken-128)*2 geeft offset
a7f8  a8         tay            ;voor tabel
a7f9  b90da0     lda  $a00d,y  ;adres van de routine op
a7fc  48         pha            ;stack zetten
a7fd  b90ca0     lda  $a00c,y
a800  48         pha
a801  4c7300     jmp  $0073    ;CHRGET plus uitvoeren
a804  4ca5a9     jmp  $a9a5    ;sprong naar LET-commando

a807  c93a       cmp  #$3a     ;dubbele punt?
a809  f0d6       beq  $a7e1    ;ja
a80b  4c08af     jmp  $af08    ;anders SYNTAX ERROR

a80e  c94b       cmp  #$4b     ;GO-code?
a810  d0f9       bne  $a80b    ;nee, dan SYNTAX ERROR
a812  207300     jsr  $0073    ;volgend teken
a815  a9a4       lda  #$a4     ;code voor 'TO'?
a817  20ffae     jsr  $aeff    ;zo niet dan SYNTAX ERROR
a81a  4ca0a8     jmp  $a8a0    ;naar GOTO-commando

-------------------------------------------------------------
--                  BASIC-commando: RESTORE                --
-------------------------------------------------------------
a81d  38         sec            ;programmateller min 1
a81e  a52b       lda  $2b
a820  e901       sbc  #$01
a822  a42c       ldy  $2c
a824  b001       bcs  $a827
a826  88         dey
a827  8541       sta  $41      ;als DATA-pointer opslaan
a829  8442       sty  $42
a82b  60         rts

-------------------------------------------------------------
--   BASIC-commando's: END en STOP; uitlezen STOP-toets    --
-------------------------------------------------------------
a82c  20e1ff     jsr  $ffe1    ;STOP-toets lezen
a82f  b001       bcs  $a832    ;naar STOP-commando
a831  18         clc            ;vlag voor END
a832  d03c       bne  $a870    ;STOP-toets niet gedrukt
```

```
a834  a57a      lda  $7a      ;CHRGET-pointer naar A/Y
a836  a47b      ldy  $7b
a838  a63a      ldx  $3a      ;regelnummer high naar X
a83a  e8        inx           ;direct mode? (255)
a83b  f00c      beq  $a849    ;ja
a83d  853d      sta  $3d      ;CHRGET-pointer naar $3D/$3E
a83f  843e      sty  $3e
a841  a539      lda  $39      ;regelnr. voor CONT vastleggen
a843  a43a      ldy  $3a
a845  853b      sta  $3b
a847  843c      sty  $3c
a849  68        pla           ;terugkeeradres wissen
a84a  68        pla
a84b  a981      lda  #$81     ;startadres 'BREAK' naar A/Y
a84d  a0a3      ldy  #$a3     ;(=$A381)
a84f  9003      bcc  $a854    ;END?
a851  4c69a4    jmp  $a469    ;nee, dan 'BREAK' afdrukken
a854  4c86e3    jmp  $e386    ;warm start (READY)

--------------------------------------------------------------
--                    BASIC-commando: CONT                  --
--------------------------------------------------------------
a857  d017      bne  $a870    ;nog tekens, dan SYNTAX ERROR
a859  a21a      ldx  #$1a     ;code voor CAN'T CONTINUE
a85b  a43e      ldy  $3e      ;CONT toegestaan?
a85d  d003      bne  $a862    ;ja
a85f  4c37a4    jmp  $a437    ;nee, dan foutmelding
a862  a53d      lda  $3d      ;CHRGET-pointer := CONT-pointer
a864  857a      sta  $7a
a866  847b      sty  $7b
a868  a53b      lda  $3b
a86a  a43c      ldy  $3c      ;regelnummer herstellen
a86c  8539      sta  $39
a86e  843a      sty  $3a
a870  60        rts
```

```
;-----------------------------------------------------------------
;---                  BASIC-commando: RUN                      ---
;-----------------------------------------------------------------
a871  08        php           ;vlaggen redden
a872  a900      lda  #$00     ;vlag voor program-mode
a874  2090ff    jsr  $ff90    ;zetten
a877  28        plp           ;vlaggen terughalen
a878  d003      bne  $a87d    ;nog regelnummers?
a87a  4c59a6    jmp  $a659    ;nee, dan CLR
a87d  2060a6    jsr  $a660    ;anders CLR en
a880  4c97a8    jmp  $a897    ;GOTO

;-----------------------------------------------------------------
;---                  BASIC-commando: GOSUB                    ---
;-----------------------------------------------------------------
a883  a903      lda  #$03     ;6 vrije stack-plaatsen?
a885  20fba3    jsr  $a3fb    ;testen
a888  a57b      lda  $7b      ;program-counter
a88a  48        pha           ;opbergen
a88b  a57a      lda  $7a
a88d  48        pha
a88e  a53a      lda  $3a      ;regelnummer
a890  48        pha           ;opbergen
a891  a539      lda  $39
a893  48        pha
a894  a98d      lda  #$8d     ;code voor GOSUB
a896  48        pha           ;op de stack zetten
a897  207900    jsr  $0079    ;CHRGOT (laatste teken)
a89a  20a0a8    jsr  $a8a0    ;GOTO
a89d  4caea7    jmp  $a7ae    ;en naar de interpreter

;-----------------------------------------------------------------
;---                  BASIC-commando: GOTO                     ---
;-----------------------------------------------------------------
a8a0  206ba9    jsr  $a96b    ;regelnummer lezen
a8a3  2009a9    jsr  $a909    ;en opzoeken
a8a6  38        sec           ;vergelijk huidige regel
a8a7  a539      lda  $39      ;met GOTO-regel
a8a9  e514      sbc  $14      ;kleiner?
a8ab  a53a      lda  $3a
a8ad  e515      sbc  $15
a8af  b00b      bcs  $a8bc    ;nee
a8b1  98        tya
a8b2  38        sec
a8b3  657a      adc  $7a      ;dus wordt er gezocht
a8b5  a67b      ldx  $7b      ;vanaf huidige regel
a8b7  9007      bcc  $a8c0
a8b9  e8        inx
a8ba  b004      bcs  $a8c0
```

```
a8bc   a52b       lda   $2b       ;en anders vanaf de
a8be   a62c       ldx   $2c       ;eerste regel
a8c0   2017a6     jsr   $a617     ;zoeken
a8c3   901e       bcc   $a8e3     ;niet gevonden, dan fout
a8c5   a55f       lda   $5f       ;anders program-counter
a8c7   e901       sbc   #$01      ;op nieuwe regel zetten
a8c9   857a       sta   $7a
a8cb   a560       lda   $60
a8cd   e900       sbc   #$00
a8cf   857b       sta   $7b
a8d1   60         rts
```

```
---                    BASIC-commando: RETURN                    ---
```

```
a8d2   d0fd       bne   $a8d1     ;nog tekens? dan weg
a8d4   a9ff       lda   #$ff
a8d6   854a       sta   $4a
a8d8   208aa3     jsr   $a38a     ;GOSUB zoeken (op stack)
a8db   9a         txs
a8dc   c98d       cmp   #$8d      ;vergelijken met GOSUB-
a8de   f00b       beq   $a8eb     ;token
a8e0   a20c       ldx   #$0c      ;niet? dan RETURN WITHOUT
                                   GOSUB ERROR
```

```
---                    Enige foutmeldingen                       ---
```

```
a8e2   2c         byt   $2c
a8e3   a211       ldx   #$02      ;UNDEF'D STATEMENT
a8e5   4c37a4     jmp   $a437     ;foutmelden
a8e8   4c08af     jmp   $af08     ;SYNTAX ERROR
```

```
---                    RETURN uitvoeren                          ---
```

```
a8eb   68         pla               ;code voor GOSUB
a8ec   68         pla               ;regelnummer van stack
a8ed   8539       sta   $39         ;opbergen
a8ef   68         pla
a8f0   853a       sta   $3a
a8f2   68         pla               ;program-counter idem
a8f3   857a       sta   $7a         ;opbergen
a8f5   68         pla
a8f6   857b       sta   $7b
```

```
---                    BASIC-commando: DATA                      ---
```

```
a8f8   2006a9     jsr   $a906     ;volgend commando zoeken
a8fb   98         tya             ;offset optellen bij
```

```
a8fc  18        clc             ;program-counter
a8fd  657a      adc  $7a
a8ff  857a      sta  $7a
a901  9002      bcc  $a905
a903  e67b      inc  $7b
a905  60        rts             ;klaar

--------------------------------------------------------------------------------
---  Offset voor volgende dubbele punt (:) vinden  ---
--------------------------------------------------------------------------------
a906  a23a      ldx  #$3a       ;": (dubbele punt)
a908  2c        byt  $2c

--------------------------------------------------------------------------------
---          Offset voor regeleinde (0) vinden          ---
--------------------------------------------------------------------------------
a909  a200      ldx  #$00       ;0 (regeleinde)
a90b  8607      stx  $07        ;zoekwaarde
a90d  a000      ldy  #$00       ;Y bevat offset
a90f  8408      sty  $08        ;opbergen
a911  a508      lda  $08        ;begin de zoeklus
a913  a607      ldx  $07
a915  8507      sta  $07
a917  8608      stx  $08
a919  b17a      lda  ($7a),y ;teken uit regel
a91b  f0e8      beq  $a905      ;regeleinde?
a91d  c508      cmp  $08        ;nee, dan vergelijken
a91f  f0e4      beq  $a905      ;gevonden
a921  c8        iny             ;niet gevonden
a922  c922      cmp  #$22       ;quote (")?
a924  d0f3      bne  $a919      ;nee
a926  f0e9      beq  $a911      ;ja, dan opbergen en naar
                                 volgende quote zoeken

--------------------------------------------------------------------------------
---                    BASIC-commando: IF                    ---
--------------------------------------------------------------------------------
a928  209ead    jsr  $ad9e      ;FRMEVL: berekenen
a92b  207900    jsr  $0079      ;CHRGOT
a92e  c989      cmp  #$89       ;vgl. met GOTO
a930  f005      beq  $a937      ;ja
a932  a9a7      lda  #$a7       ;anders checken op THEN
a934  20ffae    jsr  $aeff      ;fout indien heen THEN
a937  a561      lda  $61        ;uitkomst IF
a939  d005      bne  $a940      ;waar? dan uitvoeren

--------------------------------------------------------------------------------
---                    BASIC-commando: REM                   ---
--------------------------------------------------------------------------------
a93b  2009a9    jsr  $a909      ;niet? dan volg. regel
```

```
a93e   f0bb      beq   $a8fb    ;programcounter zetten
a940   207900    jsr   $0079    ;CHRGOT
a943   b003      bcs   $a948    ;cijfer?
a945   4ca0a8    jmp   $a8a0    ;dan GOTO
a948   4ceda7    jmp   $a7ed    ;en anders commando's

-------------------------------------------------------------------------------
---                        BASIC-commando: ON                               ---
-------------------------------------------------------------------------------
a94b   209eb7    jsr   $b79e    ;byte-waarde lezen
a94e   48        pha             ;token op stack
a94f   c98d      cmp   #$8d     ;GOSUB?
a951   f004      beq   $a957    ;ja
a953   c989      cmp   #$89     ;GOTO?
a955   d091      bne   $a8e8    ;nee, dan SYNTAX ERROR
a957   c665      dec   $65      ;ON-teller verlagen
a959   d004      bne   $a95f    ;ongelijk 0
a95b   68        pla             ;anders token terughalen
a95c   4cefa7    jmp   $a7ef    ;en commando uitvoeren

a95f   207300    jsr   $0073    ;volgend teken lezen
a962   206ba9    jsr   $a96b    ;regelnummer halen
a965   c92c      cmp   #$2c     ;volgend teken een ','?
a967   f0ee      beq   $a957    ;ja, dan klaar
a969   68        pla             ;anders token weghalen en
a96a   60        rts             ;commando beeindigen

-------------------------------------------------------------------------------
--    Regelnummer naar 16-bits getal omzetten                               --
-------------------------------------------------------------------------------
a96b   a200      ldx   #$00     ;beginnen met 0
a96d   8614      stx   $14      ;opbergen
a96f   8615      stx   $15
a971   b0f7      bcs   $a96a
a973   e92f      sbc   #$2f     ;min $30 (carry=0)
a975   8507      sta   $07
a977   a515      lda   $15      ;highbyte
a979   8522      sta   $22
a97b   c919      cmp   #$19     ;regel>64000?
a97d   b0d4      bcs   $a953    ;ja
a97f   a514      lda   $14      ;hele 16-bits-getal maal
a981   0a        asl   a        ;4
a982   2622      rol   $22
a984   0a        asl   a
a985   2622      rol   $22
a987   6514      adc   $14      ;plus nog eenmaal er bij
a989   8514      sta   $14      ;optellen geeft maal 5
a98b   a522      lda   $22
a98d   6515      adc   $15
a98f   8515      sta   $15
```

```
a991  0614      asl  $14      ;en nog eens maal 2 is
a993  2615      rol  $15      ;maal 10
a995  a514      lda  $14
a997  6507      adc  $07      ;+nummer
a999  8514      sta  $14
a99b  9002      bcc  $a99f
a99d  e615      inc  $15
a99f  207300    jsr  $0073    ;volgend teken
a9a2  4c71a9    jmp  $a971    ;en klaar

--------------------------------------------------------------------------------
---                        BASIC-commando: LET                               ---
--------------------------------------------------------------------------------
a9a5  208bb0    jsr  $b08b    ;zoek de variabele
a9a8  8549      sta  $49      ;adres opbergen
a9aa  844a      sty  $4a
a9ac  a9b2      lda  ##b2     ;volgend teken moet een
a9ae  20ffae    jsr  $aeff    ;'=' zijn, andres fout
a9b1  a50e      lda  $0e      ;integer-vlag
a9b3  48        pha           ;en
a9b4  a50d      lda  $0d      ;type-vlag bewaren
a9b6  48        pha
a9b7  209ead    jsr  $ad9e    ;FRMEVL: berekenen
a9ba  68        pla           ;type-vlag terughalen
a9bb  2a        rol  a
a9bc  2090ad    jsr  $ad90    ;en controleren
a9bf  d018      bne  $a9d9    ;string
a9c1  68        pla           ;integer?
a9c2  1012      bpl  $a9d6    ;of real?

--------------------------------------------------------------------------------
---                     INTEGER-waarde toekennen                             ---
--------------------------------------------------------------------------------
a9c4  201bbc    jsr  $bc1b    ;FAC afronden
a9c7  20bfb1    jsr  $b1bf    ;en tot integer maken
a9ca  a000      ldy  ##00     ;waarde aan
a9cc  a564      lda  $64      ;variabele
a9ce  9149      sta  ($49),y  ;toekennen
a9d0  c8        iny
a9d1  a565      lda  $65
a9d3  9149      sta  ($49),y
a9d5  60        rts

--------------------------------------------------------------------------------
---                       REAL-waarde toekennen                              ---
--------------------------------------------------------------------------------
a9d6  4cd0bb    jmp  $bbd0    ;REAL naar variabele

--------------------------------------------------------------------------------
---                     STRING-waarde toekennen                              ---
```

```
a9d9  68        pla           ;INT-vlag weghalen
a9da  a44a      ldy  $4a      ;var-adres (highbyte)
a9dc  c0bf      cpy  #$bf     ;vgl TI$
a9de  d04c      bne  $aa2c    ;nee

---------------------------------------------------------------
--                  TI$ een waarde toekennen                 --
---------------------------------------------------------------
a9e0  20a6b6    jsr  $b6a6    ;ja, dan string lezen
a9e3  c906      cmp  #$06     ;vergelijk lengte met 6
a9e5  d03d      bne  $aa24    ;ILLEGAL QUANTITY
a9e7  a000      ldy  #$00     ;tijdelijke geheugen-
a9e9  8461      sty  $61      ;plaatsen wissen
a9eb  8466      sty  $66
a9ed  8471      sty  $71      ;(positieteller)
a9ef  201daa    jsr  $aa1d    ;volgt er een cijfer?
a9f2  20e2ba    jsr  $bae2    ;FAC met 10 vermenigv.
a9f5  e671      inc  $71      ;positieteller verhogen
a9f7  a471      ldy  $71
a9f9  201daa    jsr  $aa1d    ;test op cijfer
a9fc  200cbc    jsr  $bc0c    ;FAC naar ARG
a9ff  aa        tax           ;A redden in X
aa00  f005      beq  $aa07    ;FAC = 0?
aa02  e8        inx           ;accu + 1
aa03  8a        txa           ;terug naar accu
aa04  20edba    jsr  $baed    ;FAC:=FAC+ARG
aa07  a471      ldy  $71      ;teller
aa09  c8        iny
aa0a  c006      cpy  #$06     ;al zes posities?
aa0c  d0df      bne  $a9ed    ;nee
aa0e  20e2ba    jsr  $bae2    ;FAC nog eenmaal * 10
aa11  209bbc    jsr  $bc9b    ;FAC afronden
aa14  a664      ldx  $64      ;Tijd in 63/64/65
aa16  a463      ldy  $63
aa18  a565      lda  $65
aa1a  4cdbff    jmp  $ffdb    ;tijd zetten

---------------------------------------------------------------
--            Testen of teken een cijfer is                  --
---------------------------------------------------------------
aa1d  b122      lda  ($22),y ;lees teken
aa1f  208000    jsr  $0080   ;test op cijfer
aa22  9003      bcc  $aa27   ;ja
aa24  4c48b2    jmp  $b248   ;nee, dan ILL. QUANTITY
aa27  e92f      sbc  #$2f    ;-$30 (carry=0)
aa29  4c7ebd    jmp  $bd7e   ;naar FAC en ARG

---------------------------------------------------------------
--                STRING een waarde toekennen                --
```

```
aa2c  a002      ldy  ##$02     ;offset voor stringadres
aa2e  b164      lda  ($64),y   ;stringadres high
aa30  c534      cmp  $34       ;vergelijken met $-start
aa32  9017      bcc  $aa4b     ;kleiner, dan in prog
aa34  d007      bne  $aa3d     ;groter
aa36  88        dey            ;indien gelijk, dan
aa37  b164      lda  ($64),y   ;lowbyte testen
aa39  c533      cmp  $33       ;en vergelijken
aa3b  900e      bcc  $aa4b     ;kleiner, dus in prog
aa3d  a465      ldy  $65       ;var-adres van string
aa3f  c42e      cpy  $2e       ;vegelijk met var-start
aa41  9008      bcc  $aa4b     ;kleiner
aa43  d00d      bne  $aa52
aa45  a564      lda  $64       ;idem voor low-byte
aa47  c52d      cmp  $2d
aa49  b007      bcs  $aa52
aa4b  a564      lda  $64
aa4d  a465      ldy  $65
aa4f  4c68aa    jmp  $aa68     ;toekennen uit prog
aa52  a000      ldy  ##$00     ;offset voor lengte
aa54  b164      lda  ($64),y   ;lengte lezen
aa56  2075b4    jsr  $b475     ;geheugenruimte testen
aa59  a550      lda  $50
aa5b  a451      ldy  $51
aa5d  856f      sta  $6f
aa5f  8470      sty  $70
aa61  207ab6    jsr  $b67a     ;string kopieren
aa64  a961      lda  ##$61
aa66  a000      ldy  ##$00
aa68  8550      sta  $50
aa6a  8451      sty  $51
aa6c  20dbb6    jsr  $b6db     ;stringstack op orde
aa6f  a000      ldy  ##$00     ;brengen
aa71  b150      lda  ($50),y   ;lengte naar variabele
aa73  9149      sta  ($49),y   ;kopieren
aa75  c8        iny
aa76  b150      lda  ($50),y   ;idem voor low-adres
aa78  9149      sta  ($49),y
aa7a  c8        iny
aa7b  b150      lda  ($50),y   ;en high-adres
aa7d  9149      sta  ($49),y
aa7f  60        rts

--                  BASIC-commando: PRINT#                  --

aa80  2086aa    jsr  $aa86     ;CMD-commando
aa83  4cb5ab    jmp  $abb5     ;CLRCHN
```

```
;-----------------------------------------------------------------------
;--                    BASIC-commando: CMD                            --
;-----------------------------------------------------------------------
aa86  209eb7    jsr  $b79e    ;haal bytewaarde op
aa89  f005      beq  $aa90    ;geen verdere tekens
aa8b  a92c      lda  ##$2c    ;of een komma
aa8d  20ffae    jsr  $aeff    ;anders SYNTAX ERROR
aa90  08        php           ;vlaggen op stack
aa91  8613      stx  $13      ;uitvoerkanaal naar $13
aa93  2018e1    jsr  $e118    ;CKOUT, zet uitvoer
aa96  28        plp           ;vlaggen terug
aa97  4ca0aa    jmp  $aaa0    ;en naar PRINT-commando
aa9a  2021ab    jsr  $ab21    ;string afdrukken
aa9d  207900    jsr  $0079    ;CHRGOT

;-----------------------------------------------------------------------
;--                    BASIC-commando: PRINT                          --
;-----------------------------------------------------------------------
aaa0  f035      beq  $aad7    ;geen tekens, dan CR
aaa2  f043      beq  $aae7    ;vervolg
aaa4  c9a3      cmp  ##$a3    ;TAB-token
aaa6  f050      beq  $aaf8    ;naar TAB/SPC met C=1
aaa8  c9a6      cmp  ##$a6    ;SPC-token
aaaa  18        clc           ;C=0
aaab  f04b      beq  $aaf8    ;naar TAB/SPC
aaad  c92c      cmp  ##$2c    ;komma?
aaaf  f037      beq  $aae8    ;ja, dan naar kolom
aab1  c93b      cmp  ##$3b    ;puntkomma
aab3  f05e      beq  $ab13    ;dan weg
aab5  209ead    jsr  $ad9e    ;FRMEVL: uitdrukking halen
aab8  240d      bit  $0d      ;test type
aaba  30de      bmi  $aa9a    ;string?
aabc  20ddbd    jsr  $bddd    ;FAC naar ASCII-string
aabf  2087b4    jsr  $b487    ;stringparameters lezen
aac2  2021ab    jsr  $ab21    ;en afdrukken
aac5  203bab    jsr  $ab3b    ;cursor right (jammer...)
aac8  d0d3      bne  $aa9d    ;en evt. verder
aaca  a900      lda  ##$00    ;input-buffer met 0 af-
aacc  9d0002    sta  $0200,x  ;sluiten
aacf  a2ff      ldx  ##$ff    ;pointer naar inputbuffer
aad1  a001      ldy  ##$01    ;laten wijzen
aad3  a513      lda  $13      ;uitvoerkanaal
aad5  d010      bne  $aae7    ;indien 0 dan weg
aad7  a90d      lda  ##$0d    ;return afdrukken
aad9  2047ab    jsr  $ab47
aadc  2413      bit  $13      ;kanaalnr>=128?
aade  1005      bpl  $aae5    ;nee
aae0  a90a      lda  ##$0a    ;ja, dan ook LF afdrukken
aae2  2047ab    jsr  $ab47
aae5  49ff      eor  ##$ff
aae7  60        rts
```

```
;-----------------------------------------------------------------
;--                  Naar TAB-kolom springen                    --
;-----------------------------------------------------------------
aae8  38       sec
aae9  20f0ff   jsr  $fff0   ;cursorpositie halen
aaec  98       tya          ;kolomwaarde
aaed  38       sec
aaee  e90a     sbc  #$0a    ;-10
aaf0  b0fc     bcs  $aaee   ;zolang >=0
aaf2  49ff     eor  #$ff    ;inverteren (-3:=+3)
aaf4  6901     adc  #$01    ;+1 (kolom1 = 0)
aaf6  d016     bne  $ab0e   ;en verder

;-----------------------------------------------------------------
;--          BASIC-commando's: TAB( en SPC(                     --
;-----------------------------------------------------------------
aaf8  08       php
aaf9  38       sec
aafa  20f0ff   jsr  $fff0   ;cursorpositie halen
aafd  8409     sty  $09     ;kolom bewaren
aaff  209bb7   jsr  $b79b   ;byte lezen
ab02  c929     cmp  #$29    ;testen op haakje ')'
ab04  d059     bne  $ab5f   ;nee, dan SYNTAX ERROR
ab06  28       plp          ;TAB of SPC?
ab07  9006     bcc  $ab0f   ;SPC
ab09  8a       txa          ;TAB-waarde naar accu
ab0a  e509     sbc  $09     ;en vergelijken
ab0c  9005     bcc  $ab13   ;kleiner, dan verder
ab0e  aa       tax          ;TAB-ben
ab0f  e8       inx          ;teller verhogen
ab10  ca       dex          ;en terug
ab11  d006     bne  $ab19   ;zolang OK spatie uit
ab13  207300   jsr  $0073   ;volgende teken
ab16  4ca2aa   jmp  $aaa2   ;en PRINTen
ab19  203bab   jsr  $ab3b   ;spatie afdrukken
ab1c  d0f2     bne  $ab10   ;en terug

;-----------------------------------------------------------------
;--      STRING afdrukken: A/Y=start, eind=#0                   --
;-----------------------------------------------------------------
ab1e  2087b4   jsr  $b487   ;parameters halen
ab21  20a6b6   jsr  $b6a6   ;verwerken
ab24  aa       tax          ;lengte naar X
ab25  a000     ldy  #$00    ;teller
ab27  e8       inx          ;lengte van string
ab28  ca       dex
ab29  f0bc     beq  $aae7   ;einde bereikt
ab2b  b122     lda  ($22),y ;teken lezen
ab2d  2047ab   jsr  $ab47   ;en afdrukken
ab30  c8       iny          ;volgende teken
ab31  c90d     cmp  #$0d    ;RETURN?
ab33  d0f3     bne  $ab28   ;nee
ab35  20e5aa   jsr  $aae5
ab38  4c28ab   jmp  $ab28   ;en verder
```

```
;-----------------------------------------------------------------
;--                    Spatie afdrukken                         --
;-----------------------------------------------------------------
43   ab3b  a513      lda  $13      ;INPUT-prompt
44   ab3d  f003      beq  $ab42    ;=0, dan cursor right
45   ab3f  a920      lda  ##$20    ;anders spatie
46   ab41  2c        byt  $2c
     ab42  a91d      lda  ##$1d    ;cursor right
47   ab44  2c        byt  $2c
     ab45  a93f      lda  ##$3f    ;vraagteken voor INPUT
48   ab47  200ce1    jsr  $e10c    ;en afdrukken
49   ab4a  29ff      and  ##$ff    ;vlaggen updaten
50   ab4c  60        rts           ;en klaar

;-----------------------------------------------------------------
;--          Foutafhandeling bij READ/GET/INPUT                 --
;-----------------------------------------------------------------
51   ab4d  a511      lda  $11      ;vlag bij READ/GET/INPUT
52   ab4f  f011      beq  $ab62    ;INPUT
53   ab51  3004      bmi  $ab57    ;READ
54   ab53  a0ff      ldy  ##$ff
55   ab55  d004      bne  $ab5b    ;GET
56   ab57  a53f      lda  $3f      ;DATA-regel low
57   ab59  a440      ldy  $40      ;DATA-regel hi
58   ab5b  8539      sta  $39      ;naar foutregel
59   ab5d  843a      sty  $3a
60   ab5f  4c08af    jmp  $af08    ;en SYNTAX ERROR IN
61   ab62  a513      lda  $13      ;inputkanaal
62   ab64  f005      beq  $ab6b    ;toetsenbord
63   ab66  a218      ldx  ##$18    ;FILE DATA
64   ab68  4c37a4    jmp  $a437    ;ERROR
65   ab6b  a90c      lda  ##$0c    ;$ADOC=REDO FROM START
66   ab6d  a0ad      ldy  ##$ad
67   ab6f  201eab    jsr  $ab1e    ;afdrukken
68   ab72  a53d      lda  $3d      ;programcounter low
69   ab74  a43e      ldy  $3e      ;en high
70   ab76  857a      sta  $7a      ;naar INPUT laten wijzen
71   ab78  847b      sty  $7b
72   ab7a  60        rts           ;klaar

;-----------------------------------------------------------------
;--                    BASIC-commando: GET                      --
;-----------------------------------------------------------------
73   ab7b  20a6b3    jsr  $b3a6    ;indien direct-mode: fout
74   ab7e  c923      cmp  ##$23    ;# (GET#)?
75   ab80  d010      bne  $ab92    ;nee
76   ab82  207300    jsr  $0073    ;ja, dan kanaalnr. lezen
77   ab85  209eb7    jsr  $b79e    ;bytewaarde lezen
78   ab88  a92c      lda  ##$2c    ;test op komma (foutje:
```

```
 79  ab8a  20ffae     jsr  $aeff    ;AEFD spaart 2 bytes!)
 80  ab8d  8613       stx  $13
 81  ab8f  201ee1     jsr  $e11e    ;teken GETten
 82  ab92  a201       ldx  ##$01    ;$0102 is inputbuffer
 83  ab94  a002       ldy  ##$02
 84  ab96  a900       lda  ##$00    ;buffer met 0 afsluiten
 85  ab98  8d0102     sta  $0201
 86  ab9b  a940       lda  ##$40    ;GET-vlag
 87  ab9d  200fac     jsr  $ac0f    ;variabele toekennen
 88  aba0  a613       ldx  $13      ;invoerkanaal
 89  aba2  d013       bne  $abb7    ;geen toetsenbord
 90  aba4  60         rts            ;klaar

     ------------------------------------------------------------
     --                BASIC-commando: INPUT#                  --
     ------------------------------------------------------------
 91  aba5  209eb7     jsr  $b79e    ;kanaal lezen
 92  aba8  a92c       lda  ##$2c    ;check op komma
 93  abaa  20ffae     jsr  $aeff
 94  abad  8613       stx  $13      ;opslaan als invoer
 95  abaf  201ee1     jsr  $e11e    ;teken GETten
 96  abb2  20ceab     jsr  $abce    ;direct INPUTten
 97  abb5  a513       lda  $13      ;invoerkanaal
 98  abb7  20ccff     jsr  $ffcc    ;terugzetten via CLRCHN
 99  abba  a200       ldx  ##$00    ;en op toetsen zetten
100  abbc  8613       stx  $13
101  abbe  60         rts

     ------------------------------------------------------------
     --                BASIC-commando: INPUT                   --
     ------------------------------------------------------------
102  abbf  c922       cmp  ##$22    ;test op quote (")
103  abc1  d00b       bne  $abce    ;nee
104  abc3  20bdae     jsr  $aebd    ;ja, dan string lezen
105  abc6  a93b       lda  ##$3b    ;na string op ';'
106  abc8  20ffae     jsr  $aeff    ;(puntkomma) testen
107  abcb  2021ab     jsr  $ab21    ;string afdrukken
108  abce  20a6b3     jsr  $b3a6    ;direct-mode testen
109  abd1  a92c       lda  ##$2c    ;komma
110  abd3  8dff01     sta  $01ff    ;in inputbuffer zetten
111  abd6  20f9ab     jsr  $abf9    ;vraagteken afdrukken
112  abd9  a513       lda  $13      ;invoerkanaal
113  abdb  f00d       beq  $abea    ;toetsenbord
114  abdd  20b7ff     jsr  $ffb7    ;nee, dan status lezen
115  abe0  2902       and  ##$02    ;bit filteren
116  abe2  f006       beq  $abea    ;time-out?
117  abe4  20b5ab     jsr  $abb5    ;ja, dan weer toetsenbord
118  abe7  4cf8a8     jmp  $a8f8    ;en volgend commando
119  abea  ad0002     lda  $0200    ;teken uit inputbuffer
120  abed  d01e       bne  $ac0d    ;einde?
121  abef  a513       lda  $13      ;invoerkanaal
122  abf1  d0e3       bne  $abd6    ;niet toetsenbord
```

```
123   abf3   2006a9     jsr   $a906    ;volgende statement
124   abf6   4cfba8     jmp   $a8fb    ;programcounter verhogen
125   abf9   a513       lda   $13
126   abfb   d006       bne   $ac03
127   abfd   2045ab     jsr   $ab45    ;vraagteken afdrukken
128   ac00   203bab     jsr   $ab3b    ;spatie
129   ac03   4c60a5     jmp   $a560    ;en input lezen

       ---------------------------------------------------------------
       --                  BASIC-commando: READ                     --
       ---------------------------------------------------------------
130   ac06   a641       ldx   $41      ;pointer naar DATA low
131   ac08   a442       ldy   $42      ;en high
132   ac0a   a998       lda   ##$98    ;vlag voor READ
133   ac0c   2c         byt   $2c      ;  ($40=GET)
      ac0d   a900       lda   ##$00    ;vlag voor INPUT
134   ac0f   8511       sta   $11      ;opbergen
135   ac11   8643       stx   $43      ;pointer zetten
136   ac13   8444       sty   $44
137   ac15   208bb0     jsr   $b08b    ;variabele zoeken
138   ac18   8549       sta   $49      ;var-adres low
139   ac1a   844a       sty   $4a      ;en high
140   ac1c   a57a       lda   $7a      ;programcounter
141   ac1e   a47b       ldy   $7b
142   ac20   854b       sta   $4b      ;bewaren
143   ac22   844c       sty   $4c
144   ac24   a643       ldx   $43      ;inputpointer gelijk
145   ac26   a444       ldy   $44      ;aan de
146   ac28   867a       stx   $7a      ;programcounter maken
147   ac2a   847b       sty   $7b
148   ac2c   207900     jsr   $0079    ;CHRGOT
149   ac2f   d020       bne   $ac51    ;einde
150   ac31   2411       bit   $11      ;vlag testen
151   ac33   500c       bvc   $ac41    ;ingeval van GET
152   ac35   2024e1     jsr   $e124    ;GETIN: teken lezen
153   ac38   8d0002     sta   $0200    ;naar de buffer
154   ac3b   a2ff       ldx   ##$ff    ;pointer naar buffer
155   ac3d   a001       ldy   ##$01    ;(01FF)
156   ac3f   d00c       bne   $ac4d    ;onconditioneel
157   ac41   3075       bmi   $acb8    ;van AC33
158   ac43   a513       lda   $13
159   ac45   d003       bne   $ac4a
160   ac47   2045ab     jsr   $ab45    ;vraagteken afdrukken
161   ac4a   20f9ab     jsr   $abf9    ;en nog een
162   ac4d   867a       stx   $7a      ;programcounter
163   ac4f   847b       sty   $7b
164   ac51   207300     jsr   $0073    ;CHRGET
165   ac54   240d       bit   $0d      ;type
166   ac56   1031       bpl   $ac89    ;numeriek
167   ac58   2411       bit   $11      ;READ/GET/INPUT-vlag
168   ac5a   5009       bvc   $ac65    ;GET
169   ac5c   e8         inx            ;PC:=PC+1
```

```
170  ac5d  867a      stx  $7a
171  ac5f  a900      lda  ##$00    ;zoekteken
172  ac61  8507      sta  $07
173  ac63  f00c      beq  $ac71    ;eerste maal oncond.
174  ac65  8507      sta  $07
175  ac67  c922      cmp  ##$22    ;vgl. quote (")
176  ac69  f007      beq  $ac72    ;gevonden
177  ac6b  a93a      lda  ##$3a    ;dubbele punt
178  ac6d  8507      sta  $07      ;als zoekteken
179  ac6f  a92c      lda  ##$2c    ;komma (,)
180  ac71  18        clc
181  ac72  8508      sta  $08
182  ac74  a57a      lda  $7a      ;programcounter
183  ac76  a47b      ldy  $7b
184  ac78  6900      adc  ##$00    ;+1
185  ac7a  9001      bcc  $ac7d
186  ac7c  c8        iny           ;high byte
187  ac7d  208db4    jsr  $b48d    ;string lezen
188  ac80  20e2b7    jsr  $b7e2    ;PC achter string zetten
189  ac83  20daa9    jsr  $a9da    ;var:=string
190  ac86  4c91ac    jmp  $ac91    ;en verder

191  ac89  20f3bc    jsr  $bcf3    ;numeriek: naar FAC
192  ac8c  a50e      lda  $0e      ;vlag int/real
193  ac8e  20c2a9    jsr  $a9c2    ;var:=FAC
194  ac91  207900    jsr  $0079    ;CHRGOT
195  ac94  f007      beq  $ac9d    ;klaar?
196  ac96  c92c      cmp  ##$2c    ;nee, volgt een komma?
197  ac98  f003      beq  $ac9d    ;ja
198  ac9a  4c4dab    jmp  $ab4d    ;anders fout
199  ac9d  a57a      lda  $7a      ;PC:=DATA-pointer
200  ac9f  a47b      ldy  $7b
201  aca1  8543      sta  $43
202  aca3  8444      sty  $44
203  aca5  a54b      lda  $4b      ;PC weer terugzetten
204  aca7  a44c      ldy  $4c
205  aca9  857a      sta  $7a
206  acab  847b      sty  $7b
207  acad  207900    jsr  $0079    ;CHRGOT
208  acb0  f02d      beq  $acdf    ;einde, dan klaar
209  acb2  20fdae    jsr  $aefd    ;check op komma
210  acb5  4c15ac    jmp  $ac15    ;en opnieuw
211  acb8  2006a9    jsr  $a906    ;volgend commando zoeken
212  acbb  c8        iny
213  acbc  aa        tax           ;einde regel bereikt?
214  acbd  d012      bne  $acd1    ;nee
215  acbf  a20d      ldx  ##$0d    ;ja
216  acc1  c8        iny
217  acc2  b17a      lda  ($7a),y  ;test op prog-eind
218  acc4  f06c      beq  $ad32    ;ja, dan OUT of DATA
219  acc6  c8        iny
220  acc7  b17a      lda  ($7a),y  ;DATA-regel low
```

```
221  acc9  853f      sta  $3f      ;opslaan
222  accb  c8        iny
223  accc  b17a      lda  ($7a),y ;idem voor high-byte
224  acce  c8        iny
225  accf  8540      sta  $40
226  acd1  20fba8    jsr  $a8fb   ;PC naar volg. commando
227  acd4  207900    jsr  $0079   ;CHRGOT
228  acd7  aa        tax          ;naar X
229  acd8  e083      cpx  #$83    ;gelijk aan 'DATA'?
230  acda  d0dc      bne  $acb8   ;nee
231  acdc  4c51ac    jmp  $ac51   ;ja, dan DATA lezen
232  acdf  a543      lda  $43     ;input-pointer
233  ace1  a444      ldy  $44
234  ace3  a611      ldx  $11     ;vlag
235  ace5  1003      bpl  $acea   ;geen DATA?
236  ace7  4c27a8    jmp  $a827   ;DATA-pointer zetten
237  acea  a000      ldy  #$00
238  acec  b143      lda  ($43),y ;en lezen
239  acee  f00b      beq  $acfb   ;einde bereikt?
240  acf0  a513      lda  $13     ;inputkanaal
241  acf2  d007      bne  $acfb   ;geen toetsenbord
242  acf4  a9fc      lda  #$fc    ;EXTRA IGNORED
243  acf6  a0ac      ldy  #$ac
244  acf8  4c1eab    jmp  $ab1e   ;afdrukken
245  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 4d 20 53 54 41 52 54
     ad1c  0d 00

     ------------------------------------------------------------
     --              BASIC-commando: NEXT                      --
     ------------------------------------------------------------
  1  ad1e  d004      bne  $ad24   ;variabelenaam?
  2  ad20  a000      ldy  #$00    ;nee
  3  ad22  f003      beq  $ad27   ;onconditioneel
  4  ad24  208bb0    jsr  $b08b   ;variabele zoeken
  5  ad27  8549      sta  $49     ;adres opslaan
  6  ad29  844a      sty  $4a
  7  ad2b  208aa3    jsr  $a38a   ;zoeken in stack
  8  ad2e  f005      beq  $ad35   ;gevonden
  9  ad30  a20a      ldx  #$0a    ;niet, NEXT WITHOUT FOR
 10  ad32  4c37a4    jmp  $a437   ;en afdrukken
 11  ad35  9a        txs          ;SP:=X
 12  ad36  8a        txa
 13  ad37  18        clc
 14  ad38  6904      adc  #$04    ;+4
 15  ad3a  48        pha
 16  ad3b  6906      adc  #$06
 17  ad3d  8524      sta  $24
```

```
18  ad3f  68        pla
19  ad40  a001      ldy  ##$01
20  ad42  20a2bb    jsr  $bba2   ;var van stack halen
21  ad45  ba        tsx          ;X:=SP
22  ad46  bd0901    lda  $0109,x ;stack uitlezen
23  ad49  8566      sta  $66     ;teken
24  ad4b  a549      lda  $49     ;var-adres
25  ad4d  a44a      ldy  $4a
26  ad4f  2067b8    jsr  $b867   ;naar ARG
27  ad52  20d0bb    jsr  $bbd0   ;en naar var
28  ad55  a001      ldy  ##$01   ;STEP met FAC verg.
29  ad57  205dbc    jsr  $bc5d
30  ad5a  ba        tsx
31  ad5b  38        sec
32  ad5c  fd0901    sbc  $0109,x ;d.m.v. aftrekken
33  ad5f  f017      beq  $ad78   ;gelijk
34  ad61  bd0f01    lda  $010f,x
35  ad64  8539      sta  $39
36  ad66  bd1001    lda  $0110,x
37  ad69  853a      sta  $3a
38  ad6b  bd1201    lda  $0112,x
39  ad6e  857a      sta  $7a
40  ad70  bd1101    lda  $0111,x
41  ad73  857b      sta  $7b
42  ad75  4caea7    jmp  $a7ae   ;terug naar interpreter
43  ad78  8a        txa          ;oude stackpointer
44  ad79  6911      adc  ##$11   ;terugzetten
45  ad7b  aa        tax          ;na optelling
46  ad7c  9a        txs          ;voila
47  ad7d  207900    jsr  $0079   ;laatste teken
48  ad80  c92c      cmp  ##$2c   ;komma?
49  ad82  d0f1      bne  $ad75   ;nee
50  ad84  207300    jsr  $0073   ;ja, dan volgende var
51  ad87  2024ad    jsr  $ad24   ;halen

    ------------------------------------------------------------
    --              Uitdrukking ophalen                       --
    ------------------------------------------------------------
52  ad8a  209ead    jsr  $ad9e   ;uitdrukking lezen
53  ad8d  18        clc          ;op numeriek testen
54  ad8e  24        byt  $24
    ad8f  38        sec          ;of op string
55  ad90  240d      bit  $0d     ;typeflag
56  ad92  3003      bmi  $ad97   ;string?
57  ad94  b003      bcs  $ad99   ;nee, dan dus fout
58  ad96  60        rts
59  ad97  b0fd      bcs  $ad96   ;en omgekeerd
60  ad99  a216      ldx  ##$16   ;TYPE MISMATCH
61  ad9b  4c37a4    jmp  $a437   ;afdrukken
```

```
      ----------------------------------------------------------------------
      --                    Uitdrukking evalueren                         --
      ----------------------------------------------------------------------
 62   ad9e  a67a      ldx  $7a      ;PC:=PC-1
 63   ada0  d002      bne  $ada4
 64   ada2  c67b      dec  $7b
 65   ada4  c67a      dec  $7a
 66   ada6  a200      ldx  #$00
 67   ada8  24        byt  $24
      ada9  48        pha
 68   adaa  8a        txa
 69   adab  48        pha
 70   adac  a901      lda  #$01
 71   adae  20fba3    jsr  $a3fb    ;test op stack-plaats
 72   adb1  2083ae    jsr  $ae83    ;volgende element halen
 73   adb4  a900      lda  #$00     ;masker
 74   adb6  854d      sta  $4d
 75   adb8  207900    jsr  $0079    ;CHRGOT
 76   adbb  38        sec
 77   adbc  e9b1      sbc  #$b1     ;welke bewerking?
 78   adbe  9017      bcc  $add7
 79   adc0  c903      cmp  #$03
 80   adc2  b013      bcs  $add7
 81   adc4  c901      cmp  #$01
 82   adc6  2a        rol  a        ;<, = of >
 83   adc7  4901      eor  #$01
 84   adc9  454d      eor  $4d
 85   adcb  c54d      cmp  $4d
 86   adcd  9061      bcc  $ae30
 87   adcf  854d      sta  $4d
 88   add1  207300    jsr  $0073    ;volgende teken
 89   add4  4cbbad    jmp  $adbb    ;en opnieuw
 90   add7  a64d      ldx  $4d      ;masker laden
 91   add9  d02c      bne  $ae07    ;<>0?
 92   addb  b07b      bcs  $ae58
 93   addd  6907      adc  #$07
 94   addf  9077      bcc  $ae58
 95   ade1  650d      adc  $0d      ;type-vlag
 96   ade3  d003      bne  $ade8
 97   ade5  4c3db6    jmp  $b63d    ;concatenatie
 98   ade8  69ff      adc  #$ff
 99   adea  8522      sta  $22
100   adec  0a        asl  a
101   aded  6522      adc  $22      ;maal 3
102   adef  a8        tay
103   adf0  68        pla
104   adf1  d980a0    cmp  $a080,y  ;vergelijken met
105   adf4  b067      bcs  $ae5d    ;hierarchie
106   adf6  208dad    jsr  $ad8d    ;op numeriek testen
107   adf9  48        pha
108   adfa  2020ae    jsr  $ae20    ;naar stack sturen
109   adfd  68        pla
110   adfe  a44b      ldy  $4b
111   ae00  1017      bpl  $ae19
```

```
112  ae02  aa        tax
113  ae03  f056      beq  $ae5b
114  ae05  d05f      bne  $ae66
115  ae07  460d      lsr  $0d      ;stringvlag wissen
116  ae09  8a        txa
117  ae0a  2a        rol  a        ;maal 2
118  ae0b  a67a      ldx  $7a      ;PC:=PC-1
119  ae0d  d002      bne  $ae11
120  ae0f  c67b      dec  $7b
121  ae11  c67a      dec  $7a
122  ae13  a01b      ldy  ##$1b
123  ae15  854d      sta  $4d      ;hierarchievlag zetten
124  ae17  d0d7      bne  $adf0    ;<>0
125  ae19  d980a0    cmp  $a080,y ;vgl. met tabel
126  ae1c  b048      bcs  $ae66
127  ae1e  90d9      bcc  $adf9
128  ae20  b982a0    lda  $a082,y ;adres van bewerking
129  ae23  48        pha           ;naar de stack
130  ae24  b981a0    lda  $a081,y ;(low-byte ook)
131  ae27  48        pha
132  ae28  2033ae    jsr  $ae33    ;operanden naar stack
133  ae2b  a54d      lda  $4d      ;masker
134  ae2d  4ca9ad    jmp  $ada9    ;en terug
135  ae30  4c08af    jmp  $af08    ;SYNTAX ERROR
136  ae33  a566      lda  $66      ;teken
137  ae35  be80a0    ldx  $a080,y ;tabel
138  ae38  a8        tay           ;teken naar Y
139  ae39  68        pla           ;terugkeeradres
140  ae3a  8522      sta  $22      ;opbergen
141  ae3c  e622      inc  $22
142  ae3e  68        pla
143  ae3f  8523      sta  $23
144  ae41  98        tya
145  ae42  48        pha
146  ae43  201bbc    jsr  $bc1b    ;FAC afronden
147  ae46  a565      lda  $65      ;en op de stack
148  ae48  48        pha           ;plaatsen
149  ae49  a564      lda  $64
150  ae4b  48        pha
151  ae4c  a563      lda  $63
152  ae4e  48        pha
153  ae4f  a562      lda  $62
154  ae51  48        pha
155  ae52  a561      lda  $61
156  ae54  48        pha
157  ae55  6c2200    jmp  ($0022) ;en operatie uitvoeren
158  ae58  a0ff      ldy  ##$ff
159  ae5a  68        pla           ;teken?
160  ae5b  f023      beq  $ae80    ;0, dan klaar
161  ae5d  c964      cmp  ##$64
162  ae5f  f003      beq  $ae64
163  ae61  208dad    jsr  $ad8d    ;op numeriek testen
```

```
164  ae64  844b       sty  $4b
165  ae66  68         pla
166  ae67  4a         lsr  a
167  ae68  8512       sta  $12
168  ae6a  68         pla            ;ARG van stack halen
169  ae6b  8569       sta  $69
170  ae6d  68         pla
171  ae6e  856a       sta  $6a
172  ae70  68         pla
173  ae71  856b       sta  $6b
174  ae73  68         pla
175  ae74  856c       sta  $6c
176  ae76  68         pla
177  ae77  856d       sta  $6d
178  ae79  68         pla
179  ae7a  856e       sta  $6e
180  ae7c  4566       eor  $66
181  ae7e  856f       sta  $6f
182  ae80  a561       lda  $61
183  ae82  60         rts

     ------------------------------------------------------------
     --          Rekenkundige uitdrukking ophalen            --
     ------------------------------------------------------------
184  ae83  6c0a03     jmp  ($030a) ;naar $AE86
185  ae86  a900       lda  #$00    ;typevalg op 0
186  ae88  850d       sta  $0d     ;(numeriek)
187  ae8a  207300     jsr  $0073   ;volgend teken
188  ae8d  b003       bcs  $ae92   ;cijfer?
189  ae8f  4cf3bc     jmp  $bcf3   ;waarde naar FAC
190  ae92  2013b1     jsr  $b113   ;test op letter
191  ae95  9003       bcc  $ae9a   ;geen letter
192  ae97  4c28af     jmp  $af28   ;wel, dan variabele lezen
193  ae9a  c9ff       cmp  #$ff    ;PI?
194  ae9c  d00f       bne  $aead   ;nee, dan verder
195  ae9e  a9a8       lda  #$a8    ;ja, dan pointer naar PI
196  aea0  a0ae       ldy  #$ae    ;(=$aea8)
197  aea2  20a2bb     jsr  $bba2   ;en naar FAC
198  aea5  4c7300     jmp  $0073   ;volgend teken

199  aea8  82 49 0f da a1          ;constante PI

202  aead  c92e       cmp  #$2e    ;'.'?
203  aeaf  f0de       beq  $ae8f   ;dan alsnog halen
204  aeb1  c9ab       cmp  #$ab    ;'-'?
205  aeb3  f058       beq  $af0d   ;teken omgooien
206  aeb5  c9aa       cmp  #$aa    ;'+'?
207  aeb7  f0d1       beq  $ae8a   ;dan gewoon verder
208  aeb9  c922       cmp  #$22    ;'"'?
209  aebb  d00f       bne  $aecc   ;nee
210  aebd  a57a       lda  $7a     ;ja, dan PC instellen
211  aebf  a47b       ldy  $7b
```

```
212  aec1  6900      adc  ##$00
213  aec3  9001      bcc  $aec6
214  aec5  c8        iny
215  aec6  2087b4    jsr  $b487    ;en string lezen
216  aec9  4ce2b7    jmp  $b7e2    ;PC:=stringeind+1
217  aecc  c9a8      cmp  ##$a8    ;NOT-code
218  aece  d013      bne  $aee3    ;nee
219  aed0  a018      ldy  ##$18    ;hierarchievlag
220  aed2  d03b      bne  $af0f    ;oncond. omdraaien

      ------------------------------------------------------------
      --              BASIC-commando: NOT                       --
      ------------------------------------------------------------

221  aed4  20bfb1    jsr  $b1bf    ;FAN:=INT(FAC)
222  aed7  a565      lda  $65      ;bits omdraaien
223  aed9  49ff      eor  ##$ff    ;met EOR-functie
224  aedb  a8        tay
225  aedc  a564      lda  $64
226  aede  49ff      eor  ##$ff
227  aee0  4c91b3    jmp  $b391    ;en terug naar FP

      ------------------------------------------------------------
      --              BASIC-commando: FN                        --
      ------------------------------------------------------------

228  aee3  c9a5      cmp  ##$a5    ;FN-code?
229  aee5  d003      bne  $aeea    ;nee
230  aee7  4cf4b3    jmp  $b3f4    ;ja: uitvoeren

      ------------------------------------------------------------
      --              BASIC-commando: SGN                       --
      ------------------------------------------------------------

231  aeea  c9b4      cmp  ##$b4    ;SGN-code?
232  aeec  9003      bcc  $aef1    ;nee, dan geen functie
233  aeee  4ca7af    jmp  $afa7    ;ja: uitvoeren

234  aef1  20faae    jsr  $aefa    ;test op '('
235  aef4  209ead    jsr  $ad9e    ;uitdrukking tussen ()
                                   halen
      ------------------------------------------------------------
      --            Testen op diverse tekens                    --
      ------------------------------------------------------------

236  aef7  a929      lda  ##$29    ;haakje sluiten ')'
237  aef9  2c        byt  $2c
     aefa  a928      lda  ##$28    ;haakje openen '('
238  aefc  2c        byt  $2c
     aefd  a92c      lda  ##$2c    ;komma ','
239  aeff  a000      ldy  ##$00    ;teller
240  af01  d17a      cmp  ($7a),y  ;verg. met huidig teken
241  af03  d003      bne  $af08    ;nee, dan SYNTAX ERROR
242  af05  4c7300    jmp  $0073    ;ja, volgende teken
243  af08  a20b      ldx  ##$0b    ;code voor SYNTAX ERROR
244  af0a  4c37a4    jmp  $a437    ;afdrukken
```

```
;--------------------------------------------------------------------
;--                      Teken wisselen                            --
;--------------------------------------------------------------------
245  af0d  a015      ldy  ##$15     ;hierarchie-code
246  af0f  68        pla
247  af10  68        pla
248  af11  4cfaad    jmp  $adfa     ;en uitvoeren

;--------------------------------------------------------------------
;--                   Test voor stringruimte                       --
;--------------------------------------------------------------------
249  af14  38        sec
250  af15  a564      lda  $64
251  af17  e900      sbc  ##$00     ;$a000=memtop
252  af19  a565      lda  $65
253  af1b  e9a0      sbc  ##$a0
254  af1d  9008      bcc  $af27
255  af1f  a9a2      lda  ##$a2
256  af21  e564      sbc  $64
257  af23  a9e3      lda  ##$e3
258  af25  e565      sbc  $65
259  af27  60        rts

;--------------------------------------------------------------------
;--                    Variabele halen                             --
;--------------------------------------------------------------------
260  af28  208bb0    jsr  $b08b     ;variabele zoeken
261  af2b  8564      sta  $64       ;opslaan
262  af2d  8465      sty  $65
263  af2f  a645      ldx  $45       ;variabele-naam
264  af31  a446      ldy  $46
265  af33  a50d      lda  $0d       ;type
266  af35  f026      beq  $af5d     ;numeriek
267  af37  a900      lda  ##$00
268  af39  8570      sta  $70
269  af3b  2014af    jsr  $af14     ;testen
270  af3e  901c      bcc  $af5c
271  af40  e054      cpx  ##$54     ;'T'?
272  af42  d018      bne  $af5c     ;nee
273  af44  c0c9      cpy  ##$c9     ;'I$'?
274  af46  d014      bne  $af5c     ;nee
275  af48  2084af    jsr  $af84     ;ja, dan tijd halen
276  af4b  845e      sty  $5e
277  af4d  88        dey
278  af4e  8471      sty  $71
279  af50  a006      ldy  ##$06
280  af52  845d      sty  $5d
281  af54  a024      ldy  ##$24
282  af56  2068be    jsr  $be68     ;string evalueren
283  af59  4c6fb4    jmp  $b46f
284  af5c  60        rts            ;klaar
```

```
      ---------------------------------------------------------------
      --                    Numerieke variabele                    --
      ---------------------------------------------------------------
285   af5d  240e      bit  $0e      ;vlag integer/real
286   af5f  100d      bpl  $af6e    ;real?
287   af61  a000      ldy  #$00     ;nee
288   af63  b164      lda  ($64),y ;haal integer
289   af65  aa        tax
290   af66  c8        iny
291   af67  b164      lda  ($64),y
292   af69  a8        tay
293   af6a  8a        txa
294   af6b  4c91b3    jmp  $b391    ;en naar FP
295   af6e  2014af    jsr  $af14    ;testen
296   af71  902d      bcc  $afa0
297   af73  e054      cpx  #$54     ;test op 'T'
298   af75  d01b      bne  $af92
299   af77  c049      cpy  #$49     ;en op 'I'
300   af79  d025      bne  $afa0
301   af7b  2084af    jsr  $af84    ;tijd halen
302   af7e  98        tya
303   af7f  a2a0      ldx  #$a0
304   af81  4c4fbc    jmp  $bc4f    ;en weg
```

```
---                    Tijd naar FAC halen                    ---

af84  20deff    jsr  $ffde    ;kernal gettime
af87  8664      stx  $64      ;bewaren in FAC
af89  8463      sty  $63
af8b  8565      sta  $65
af8d  a000      ldy  #$00     ;ongebruikt
af8f  8462      sty  $62
af91  60        rts           ;klaar

af92  e053      cpx  #$53     ;check op ST: 'S'
af94  d00a      bne  $afa0
af96  c054      cpy  #$54     ;'T'
af98  d006      bne  $afa0
af9a  20b7ff    jsr  $ffb7    ;status halen
af9d  4c3cbc    jmp  $bc3c    ;naar floating point
afa0  a564      lda  $64      ;adres van de variabele
afa2  a465      ldy  $65
afa4  4ca2bb    jmp  $bba2    ;naar FAC
afa7  0a        asl  a        ;maal twee
afa8  48        pha           ;op de stack
afa9  aa        tax           ;en naar X
afaa  207300    jsr  $0073    ;volgende teken via CHRGET
afad  e08f      cpx  #$8f     ;functiecode?
afaf  9020      bcc  $afd1    ;ja
afb1  20faae    jsr  $aefa    ;haakje openen testen
afb4  209ead    jsr  $ad9e    ;term halen (FRMEVL)
afb7  20fdae    jsr  $aefd    ;check op komma
afba  208fad    jsr  $ad8f    ;check op string
afbd  68        pla           ;terug van stack
afbe  aa        tax
afbf  a565      lda  $65      ;adres van de stringdescr.
afc1  48        pha           ;op de stack
afc2  a564      lda  $64      ;lowbyte ook
afc4  48        pha
afc5  8a        txa
afc6  48        pha
afc7  209eb7    jsr  $b79e    ;byte naar .X
afca  68        pla
afcb  a8        tay
afcc  8a        txa
afcd  48        pha
afce  4cd6af    jmp  $afd6    ;uitvoeren
afd1  20f1ae    jsr  $aef1    ;term tussen haakjes halen
afd4  68        pla           ;funktiecode terug
afd5  a8        tay
afd6  b9ea9f    lda  $9fea,y
afd9  8555      sta  $55
```

```
afdb  b9eb9f   lda  $9feb,y
afde  8556     sta  $56      ;funktieberekening
afe0  205400   jsr  $0054    ;functie uitvoeren
afe3  4c8dad   jmp  $ad8d    test op getal
```

BASIC-commando OR

```
afe6  a0ff     ldy  #$ff     ;vlag voor OR
afe8  2c       .byt $2c
```

BASIC-commando AND

```
afe9  a000     ldy  #$00     ;vlag voor AND
afeb  840b     sty  $0b      ;opslaan
afed  20bfb1   jsr  $b1bf    ;FAC naar integer
aff0  a564     lda  $64      ;lowbyte
aff2  450b     eor  $0b      ;x-or met vlag
aff4  8507     sta  $07      ;opslaan
aff6  a565     lda  $65      ;highbyte idem
aff8  450b     eor  $0b
affa  8508     sta  $08
affc  20fcbb   jsr  $bbfc    ;ARG naar FAC
afff  20bfb1   jsr  $b1bf    ;en weer naar integer
b002  a565     lda  $65      ;nogmaals
b004  450b     eor  $0b
b006  2508     and  $08
b008  450b     eor  $0b
b00a  a8       tay
b00b  a564     lda  $64
b00d  450b     eor  $0b
b00f  2507     and  $07
b011  450b     eor  $0b
b013  4c91b3   jmp  $b391    ;en weer naar floating point
```

Vergelijking

```
b016  2090ad   jsr  $ad90    ;check op gelijke typen
b019  b013     bcs  $b02e    ;stringtype? dan iets verder
b01b  a56e     lda  $6e
b01d  097f     ora  #$7f     ;ARG
b01f  256a     and  $6a
b021  856a     sta  $6a
b023  a969     lda  #$69     ;adres van ARG
b025  a000     ldy  #$00
b027  205bbc   jsr  $bc5b    ;vergelijken met FAC
```

```
b02a   aa          tax
b02b   4c61b0      jmp   $b061    ;uitkomst naar FAC

---------------------------------------------------------------
---                  String-vergelijking                    ---
---------------------------------------------------------------

b02e   a900        lda   #$00     ;stringvlag op 0 zetten
b030   850d        sta   $0d
b032   c64d        dec   $4d
b034   20a6b6      jsr   $b6a6    ;verwerken met FRESTR
b037   8561        sta   $61      ;lengte
b039   8662        stx   $62      ;en het adres
b03b   8463        sty   $63
b03d   a56c        lda   $6c      ;pointer naar tweede string
b03f   a46d        ldy   $6d
b041   20aab6      jsr   $b6aa    ;FRESTR aanroepen
b044   866c        stx   $6c      ;adres van string 2
b046   846d        sty   $6d
b048   aa          tax
b049   38          sec
b04a   e561        sbc   $61      ;lengte vergelijken
b04c   f008        beq   $b056    ;gelijk?
b04e   a901        lda   #$01     ;nee
b050   9004        bcc   $b056    ;string 2 korter
b052   a661        ldx   $61      ;lengte van string 1
b054   a9ff        lda   #$ff
b056   8566        sta   $66
b058   a0ff        ldy   #$ff
b05a   e8          inx
b05b   c8          iny
b05c   ca          dex            ;byte voor byte
b05d   d007        bne   $b066    ;vergelijken
b05f   a666        ldx   $66
b061   300f        bmi   $b072
b063   18          clc
b064   900c        bcc   $b072
b066   b16c        lda   ($6c),y  ;daar gaat-ie dan
b068   d162        cmp   ($62),y  ;gelijk?
b06a   f0ef        beq   $b05b    ;ja, dan terug
b06c   a2ff        ldx   #$ff     ;nee
b06e   b002        bcs   $b072
b070   a201        ldx   #$01
b072   e8          inx
b073   8a          txa
b074   2a          rol   a
b075   2512        and   $12
b077   f002        beq   $b07b
b079   a9ff        lda   #$ff
b07b   4c3cbc      jmp   $bc3c    ;resultaat naar FAC
```

```
b07e  20fdae    jsr  $aefd    ;check op komma

------------------------------------------------------------------------
----                    BASIC-commando DIM                          ----
------------------------------------------------------------------------

b081  aa        tax
b082  2090b0    jsr  $b090    ;dimensioneren
b085  207900    jsr  $0079    ;laatste teken halen
b088  d0f4      bne  $b07e    ;nog niet op?
b08a  60        rts           ;alles gehad

b08b  a200      ldx  #$00     ;niet dimensioneren-vlag
b08d  207900    jsr  $0079    ;laatste teken
b090  860c      stx  $0c      ;vlag voor DIM zetten
b092  8545      sta  $45      ;variabelenaam opslaan
b094  207900    jsr  $0079    ;wederom een CHRGOT
b097  2013b1    jsr  $b113    test op letter
b09a  b003      bcs  $b09f    ;okee
b09c  4c08af    jmp  $af08    ;anders een syntax error
b09f  a200      ldx  #$00     ;stringvlag wissen
b0a1  860d      stx  $0d
b0a3  860e      stx  $0e
b0a5  207300    jsr  $0073    ;volgend teken
b0a8  9005      bcc  $b0af    ;cijfer?
b0aa  2013b1    jsr  $b113    ;test op teken
b0ad  900b      bcc  $b0ba    ;nee?
b0af  aa        tax           ;volgende teken van de naam
b0b0  207300    jsr  $0073    ;via CHRGET
b0b3  90fb      bcc  $b0b0    ;cijfer?
b0b5  2013b1    jsr  $b113    ;test op letter
b0b8  b0f6      bcs  $b0b0    ;rest overslaan
b0ba  c924      cmp  #$24     ;is het een '$'?
b0bc  d006      bne  $b0c4    ;nee
b0be  a9ff      lda  #$ff     ;ja, dan stringvlag zetten
b0c0  850d      sta  $0d
b0c2  d010      bne  $b0d4    ;en wegwezen
b0c4  c925      cmp  #$25     ;een procentje dan (%)?
b0c6  d013      bne  $b0db    ;ook niet
b0c8  a510      lda  $10      ;wel, dan integercheck
b0ca  d0d0      bne  $b09c    ;niet toegestaan? fout dan
b0cc  a980      lda  #$80     ;integervlag zetten
b0ce  850e      sta  $0e
b0d0  0545      ora  $45      ;7e bit zetten
b0d2  8545      sta  $45
b0d4  8a        txa
b0d5  0980      ora  #$80     ;in tweede teken
b0d7  aa        tax
b0d8  207300    jsr  $0073    ;volgene teken weer
b0db  8646      stx  $46      ;opslaan
```

```
b0dd  38        sec
b0de  0510      ora  $10
b0e0  e928      sbc  ##28     ;linker haakje?
b0e2  d003      bne  $b0e7    ;nee
b0e4  4cd1b1    jmp  $b1d1    ;dimensioneren
b0e7  a000      ldy  ##00     ;maar eerst opzoeken
b0e9  8410      sty  $10
b0eb  a52d      lda  $2d      ;start van variabelen
b0ed  a62e      ldx  $2e
b0ef  8660      stx  $60      ;opslaan
b0f1  855f      sta  $5f
b0f3  e430      cpx  $30
b0f5  d004      bne  $b0fb
b0f7  c52f      cmp  $2f      ;eind bereikt?
b0f9  f022      beq  $b11d    ;ja, bestaat dus niet
b0fb  a545      lda  $45      ;eerste letter
b0fd  d15f      cmp  ($5f),y  ;vergelijken
b0ff  d008      bne  $b109    ;ongelijk
b101  a546      lda  $46      ;tweede teken
b103  c8        iny
b104  d15f      cmp  ($5f),y  ;ook vergelijken
b106  f07d      beq  $b185    ;gelijk
b108  88        dey
b109  18        clc
b10a  a55f      lda  $5f      ;verhogen met
b10c  6907      adc  ##07     ;7 bytes per variabele
b10e  90e1      bcc  $b0f1
b110  e8        inx
b111  d0dc      bne  $b0ef    ;en verder zoeken

;---------------------------------------------------------------
;---                  Testen op een letter                   ---
;---------------------------------------------------------------

b113  c941      cmp  ##41     ;letter 'A'
b115  9005      bcc  $b11c    ;nee
b117  e95b      sbc  ##5b     ;verlagen
b119  38        sec           ;carry wissen
b11a  e9a5      sbc  ##a5     ;en nogmaal verlagen
b11c  60        rts           ;klaar; C=1:ja, C=0:nee

;---------------------------------------------------------------
;---                BASIC-variabelen testen                  ---
;---------------------------------------------------------------

b11d  68        pla
b11e  48        pha           ;check oproepadres
b11f  c92a      cmp  ##2a     ;FRMEVL?
b121  d005      bne  $b128    ;nee
b123  a913      lda  ##13
```

```
b125  a0bf      ldy  ##$bf
b127  60        rts
b128  a545      lda  $45
b12a  a446      ldy  $46      ;naam van variabele
b12c  c954      cmp  ##$54    ;'T'
b12e  d00b      bne  $b13b    ;nee
b130  c0c9      cpy  ##$c9    ;'I$'
b132  f0ef      beq  $b123    ;ja, dus TI$
b134  c049      cpy  ##$49    ;I
b136  d003      bne  $b13b    ;nee, geen TI
b138  4c08af    jmp  $af08    ;dan een syntax error
b13b  c953      cmp  ##$53    ;'S'
b13d  d004      bne  $b143    ;nee
b13f  c054      cpy  ##$54    ;en een 'T'
b141  f0f5      beq  $b138    ;ST, dan ook een foutmelding
b143  a52f      lda  $2f
b145  a430      ldy  $30      ;pointer op array
b147  855f      sta  $5f      ;opslaan
b149  8460      sty  $60
b14b  a531      lda  $31
b14d  a432      ldy  $32      ;pointer naar eind
b14f  855a      sta  $5a      ;idem
b151  845b      sty  $5b
b153  18        clc           ;ruimte: 7 bytes
b154  6907      adc  ##$07
b156  9001      bcc  $b159
b158  c8        iny
b159  8558      sta  $58      ;nieuw blok
b15b  8459      sty  $59
b15d  20b8a3    jsr  $a3b8    ;en verschuiven maar
b160  a558      lda  $58
b162  a459      ldy  $59
b164  c8        iny
b165  852f      sta  $2f      ;arrayta.opnieuw instellen
b167  8430      sty  $30
b169  a000      ldy  ##$00
b16b  a545      lda  $45      ;eerste letter van de naam
b16d  915f      sta  ($5f),y
b16f  c8        iny           ;tweede ook
b170  a546      lda  $46
b172  915f      sta  ($5f),y
b174  a900      lda  ##$00
b176  c8        iny           ;en nog vijfmaal voor
b177  915f      sta  ($5f),y  ;de waarde
b179  c8        iny
b17a  915f      sta  ($5f),y
b17c  c8        iny
b17d  915f      sta  ($5f),y
b17f  c8        iny
b180  915f      sta  ($5f),y
```

```
b182  c8        iny
b183  915f      sta  ($5f),y
b185  a55f      lda  $5f
b187  18        clc
b188  6902      adc  #$02
b18a  a460      ldy  $60
b18c  9001      bcc  $b18f
b18e  c8        iny
b18f  8547      sta  $47      ;pointer opslaan
b191  8448      sty  $48
b193  60        rts           ;klaar
```

```
---   Pointer naar eerste arrayelement berekenen   ---
```

```
b194  a50b      lda  $0b      ;aantal dimensies
b196  0a        asl  a        ;maal 2
b197  6905      adc  #$05     ;plus 5
b199  655f      adc  $5f
b19b  a460      ldy  $60      ;optellen bij $5f/$60
b19d  9001      bcc  $b1a0
b19f  c8        iny
b1a0  8558      sta  $58      ;resultaat
b1a2  8459      sty  $59
b1a4  60        rts
```

```
---                  Constante -32768                  ---
```

```
b1a5  90 80 00 00 00
```

```
b1aa  20bfb1    jsr  $b1bf    ;FAC naar integer
b1ad  a564      lda  $64
b1af  a465      ldy  $65
b1b1  60        rts           ;klaar
```

```
---   Floating point naar integer vertalen   ---
```

```
b1b2  207300    jsr  $0073    ;volgend teken halen
b1b5  209ead    jsr  $ad9e    ;evalueren
b1b8  208dad    jsr  $ad8d    ;numeriek?
b1bb  a566      lda  $66      ;voorteken
b1bd  300d      bmi  $b1cc    ;negatief: ill. quantity
b1bf  a561      lda  $61      ;exponent
b1c1  c990      cmp  #$90     ;>32768
b1c3  9009      bcc  $b1ce    ;nee
```

```
b1c5  a9a5      lda  ##a5
b1c7  a0b1      ldy  ##b1     ;pointer naar 32768
b1c9  205bbc    jsr  $bc5b    ;vergelijken met FAC
b1cc  d07a      bne  $b248    ;ongelijk: ill.quantity
b1ce  4c9bbc    jmp  $bc9b    ;float naar integer;
```

--- Gedimensioneerde variabele verwerken ---

```
b1d1  a50c      lda  $0c      ;vlag voor DIM
b1d3  050e      ora  $0e      ;OR met integer
b1d5  48        pha
b1d6  a50d      lda  $0d      ;stringvlag checken
b1d8  48        pha
b1d9  a000      ldy  ##00
b1db  98        tya
b1dc  48        pha
b1dd  a546      lda  $46      ;2e letter van de naam
b1df  48        pha
b1e0  a545      lda  $45      ;eerste letter...
b1e2  48        pha
b1e3  20b2b1    jsr  $b1b2    ;index omzetten naar int.
b1e6  68        pla
b1e7  8545      sta  $45      ;varnaam terughalen
b1e9  68        pla
b1ea  8546      sta  $46
b1ec  68        pla
b1ed  a8        tay
b1ee  ba        tsx
b1ef  bd0201    lda  $0102,x  ;vlaggen van de stack halen
b1f2  48        pha
b1f3  bd0101    lda  $0101,x
b1f6  48        pha
b1f7  a564      lda  $64
b1f9  9d0201    sta  $0102,x  ;indices op de stack
b1fc  a565      lda  $65
b1fe  9d0101    sta  $0101,x
b201  c8        iny
b202  207900    jsr  $0079    ;laatste teken halen
b205  c92c      cmp  ##2c     ;','?
b207  f0d2      beq  $b1db    ;ja, dan volg. index
b209  840b      sty  $0b      ;nee, aantal indices opslaan
b20b  20f7ae    jsr  $aef7    ;haakje sluiten lezen
b20e  68        pla
b20f  850d      sta  $0d
b211  68        pla           ;vlaggen terug
b212  850e      sta  $0e
b214  297f      and  ##7f
b216  850c      sta  $0c
```

```
b218  a62f      ldx  $2f
b21a  a530      lda  $30      ;pointer op arraytabel
b21c  865f      stx  $5f
b21e  8560      sta  $60      ;opslaan
b220  c532      cmp  $32
b222  d004      bne  $b228
b224  e431      cpx  $31      ;vergelijken met tabeleind
b226  f039      beq  $b261    ;einde? dan niet gevonden
b228  a000      ldy  ##00
b22a  b15f      lda  ($5f),y ;tabelnaam
b22c  c8        iny
b22d  c545      cmp  $45      ;vergelijken
b22f  d006      bne  $b237
b231  a546      lda  $46
b233  d15f      cmp  ($5f),y ;tweede teken ook
b235  f016      beq  $b24d    ;en gevonden
b237  c8        iny
b238  b15f      lda  ($5f),y
b23a  18        clc
b23b  655f      adc  $5f      ;veldlengte bij optellen
b23d  aa        tax
b23e  c8        iny
b23f  b15f      lda  ($5f),y
b241  6560      adc  $60
b243  90d7      bcc  $b21c    ;en verder zoeken

b245  a212      ldx  ##$12    ;bad subscript error
b247  2c        .by  $2c
b248  a20e      ldx  ##$0e    ;illegal quantity
b24a  4c37a4    jmp  $a437    ;foutmelding afdrukken

b24d  a213      ldx  ##$13    ;redim'd array
b24f  a50c      lda  $0c      ;dim-vlag 0?
b251  d0f7      bne  $b24a    ;nee, foutmelding
b253  2094b1    jsr  $b194    ;zet pointer op eerste elem.
b256  a50b      lda  $0b      ;aantal gevonden dim's
b258  a004      ldy  ##$04
b25a  d15f      cmp  ($5f),y ;vergelijken
b25c  d0e7      bne  $b245    ;ongelijk: bad subscript
b25e  4ceab2    jmp  $b2ea    ;arrayelement zoeken
```

```
---------------------------------------------------------------------
---                    Arrayvariabele creeren                     ---
---------------------------------------------------------------------

b261  2094b1    jsr  $b194    ;ptr n. arrayelem. berekenen
b264  2008a4    jsr  $a408    ;vrij geheugen checken
b267  a000      ldy  #$00
b269  8472      sty  $72
b26b  a205      ldx  #$05
b26d  a545      lda  $45      ;eerste letter v/d naam
b26f  915f      sta  ($5f),y  ;in tabel plaatsen
b271  1001      bpl  $b274    ;integer?
b273  ca        dex
b274  c8        iny
b275  a546      lda  $46      ;tweede letter
b277  915f      sta  ($5f),y  ;in tabel plaatsen
b279  1002      bpl  $b27d    ;integer/string?
b27b  ca        dex
b27c  ca        dex
b27d  8671      stx  $71      ;varlengte: 2, 3 of 5
b27f  a50b      lda  $0b      ;aantal DIM's
b281  c8        iny
b282  c8        iny
b283  c8        iny
b284  915f      sta  ($5f),y  ;in tabel plaatsen
b286  a20b      ldx  #$0b     ;default-DIM: 11 (0..10)
b288  a900      lda  #$00     ;DIM-commando?
b28a  240c      bit  $0c
b28c  5008      bvc  $b296    ;nee
b28e  68        pla           ;DIM van stack halen
b28f  18        clc
b290  6901      adc  #$01     ;plus 1
b292  aa        tax           ;naar X
b293  68        pla
b294  6900      adc  #$00     ;carry optellen
b296  c8        iny
b297  915f      sta  ($5f),y  ;in tabel
b299  c8        iny
b29a  8a        txa           ;terug
b29b  915f      sta  ($5f),y  ;en ook naar tabel
b29d  204cb3    jsr  $b34c    ;ruimte berekenen
b2a0  8671      stx  $71      ;opslaan in var-einde
b2a2  8572      sta  $72
b2a4  a422      ldy  $22
b2a6  c60b      dec  $0b      ;meer te dimensioneren?
b2a8  d0dc      bne  $b286    ;ja
b2aa  6559      adc  $59
b2ac  b05d      bcs  $b30b    ;veldlengte+startadres
b2ae  8559      sta  $59
b2b0  a8        tay
```

```
b2b1  8a        txa
b2b2  6558      adc  $58
b2b4  9003      bcc  $b2b9
b2b6  c8        iny
b2b7  f052      beq  $b30b
b2b9  2008a4    jsr  $a408    ;check op ruimte
b2bc  8531      sta  $31      ;einde array-tabel
b2be  8432      sty  $32
b2c0  a900      lda  #$00     ;array 'legen'
b2c2  e672      inc  $72
b2c4  a471      ldy  $71
b2c6  f005      beq  $b2cd
b2c8  88        dey
b2c9  9158      sta  ($58),y ;0 maken
b2cb  d0fb      bne  $b2c8    ;als y<>0
b2cd  c659      dec  $59
b2cf  c672      dec  $72
b2d1  d0f5      bne  $b2c8
b2d3  e659      inc  $59
b2d5  38        sec
b2d6  a531      lda  $31
b2d8  e55f      sbc  $5f
b2da  a002      ldy  #$02
b2dc  915f      sta  ($5f),y ;arraylengte
b2de  a532      lda  $32
b2e0  c8        iny
b2e1  e560      sbc  $60
b2e3  915f      sta  ($5f),y ;idem
b2e5  a50c      lda  $0c      ;DIM-commando?
b2e7  d062      bne  $b34b    ;ja: RTS
```

Arrayelement zoeken

```
b2e9  c8        iny
b2ea  b15f      lda  ($5f),y ;aantal dimensies
b2ec  850b      sta  $0b      ;opslaan
b2ee  a900      lda  #$00
b2f0  8571      sta  $71      ;var-pointer
b2f2  8572      sta  $72
b2f4  c8        iny
b2f5  68        pla           ;index van stack
b2f6  aa        tax
b2f7  8564      sta  $64
b2f9  68        pla
b2fa  8565      sta  $65
b2fc  d15f      cmp  ($5f),y ;vgl. met arraywaarde
b2fe  900e      bcc  $b30e    ;kleiner
b300  d006      bne  $b308    ;te groot: BAD SUBSCRIPT
```

```
b302  c8        iny
b303  8a        txa
b304  d15f      cmp   ($5f),y ;check low-byte
b306  9007      bcc   $b30f   ;kleiner
b308  4c45b2    jmp   $b245   ;fout
b30b  4c35a4    jmp   $a435   ;OUT OF MEMORY
```

```
---          Adresberekening van array-element          ---
```

```
b30e  c8        iny
b30f  a572      lda   $72
b311  0571      ora   $71
b313  18        clc
b314  f00a      beq   $b320
b316  204cb3    jsr   $b34c   ;berekenen
b319  8a        txa
b31a  6564      adc   $64
b31c  aa        tax
b31d  98        tya
b31e  a422      ldy   $22
b320  6565      adc   $65
b322  8671      stx   $71
b324  c60b      dec   $0b     ;aantal DIM's
b326  d0ca      bne   $b2f2   ;en verder
b328  8572      sta   $72
b32a  a205      ldx   #$05
b32c  a545      lda   $45     ;eerste teken
b32e  1001      bpl   $b331   ;geen integer?
b330  ca        dex
b331  a546      lda   $46     ;tweede teken
b333  1002      bpl   $b337   ;integer/string?
b335  ca        dex
b336  ca        dex
b337  8628      stx   $28     ;lengte: 2, 3 of 5
b339  a900      lda   #$00
b33b  2055b3    jsr   $b355   ;offset berekenen
b33e  8a        txa
b33f  6558      adc   $58
b341  8547      sta   $47
b343  98        tya
b344  6559      adc   $59
b346  8548      sta   $48
b348  a8        tay
b349  a547      lda   $47
b34b  60        rts           ;klaar
```

```
;---------------------------------------------------------------
;                Hulproutine voor array-berekeningen
;---------------------------------------------------------------

b34c  8422      sty  $22
b34e  b15f      lda  ($5f),y ;uit tabel
b350  8528      sta  $28     ;bewaren
b352  88        dey
b353  b15f      lda  ($5f),y ;vorige
b355  8529      sta  $29     ;idem
b357  a910      lda  #$10
b359  855d      sta  $5d
b35b  a200      ldx  #$00
b35d  a000      ldy  #$00
b35f  8a        txa          ;vermenigvuldigen
b360  0a        asl  a
b361  aa        tax
b362  98        tya
b363  2a        rol  a
b364  a8        tay
b365  b0a4      bcs  $b30b
b367  0671      asl  $71
b369  2672      rol  $72
b36b  900b      bcc  $b378
b36d  18        clc
b36e  8a        txa
b36f  6528      adc  $28
b371  aa        tax
b372  98        tya
b373  6529      adc  $29
b375  a8        tay
b376  b093      bcs  $b30b
b378  c65d      dec  $5d
b37a  d0e3      bne  $b35f
b37c  60        rts          ;klaar

;---------------------------------------------------------------
;                      BASIC-functie FRE
;---------------------------------------------------------------

b37d  a50d      lda  $0d     ;type-vlag
b37f  f003      beq  $b384   ;geen string
b381  20a6b6    jsr  $b6a6   ;FRESTR uitvoeren
b384  2026b5    jsr  $b526   ;en garbage collecten
b387  38        sec
b388  a533      lda  $33
b38a  e531      sbc  $31     ;begin van de string
b38c  a8        tay
b38d  a534      lda  $34
b38f  e532      sbc  $32     ;variabelenstart vanaf
b391  a200      ldx  #$00    ;vlag: numeriek
b393  860d      stx  $0d
```

```
b395  8562      sta  $62      ;uitkomst opslaan
b397  8463      sty  $63
b399  a290      ldx  ##$90
b39b  4c44bc    jmp  $bc44    ;naar floatingpoint omzetten

-------------------------------------------------------------
- -                 BASIC-functie POS                      ---
-------------------------------------------------------------

b39e  38        sec           ;vlag: cursospos halen
b39f  20f0ff    jsr  $fff0    ;uitvoeren
b3a2  a900      lda  ##$00
b3a4  f0eb      beq  $b391    ;klaar

-------------------------------------------------------------
---               Directe mode testen                    ---
-------------------------------------------------------------

b3a6  a63a      ldx  $3a      ;vlag
b3a8  e8        inx           ;+1
b3a9  d0a0      bne  $b34b    ;=0? nee, dan klaar
b3ab  a215      ldx  ##$15    ;ILLEGAL DIRECT
b3ad  2c        byt  $2c      ;bekend truukje
b3ae  a21b      ldx  #1b      ;UNDEF'D FUNCTION
b3b0  4c37a4    jmp  $a437    ;foutmelding genereren

-------------------------------------------------------------
---               BASIC-commando DEF FN                  ---
-------------------------------------------------------------

b3b3  20e1b3    jsr  $b3e1    ;check FN-syntax
b3b6  20a6b3    jsr  $b3a6    ;check directe mode
b3b9  20faae    jsr  $aefa    ;check op haakje openen
b3bc  a980      lda  ##$80    ;INT-waarde niet toestaan
b3be  8510      sta  $10
b3c0  208bb0    jsr  $b08b    ;variabele zoeken
b3c3  208dad    jsr  $ad8d    ;check op numeriek
b3c6  20f7ae    jsr  $aef7    ;check op haakje sluiten
b3c9  a9b2      lda  ##$b2    ;code voor '='
b3cb  20ffae    jsr  $aeff    ;check
b3ce  48        pha
b3cf  a548      lda  $48      ;var-adres op stack zetten
b3d1  48        pha
b3d2  a547      lda  $47
b3d4  48        pha
b3d5  a57b      lda  $7b      ;prog-wijzer ook op stack
b3d7  48        pha           ;zetten
b3d8  a57a      lda  $7a
b3da  48        pha
b3db  20f8a8    jsr  $a8f8    ;en naar volgende commando
```

```
b3de  4c4fb4    jmp  $b44f   ;FN-var van stack halen

---------------------------------------------------------
---                FN-syntax testen                   ---
---------------------------------------------------------

b3e1  a9a5      lda  #$a5    ;code voor FN
b3e3  20ffae    jsr  $aeff   ;check
b3e6  0980      ora  #$80    ;integer tegenhouden
b3e8  8510      sta  $10
b3ea  2092b0    jsr  $b092   ;variabele zoeken
b3ed  854e      sta  $4e
b3ef  844f      sty  $4f     ;FN varpointer instellen
b3f1  4c8dad    jmp  $ad8d   ;check op numeriek

---------------------------------------------------------
---                BASIC-functie FN                   ---
---------------------------------------------------------

b3f4  20e1b3    jsr  $b3e1   ;check op FN-syntax
b3f7  a54f      lda  $4f     ;FN-varpointer
b3f9  48        pha          ;op stack zetten
b3fa  a54e      lda  $4e
b3fc  48        pha
b3fd  20f1ae    jsr  $aef1   ;tekst tussen () halen
b400  208dad    jsr  $ad8d   ;check op numeriek
b403  68        pla          ;pointer terug van stack
b404  854e      sta  $4e     ;halen
b406  68        pla
b407  854f      sta  $4f
b409  a002      ldy  #$02    ;zoeken
b40b  b14e      lda  ($4e),y
b40d  8547      sta  $47
b40f  aa        tax
b410  c8        iny
b411  b14e      lda  ($4e),y
b413  f099      beq  $b3ae   ;UNDEF'D FUNCTION
b415  8548      sta  $48
b417  c8        iny
b418  b147      lda  ($47),y
b41a  48        pha
b41b  88        dey
b41c  10fa      bpl  $b418
b41e  a448      ldy  $48
b420  20d4bb    jsr  $bbd4   ;FAC naar FN-var
b423  a57b      lda  $7b     ;prog-pointer bewaren
b425  48        pha
b426  a57a      lda  $7a
b428  48        pha
b429  b14e      lda  ($4e),y ;progpointer op FN-functie
```

```
b42b  857a      sta  $7a
b42d  c8        iny
b42e  b14e      lda  ($4e),y
b430  857b      sta  $7b
b432  a548      lda  $48
b434  48        pha
b435  a547      lda  $47
b437  48        pha
b438  208aad    jsr  $ad8a    ;numerieke waarde halen
b43b  68        pla
b43c  854e      sta  $4e
b43e  68        pla
b43f  854f      sta  $4f
b441  207900    jsr  $0079    ;CHRGOT
b444  f003      beq  $b449    ;meer tekens?
b446  4c08af    jmp  $af08    ;dan SYNTAX ERROR
b449  68        pla           ;prog-wijzer weer terug
b44a  857a      sta  $7a
b44c  68        pla
b44d  857b      sta  $7b
b44f  a000      ldy  #$00
b451  68        pla
b452  914e      sta  ($4e),y
b454  68        pla
b455  c8        iny
b456  914e      sta  ($4e),y
b458  68        pla           ;en FN-var van stack
b459  c8        iny           ;halen
b45a  914e      sta  ($4e),y
b45c  68        pla
b45d  c8        iny
b45e  914e      sta  ($4e),y
b460  68        pla
b461  c8        iny
b462  914e      sta  ($4e),y
b464  60        rts           ;klaar
```

BASIC-functie STR$

```
b465  208dad    jsr  $ad8d    ;check op numeriek
b468  a000      ldy  #$00
b46a  20dfbd    jsr  $bddf    ;FAC naar ASCII
b46d  68        pla           ;terugkeeradres weghalen
b46e  68        pla
b46f  a9ff      lda  #$ff
b471  a000      ldy  #$00
b473  f012      beq  $b487    ;en verder
```

```
                    Verwerking van strings

b475  a664      ldx  $64
b477  a465      ldy  $65
b479  8650      stx  $50      ;wijzer naar descriptor
b47b  8451      sty  $51
b47d  20f4b4    jsr  $b4f4    ;check geheugenruimte
b480  8662      stx  $62
b482  8463      sty  $63
b484  8561      sta  $61

b486  60        rts           ;klaar
b487  a222      ldx  ##$22    ;code voor '"'
b489  8607      stx  $07      ;als vlag bewaren
b48b  8608      stx  $08
b48d  856f      sta  $6f      ;start van de string
b48f  8470      sty  $70
b491  8562      sta  $62
b493  8463      sty  $63
b495  a0ff      ldy  ##ff     ;teller
b497  c8        iny           ;bij 0 beginnen
b498  b16f      lda  ($6f),y  ;teken uit de string
b49a  f00c      beq  $b4a8    ;einde?
b49c  c507      cmp  $07      ;vgl met vlag (")
b49e  f004      beq  $b4a4
b4a0  c508      cmp  $08      ;en met andere vlag
b4a2  d0f3      bne  $b497
b4a4  c922      cmp  ##$22    ;vgl met "
b4a6  f001      beq  $b4a9
b4a8  18        clc
b4a9  8461      sty  $61      ;lengte van de string
b4ab  98        tya
b4ac  656f      adc  $6f
b4ae  8571      sta  $71      ;eindadres low + 1
b4b0  a670      ldx  $70
b4b2  9001      bcc  $b4b5
b4b4  e8        inx
b4b5  8672      stx  $72      ;eindadres high + 1
b4b7  a570      lda  $70      ;startadres
b4b9  f004      beq  $b4bf    ;0?
b4bb  c902      cmp  ##$02    ;of 2?
b4bd  d00b      bne  $b4ca    ;nee dus
b4bf  98        tya
b4c0  2075b4    jsr  $b475    ;stringdescr. halen; len=A,
b4c3  a66f      ldx  $6f      ;X/Y geven adres
b4c5  a470      ldy  $70      ;startadres halen
b4c7  2088b6    jsr  $b688    ;string kopieren
b4ca  a616      ldx  $16      ;pointer
```

```
b4cc  e022      cpx  #$22     ;stack vol?
b4ce  d005      bne  $b4d5     ;nee
b4d0  a219      ldx  #$19      ;ja: FORMULA TOO COMPLEX
b4d2  4c37a4    jmp  $a437     ;foutmelding
b4d5  a561      lda  $61
b4d7  9500      sta  $00,x     ;stringlengte
b4d9  a562      lda  $62       ;en
b4db  9501      sta  $01,x     ;adres
b4dd  a563      lda  $63       ;naar
b4df  9502      sta  $02,x     ;stringstack
b4e1  a000      ldy  #$00
b4e3  8664      stx  $64
b4e5  8465      sty  $65
b4e7  8470      sty  $70
b4e9  88        dey            ;Y:=255
b4ea  840d      sty  $0d       ;vlag instellen
b4ec  8617      stx  $17       ;laatste stringdescr.
b4ee  e8        inx
b4ef  e8        inx
b4f0  e8        inx            ;plus 3
b4f1  8616      stx  $16       ;nieuwe index maken
b4f3  60        rts            ;klaar

b4f4  460f      lsr  $0f       ;garb.coll-vlag terugzetten
b4f6  48        pha            ;stringlengte
b4f7  49ff      eor  #$ff
b4f9  38        sec
b4fa  6533      adc  $33
b4fc  a434      ldy  $34
b4fe  b001      bcs  $b501
b500  88        dey
b501  c432      cpy  $32
b503  9011      bcc  $b516
b505  d004      bne  $b50b
b507  c531      cmp  $31
b509  900b      bcc  $b516
b50b  8533      sta  $33
b50d  8434      sty  $34
b50f  8535      sta  $35
b511  8436      sty  $36
b513  aa        tax
b514  68        pla
b515  60        rts            ;en weer terug

;---------------------------------------------------------------
;--                Check op garbage-collection                ---
;---------------------------------------------------------------

b516  a210      ldx  #$10      ;OUT OF MEMORY
b518  a50f      lda  $0f       ;garb.coll.-vlag
```

```
b51a   30b6       bmi   $b4d2    ;al gedaan? dan fout
b51c   2026b5     jsr   $b526    ;garb. coll. uitvoeren
b51f   a980       lda   #$80     ;vlag zetten
b521   850f       sta   $0f
b523   68         pla            ;lengte terug
b524   d0d0       bne   $b4f6    ;klaar
```

```
---                Garbage-collection uitvoeren                ---
```

```
b526   a637       ldx   $37      ;niet-gebruikte stringruimte
b528   a538       lda   $38      ;wordt m.b.v. garbage-
b52a   8633       stx   $33      ;collection (rommel ophalen)
b52c   8534       sta   $34      ;weer vrijgegeven
b52e   a000       ldy   #$00     ;(niet gedocumenteerd)
b530   844f       sty   $4f
b532   844e       sty   $4e
b534   a531       lda   $31
b536   a632       ldx   $32
b538   855f       sta   $5f
b53a   8660       stx   $60
b53c   a919       lda   #$19
b53e   a200       ldx   #$00
b540   8522       sta   $22
b542   8623       stx   $23
b544   c516       cmp   $16
b546   f005       beq   $b54d
b548   20c7b5     jsr   $b5c7
b54b   f0f7       beq   $b544
b54d   a907       lda   #$07
b54f   8553       sta   $53
b551   a52d       lda   $2d
b553   a62e       ldx   $2e
b555   8522       sta   $22
b557   8623       stx   $23
b559   e430       cpx   $30
b55b   d004       bne   $b561
b55d   c52f       cmp   $2f
b55f   f005       beq   $b566
b561   20bdb5     jsr   $b5bd
b564   f0f3       beq   $b559
b566   8558       sta   $58
b568   8659       stx   $59
b56a   a903       lda   #$03
b56c   8553       sta   $53
b56e   a558       lda   $58
b570   a659       ldx   $59
b572   e432       cpx   $32
b574   d007       bne   $b57d
```

```
b576  c531      cmp  $31
b578  d003      bne  $b57d
b57a  4c06b6    jmp  $b606
b57d  8522      sta  $22
b57f  8623      stx  $23
b581  a000      ldy  #$00
b583  b122      lda  ($22),y
b585  aa        tax
b586  c8        iny
b587  b122      lda  ($22),y
b589  08        php
b58a  c8        iny
b58b  b122      lda  ($22),y
b58d  6558      adc  $58
b58f  8558      sta  $58
b591  c8        iny
b592  b122      lda  ($22),y
b594  6559      adc  $59
b596  8559      sta  $59
b598  28        plp
b599  10d3      bpl  $b56e
b59b  8a        txa
b59c  30d0      bmi  $b56e
b59e  c8        iny
b59f  b122      lda  ($22),y
b5a1  a000      ldy  #$00
b5a3  0a        asl  a
b5a4  6905      adc  #$05
b5a6  6522      adc  $22
b5a8  8522      sta  $22
b5aa  9002      bcc  $b5ae
b5ac  e623      inc  $23
b5ae  a623      ldx  $23
b5b0  e459      cpx  $59
b5b2  d004      bne  $b5b8
b5b4  c558      cmp  $58
b5b6  f0ba      beq  $b572
b5b8  20c7b5    jsr  $b5c7
b5bb  f0f3      beq  $b5b0
b5bd  b122      lda  ($22),y
b5bf  3035      bmi  $b5f6
b5c1  c8        iny
b5c2  b122      lda  ($22),y
b5c4  1030      bpl  $b5f6
b5c6  c8        iny
b5c7  b122      lda  ($22),y
b5c9  f02b      beq  $b5f6
b5cb  c8        iny
b5cc  b122      lda  ($22),y
b5ce  aa        tax
```

```
b5cf  c8        iny
b5d0  b122      lda  ($22),y
b5d2  c534      cmp  $34
b5d4  9006      bcc  $b5dc
b5d6  d01e      bne  $b5f6
b5d8  e433      cpx  $33
b5da  b01a      bcs  $b5f6
b5dc  c560      cmp  $60
b5de  9016      bcc  $b5f6
b5e0  d004      bne  $b5e6
b5e2  e45f      cpx  $5f
b5e4  9010      bcc  $b5f6
b5e6  865f      stx  $5f
b5e8  8560      sta  $60
b5ea  a522      lda  $22
b5ec  a623      ldx  $23
b5ee  854e      sta  $4e
b5f0  864f      stx  $4f
b5f2  a553      lda  $53
b5f4  8555      sta  $55
b5f6  a553      lda  $53
b5f8  18        clc
b5f9  6522      adc  $22
b5fb  8522      sta  $22
b5fd  9002      bcc  $b601
b5ff  e623      inc  $23
b601  a623      ldx  $23
b603  a000      ldy  #$00
b605  60        rts
b606  a54f      lda  $4f
b608  054e      ora  $4e
b60a  f0f5      beq  $b601
b60c  a555      lda  $55
b60e  2904      and  #$04
b610  4a        lsr  a
b611  a8        tay
b612  8555      sta  $55
b614  b14e      lda  ($4e),y
b616  655f      adc  $5f
b618  855a      sta  $5a
b61a  a560      lda  $60
b61c  6900      adc  #$00
b61e  855b      sta  $5b
b620  a533      lda  $33
b622  a634      ldx  $34
b624  8558      sta  $58
b626  8659      stx  $59
b628  20bfa3    jsr  $a3bf
b62b  a455      ldy  $55
b62d  c8        iny
```

```
b62e  a558      lda  $58
b630  914e      sta  ($4e),y
b632  aa        tax
b633  e659      inc  $59
b635  a559      lda  $59
b637  c8        iny
b638  914e      sta  ($4e),y
b63a  4c2ab5    jmp  $b52a

;---------------------------------------------------------------
;---              Stringconcatenatie '+'                     ---
;---------------------------------------------------------------

b63d  a565      lda  $65     ;stringdescriptor 1e string
b63f  48        pha          ;onthouden
b640  a564      lda  $64
b642  48        pha
b643  2083ae    jsr  $ae83   ;adres 2e descr. ophalen
b646  208fad    jsr  $ad8f   ;check op stringvariabele
b649  68        pla          ;descriptor weer teruglezen
b64a  856f      sta  $6f
b64c  68        pla
b64d  8570      sta  $70
b64f  a000      ldy  #$00    ;teller
b651  b16f      lda  ($6f),y ;lengtebyte
b653  18        clc          ;lengte 2e string erbij
b654  7164      adc  ($64),y ;optellen
b656  9005      bcc  $b65d   ;<256
b658  a217      ldx  #$17    ;STRING TOO LONG
b65a  4c37a4    jmp  $a437   ;foutmelding
b65d  2075b4    jsr  $b475   ;ruimte reserveren
b660  207ab6    jsr  $b67a   ;eerste string daar plaatsen
b663  a550      lda  $50     ;pointer naar 2e descriptor
b665  a451      ldy  $51
b667  20aab6    jsr  $b6aa   ;FRESTR uitvoeren
b66a  208cb6    jsr  $b68c   ;strings koppelen
b66d  a56f      lda  $6f
b66f  a470      ldy  $70
b671  20aab6    jsr  $b6aa   ;FRESTR
b674  20cab4    jsr  $b4ca   ;descriptor op stack zetten
b677  4cb8ad    jsr  $adb8   ;terug naar berekening

b67a  a000      ldy  #$00    ;teller
b67c  b16f      lda  ($6f),y ;stringlengte
b67e  48        pha          ;onthouden
b67f  c8        iny
b680  b16f      lda  ($6f),y ;stringadres low
b682  aa        tax
b683  c8        iny
b684  b16f      lda  ($6f),y ;stringadres high
```

```
b686  a8        tay
b687  68        pla           ;lengte
b688  8622      stx  $22      ;stringpointer
b68a  8423      sty  $23
b68c  a8        tay
b68d  f00a      beq  $b699    ;0, dan klaar
b68f  48        pha
b690  88        dey
b691  b122      lda  ($22),y ;string kopieren
b693  9135      sta  ($35),y ;teken voor teken
b695  98        tya
b696  d0f8      bne  $b690    ;klaar?
b698  68        pla           ;ja
b699  18        clc
b69a  6535      adc  $35
b69c  8535      sta  $35      ;pointer plus lengte
b69e  9002      bcc  $b6a2
b6a0  e636      inc  $36
b6a2  60        rts           ;klaar
```

Stringroutine FRESTR

```
b6a3  208fad    jsr  $ad8f    ;check op stringvariabele
b6a6  a564      lda  $64      ;pointer stringdescriptor
b6a8  a465      ldy  $65
b6aa  8522      sta  $22
b6ac  8423      sty  $23
b6ae  20dbb6    jsr  $b6db    ;descr. van stack halen
b6b1  08        php
b6b2  a000      ldy  #$00     ;teller
b6b4  b122      lda  ($22),y ;lengtebyte
b6b6  48        pha           ;bewaren
b6b7  c8        iny
b6b8  b122      lda  ($22),y ;lowbyte
b6ba  aa        tax
b6bb  c8        iny
b6bc  b122      lda  ($22),y ;highbyte
b6be  a8        tay
b6bf  68        pla           ;lengte
b6c0  28        plp           ;status
b6c1  d013      bne  $b6d6    ;string niet op stingstack
b6c3  c434      cpy  $34
b6c5  d00f      bne  $b6d6
b6c7  e433      cpx  $33
b6c9  d00b      bne  $b6d6
b6cb  48        pha
b6cc  18        clc
b6cd  6533      adc  $33      ;$33/$34 op stringstart
```

```
b6cf  8533      sta  $33
b6d1  9002      bcc  $b6d5
b6d3  e634      inc  $34
b6d5  68        pla
b6d6  8622      stx  $22
b6d8  8423      sty  $23
b6da  60        rts
b6db  c418      cpy  $18
b6dd  d00c      bne  $b6eb
b6df  c517      cmp  $17      ;descr. in stack?
b6e1  d008      bne  $b6eb    ;nee
b6e3  8516      sta  $16
b6e5  e903      sbc  #$03     ;ja, dan verwijderen
b6e7  8517      sta  $17
b6e9  a000      ldy  #$00
b6eb  60        rts
```

```
---                    BASIC-functie CHR$                    ---

b6ec  20a1b7    jsr  $b7a1
b6ef  8a        txa
b6f0  48        pha
b6f1  a901      lda  #$01
b6f3  207db4    jsr  $b47d
b6f6  68        pla
b6f7  a000      ldy  #$00
b6f9  9162      sta  ($62),y
b6fb  68        pla
b6fc  68        pla
b6fd  4ccab4    jmp  $b4ca

---                    BASIC-functie LEFT$                   ---

b700  2061b7    jsr  $b761
b703  d150      cmp  ($50),y
b705  98        tya
b706  9004      bcc  $b70c
b708  b150      lda  ($50),y
b70a  aa        tax
b70b  98        tya
b70c  48        pha
b70d  8a        txa
b70e  48        pha
b70f  207db4    jsr  $b47d
b712  a550      lda  $50
b714  a451      ldy  $51
b716  20aab6    jsr  $b6aa
b719  68        pla
b71a  a8        tay
b71b  68        pla
b71c  18        clc
b71d  6522      adc  $22
b71f  8522      sta  $22
b721  9002      bcc  $b725
b723  e623      inc  $23
b725  98        tya
b726  208cb6    jsr  $b68c
b729  4ccab4    jmp  $b4ca

---                    BASIC-functie RIGHT$ .                ---

b72c  2061b7    jsr  $b761
```

```
b72f  18        clc
b730  f150      sbc  ($50),y
b732  49ff      eor  ##$ff
b734  4c06b7    jmp  $b706

--------------------------------------------------------------------------
---                    BASIC-functie MID$                              ---
--------------------------------------------------------------------------

b737  a9ff      lda  ##$ff
b739  8565      sta  $65
b73b  207900    jsr  $0079
b73e  c929      cmp  ##$29
b740  f006      beq  $b748
b742  20fdae    jsr  $aefd
b745  209eb7    jsr  $b79e
b748  2061b7    jsr  $b761
b74b  f04b      beq  $b798
b74d  ca        dex
b74e  8a        txa
b74f  48        pha
b750  18        clc
b751  a200      ldx  ##$00
b753  f150      sbc  ($50),y
b755  b0b6      bcs  $b70d
b757  49ff      eor  ##$ff
b759  c565      cmp  $65
b75b  90b1      bcc  $b70e
b75d  a565      lda  $65
b75f  b0ad      bcs  $b70e
b761  20f7ae    jsr  $aef7
b764  68        pla
b765  a8        tay
b766  68        pla
b767  8555      sta  $55
b769  68        pla
b76a  68        pla
b76b  68        pla
b76c  aa        tax
b76d  68        pla
b76e  8550      sta  $50
b770  68        pla
b771  8551      sta  $51
b773  a555      lda  $55
b775  48        pha
b776  98        tya
b777  48        pha
b778  a000      ldy  ##$00
b77a  8a        txa
b77b  60        rts
```

```
---------------------------------------------------------------
---                  BASIC-functie LEN                      ---
---------------------------------------------------------------

b77c   2082b7     jsr   $b782
b77f   4ca2b3     jmp   $b3a2

---------------------------------------------------------------
---                  Stringparameters halen                 ---
---------------------------------------------------------------

b782   20a3b6     jsr   $b6a3
b785   a200       ldx   #$00
b787   860d       stx   $0d
b789   a8         tay
b78a   60         rts

---------------------------------------------------------------
---                  BASIC-functie ASC                      ---
---------------------------------------------------------------

b78b   2082b7     jsr   $b782
b78e   f008       beq   $b798
b790   a000       ldy   #$00
b792   b122       lda   ($22),y
b794   a8         tay
b795   4ca2b3     jmp   $b3a2
b798   4c48b2     jmp   $b248

---------------------------------------------------------------
---                  Byte in X-register lezen               ---
---------------------------------------------------------------

b79b   207300     jsr   $0073
b79e   208aad     jsr   $ad8a
b7a1   20b8b1     jsr   $b1b8
b7a4   a664       ldx   $64
b7a6   d0f0       bne   $b798
b7a8   a665       ldx   $65
b7aa   4c7900     jmp   $0079

---------------------------------------------------------------
---                  BASIC-functie VAL                      ---
---------------------------------------------------------------

b7ad   2082b7     jsr   $b782
b7b0   d003       bne   $b7b5
b7b2   4cf7b8     jmp   $b8f7
b7b5   a67a       ldx   $7a
b7b7   a47b       ldy   $7b
b7b9   8671       stx   $71
```

```
b7bb  8472      sty  $72
b7bd  a622      ldx  $22
b7bf  867a      stx  $7a
b7c1  18        clc
b7c2  6522      adc  $22
b7c4  8524      sta  $24
b7c6  a623      ldx  $23
b7c8  867b      stx  $7b
b7ca  9001      bcc  $b7cd
b7cc  e8        inx
b7cd  8625      stx  $25
b7cf  a000      ldy  ##$00
b7d1  b124      lda  ($24),y
b7d3  48        pha
b7d4  98        tya
b7d5  9124      sta  ($24),y
b7d7  207900    jsr  $0079
b7da  20f3bc    jsr  $bcf3
b7dd  68        pla
b7de  a000      ldy  ##$00
b7e0  9124      sta  ($24),y
b7e2  a671      ldx  $71
b7e4  a472      ldy  $72
b7e6  867a      stx  $7a
b7e8  847b      sty  $7b
b7ea  60        rts

--------------------------------------------------------------------------------
---          GETADR/GETBYT: word/byte inlezen                                ---
--------------------------------------------------------------------------------

b7eb  208aad    jsr  $ad8a
b7ee  20f7b7    jsr  $b7f7
b7f1  20fdae    jsr  $aefd
b7f4  4c9eb7    jmp  $b79e

--------------------------------------------------------------------------------
---          FAC naar 16-bits (pos.) omzetten                                ---
--------------------------------------------------------------------------------

b7f7  a566      lda  $66
b7f9  309d      bmi  $b798
b7fb  a561      lda  $61
b7fd  c991      cmp  ##$91
b7ff  b097      bcs  $b798
b801  209bbc    jsr  $bc9b
b804  a564      lda  $64
b806  a465      ldy  $65
b808  8414      sty  $14
b80a  8515      sta  $15
b80c  60        rts
```

```
---------------------------------------------------------------------
---                    BASIC-functie PEEK                         ---
---------------------------------------------------------------------

b80d   a515       lda  $15
b80f   48         pha
b810   a514       lda  $14
b812   48         pha
b813   20f7b7     jsr  $b7f7
b816   a000       ldy  #$00
b818   b114       lda  ($14),y
b81a   a8         tay
b81b   68         pla
b81c   8514       sta  $14
b81e   68         pla
b81f   8515       sta  $15
b821   4ca2b3     jmp  $b3a2

---------------------------------------------------------------------
---                    BASIC-commando POKE                        ---
---------------------------------------------------------------------

b824   20ebb7     jsr  $b7eb
b827   8a         txa
b828   a000       ldy  #$00
b82a   9114       sta  ($14),y
b82c   60         rts

---------------------------------------------------------------------
---                    BASIC-commando WAIT                        ---
---------------------------------------------------------------------

b82d   20ebb7     jsr  $b7eb
b830   8649       stx  $49
b832   a200       ldx  #$00
b834   207900     jsr  $0079
b837   f003       beq  $b83c
b839   20f1b7     jsr  $b7f1
b83c   864a       stx  $4a
b83e   a000       ldy  #$00
b840   b114       lda  ($14),y
b842   454a       eor  $4a
b844   2549       and  $49
b846   f0f8       beq  $b840
b848   60         rts

---------------------------------------------------------------------
---                    FAC:=FAC+0.5                               ---
---------------------------------------------------------------------
```

```
b849  a911      lda  ##$11
b84b  a0bf      ldy  ##$bf
b84d  4c67b8    jmp  $b867

-------------------------------------------------------------------------------
---                    FAC:=A/Y-FAC                                         ---
-------------------------------------------------------------------------------

b850  208cba    jsr  $ba8c

-------------------------------------------------------------------------------
---                    FAC:=ARG-FAC                                         ---
-------------------------------------------------------------------------------

b853  a566      lda  $66
b855  49ff      eor  ##$ff
b857  8566      sta  $66
b859  456e      eor  $6e
b85b  856f      sta  $6f
b85d  a561      lda  $61
b85f  4c6ab8    jmp  $b86a

-------------------------------------------------------------------------------
---          Exponenten FAC&ARG normaliseren                                ---
-------------------------------------------------------------------------------

b862  2099b9    jsr  $b999
b865  903c      bcc  $b8a3

-------------------------------------------------------------------------------
---                    FAC:=A/Y+FAC                                         ---
-------------------------------------------------------------------------------

b867  208cba    jsr  $ba8c

-------------------------------------------------------------------------------
---                    FAC:=ARG+FAC                                         ---
-------------------------------------------------------------------------------

b86a  d003      bne  $b86f
b86c  4cfcbb    jmp  $bbfc
b86f  a670      ldx  $70
b871  8656      stx  $56
b873  a269      ldx  ##$69
b875  a569      lda  $69
b877  a8        tay
b878  f0ce      beq  $b848
b87a  38        sec
b87b  e561      sbc  $61
b87d  f024      beq  $b8a3
b87f  9012      bcc  $b893
```

```
b881  8461      sty  $61
b883  a46e      ldy  $6e
b885  8466      sty  $66
b887  49ff      eor  #$ff
b889  6900      adc  #$00
b88b  a000      ldy  #$00
b88d  8456      sty  $56
b88f  a261      ldx  #$61
b891  d004      bne  $b897
b893  a000      ldy  #$00
b895  8470      sty  $70
b897  c9f9      cmp  #$f9
b899  30c7      bmi  $b862
b89b  a8        tay
b89c  a570      lda  $70
b89e  5601      lsr  $01,x
b8a0  20b0b9    jsr  $b9b0
b8a3  246f      bit  $6f
b8a5  1057      bpl  $b8fe
b8a7  a061      ldy  #$61
b8a9  e069      cpx  #$69
b8ab  f002      beq  $b8af
b8ad  a069      ldy  #$69
b8af  38        sec
b8b0  49ff      eor  #$ff
b8b2  6556      adc  $56
b8b4  8570      sta  $70
b8b6  b90400    lda  $0004,y
b8b9  f504      sbc  $04,x
b8bb  8565      sta  $65
b8bd  b90300    lda  $0003,y
b8c0  f503      sbc  $03,x
b8c2  8564      sta  $64
b8c4  b90200    lda  $0002,y
b8c7  f502      sbc  $02,x
b8c9  8563      sta  $63
b8cb  b90100    lda  $0001,y
b8ce  f501      sbc  $01,x
b8d0  8562      sta  $62
b8d2  b003      bcs  $b8d7
b8d4  2047b9    jsr  $b947
b8d7  a000      ldy  #$00
b8d9  98        tya
b8da  18        clc
b8db  a662      ldx  $62
b8dd  d04a      bne  $b929
b8df  a663      ldx  $63
b8e1  8662      stx  $62
b8e3  a664      ldx  $64
b8e5  8663      stx  $63
b8e7  a665      ldx  $65
```

```
b8e9  8664      stx  $64
b8eb  a670      ldx  $70
b8ed  8665      stx  $65
b8ef  8470      sty  $70
b8f1  6908      adc  #$08
b8f3  c920      cmp  #$20
b8f5  d0e4      bne  $b8db
b8f7  a900      lda  #$00
b8f9  8561      sta  $61
b8fb  8566      sta  $66
b8fd  60        rts
b8fe  6556      adc  $56
b900  8570      sta  $70
b902  a565      lda  $65
b904  656d      adc  $6d
b906  8565      sta  $65
b908  a564      lda  $64
b90a  656c      adc  $6c
b90c  8564      sta  $64
b90e  a563      lda  $63
b910  656b      adc  $6b
b912  8563      sta  $63
b914  a562      lda  $62
b916  656a      adc  $6a
b918  8562      sta  $62
b91a  4c36b9    jmp  $b936
b91d  6901      adc  #$01
b91f  0670      asl  $70
b921  2665      rol  $65
b923  2664      rol  $64
b925  2663      rol  $63
b927  2662      rol  $62
b929  10f2      bpl  $b91d
b92b  38        sec
b92c  e561      sbc  $61
b92e  b0c7      bcs  $b8f7
b930  49ff      eor  #$ff
b932  6901      adc  #$01
b934  8561      sta  $61
b936  900e      bcc  $b946
b938  e661      inc  $61
b93a  f042      beq  $b97e
b93c  6662      ror  $62
b93e  6663      ror  $63
b940  6664      ror  $64
b942  6665      ror  $65
b944  6670      ror  $70
b946  60        rts
```

6

CBM 64, de uitbreidingen

Inhoud

[1)] Dit hoofdstuk heeft een eigen inhoudsopgave.

De wereldwijde populariteit van de Commodore 64 heeft er toe geleid dat er zeer veel fabrikanten uitbreidingen voor deze machine zijn gaan produceren. Joysticks, datarecorders, diskdrives, printers, modems, graphic tablets en zelfs video-interfaces, noem maar op.
Om al deze extra's zinvol te kunnen gebruiken moet men behalve de standaard BASIC 2.0 ook de eigenschappen van deze uitbreidingen kennen en daar gaat dit hoofdstuk dan ook over.

De BASIC commando's waar dit soort randapparatuur mee bestuurd wordt zijn weliswaar ook in hoofdstuk 7, BASIC, behandeld, maar de volledige uitleg, aangevuld met voorbeelden, staat in dit hoofdstuk. De ingewikkeldheid van deze commando's, gecombineerd met het feit dat de vele verschillende randapparaten vaak net anders werken dan men zou verwachten, vormt de reden voor deze aparte behandeling. Ook de principes van de werking van de verschillende soorten uitbreidingen komen aan de orde. Hoe staat de informatie nu precies op cassette of diskette, of hoe werkt een RS232 aansluiting.

Gelukkig kan vaak volstaan worden met algemene informatie en wat programmeervoorbeelden. Bijvoorbeeld zoals in het geval van de joystick, die weliswaar in vele maten en soorten te koop is, maar altijd op precies dezelfde manier werkt. Informatie over al die verschillende uitvoeringen staat dan niet in dit hoofdstuk, maar in hoofdstuk 14, Produktinformatie. Soms zal het echter nodig zijn om dieper op de specifieke eigenschappen van een bepaald produkt in te gaan, zoals bij de diverse printers en printer-interfaces. Hoewel deze allemaal wel op elkaar lijken kunnen er toch grote verschillen in gebruik en programmering bestaan, die dan hier uit de doeken gedaan zullen worden.

De informatie zal zo algemeen mogelijk gepresenteerd worden. alleen als het niet anders kan zullen de details van een bepaald apparaat aan de orde komen, zoals bij de Commodore printers die nu eenmaal nogal verschillende mogelijkheden en aansturingen daarvan kennen.

6/1

Gegevensopslag

Inhoud

Bij iedere computer is een methode nodig om de gegevens en programma's te kunnen bewaren. Tenzij iemand de computer alleen gebruikt om cartridge-spellen mee te spelen valt daar niet aan te ontkomen. De meest voor de hand liggende opslagmedia zijn cassette en diskette. Beiden gebaseerd op magnetisch materiaal, maar verder volkomen verschillend in hun werking en mogelijkheden. De cassetterecorder is goedkoop in aanschaf en ook de bandjes zijn niet duur. De nadelen: traag en beperkt in de mogelijkheden. De diskdrive is weliswaar veel duurder, maar heeft dan ook wel wat meer te bieden. De snelheid waarmee gegevens of programma's worden weggeschreven of ingelezen ligt veel hoger. Ook de betrouwbaarheid is beter dan die van de cassetterecorder. Bovendien heeft een gevorderd programmeur met een diskdrive veel meer mogelijkheden tot zijn of haar beschikking dan met een Datasette.

Behalve deze twee meest gebruikte mogelijkheden van informatieopslag zijn er nog meer, zoals EPROM, QuickDisk, tapestreamers of zelfs hard-disk. Ze worden echter niet vaak toegepast bij de 64 – behalve dan de EPROM in programmacartridges – en zullen dan ook pas later aan de orde komen.

6/1.1

Cassetterecorder

Inhoud

1.1 Cassetterecorder

De cassetterecorder is het meest verbreide opslagapparaat bij 64 bezitters. Voor de meeste toepassingen zijn de beperkingen niet storend, terwijl de prijs van zowel de recorder als de cassette's heel redelijk is.

Commodore computers vereisen echter wel een speciale recorder, het is niet mogelijk om zo maar een kleine cassetterecorder aan te sluiten. De computer bestuurt de motor van de recorder, door de stroom aan- en uit te schakelen. Bovendien kan de computer vaststellen of er een toets op de recorder ingedrukt is.
Speciaal voor gebruik met de 64 gebouwde recorders hebben dan ook geen eigen stroomvoorziening, maar betrekken hun spanning uit de 64 zelf. Ook de aansluitplug is nogal bijzonder, de recorder wordt met een platte, zespolige stekker op de computer aangesloten.

Behalve Commodore's eigen model, de 1530 Datasette C2N, zijn er vele andere fabrikanten op de markt gekomen met direct aansluitbare recorders. De enige eis die aan zo'n recorder gesteld moet worden - behalve dan dat het mechanisme en de elektronica betrouwbaar moeten zijn - is de aanwezigheid van een bandteller. Zonder zo'n bandteller wordt het al gauw een hopeloze zaak om een bepaald programma op een cassette terug te vinden. Naast deze direct aansluitbare recordertjes zijn er ook interfaces op de markt waarmee een willekeurige recorder op dde 64 aangesloten kan worden. Deze 'verloopstekkers' zorgen dan voor aanpassing van de signalen, zodat een audio-recorder in staat is om het computersignaal te verwerken. Soms, als de recorder deze mogelijkheid tenminste biedt, zal zo'n interface ook de motorbesturing regelen. Maar een interface is nooit in staat om aan de computer te melden of er al dan niet een toets op de recorder ingedrukt is. Meestal sluiten ze dit signaal kort, zodat de 64 altijd het 'toets ingedrukt' signaal krijgt. De gebruiker moet er dan maar voor zorgen dat dit ook inderdaad zo is.
Al met al is zo'n interface wel bruikbaar, tenminste als tijdelijke oplossing. Maar een direct aansluitbare recorder biedt een veel groter bedieningsgemak en bovendien meestal een betere signaalkwaliteit.

Cassette-keuze

De kwaliteit van de gebruikte cassette is natuurlijk ook van belang voor de betrouwbaarheid van de programma- en gegevens-opslag. Cassette's zijn in vele kwaliteiten te koop, vanaf goedkope partijen die onder de een of andere fantasienaam op de markt gedumpt worden tot de kostbaarste merk-cassette's aan toe. Ook zijn er speciale computer-cassette's te koop.

Bij de standaard schrijfmethode van de 64 maakt het allemaal niet zoveel uit welke cassette er gebruikt wordt. Het is echter nooit verstandig om zo'n goedkope dumpcassette te gebruiken, daar dit meestal afgekeurde partijen van de een of andere merk-fabrikant zijn. Vaak vertonen dit soort cassette's kleine 'drop-outs', plekjes waar het magnetisch materiaal niet goed is. Dit soort fouten zijn bij gebruik voor muziek alleen maar hinderlijk, maar bij computergebruik ronduit desastreus. Opeens worden een of meer tekens niet opgenomen en bij het later opnieuw inlezen dus ook niet gelezen. De gevolgen laten zich raden. De kwaliteit van de merk-cassette's is echter ruimschoots vol-

doende, ook de goedkoopste types functioneren heel behoorlijk. Vaak zelfs zijn de duurdere types juist minder goed bruikbaar, chroomdioxide- en metal-cassette's vereisen speciale opname instellingen die de simpele datarecorder niet heeft.

De bandlengte is weer een verhaal apart; langere (en dus dunnere) cassetteband heeft minder mechanische stevigheid. Bij de band die in een C120 cassette wordt gebruikt zal eerder rek optreden dan bij de tape in een C30. Bovendien wordt het spoelen en zoeken op die lange cassette's een tijdrovend karwei. Gebruik dus bij voorkeur korte bandlengtes, C30 voldoet goed.
De speciale computercassette's zijn meestal nog korter, C10 en C15 zijn heel gebruikelijke lengtes. Op die 5 respectievelijk 7.5 minuten per kant passen echter meer dan genoeg programma's. Een tweede voordeel van deze cassette's is het ontbreken van de lange aanloopstrook. Bij een standaard geluidscassette moet er altijd op gelet worden dat die aanloopband voorbij de kop is als er een computersignaal opgenomen gaat worden; er moet altijd eerst een klein stukje doorgespoeld worden. De computercassette kent dit probleem niet; een zorg minder voor de gebruiker.

Omdat de lees- en schrijfsnelheid van de 64 op cassette tamelijk traag zijn, wordt er vaak met allerlei hulpprogramma's gewerkt die deze snelheid opvoeren. Soms wel tot 10 keer sneller.
De 'informatie-dichtheid' op de cassetteband wordt dan ook groter, zodat er hogere eisen aan de cassette gesteld moeten worden. Gebruik in zo'n geval liefst een speciale computercassette die bovendien van een goed merk moet zijn. Het kan lang goed gaan met standaard cassette's, maar vroeger of later .....

Veiligheid

Maar ook met die speciale computercassette's gaat het ooit eens gegarandeerd mis. Opeens blijkt om welke reden dan ook een bepaalde cassette niet meer in te lezen. Die reden kan overigens heel duidelijk zijn, bijvoorbeeld een omgestoten kop koffie.
Om dergelijke problemen voor te zijn moet er altijd een 'back-up' van belangrijke programma's en bestanden gemaakt worden. Schrijf de gegevens twee keer weg, een keer op een werk-cassette en nog een keer op een back-up cassette. Die laatste wordt dan op een veilige plek opgeborgen, tot er iets fout gaat met het werk exemplaar.

6/1.1.1

Basisbegrippen

Tot nog toe is er geen onderscheid gemaakt tussen programma's enerzijds en bestanden met allerlei gegevens anderzijds. Beide laten zich zonder problemen op een cassette wegschrijven en later weer teruglezen. De gebruikte techniek wijkt echter sterk af; een programma bestaat uit een aaneengesloten stuk informatie terwijl een bestand uit vele losse stukjes kan bestaan, die met tussenpozen worden weggeschreven.

BASIC-programma's

Een BASIC-programma staat als een blok in het geheugen van de 64, het beslaat een bepaalde lengte en begint op een vast adres. Als zo'n programma naar cassette wordt weggeschreven wordt in feite de inhoud van dat geheugenblok weggeschreven. Geheugenplaats na geheugenplaats wordt uitgelezen en op de band gezet.
Om dat allemaal later weer terug te lezen is relatief simpel; het beginadres van een BASIC-programma is bekend. Door alle bytes die ingelezen worden vanaf dat beginadres weer in RAM te zetten wordt het programma geladen. Daarna moet de Kernal nog wat wijzers aanpassen en klaar is Kees.

Machinetaalprogramma's

Behalve BASIC-programma's kunnen ook machinetaalprogramma's worden weggeschreven en teruggelezen. Ook hier is er weer sprake van een aaneengesloten stuk geheugen, dat byte voor byte behandeld wordt. Alleen is het beginadres niet bekend, dat kan voor ieder ML programma verschillend zijn. Voor het teruglezen van zo'n stuk geheugen zonder vast beginadres is een speciale vorm van het leescommando aanwezig, dat nog behandeld zal worden. Wegschrijven gaat iets minder gemakkelijk, maar ook daar is een mouw aan te passen. Overigens hoeft zo'n willekeurig stuk geheugen geen machinetaal programma te bevatten, het kan ook bijvoorbeeld de inhoud van een beeldscherm in hoog oplossend vermogen zijn.

Bestanden

Bij gegevens is de situatie anders, zo'n bestand wordt vanuit een programma met allerlei verschillende zaken gevuld. Dat kunnen stukken tekst zijn, maar ook integer of real getallen. Bovendien hoeft dat schrijven vanuit het programma niet aan een stuk te gebeuren, het is heel wel denkbaar dat er tussen de verschillende schrijfopdrachten de nodige tijd verloopt. Om de cassette dan de hele tijd door te laten lopen zou erg veel ruimte op die band verspillen, ruimte die onbeschreven blijft. Vandaar dat een schrijfopdracht naar een bestand via een buffer verloopt. Als er iets geschreven wordt, dan wordt die informatie eerst in de cassettebuffer

1.1 Cassetterecorder

opgeborgen. Deze buffer bevindt zich in het geheugen vanaf adres 828 en is 192 posities lang. Pas als deze buffer vol is - of het programma afgelopen zodat het bestand gesloten moet worden - wordt dit blok van 192 tekens naar cassette weggeschreven.
Het teruglezen van zo'n bestand gaat op een vergelijkbare manier; de 64 leest steeds een hele buffer in; waarna de afzonderlijke leesopdrachten in het programma de diverse variabelen weer uit de buffer lezen. Het is hierbij wel zaak om op te passen, het per ongeluk teruglezen van een getal als tekst levert gegarandeerd problemen op.

6/1.1.2

Gebruik

Laden, saven en verifiëren van BASIC-programma's
Hiertoe zijn een drietal commando's beschikbaar, LOAD, SAVE en VERIFY. Daar dezelfde commando's ook gebruikt worden voor de diskdrive zijn er nogal wat verschillende vormen van deze commando's. Voor cassettegebruikers zijn ze echter heel simpel; de 64 neemt aan dat deze commando's op de cassette slaan als er verder niets wordt aangegeven.
Meestal wordt de volgende vorm van het SAVE commando gebruikt:
SAVE "programmanaam",1
waarbij de naam maximaal 16 tekens lang mag zijn. De 1 achter het commando geeft aan dat het programma naar de cassetterecorder weggeschreven moet worden, maar als die 1 weggelaten wordt gaat het ook goed. De computer neemt aan dat de cassetterecorder bedoeld wordt als in- en uitvoer-apparaat, als er verder niets opgegeven wordt Ook de naam van het programma mag eventueel weggelaten worden, maar dat is niet verstandig. Zonder die programmanaam wordt het lastig om later het juiste programma terug te laden.
In antwoord op het SAVE commando geeft de 64 de boodschap: PRESS RECORD & PLAY ON TAPE
Zodra er een (willekeurige) toets op de recorder ingedrukt wordt zal de boodschap OK op het scherm gezet worden en begint de 64 met wegschrijven. Het is dus nodig om van te voren de cassette op de juiste positie te zetten.
Tijdens dit wegschrijven wordt het beeldscherm blanco, dit is normaal bij alle cassette-operaties. Waarom dit zo is zal later aan de orde komen. Pas als het programma geheel weggeschreven is meldt de 64 zich weer met 'READY'.
Het programma is nu weliswaar weggeschreven, maar er bestaat altijd een zekere kans dat er daarbij iets fout gegaan is. Een slecht plekje op de band, of een slecht lopende cassette zijn maar twee van de mogelijkheden waarom een gesaved programma niet meer terug te laden kan blijken.

Om de opname te controleren is er een speciaal commando, VERIFY. Het ziet er als volgt uit:
VERIFY "programmanaam",1
waarbij alweer zowel de programmanaam als de 1, die aangeeft dat het om apparaat nummer 1 (de cassetterecorder) gaat, weggelaten mogen worden.
In de praktijk wordt dat ook bijna altijd gedaan, omdat VERIFY onmiddelijk na SAVE gebruikt wordt. Eerst terugspoelen tot even voor het begin van de opname (lastig zonder bandteller), en dan VERIFY intikken. De computer vraagt:
PRESS PLAY ON TAPE
en zal zodra dit gebeurd is gaan verifieren. Dit houdt in dat het eerste programma op de cassette (als er althans geen naam opgegeven was) wordt opgezocht, wat wordt aangegeven met de melding 'SEARCHING', of 'SEARCHING FOR

1.1 Cassetterecorder

Programmanaam', als er wel een programmanaam was opgegeven. Als er een programma gevonden is wordt ook dit gemeld, met 'FOUND programmanaam'. Deze melding blijft even op het scherm staan, het duurt ongeveer tien seconden voordat de computer verdergaat met de volgende stap. Eventueel valt deze wachttijd te bekorten met de Commodore-toest, als deze tijdens de wachttijd wordt ingedrukt gaat de computer meteen verder. De spatiebalk, de CRL toets en de toets met het pijltje naar links (linksboven op het toestenbord) hebben overigens hetzelfde effect. Althans, zo gaat dat bij bijna alle 64 computers, maar bij de allereerste modellen was er een verschil; deze bleven altijd wachten tot er op de Commodore-toets werd gedrukt.
Als de naam van dit op tape gevonden programma overeenstemt met de opgegeven naam, of als er geen naam opgegeven was, dan wordt dit programma byte voor byte vergeleken wordt met het programma in het geheugen. Op het scherm wordt dit gemeld met de boodschap 'VERIFYING'. Worden er hierbij geen verschillen gevonden dan meldt de 64 dit met 'OK', anders verschijnt de melding 'VERIFY ERROR'. In dat laatste geval is het aan te raden om het met een andere cassette nogmaals te proberen. Waarschijnlijk is de zojuist gebruikte cassette defect, of althans niet geschikt om als datacassette te worden gebruikt.

Om een BASIC-programma te laden zijn er twee mogelijkheden. De simpelste hiervan is tegelijkertijd de Commodore en de RUN/STOP toetsen indrukken, dan wordt het eerste programma op de cassette automatisch geladen en gestart. De SHIFT RUN/STOP combinatie heeft hetzelfde effect.

Daarnaast bestaat er ook een LOAD opdracht, die ziet er voor BASIC-programma's op cassette als volgt uit:
LOAD "programmanaam",1
Alweer, zowel de programmanaam als de 1 (=cassetterecorder) mogen worden weggelaten.
Welke manier van laden ook gekozen wordt, de twee toets methode of het LOAD commando, ook hier vindt weer een 'dialoog' met de computer plaats. Achtereenvolgens verschijnen de meldingen 'PRESS PLAY ON TAPE', 'OK', 'SEARCHING' of 'SEARCHING FOR PROGRAMMANAAM', 'FOUND PROGRAMMANAAM' en 'LOA DING'. Net als bij het verifieren wordt na de FOUND melding weer tien seconden gewacht, hetgeen eventueel met de spatiebalk of de Commodore-, CTRL- of pijltje links-toets bekort kan worden.
Na het laden verschijnt meestal de normale 'READY' melding, ten teken dat het programma zonder fouten geladen is. Als er wel laadfouten opgetreden zijn wordt dit met '?LOAD ERROR' aangegeven. In dat geval valt het maar te hopen dat een tweede poging beter afloopt, of dat er een backup beschikbaar is.

Laadfouten kunnen vele oorzaken hebben, in de paragraaf 6/1.1.4, Tips, worden wat trucs besproken om ze te vermijden of zelfs te omzeilen.

Bestanden

Om vanuit een programma allerlei gegevens op cassette te schrijven en weer terug te lezen wordt een hele andere route bewandeld. In zo'n geval worden bijzondere vormen van de in- en uitvoer commando's gebruikt, zoals PRINT, INPUT en GET. Er wordt dan niet naar het scherm geschreven, maar naar de cassetterecorder. Of eigenlijk naar een bestand,

1.1 Cassetterecorder

een file, dat op die cassetterecorder geopend is. Voor zo'n file beschreven kan worden moet het dus eerst geopend worden, met het OPEN commando.

Voor apparaat nummer 1, de cassetterecorder, ziet dat er als volgt uit:
OPEN filenummer,apparaatnummer ,stuurcode,FILENAAM
Het apparaatnummer is natuurlijk 1, het filenummer kan tussen de 1 en de 127 liggen. De stuurcode - meestal secundair adres genoemd - geeft aan of het bestand gelezen of geschreven gaat worden. Een 0 betekent lezen, een 1 schrijven. De filenaam is net als bij programma's maximaal 16 tekens lang en mag desgewenst worden weggelaten. Als het secundaire adres niet wordt ingevuld wordt er de waarde 0 voor genomen; OPEN 1 opent het eerste databestand op cassette voor lezen.
Na een OPEN opdracht met 0 als secundair adres, waarbij een bestand geopend wordt om te lezen, zal de computer dit file op de cassette gaan zoeken. Als er een naam was opgegeven zal er gezocht worden naar een bestand met die naam, anders zal het eerste bestand dat zich op de cassette bevindt geopend worden. Daarbij zal de bekende 'PRESS PLAY ON TAPE' melding verschijnen. Een OPEN commando met 1 als secundair adres geeft 'PRESS RECORD & PLAY ON TAPE'.
Na een OPEN commando wordt eerst het bestand ook daadwerkelijk geopend, een te lezen file wordt opgezocht waarna de eerste buffer wordt ingelezen, voor een te schrijven file wordt een zogenaamde 'header' waarin onder andere de bestandsnaam staat geschreven. Pas dan gaat het programma verder.

Het filenummer achter het OPEN commando kan misverstanden geven, soms denkt men dan ook meerdere bestanden tegelijk te kunnen openen. Dat is echter bij de cassetterecorder niet mogelijk, hoewel er geen foutmelding wordt gegeven. Alle lees- en schrijfopdrachten naar een cassettebestand gaan via de cassettebuffer en daar is er maar een van. Het openen van een reeds in gebruik zijnd filenummer geeft wel een foutmelding, 'FILE OPEN'.

Het schrijven naar een geopend file is simpel, de opdracht:
PRINT#filenummer, variabele of constante
zal de genoemde waarden in het met filenummer aangeduide bestand plaatsen. Alle in BASIC 2.0 PRINT statements toegestane constructies mogen gebruikt worden. Zowel constanten, zoals "tekst" als 12345.12 als variabelen, zoals A, A$ of A% mogen. Het PRINT# commando lijkt als twee druppels water op het gewone PRINT commando. Ook constructies als:
PRINT#1,"Dit is een tekst";N
zijn toegestaan, waarbij N voor een getalvariabele staat.

Dit leidt echter wel tot allerlei problemen bij het teruglezen van de informatie van een bestand. Het daartoe gebruikte
INPUT#filenummer,variabelelijst
commando lijkt namelijk ook sterk op het overeenkomstige INPUT commando. De diverse in te lezen waarden dienen op een juiste manier van elkaar gescheiden te worden. Bovendien is het ook bij bestanden onmogelijk om bijvoorbeeld een tekst in een getalsvariabele in te lezen.
Bij een rechtstreekse INPUT vanaf het toetsenbord kent de 64 bij zulke fouten allerlei beveiligingen en waarschuwingen,

maar bij bestanden moet de programmeur dat van te voren goed plannen. Anders gaat er van alles mis!

De simpelste wijze om deze problemen te omzeilen is als volgt: scheidt alle weer in te lezen stukjes informatie met een <RETURN>, oftewel een CHR$(13).
Dat kan door alle variabelen of constantes met een eigen PRINT# opdracht op tape te zetten waarbij de puntkomma of de komma als scheiders uit den boze zijn. De puntkomma laat immers de geprinte informatie keurig aansluiten, terwijl de komma het nog bonter maakt en de geprinte gegevens in kolommen met 10 plaatsen tussenruimte verdeelt. Er zijn weliswaar best manieren om dit soort problemen te omzeilen, maar die komen later nog wel aan bod.
PRINT#1,"tekst","meer tekst"
geeft ook in een bestand:
tekst meer tekst
Niet alleen mist de scheider, maar er zijn zelfs overbodige spaties in het bestand geschreven. Om een bestand weer goed terug te kunnen lezen moet ieder stukje informatie van alle andere stukjes gescheiden worden. In de voorbeelden, 6/1.1.3, wordt uit de doeken gedaan hoe dit het beste kan gebeuren.

Een tweede probleem bij het weer inlezen van de gegevens is hoe te bepalen wanneer de gegevens 'op' zijn. Het is natuurlijk onmogelijk om meer gegevens uit een bestand te lezen dan erin gezet zijn, maar het is aan de programmeur om ervoor te zorgen dat het inlezen inderdaad op tijd stopt. Daarom wordt na iedere I/O operatie - dat staat voor input/output, iedere handeling met een randapparaat - de status van die operatie in de systeemvariabele ST gezet. Deze variabele kan alleen maar gelezen worden. De ST wordt voor vele doeleinden gebruikt, niet alleen voor cassette, en ook tijdens cassette I/O kan de ST meerdere zaken aangeven. Als na een lees- of schrijfopdracht de ST gelijk aan 0 is, dan is alles goed gegaan. Om te ontdekken of onder het lezen het einde van een bestand bereikt is kan het beste de test:
IF ST AND 64 THEN GOTO einde bestand afhandeling
gebruikt worden. Op de andere mogelijke waarden die ST aan kan nemen wordt later teruggekomen. Overigens kan de bovenvermelde programmaregel aanleiding geven tot zeer frustrerende foutmeldingen. Als de spaties weggelaten zouden worden, wat vaak gedaan wordt, gaat het er als volgt uitzien: IFSTAND64THEN-GOTOregelnr
hetgeen op een zeer hardnekkige 'SYNTAX ERROR' uitdraait. De letters 4 t/m 6 uit deze regel vormen namelijk het woord TAN en dat is een legale BASIC-functie. Die overigens in deze regel inderdaad een taalfout voor BASIC is. Wel iets om op te letten!

Die 'einde bestand' routine is heel simpel; het bestand moet weer gesloten worden met de CLOSE opdracht.
CLOSE filenummer
Na het CLOSE commando is het betreffende filenummer weer beschikbaar om eventueel opnieuw te gebruiken met een ander bestand. Bovendien wordt een bestand dat beschreven was door de CLOSE netjes afgesloten, de resterende gegevens in de buffer worden alsnog op de cassette gezet. Zonder de CLOSE zou dit niet gebeuren en blijkt later bij inlezen dat de laatste gegevens nooit weggeschreven zijn.

1.1 Cassetterecorder

Het commando INPUT#1,A$ heeft in BASIC 2.0 een beperking; de aldus ingelezen string mag niet langer dan 80 tekens zijn. De maximale lengte van een stringvariabele is echter veel groter. Zo'n lange string is zonder problemen te printen, ook naar een bestand. Er kunnen dus langere strings in een bestand staan dan er met INPUT# kunnen worden ingelezen. Om die lange strings alsnog terug te kunnen lezen is er het GET# commando:
GET# filenummer, stringvariabelelijst
Voor iedere variabelenaam uit de lijst - meestal wordt er slechts een gebruikt - leest de GET# een enkel teken van band. Alle tekens worden gelezen, ook de scheiders als CHR$(13) en de komma. Door nu met GET# de gegevens in te lezen en ze daarna weer tot langere strings samen te voegen kunnen ook lange strings weer opgebouwd worden. Een waarschuwing: een teken met code 0 - CHR$(0) - wordt niet als zodanig ingelezen, maar als een lege string. GET#1,A$:PRINT ASC(A$) kan de foutmelding 'ILLEGAL QUANTITY ERROR' opleveren. Er zijn meerdere mogelijkheden om dit te omzeilen, zie verder 6/1.1.3, voorbeelden.
Het filenummer na PRINT# en INPUT# moet al bekend zijn uit een eerder OPEN commando, anders volgt de foutmelding: '?FILE NOT OPEN'.

Een laatste commando dat bij bestanden gebruikt kan worden is:
CMD filenummer,tekst
Dit heeft tot gevolg dat alle uitvoer die normaal naar het beeldscherm gezonden wordt, nu naar het met filenummer aangegeven bestand gaat. De eventueel achter het filenummer opgegeven tekst wordt als eerste regel naar dat file gezonden.
Met CMD kan bijvoorbeeld de listing van een programma in een cassettebestand gezet worden, in hetzelfde formaat als die listing ook op het beeldscherm verschijnt. Zo'n bestand zou dan later weer gebruikt kunnen worden door een ander programma. Na de laatste uitvoer die via CMD naar een bestand gestuurd is moet er altijd nog een lege regel geprint worden, om het bestand goed af te sluiten. Zie de voorbeelden.

6/1.1.3

Voorbeelden

Nu dan wat voorbeelden van de programmering van bestanden op de cassetterecorder. Het laden en saven van BASIC programma's is zo simpel dat er verder geen voorbeelden bij nodig zijn, maar met bestanden ligt dat wel even anders.

```
   REM NUMERIEKE VARIABELE IN
   BESTAND SCHRIJVEN
10 OPEN1,1,1,"TEST1"
20 A=12345
30 PRINT#1,A
40 CLOSE1
```

Simpel genoeg, dit eerste begin. Zet overigens eerst de bandteller op 0, dat is zo meteen makkelijk.
In regel 10 wordt er een bestand met de naam TEST1 met filenummer 1 geopend op cassette om te beschrijven, in 30 wordt op dat filenummer 1 de variabele A geschreven. Het bestand wordt in 40 nog even netjes afgesloten. Nu dan het lezen, de waarde 12345 die zonet in TEST1 geschreven is moet er ook weer uit terug te lezen zijn. Eerst de tape terugspoelen naar 0, dat wel.

```
   REM NUMERIEKE VARIABELE
   UIT BESTAND LEZEN
10 OPEN1,1,0,"TEST1"
20 INPUT#1,A
30 PRINTA
40 CLOSE1
```

Alweer eenvoudig. De derde parameter achter het OPEN commando is dit keer een 0, waardoor TEST1 voor lezen wordt geopend. De enige waarde die in TEST1, bestand nummer 1, staat wordt daarna uitgelezen en afgedrukt. Tenslotte weer netjes de deur dichtdoen: CLOSE1.
Met teksten kan het ook, Kijk maar:

```
   REM ALFANUMERIEKE VARIABELE
   IN BESTAND SCHRIJVEN
10 OPEN1,1,1,"TEST2"
20 A$="CASSETTE-TEXT"
30 PRINT#1,A$
40 CLOSE1
```

Dit lijkt zo sterk op het eerste voorbeeld dat het niet verder uitgelegd hoeft te worden. Dat geldt ook voor het volgende programma, dat die tekst weer terugleest:

```
   REM ALFANUMERIEKE VARIABELE
   UIT BESTAND LEZEN
10 OPEN1,1,0,"TEST2"
20 INPUT#1,A$
30 PRINTA$
40 CLOSE1
```

Het kan overigens nog simpeler. Zet het laatste bestandje, TEST2, maar weer klaar om te lezen en probeer:

```
   REM ALFANUMERIEKE VARIABELE
   UIT BESTAND LEZEN
10 OPEN1
20 INPUT#1,A$
30 PRINTA$
40 CLOSE1
```

Het OPEN commando in regel 10 hoeft in feite alleen maar het filenummer te weten. De tweede parameter, het apparaatnummer, wordt automatisch op 1 - de cassette dus - gezet als deze wordt weggelaten. De derde parameter wordt indien niet ingevuld als 0 gezien, lezen dus. De bestandsnaam kan ook al vervallen, maar dat heeft een nadeel. Dan wordt namelijk het eerste bestand op de cassette gelezen, zonder dat er een controle is of dit wel het bedoelde bestand is. Natuurlijk kunnen er meerdere waarden in een bestand gezet worden:

```
   REM DIVERSE VARIABELEN IN
   BESTAND SCHRIJVEN
10 OPEN1,1,1,"TEST3"
20 A$="CASSETTE-TEXT":A=12345:
B$="ALWEER ALFANUMERIEK":B=54321
30 PRINT#1,A$
40 PRINT#1,A
50 PRINT#1,B$
60 PRINT#1,B
70 CLOSE1
```

Maar liefst vier tegelijkertijd en nog wel van verschillende types. Zowel teksten als getallen gaat goed, maar de echte test is pas om ze weer in te lezen:

```
   REM DIVERSE VARIABELEN
   UIT BESTAND LEZEN
10 OPEN1,1,0,"TEST3"
20 INPUT#1,X$,Y,Y$,X
30 PRINTX$
40 PRINTX
50 PRINTY$
60 PRINTY
70 CLOSE1
```

Dat lukt ook. Er lijkt weliswaar een fout op te treden, de volgorde is anders dan bij het wegschrijven, maar dat ligt aan de omdraaiing bij het teruglezen en printen. In regel 20 wordt eerst Y en dan pas X ingelezen, terwijl bij het afdrukken de X voor de Y komt. De variabelen komen dus wel degelijk in de goede volgorde uit het bestand.
Het volgende voorbeeld leest hetzelfde TEST3 bestandje.

```
   REM DIVERSE VARIABELEN UIT
   BESTAND LEZEN MET
   VOLGORDE-FOUT
10 OPEN1,1,0,"TEST3"
20 INPUT#1,X$,X,Y,Y$
30 PRINTX$
40 PRINTX
50 PRINTY$
60 PRINTY
70 CLOSE1
```

Ja, dat gaat fout. Gelukkig een fout die de 64 ontdekt, anders was het misschien niet eens opgevallen. Die melding '?FILE DATA ERROR IN 20' ligt aan het volgorde van de variabelen die er in die regel gelezen worden. Als er een string is weggeschreven, moet die ook weer in een stringvariabele worden ingelezen. Anders komt dat op deze foutmelding te staan. Overigens zijn de eerste twee variabelen, X$ en X, wel goed ingelezen. Probeer maar eens in direct mode: PRINT X$ en PRINT X. Even oppassen hiermee, want de andere kant op gaat het wel. Een getal teruglezen in een stringvariabele levert geen fouten op, dat staat de programmeur helemaal vrij. Soms is dat handig, maar het blijft dus wel opletten dat de variabelen in de juiste volgorde worden teruggelezen.

Nog eens met TEST3:

```
   REM DIVERSE VARIABELEN UIT
   BESTAND LEZEN, TEVEEL
10 OPEN1,1,0,"TEST3"
20 INPUT#1,X$,X,Y$,Y,Z$,Z
30 PRINTX$
40 PRINTX
50 PRINTY$
60 PRINTY
70 PRINTZ$
80 PRINTZ
90 CLOSE1
```

1.1 Cassetterecorder

En dat gaat ook al niet! De foutmelding '?STRING TOO LONG ERROR IN 20' is jammer genoeg wat onduidelijk, maar wel verklaarbaar. Waar er vier variabelen waren probeerde dit voorbeeld er zes in te lezen. Dat ging dan ook fout bij de vijfde, de eerste die niet in het bestand stond (X$, X, Y$ en Y zijn wel goed ingelezen). Terwijl het programma die Z$ probeerde te lezen was het bestand al leeg, maar blijkbaar kwamen er toch nog wel codes van de cassette af. Maar omdat deze codes niet werden afgesloten door een scheidingsteken, werd de ingelezen lengte groter dan 80 tekens, welke tekens dat ook waren. En 80 tekens is nu eenmaal de bovengrens voor het INPUT commando, als het om strings gaat. Daarom verschijnt deze foutmelding.

Blijkbaar staan er dus behalve de tekens die in de weggeschreven variabelen staan ook nog andere zaken op de cassette, zoals die scheidings-tekens. Dat klopt dan ook, alle programma's tot nog toe schreven de variabelen ieder met een eigen PRINT# commando weg, er stond dus automatisch een einde-regel teken, CHR$(13), achter, dat als scheider functioneerde.
Maar ook dat helpt niet bij dit probleem, het gaat er allereerst om het einde van een bestand tijdig te ontdekken. Dat kan met de systeemvariabele ST, kijk maar:

```
   REM DIVERSE VARIABELEN UIT
   BESTAND LEZEN,
   MET EINDE BESTAND TEST
10 OPEN1,1,0,"TEST3"
20 INPUT#1,A$,A
30 PRINTA$
40 PRINTA
50 IF ST=0 THEN GOTO 20
60 IF ST<>64 THEN PRINT "ER IS
   EEN FOUT OPGETREDEN": GOTO 80
70 PRINT "EINDE BESTAND BEREIKT"
80 CLOSE1
```

Hier is nogal wat veranderd vergeleken met de vorige lees-programma's. Het lezen en afdrukken van de variabelen gebeurt nu in een lus, die pas verlaten wordt als in regel 50 ST ongelijk aan 0 is. Dan zijn er meerdere mogelijkheden, ST kan gelijk aan 64 zijn, wat betekent dat het bestand 'op' is, of de waarde van ST kan de een of andere fout aanduiden. Dat laatste komt echter niet veel voor en zal pas later aan de orde komen.
Op zo'n manier kan een bestand dus worden gelezen zonder dat het aantal waarden van tevoren bekend hoeft te zijn. De logica van regel 50 is: „als het bestand nog niet leeg is, ga dan weer lezen". Wat natuurlijk wel bekend moet zijn is de soort van de te lezen variabelen, die volgorde moet vast liggen.

Tot nog toe is de beperking van maximaal 80 tekens tekst in een tekstvariabele inlezen nog geen probleem geweest. Toch zal dat vroeger of later gaan gebeuren, zeker na het volgende programma:

```
   REM EEN LANGE STRING IN
   BESTAND SCHRIJVEN
10 OPEN1,1,1,"TEST4"
20 S$=CHR$(13)
30 A$="DEZE TEKST WORDT WAT
   LANGER DAN GEBRUIKELIJK,"
40 B$="MAAR LEVERT NOG GEEN
   ECHTE PROBLEMEN OP."
50 C$="DAT KOMT ECHTER ZOMETEEN!"
60 D$=A$+B$+C$
70 PRINT#1,A$;S$;B$;S$;C$;S$;D$
80 CLOSE1
```

Er worden vier teksten in het bestand TEST4 geschreven, met een slechts een PRINT# opdracht. Daarvoor wordt een truc gebruikt, de CHR$(13), die tot nog toe vanzelf tussen de verschillende stukken in het bestand geschreven werd staat

1.1 Cassetterecorder

in S$, zie regel 20. Door tussen de verschillende stringvariabelen iedere keer ook een S$ weg te schrijven zijn ze keurig van elkaar gescheiden door de einde-regel code. Op zo'n manier kunnen er makkelijk meerdere variabelen met een PRINT# commando worden gebruikt. Let wel op de leestekens tussen de variabelenamen, de komma bijvoorbeeld levert problemen op, de puntkomma is altijd goed.
Het nu gaan teruglezen is een heel ander verhaal. De eerste drie teksten in het bestand, A$, B$ en C$ zijn niet lastig, maar D$ is een combinatie van de eerste drie, veel langer dan 80 tekens. Hoe nu? Met INPUT# gaat het niet helemaal goed, zoals het volgende voorbeeld aantoont.

```
   REM TEKST VARIABELEN UIT
   BESTAND LEZEN, MET EINDE
   BESTAND TEST
10 OPEN1,1,0,"TEST4"
20 INPUT#1,A$
30 PRINTA$
40 IF ST=0 THEN GOTO 20
50 IF ST<>64 THEN PRINT "ER IS
   EEN FOUT OPGETREDEN": GOTO 80
60 PRINT "EINDE BESTAND BEREIKT"
70 CLOSE1
```

Het programma lijkt sterk op het vorige leesprogramma, alleen worden er nu alleen strings gelezen, een per keer. Het gaat ook een heel eind goed, tot de vierde tekst aan de beurt is. Dan verschijnt er, nauwelijks onverwacht, '?STRING TOO LONG ERROR IN 20'. Blijkbaar is INPUT# niet bruikbaar voor lange teksten.
In zo'n geval brengt het GET# commando uitkomst:

```
   REM LANGE TEKST VARIABELEN
   UIT BESTAND LEZEN, MET EINDE
   BESTAND TEST
10 OPEN1,1,0,"TEST4"
20 A$=""
30 GET#1,B$: IF B$=""
   THEN GOTO 30
40 IF B$=CHR$(13) THEN GOTO 60
50 A$=A$+B$: GOTO 30
60 PRINT "INGELEZEN REGEL:"
70 PRINT A$
80 IF ST=0 THEN GOTO 20
90 IF ST<>64 THEN PRINT "ER IS
   EEN FOUT OPGETREDEN":
   GOTO 80
100 PRINT "EINDE BESTAND BEREIKT"
110 CLOSE1
```

Het eenvoudige INPUT# uit de vorige programma's is vervangen door de constructie in de regels 20 tot en met 70. Eerst wordt de variabele A$, waarin de tekst opgebouwd gaat worden, leeg gemaakt. De eerste keer is dat perse niet nodig, maar bij het gegin van het inlezen van de tweede tekst zal er al wat instaan, namelijk de eerste tekst.
In regel 30 leest GET# steeds 1 teken in, dat in B$ terechtkomt. Het kan gebeuren dat GET# een lege string oplevert, ook dat wordt in regel 30 ondervangen. In zo'n geval wordt meteen de volgende GET# uitgevoerd.
Als er wel iets ingelezen was, dan wordt in regel 40 gekeken of dat CHR$(13) was, de scheider. Zo ja, dan wordt de som van de ingelezen strings, A$, afgedrukt. Anders wordt het teken in B$ bij de tekst in opbouw in A$ gevoegd en springt het programma terug naar het lezen in regel 30.
Deze constructie heeft voor- en nadelen. Want hoewel het verreweg de veiligste manier is om een bestand te lezen waarbij er nooit fouten kunnen optreden, werkt het veel trager dan de INPUT# methode. Niet alleen omdat daar de tekst met een enkel commando werd ingelezen, maar voornamelijk omdat de hulpstring B$ zo vaak opnieuw ingelezen wordt. Dat leidt tot op de lange duur tot een 'garbage-collect', waarover in hoofdstuk 5 meer verteld wordt.

6/1.1.4

Tips

Opname bescherming

Iedere cassette heeft aan de achterzijde twee uitbreekbare nokjes. Als het linker nokje wordt verwijderd - met een kleine schroevendraaier bijvoorbeeld - is de kant die naar boven toe ligt (met de band naar voren) beveiligd tegen per ongeluk opnemen. Dat is niet alleen zo op gewone recorders, maar ook op datarecorders.
Het is zonder meer aan te raden om deze beveiliging altijd te gebruiken als een cassette kant niet verder beschreven hoeft te worden. Maar al te vaak worden stukken van computercassette's per ongeluk gewist, even niet opletten en het is te laat. Mocht de cassette later toch weer beschreven moeten worden is de beveiliging altijd weer op te heffen met een stukje plakband, dat over de plaats waar het nokje zat geplakt kan worden.

Tape-indeling

Neem altijd wat extra tussenruimte op de cassette bij het wegschrijven van programma's. Zet ze niet zo dicht mogelijk achter elkaar, maar laat de bandteller doorlopen tot een mooi rond getal. Dit vergemakkelijkt het opzoeken van een programma, tenminste als u die tellerstand wel even noteert
Een tweede voordeel van deze techniek is dat een gewijzigd programma dan met een beetje geluk weer op dezelfde cassette kan worden teruggeschreven. Zonder dat alle programma's die verderop op de cassette staan ook opnieuw gesaved moeten worden. Let er wel goed op dat zo'n programma niet te lang mag zijn geworden door de veranderingen, anders wordt het begin van het volgende programma toch nog overschreven. Voorzichtigheid is de boodschap bij deze truc!

Tape-fouten door stoorsignalen

Er kan van alles mis gaan bij het gebruik van de datarecorder. Laad-fouten, verminkte gegevens in bestanden, noem maar op.
Een mogelijke oorzaak is de opstelling van de datarecorder. Deze is namelijk nogal gevoelig voor allerlei stoorsignalen. Net zoals allerlei huishoudelijke apparaten de radio en de TV kunnen storen. Veel stoorsignalen zijn afkomstig van de computerapparauur zelf, zowel de TV of monitor als de 64 - en ook een eventuele diskdrive of printer - zijn allemaal bronnen van storingen voor de datarecorder.
Vaak helpt het al als de recorder verplaatst wordt, zover mogelijk van het beeldscherm. Het is zaak om met uitproberen de beste opstelling te vinden.

'?LOAD ERROR' omzeilen

Ook dat kan gelukkig, omdat ieder programma twee keer achter elkaar op de band gezet wordt door de computer. Dat is een van de manieren waarop de 64

laadfouten ontdekt, door die twee opnames met elkaar te vergelijken. Iedere laadoperatie heeft als het ware een ingebouwde VERIFY.
Als de fout waardoor de '?LOAD ERROR' onstaat nu in de tweede opname schuilt, staat er dus al een programma dat wel goed is in het geheugen. Maar de Kernal zet de diverse wijzers niet goed, als er een fout optreed, zodat dat programma niet te gebruiken of zelfs maar te listen valt. Die wijzers kunnen echter ook met enige goedgemikte POKE's vanaf het toetsenbord worden ingesteld. Tik onmiddelijk na het optreden van de '?LOAD ERROR', dus zonder ook maar iets anders te doen, het volgende in:

```
POKE45,PEEK(174):
POKE46,PEEK(175):CLR
```

gevolgd door <RETURN>. De kans is groot dat er dan alsnog een perfect programma verschijnt.

Cassette buffer bekijken

Als de 64 met cassettebestanden werkt, worden de geheugenplaatsen 828-1019 als buffer gebruikt. Normaal gesproken, als alles goed gaat, heeft de programmeur daar weinig te zoeken. Maar als er allerlei fouten optreden bij het inlezen van een bestand kan het nuttig zijn om eens in die buffer te kijken.
Het makkelijkst gaat dat met de volgende commandoregel, in direct mode:

```
FOR N=828 TO 1019: PRINT
CHR$(PEEK(N));: NEXT N
```

die keurig alle tekens in de buffer op het scherm zet.

6/1.2

Diskdrive

Inhoud

1.2 Diskdrive

Hoewel duurder dan de cassetterecorder kiezen toch veel mensen voor een diskdrive als opslagapparaat. De mogelijkheden – en natuurlijk de snelheid – zijn nu eenmaal veel groter.

Om eerst maar eens op die snelheid in te gaan; het tempo waarmee de standaard diskdrive voor de Commodore 64, de 1541, informatie wegschrijft en weer terugleest is inderdaad veel groter dan die van de cassetterecorder.
Zo kost het laden van een 16K groot programma vanaf cassette zo'n 330 seconden, hetzelfde programma laden van disk kost slechts 50 seconden, dat is bijna zeven keer sneller. Toch staat de 1541 diskdrive bekend als een uiterst trage drive, vergeleken met allerlei andere merken – die echter voor andere computers bestemd zijn – slaat de Commodore 1541 een slecht figuur. In de eerste instantie zal de nieuwe disk-gebruiker daar nog niet zo'n erg in hebben, maar na enige tijd gaan de lange wachttijden toch storen. Gelukkig zijn er legio mogelijkheden om hier verandering in te brengen, vanaf handige stukken software – diskturbo's – tot en met volledig andere drives. Een aantal fabrikanten maakt diskdrives die op de 64 aangesloten kunnen worden, al dan niet met behulp van een interface; sommige van deze drives geven een aanzienlijke verbetering qua snelheid. In komende aanvullingen zult u hier meer informatie over vinden, zowel in dit hoofdstuk als in 14: productinformatie.

Daar echter verreweg de meeste disk-gebruikers een standaard 1541 diskdrive hebben staan, al dan niet voorzien van soft- of hardware-matige extra's, zullen we ons eerst op dit apparaat concentreren.

Met zo'n diskdrive worden allerlei zaken veel eenvoudiger dan met een cassetterecorder. Zo is de drive zelf in staat om de diverse bestanden en programma's die op een diskette opgeslagen staan te localiseren. U hoeft er niet voor te zorgen dat een cassette op de juiste positie gepositioneerd staat.
De diskdrive is, zoals dat heet, intelligent. In feite is de 1541 een heel gespecialiseerde micro-computer. Behalve het mechanisme, dat de disk kan lezen en schrijven, zit er een microprocessor in. Een 6502, om precies te zijn, een processor die op een paar kleine verschillen na gelijk is aan de 6510 processor, die in de Commodore 64 zelf zit.
Verder heeft de diskdrive 16K ROM aan boord, dat het besturingsprogramma voor de drive bevat en ook nog eens 2K RAM, dat onder andere gebruikt wordt om gegevens te bufferen (tijdelijk op te slaan). Later zullen we leren hoe we zelf die micro-processor in de 1541 kunnen programmeren en zodoende allerlei in feite "onmogelijke" trucs kunnen uithalen, zoals het versnellen van de communicatie met de drive.

Aansluiten en aanzetten

Op het eerste gezicht zou men echter niet zeggen, dat de 1541 zoveel mogelijkheden biedt. Het is een eenvoudig kastje, waarin we een diskette kunnen steken, met aan de voorzijde verder twee waarschuwingslampjes, een rode en een groene. Aan de achterkant vinden we dan nog een aan- en uitschakelaar, een aansluiting voor het netsnoer en een tweetal "seriële" bussen.
Die twee seriële bussen zijn volstrekt gelijk, het maakt niets uit op welke van de twee de computer aangesloten wordt.

1.2 Diskdrive

De andere bus kan eventueel worden gebruikt om een tweede randapparaat op aan te sluiten, zoals een printer of een tweede diskdrive. Meer informatie over de Commodore seriele bus vindt u in hoofdstuk 4, de opbouw. Let wel op, dat u nooit ofte nimmer zo'n plug mag losmaken, terwijl een van de apparaten van het systeem aanstaat. Het kan weliswaar goed aflopen, maar de kansen dat er een duur IC de geest geeft, zijn groot.

Volgens de fabrikant bestaat dat gevaar ook als de diverse apparaten in de verkeerde volgorde zouden worden aangezet. In principe moet de computer altijd als laatste worden ingeschakeld, maar in de praktijk gebruiken veel mensen een meervoudige contactdoos met een centrale schakelaar, hetgeen vrijwel altijd uitstekend functioneert. Mocht u onverklaarbare problemen hebben met uw systeem, dan kan het nooit kwaad om zich aan de voorgeschreven volgorde te houden.

Wel iets om op te letten is, dat er nooit een diskette in de drive mag zitten tijdens het aan- of uitschakelen. Door allerlei korte spanningspieken in de elektronica kan zo'n diskette op dat moment hopeloos verminkt worden.
Als het rode waarschuwingslampje brandt geldt hetzelfde, dan wordt de disk namelijk gelezen of beschreven. Op zo'n moment het deurtje van de drive openen, kan de informatie op de diskette beschadigen.
Het groene waarschuwingslampje geeft alleen maar aan, dat de drive aanstaat, hoewel in sommige 1541 gebruiksaanwijzingen ten onrechte te lezen valt, dat de disk niet verwisseld mag worden terwijl de groene lamp brandt.

De rode lamp kan behalve voortdurend branden ook nog knipperen. Dat geeft aan, dat het besturings-programma van de 1541 een fout heeft geconstateerd. Voorlopig is het devies in zo'n geval: even rustig nadenken wat het zou kunnen zijn en daarna nog eens, maar dan wel goed, doen. Later in dit hoofdstuk zal getoond worden hoe de foutmelding uit te lezen valt, zodat men precies weet wat er aan de hand is.

Werking

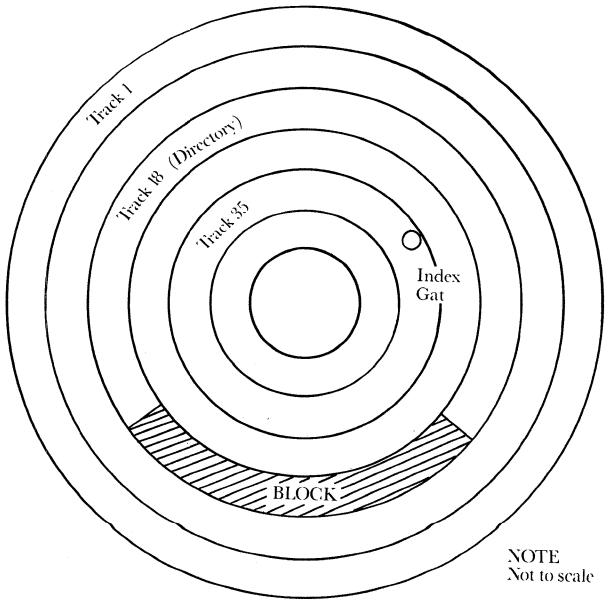
Het werkingsprincipe van de diskdrive is volledig afwijkend van dat van de cassetterecorder. Op een cassette staat alle informatie achter elkaar op een band, de informatie kan slechts in een bepaalde volgorde doorlopen worden.
Een diskette werkt weliswaar ook volgens het magnetische principe, maar de wijze waarop de informatie wordt opgeslagen is volledig anders. Op een disk wordt tijdens het zogenaamde formatteren een magnetisch patroon van blokken aangebracht, waarbinnen de diverse stukken informatie geschreven zullen worden. Daartoe wordt de diskette verdeeld in een aantal concentrische sporen, die op hun beurt weer ieder een aantal sectoren bevatten.

Tijdens het formatteren "tekent" de 1541 een patroon van 35 sporen op de disk, die afhankelijk van de plaats waar ze op de disk staan 21, 19, 18 of 17 sectoren bevatten. Dit is een van de bijzonderheden van de 1541 drive, hierdoor wordt namelijk slim gebruik gemaakt van het feit dat een spoor aan de buitenrand van de disk langer is dan een spoor aan de binnenrand.

1.2 Diskdrive

Ieder van die in totaal 683 blokken, zoals zo'n unieke combinatie van track- en sector-nummer heet, kan 256 bytes informatie bevatten. De theoretische capaciteit van de 1541 is dan ook 174848 bytes. Daar een byte overeenkomt met een letterteken komt dat overeen met zo'n 50 redelijk dicht getypte A4 velletjes.

In de praktijk ligt de capaciteit echter wel wat lager. De diskdrive moet namelijk ook bijhouden welk bestand waar staat en ook deze informatie staat op de diskette. Een spoor, nummer 18, is zelfs helemaal gereserveerd voor de floppydrive zelf, hier staat onder meer de zogenaamde directory, de inhoudsopgave. In deze directory staat, behalve de naam van de file, onder meer op welk blok dit bestand begint, waar de eerste 254 bytes informatie van dit bestand staan.
Inderdaad, slechts 254 bytes. Van ieder 256 bytes blok worden slechts 254 bytes voor de eigenlijke informatie-opslag gebruikt, de beide missende bytes zijn no



Figuur 6/1.2-1: Disk-structuur

dig om aan te geven op welke track en sector het volgende blok te vinden is. Later zal er nog zeer uitvoerig worden teruggekomen op de logische structuur van de diskette. Uiteindelijk komt het er op neer dat van de theoretische 174848 bytes er 168656 overblijven om werkelijk te gebruiken.

Mogelijkheden

De mogelijkheden van deze manier van opslag zijn veel en veel groter dan die van een cassetterecorder. De diskdrive kan, bijvoorbeeld na een laad-commando, zelf in de directory van de schijf kijken of dat bestand bestaat en het, als dat tenminste het geval is, direct laden. Alle blokken worden immers gekenmerkt door een unieke combinatie van track- en sectornummer.

Bij het wegschrijven gebeurt iets vergelijkbaars, eerst zal het bedrijfssysteem van de 1541 controleren of de naam die er bij de schrijf-opdracht meegegeven is al bestaat, als dit het geval is volgt er een foutmelding. In verreweg de meeste gevallen zal dit de reden zijn van het knipperen van het rode lampje bij een SAVE-opdracht.
Als de naam nog niet bekend was zal het programma netjes worden opgeslagen, waarbij er natuurlijk geen al in gebruik zijnde blokken mogen worden overschreven. Daartoe wordt van alle blokken bijgehouden of ze al dan niet in gebruik zijn. Dat gebeurt in de BAM, de Block Availability Map –Blok Beschikbaar Tabel – die ook op de systeemtrack 18 staat. Tijdens het wegschrijven van een bestand wordt steeds naar lege blokken gezocht in die BAM en als er een blok toegewezen wordt wordt ook de BAM

weer bijgewerkt. Daarbij worden meteen keurig alle pointers – wijzers – bijgehouden, de twee bytes in ieder blok die aangeven waar het volgende blok staat.

Relatieve bestanden

Maar een diskdrive heeft nog een heel belangrijke extra mogelijkheid. Want wat er tot nog toe beschreven is verschilt, behalve in gebruiksgemak, niet zo erg veel van de mogelijkheden die een cassetterecorder biedt. Hoewel de diskette op zich niet sequentieel is, per slot van rekening kan de lees- en schrijfkop naar ieder willekeurig blok op de schijf bewogen worden, is zo'n bestand dat juist wel.
Teruglezen gaat keurig, blok voor blok, in precies dezelfde volgorde als waarin het weggeschreven is.
Stel dat er adressen in een zo'n bestand zouden staan, met als vijfde de heer Jansen, dan zal een programma dat dit bestand leest de eerste vier personen, met hun gegevens, moeten lezen om tenslotte bij de vijfde, de heer Jansen, aan te landen.

Pure tijdverspilling in feite. Het zou immers veel sneller zijn als het programma rechtstreeks naar dat vijfde record – zoals zo'n combinatie van bij elkaar behorende gegevens genoemd wordt – kon gaan? Zonder eerst de eerste vier te moeten inlezen?
Dat is dan ook precies de extra mogelijkheid die diskdrives bieden. Behalve het sequentiele bestand, dat net als een tape in volgorde gelezen en geschreven moet worden, kent de 1541 ook relatieve bestanden, vaak ook willekeurig toegankelijke bestanden genoemd. Die tweede naam drukt eigenlijk duidelijker uit waar het om gaat, zo'n bestand kan namelijk op ieder gewenst punt worden ingelezen en beschreven. Voor een willekeurig toegankelijk bestand geldt dat het bestaat uit een aantal van tevoren gedefineerde records, die ieder uit een of meer velden van een vastgelegde lengte bestaan.
Bijvoorbeeld:
Naam, 20 posities
Adres, 25 posities
Postcode en woonplaats, 30 posities
Telefoonnummer, 10 posities

Als ieder record nu maar precies van dezelfde lengte is – niet gebruikte posities nemen dus wel ruimte op de diskette in, ze zijn met spaties gevuld – dan kan de diskdrive ieder gewenst record rechtstreeks lezen en ook weer beschrijven. Behalve snelheid – er hoeven immers niet eerst een aantal records te worden overgeslagen om bij de heer Jansen te komen – heeft dat nog een tweede voordeel.

Bij sequentiële bestanden is de totale bestandslengte in feite beperkt tot wat er in een keer in het geheugen past. Anders ontstaan er grote problemen bij het hanteren van zo'n bestand, de programmering wordt erg complex.
Bij relatieve bestanden bestaat die beperking niet meer. Het programma leest en schrijft exact dat record waar het om gaat, de maximale bestandslengte wordt alleen nog maar beperkt door de capaciteit van de diskdrive. En die is groot genoeg voor bijna alle doeleinden.
Later in dit hoofdstuk zullen de relatieve bestanden nog uitgebreid aan de orde komen.

Diskette-keuze

Er zijn diskettes in vele kwaliteiten en prijsklassen op de markt. Dat komt on-

der meer doordat de ene diskdrive veel meer informatie op een disk schrijft dan de andere; zo zijn er diskdrives die beide kanten van de diskette gebruiken. Dergelijke drives hebben zowel aan de bovenals aan de onderzijde een lees- en schrijfkop.
Bovendien kan de informatie-dichtheid behoorlijk uiteenlopen. De 1541 schrijft 35 sporen op een kant. Andere drives gebruiken meer sporen, die bovendien dichter beschreven worden, er staat meer informatie per centimeter spoor op.

Dat alles leidt ertoe, dat er inderdaad veel soorten diskettes nodig zijn. Al die verschillende types worden met een aantal afkortingen aangeduid, waarmee de fabrikant aangeeft, wat de eigenschappen zijn.

SS	Single Sided (eenzijdig)
DS	Double Sided (tweezijdig)
SD	Single Density (enkele dichtheid)
DD	Double Density (dubbele dichtheid)

Figuur 6/1.2-2: Overzicht diskette-aanduidingen

Voor de Commodore 1541 drive voldoen de goedkoopste SS/SD diskettes uitstekend, hoewel het natuurlijk nooit kwaad kan om een DD kwaliteit te gebruiken.

Sommige diskettes zijn voorzien van een verstevigingsring rond het middengat waar de drive aangrijpt (een hub-ring in het Engels). Gezien de krachten, die de diskdrive op de disk uitoefent, is dit een goed idee. Zonder die hub-ring gaat een disk veel sneller slippen, waardoor de informatie onleesbaar wordt.
Een andere kreet die soms op diskettes voorkomt is "soft-sectored". Dit slaat op de wijze waarop de indeling in tracks en sectors op de diskette wordt aangebracht tijdens het formatteren, sommige diskdrives maken daarvoor gebruik van een patroon van gaatjes in de disk zelf. Ook hierin is de 1541 een simpele kostganger, de drive brengt zelf de sectoren-indeling aan. Soft-sectored is goedkoper en daarom de beste keus, maar mochten er eens alleen maar hard-sectored diskettes verkrijgbaar zijn, ze werken net zo goed.

Algemeen gesproken geldt dat een diskette die zich zonder fouten laat formatteren goed zal werken. Bij dit formatteren worden namelijk alle blokken voorzien van de nodige informatie en daarop weer gecontroleerd. Wees alleen wat voorzichtig met de merkloze, "witte", diskettes. Dit kunnen afgekeurde partijen zijn, die produktiefouten vertonen. Vaak genoeg echter werken ook deze allergoedkoopste diskettes feilloos.

Veiligheid

Ook bij diskettes geldt wat er al voor cassettes gesteld is: vroeger of later kan het gebeuren dat een bepaalde disk niet meer te gebruiken blijkt. Daarom is het regelmatig maken van een "back-up" van het grootste belang. Het bij cassettes wel belangrijke tweemaal wegschrijven is echter minder noodzakelijk, bijna alle fouten die onder het wegschrijven kunnen ontstaan zullen meteen tot een foutmelding van de diskdrive leiden. Let dus altijd goed op het rode lampje, als dat knippert is er iets mis!

1.2 Diskdrive

Laat diskettes nooit op tafel rondslingeren. Ze zijn gevoelig voor stof en mechanische beschadigingen. Berg ze altijd meteen weer op in hun hoesje en zet ze weer in de diskettedoos.

Het etiketteren van diskettes is natuurlijk een goed idee, zo weet men tenminste wat erop staat. Het beschrijven van zo'n etiket is echter lastiger, want met een balpen op een al op de disk geplakt etiket schrijven kan zo'n diskette beschadigen. Gebruik hiervoor uitsluitend zeer zachte viltstiften of fineliners!

Wees helemaal voorzichtig voor dergelijke mechanische beschadigingen. Nooit met nietjes, paperclips o.i.d. iets aan een schijfje hechten.

Een andere bron van ellende is magnetisme, dat de informatie kan uitwissen. Wees vooral voorzichtig met de TV of de monitor, die stralen sterke magneetvelden uit. Houdt diskettes op minimaal 50 centimeter afstand ervan. Ook telefoons zijn gevaarlijk, als een telefoon overgaat komen ook daar sterke magneetvelden voor. Een diskette, die onder een telefoontoestel ligt, loopt niet alleen gevaar mechanisch beschadigd te worden, het risico dat een vriendelijke opbeller onwetend de informatie op die disk verminkt, is nog veel groter.

Het is maar al te gemakkelijk om per ongeluk een verkeerd bestand op een diskette te schrijven. Gelukkig kan een disk "write-protected", dus onbeschrijfbaar, gemaakt worden. Aan de rechterkant van de disk is een vierkant gaatje uitgespaard. Dat gaatje betekent dat een schijf beschreven mag worden. Afplakken met de bijgeleverde plakkertjes is een bescherming tegen per ongeluk beschrijven.

Dubbelzijdig gebruik

Veel diskdrive-bezitters hebben een methode gevonden om hun diskette-kosten te drukken. Zo'n disk heeft namelijk altijd aan beide kanten een magnetische laag, ook als de disk enkelzijdig verkocht wordt. Iedere diskette wordt dubbelzijdig geproduceerd, pas daarna, bij de testfase van de fabricage, worden de exemplaren met een slechte kant alsnog tot enkelzijdig bestempeld. Maar vaak genoeg worden ook diskettes, waarvan beide kanten goed zijn, als enkelzijdig gelabeld en verkocht, simpel omdat er meer vraag naar is.

De truc is nu om ook die tweede kant in gebruik te nemen, dat scheelt de helft in de disk-kosten. Op zich gaat het heel simpel. Door het vierkante gaatje dat zich aan de rechterkant van de schijf bevindt aan de linkerkant te kopiëren maakt men van een "floppy" een "flippy" (een dubbelzijdige diskette).

Toch valt die "besparing" af te raden. Een diskette heeft nu eenmaal een boven- en een onderkant en dus een bepaalde draai-richting. Daarop is de diskette ontworpen, de voering die stofjes en dergelijke van het gevoelige magnetische oppervlak verwijdert kan soms, als die draai-richting wordt omgedraaid, juist het in de hoeken van de disk-envelop verzamelde stof op het dis-oppervlak deponeren.

Maar al te vaak wordt deze verkeerde zuinigheid aangeraden door mede-hobbyïsten. De risico's zijn echter te groot, tenzij voor heel zelden gebruikte "archief"-diskettes.

1.2 Diskdrive

Figuur 6/1.2-3: Floppy-knipper

6/1.2.1

Basisbegrippen

Veel van wat er in hoofdstuk 6/1.1.1, basisbegrippen bij de cassetterecorder, vermeldt staat, gaat even goed op voor de diskdrive. Die informatie zal hier dan ook niet herhaald worden. Zeker bij BASIC-programma's en machinetaalprogramma's werken beide randapparaten, vanuit het standpunt van een BASIC-programmeur gezien, vrijwel hetzelfde.

Slechts bij de bestanden zij er echte verschillen tussen de cassetterecorder en de drive. Bij cassetterecorder-gebruik is er sprake van een cassettebuffer in het geheugen van de 64, waardoor het mogelijk wordt om ook kleine stukjes informatie op te slaan zonder dat dit onevenredig veel ruimte op de band inneemt.
Natuurlijk moeten er ook dergelijke buffers zijn, als er een diskdrive wordt gebruikt, maar deze bevinden zich niet in de 64 echter in de 1541 drive zelf. De 2K RAM in de diskdrive wordt onder meer gebruikt, om allerlei gegevens tijdelijk op te slaan.
Iedere lees- of schrijfopdracht op de diskette beslaat namelijk een heel blok van 256 bytes tegelijkertijd (254 bytes informatie en 2 pointer-bytes), terwijl zeker in het geval van een gegevens-bestand die bytes vaak slechts een voor een door een programma worden ingelezen of weggeschreven. Om de kloof tussen de eigentijdse disk lees- en schrijfoperaties in blokken en het gebruik van de enkele gegevens in BASIC op te vangen, kan de 1541 meerdere buffers tegelijkertijd in zijn eigen geheugen hebben staan, met een maximum van drie voor sequentiele bestanden.
Met andere woorden, er kunnen drie bestanden tegelijkertijd in gebruik zijn per 1541 diskdrive in plaats van het ene enkele bestand dat de cassetterecorder aankan.

Commando-kanaal

Zoals reeds gesteld: de 1541 is een intelligente diskdrive. Het apparaat kan zelfstandig allerlei taken uitvoeren. Veel van die zaken worden door BASIC geregeld; een LOAD-commando wordt automatisch uitgevoerd. Maar er zijn een aantal mogelijkheden die vereisen dat er rechtstreeks met de diskdrive gecommuniceerd kan worden; daarvoor dient het speciale commando-kanaal.
Dit wordt net als een gewoon bestand geopend, maar is echter niet aan het een of andere bestand op de diskette gekoppeld. Alle op dit bestand gePRINTe teksten worden door de floppydrive als rechtstreekse commando's geïnterpreteerd. De eventuele foutmeldingen van het Disk Operating system kunnen uit dit commando-kanaal gelezen worden.

6/1.2.2

Gebruik

Formatteren
Voordat een nieuwe diskette gebruikt kan worden om er informatie op op te slaan moet deze eerst voorzien worden van het magnetische patroon van sporen en sectoren. Dit zogenaamde formatteren gebeurt door een commando aan de drive te geven via het commando-kanaal.
Bij dit formatteren gaat alle mogelijk al op de disk aanwezige informatie verloren; wees er dus heel erg voorzichtig mee. Formatteer een schijf alleen dan als u werkelijk zeker bent dat er nog niets op staat, of als de al aanwezige informatie zeker niet meer nodig is!
Om een schijf te formatteren worden de volgende commando's gebruikt (later zullen deze diepgaand behandeld worden:

```
OPEN 15,8,15
PRINT#15,"NEW0:disknaam,id"
CLOSE 15
```

Na dit commando zal de diskdrive ongeveer honderd seconden actief zijn; het eigenlijke formatteren vindt dan plaats. Een doodenkele keer kan het voorkomen dat er tijdens het formatteren een fout optreedt; het rode lampje op de drive begint dan te knipperen. Probeer het dan gewoon nog een keer, meestal gaat het dan wel goed. Mocht de fout hardnekkig blijken, stel dan de diskette in kwestie buiten gebruik, naar alle waarschijnlijkheid is deze beschadigd of van slechte kwaliteit.

De disknaam en de disk-id mag u zelf kiezen, waarbij geldt dat de disknaam maximaal 16 tekens mag zijn en de disk-id twee tekens lang moet wezen. Niet alle tekens zijn hierbij toegestaan, sommige hebben een speciale betekenis. Letters en cijfers mogen altijd.

De disknaam is bedoeld voor de gebruiker, om onderscheid aan te kunnen brengen tussen de diverse diskettes.
De disk-id is voor de 1541 zelf van belang. Aan de hand van deze id controleert de diskdrive of u de disk verwisseld heeft. Bij een schrijfopdracht bijvoorbeeld controleert de drive eerst deze id; als deze dezelfde is als bij de vorige schrijfopdracht zal de 1541 de BAM-kopie in zijn eigen geheugen gebruiken om vast te stellen welke blokken beschikbaar zijn. Alleen als de huidige id op schijf afwijkt van de id die bij de vorige schrijf-actie gelezen was, zal de BAM inderdaad van schijf gelezen worden.

Het onmiddellijk na elkaar gebruiken van twee diskettes met dezelfde id waarbij die tweede schijf beschreven wordt leidt dan ook onherroepelijk tot grote problemen. De 1541 zal dan namelijk niet

"weten" dat de schijf verwisseld is en de oude BAM gebruiken om te bepalen welke blokken nog beschikbaar zijn. Naar alle waarschijnlijkheid zullen daarbij reeds voor andere bestanden in gebruik zijnde blokken worden overschreven, waarna die oorspronkelijke bestanden reddeloos verloren zijn.
Gebruik daarom altijd unieke id's, het tweemaal voorkomen van hetzelfde id in het diskette-archief leidt vroeger of later tot ongelukken.

De nul na het NEW commando is een overblijfsel uit vroeger tijden. De 1541 kan slechts een enkele diskette bevatten, sommige oudere Commodore drives omvatten een tweetal drives in een enkele kast. Zo'n dubbele drive bezit slechts een enkel apparaatnummer, om onderscheid te kunnen maken tussen de twee diskettes werden deze met 0 en 1 aangeduid.
Daar het 1541-DOS ontwikkeld is uit deze oudere Disk Operating Systems kan ook hier een 0 of een 1 worden opgegeven, hoewel een 1 natuurlijk aanleiding tot een foutmelding zal zijn. De nul mag desgewenst ook worden weggelaten, in ieder geval neemt de 1541 aan dat drive 0 (de enige mogelijkheid) bedoeld is.

Ook het commando zelf mag ingekort worden tot een enkele letter. Een enkele N wordt als het NEW-commando geïnterpreteerd, zodat

PRINT#15,"N:disknaam,id"

net zo goed werkt.

Schijfinhoud opvragen

De diskdrive houdt op iedere schijf bij welke bestanden er op aanwezig zijn. Deze informatie is natuurlijk ook voor de gebruiker erg belangrijk, vandaar dat de directory ook uit te lezen is. Dit gebeurt met een speciale vorm van het LOAD-commando:

LOAD"$",8

Let op, de aldus ingelezen directory overschrijft het BASIC-programma dat op het moment in het geheugen staat!
Om het zojuist geladen directory te kunnen bekijken moet het LIST commando gebruikt worden, net als bij een BASIC-programma. Figuur 6/1.2.2-1 toont een voorbeeld van een directory.

```
0 "1541 TEST/DEMO      " ZX 2A
13    "HOW TO USE"          PRG
5     "HOW PART TWO"        PRG
4     "VIC-20 WEDGE"        PRG
1     "C-64 WEDGE"          PRG
4     "DOS 5.1"             PRG
11    "COPY/ALL"            PRG
9     "PRINTER TEST"        PRG
4     "DISK ADDR CHANGE"    PRG
4     "DIR"                 PRG
6     "VIEW BAM"            PRG
4     "CHECK DISK"          PRG
14    "DISPLAY T&S"         PRG
9     "PERFORMANCE TEST"    PRG
5     "SEQUENTIAL FILE"     PRG
13    "RANDOM FIAL"         PRG
558 BLOCKS FREE.

READY.
```

Figuur 6/1.2.2-1: Een disk-directory

Overigens is dit de directory van de test/demo schijf die bij de diskdrive wordt meegeleverd. Behalve de namen van de programma's en bestanden op

de schijf staat er nog meer informatie op.

De inverse balk bevat de disknaam (1541TEST/DEMO), de disk-id (ZX) en het versienummer van het disk-operating system waaronder deze schijf is aangemaakt, 2A. Andere Commodore diskdrives, die daar een andere code neerzetten, zijn soms niet helemaal uitwisselbaar met de 1541, maar ze komen slechts zelden voor.
Onder deze kopbalk staan alle bestanden op de schijf vermeldt, met voor de naam het aantal blokken dat ze in beslag nemen en na de naam het type van het bestand. Op deze schijf staan alleen maar PRG-files, programma-bestanden in BASIC of machinetaal.
Tenslotte vermeldt de directory hoeveel blokken er nog vrij zijn op de schijf.

Laden, saven en verifieren van BASIC-programma's

Dezelfde drie commando's als we al bij de cassetterecorder hebben leren kennen, hoofdstuk 6/1.1.2, werken ook voor de diskdrive.
Om een BASIC-programma op te slaan wordt het volgende commando gebruikt:

SAVE"programmanaam",8

waarbij de programmanaam maximaal 16 tekens lang mag zijn en moet worden opgegeven. Alweer, niet alle tekens zijn toegestaan; gebruik letters, cijfers en sommige leestekens zoals de punt en het streepje.
Het verschil met het SAVE-commando voor de recorder is alleen het apparaatnummer. Voor alle diskdrive-operaties is dit standaard 8, voor de recorder is het een 1. Weglaten van dit nummer heeft tot gevolg dat de 64 aanneemt dat de recorder bedoeld wordt.

Waar bij cassette-operaties een heel vraag- en antwoordspel op het scherm verschijnt, daar is dat bij de diskdrive niet nodig. Alle commando's worden automatisch afgehandeld, pas als de READY weer in beeld verschijnt moeten we zelf weer actie ondernemen. Nogmaals, let altijd goed op of het rode drivelampje niet knippert en op die manier een fout aangeeft.
Dit zal bijvoorbeeld het geval zijn als de programmanaam al op de schijf in gebruik is, de 1541 overschrijft programma's en bestanden niet automatisch. Om zo'n programma dan alsnog te kunnen overschrijven is er een speciale vorm van het SAVE-commando, dat de 1541 opdraagt het programma altijd weg te schrijven, zelfs al zou hierbij een al bestaande programmanaam worden overschreven:

SAVE"@:programmanaam",8

Deze vorm van SAVEn heeft een eigen naam, SAVE & REPLACE. Het is echter niet volledig betrouwbaar, het kan een heel enkele keer gebeuren dat na dit commando de inhoud van de diskette verminkt blijkt te zijn. Het gebruik ervan is dan ook af te raden.
Er zijn andere, veiliger manieren om een programma te overschrijven, die later in dit hoofdstuk nog aan de orde zullen komen.

Ook bij de 1541 is er een mogelijkheid om een zojuist opgeslagen programma te vergelijken met het exemplaar wat nog steeds in het 64-geheugen staat, met het VERIFY-commando. Dat gaat als volgt:

VERIFY"programmanaam",8

De diskdrive zal dan eerst het te controleren programma opzoeken. Dit wordt gemeld met de tekst 'SEARCHING FOR PROGRAMMANAAM'; als het programma gelocaliseerd is verschijnt: 'VERIFYING'. De tekst '?FILE NOT FOUND ERROR' betekent dat het gezochte programma zich niet op de disk bevindt, meestal valt dit aan een typfout te wijten. Als er geen fouten gevonden worden, meldt de computer zich terug met 'OK', anders verschijnt de melding 'VERIFY ERROR'. Die kans is echter bijzonder klein, in de praktijk is het niet nodig om geSAVEde programma's te VERIFYen.

Laden gaat met:

LOAD"programmanaam",8

Zowel de naam als het apparaatnummer (8) moeten te allen tijde worden opgegeven; ook dit keer is er verder geen actie van de gebruiker nodig. Op het scherm verschijnt de melding 'SEARCHING FOR PROGRAMMANAAM', gevolgd door 'LOADING' of '?FILE NOT FOUND ERROR'.
In dat laatste geval is er bijna altijd sprake van of een spelfout in de naam of er zit een verkeerde diskette in de drive.
Tijdens het laden van een programma kan de diskdrive allerlei fouten constateren. Let dan ook, zoals altijd, op het eventuele knipperen van het rode lampje.

Gedeeltelijke namen

Bij zowel het laden als het verifieren hoeft de programmanaam niet volledig ingetikt te worden. Een tweetal tekens, de '?' en het '*' hebben in een programma- of bestandsnaam namelijk een speciale betekenis.
Dit valt het beste uit te leggen aan de hand van een paar voorbeelden. Stel dat we van de 1541TEST/DEMO diskette, waarvan de directory eerder in dit hoofdstuk is afgedrukt, het programma PRINTER TEST willen laden, dan kan dat natuurlijk met:

LOAD"PRINTER TEST",8

Maar met:

LOAD"PRINT*",8

gaat het ook!

Minder typewerk, dus minder kans op fouten. Hoe het werkt is als volgt: de "*" is een van de beide speciale tekens, die gebruikt mogen worden bij het opgeven van filenamen aan de 1541 en betekent zoveel als: 'vanaf hier maakt het niet meer uit wat er staat'.
Als het gedeelte voor de '*' maar uniek is, dan zal zo'n programmanaam met speciale tekens uitstekend werken. Ook het commando:

LOAD"PR*",8

zal PRINTER TEST laden.

Als een naam door het gebruik van de '*' niet meer uniek is, dan neemt de 1541 het eerste bestand in de directory waarvan de naam overeenkomt met de opgegeven combinatie. Met andere woorden:

LOAD"P*",8

laadt nog altijd PRINTER TEST. Hoewel PERFORMANCE TEST ook vol-

1.2 Diskdrive

doet aan de specificatie staat deze naam pas later in de directory. Om PERFORMANCE TEST te laden moet minimaal:

LOAD"PE*",8

gebruikt worden.

Het tweede speciale teken heeft een ander gebruik, het '?' staat voor slechts een enkel teken in de naam. Maar op die positie kan het dan wel voor elk willekeurig teken staan, het vraagteken is de joker! Zo zal:

LOAD"??R*",8

zoeken naar een programma waarvan alleen vaststaat dat de derde letter in de naam een R is. Er komen er twee in aanmerking, namelijk DIR en PERFORMANCE TEST, waarvan DIR, die als eerste van die twee in de directory staat, gekozen en geladen zal worden.

Om het eerste programma op een diskette te laden – bij commerciële software vaak een menu of ander startpunt – kunnen we dan ook in theorie volstaan met:

LOAD"*",8

In de praktijk blijkt dit echter toch soms niet te werken. De verklaring is simpel, de 1541 zal een enkele '*' eerst interpreteren als de naam van het laatst geladen programma en pas als dat niet te vinden is inderdaad het eerste programma op de schijf laden.

Bestanden

Bestands-afhandeling op diskette lijkt weliswaar veel op de manier waarop er met cassette-bestanden wordt omgegaan, maar er zijn toch duidelijke verschillen. Dat komt onder andere door de flexibeler mogelijkheden van de 1541.
Zo zijn er meerdere types bestanden te onderscheiden, ieder met hun eigen specifieke mogelijkheden.

Type	Omschrijving
PRG	BASIC- of machinetaal-programma;
SEQ	Sequentieel databestand;
USR	Zelf te definiëren type, in principe gelijk aan SEQ;
REL	Relatief bestand, wordt pas later behandeld.

Figuur 6/1.2.2-2, overzicht bestandstypes

Van deze vier bestands-types worden het PRoGram- en het SEQuential-file verreweg het meeste gebruikt, de behandeling van de beide andere types wordt tot later in dit hoofdstuk uitgesteld.

Het PRG-bestand is, net als bij de cassette, de inhoud van een stuk 64-geheugen. Vanaf een begin-adres wordt byte voor byte opgeslagen, tot het einde van het weg te schrijven blok bereikt is. In het geval van een BASIC-programma staat dat begin-adres vast en is het eind-adres afhankelijk van de lengte van het programma; bij een ML-programma is dat begin-adres wisselend, dat kan voor ieder ML-programma weer anders liggen.
Zo'n PRG-bestand kan overigens ook iets heel anders bevatten dan een BASIC- of een ML-programma. Het kan een willekeurig blok geheugen zijn met bijvoorbeeld sprite-definities.

Het SEQ-bestand, het sequentiële be-

1.2 Diskdrive

stand, wordt vanuit een programma gelezen en beschreven.
Om zo'n bestand te kunnen gebruiken moet het eerst geOPENd worden, met het volgende commando:

OPEN filenummer,apparaatnummer, kanaalnummer,"0:FILENAAM,type, richting"

Het *filenummer* kan vrij gekozen worden tussen de 1 en de 127, het *apparaatnummer* is standaard 8 voor de diskdrive. Het *kanaalnummer* mag tussen de 2 en de 14 liggen (de nummers 0, 1 en 15 hebben een speciale betekenis) en geeft aan welke diskbuffer er gebruikt moet worden. Het is natuurlijk niet mogelijk om tegelijkerijd twee bestanden met hetzelfde kanaalnummer geOPENd te hebben; twee bestanden kunnen ook op de 1541 niet dezelfde buffer delen.

De FILENAAM moet opgegeven worden en geeft de naam van het bestand aan, maximaal 16 tekens waarbij natuurlijk de '*' en het '?' (de speciale tekens) gebruikt mogen worden. Alleen de '0:' voor de filenaam geeft in principe aan welke disk er gebruikt moet worden. Daar de 1541 een enkele drive bevat mag de 0: worden weggelaten.

Met *'type'* wordt opgegeven welk bestands-type we bedoelen, zie figuur 6/1.2.2-2, waarbij zowel de volledige naam als slechts de eerste letter gebruikt mogen worden; het disk operating system kijkt alleen maar naar die eerste letter.

De *richting* tenslotte geeft aan of het bestand geopend moet worden om te schrijven (write) of om te lezen (read). De 1541 kijkt alleen naar de eerste letter. Om een bestand te gaan lezen mag hier dus zowel 'read' als 'r' (of eventueel 'rood') worden gelezen.

Het is, althans bij niet-relatieve files, niet mogelijk om tegelijkertijd te lezen en te schrijven op een en hetzelfde bestand. Als een bestand binnen een programma eerst gelezen en daarna beschreven moet worden is het nodig om dat bestand na het lezen eerst af te sluiten en dan weer als schrijf-bestand te openen.

Na de OPEN-opdracht zal de 1541 zelf het betreffende bestand proberen te openen, waarbij het natuurlijk tot een fout leidt als een lees-bestand niet op de schijf bekend is. Deze wordt echter niet op het scherm gemeld, slechts het knipperen van het rode lampje geeft aan, dat er iets fout ging.
Het OPENen van een al bestaande schrijfbestand gaat ook fout, zo'n bestand wordt dan niet automatisch overschreven, tenzij we met de 'apestaart' in het OPEN-commando aangeven dat dit wel de bedoeling is. Een voorbeeld:

OPEN1,8,15,"TEST,S,W"

opent een sequentieel bestand met de naam test om te beschrijven, maar alleen als de naam test nog niet op de diskette bekend was.

OPEN1,8,15,"@0:TEST,S,W"

zal het bestand test, als dat al bekend was, eerst weggooien en het daarna weer openen.
Pas op, deze laatste vorm, het 'OPEN en REPLACE' commando, is niet volledig

1.2 Diskdrive

betrouwbaar. Het gebeurt niet vaak, maar het is mogelijk dat na zo'n 'apestaart'-commando de diskette beschadigd blijkt te zijn of dat de structuur van directory, blokken en BAM niet meer klopt.

De enige controle die de 64 kan uitoefenen op dit hele gebeuren van OPENen is of het is het filenummer niet al in gebruik is. Mocht dit het geval zijn, dan verschijnt de foutmelding 'FILE OPEN'. Alle andere fouten worden door de diskdrive opgemerkt en leiden niet tot foutmeldingen in BASIC.

Het schrijven naar een eenmaal geopende file gebeurt met een speciale vorm van het PRINT-commando:

PRINT#filenummer,variabele of constante

zal de genoemde waarden in het met filenummer aangeduide bestand plaatsen. Alle in BASIC 2.0 PRINT statements toegestane constructies mogen gebruikt worden. Zowel constanten, zoals "tekst" als 12345.12 als variabelen, zoals A, A$ of A% zijn toegestaan. Meerdere constanten of variabelen kunnen ook tegelijkertijd met een PRINT@ commando worden weggeschreven, waarbij het echter wel van belang is om de diverse weggeschreven waardes goed van elkaar te scheiden, zowel in het commando als in het bestand.

Bij het teruglezen van de informatie van een bestand wordt het

INPUT#filenummer,variabelelijst

commando gebruikt, wat sterk op het overeenkomstige INPUT commando lijkt. De diverse in te lezen waardes dienen op een juiste manier van elkaar gescheiden te zijn. Bovendien is het ook bij bestanden onmogelijk om bijvoorbeeld een tekst in een numerieke variabele te lezen.

Bij een rechtstreekse INPUT vanaf het toetsenbord kent de 64 bij zulke fouten allerlei beveiligingen en waarschuwingen. Bij bestanden moet de programmeur het zelf regelen.
De eenvoudigste manier om deze problemen te omzeilen is alle weer apart in te lezen stukjes informatie met een ‹RETURN›, oftewel een CHR$(13) van elkaar te scheiden. Dat kan bijvoorbeeld door alle variabelen of constanten met een eigen PRINT# opdracht weg te schrijven.

Als er wel meerdere waardes met een PRINT-commando worden weggeschreven, dan moet er goed op de scheidingstekens in dat commando gelet worden. De puntkomma laat immers de geprinte informatie keurig aansluiten, terwijl de komma het nog bonter maakt en de geprinte gegevens in kolommen met 10 plaatsen tussenruimte verdeelt.
PRINT@1,"tekst","meer tekst" geeft ook op diskette:

tekst meer tekst

Niet alleen ontbreekt de scheider die voor het apart teruglezen nodig is; er zijn zelfs overbodige spaties in het bestand gezet. Om een bestand weer goed terug te kunnen lezen moet ieder stukje informatie van alle andere stukjes gescheiden worden. In de voorbeelden, 6/1.2.3, wordt getoond hoe dit het beste kan gebeuren.

1.2 Diskdrive

Een tweede probleem bij het weer inlezen van de gegevens is hoe te bepalen wanneer het bestand leeg is. Het is niet mogelijk om meer gegevens uit een bestand te lezen dan erin staan, maar het is aan de programmeur om ervoor te zorgen dat het inlezen inderdaad op tijd stopt.
Daarom wordt na ieder I/O operatie de status van die operatie in de systeemvariabele ST gezet, die alleen maar gelezen kan worden. ST wordt voor vele doeleinden gebruikt, niet alleen voor disk-I/O.

Algemeen gesproken geldt dat als na een lees- of schrijfopdracht de ST gelijk aan 0 is, alles goed gegaan is. Om te ontdekken of onder het lezen het einde van een bestand bereikt is kan het beste de tekst:

IF ST AND 64 THEN GOTO einde bestand afhandeling

gebruikt worden. Op de andere mogelijke waarden die ST aan kan nemen wordt later teruggekomen.
Overigens kan de bovenvermelde programmaregel aanleiding geven tot zeer frusterende foutmeldingen. Als de spaties weggelaten zouden worden, wat vaak gedaan wordt, gaat het er als volgt uitzien:
IFSTAND64THENGOTOregelnr, hetgeen op een zeer hardnekkige 'SYNTAX ERROR' uitdraait. De vierde, vijfde en zesde letters uit deze regel vormen namelijk het woord TAN en dat is weer een legale BASIC-functie, die echter in deze regel inderdaad een taalfout voor BASIC is.

Die 'einde bestand' routine is heel simpel; het bestand moet weer gesloten worden met de CLOSE opdracht:

CLOSE filenummer

Na het CLOSE commando is het betreffende filenummer weer beschikbaar om eventueel opnieuw te gebruiken met een ander bestand. Bovendien wordt een bestand dat beschreven was door de CLOSE netjes afgesloten, de resterende gegevens in de buffer worden alsnog op de schijf gezet. Zonder de CLOSE zou dit niet gebeuren en blijkt later bij inlezen dat de laatste gegevens nooit weggeschreven zijn.

Dit afsluiten kan soms problemen opleveren bij disk-I/O. Het feit dat een bestand in BASIC is afgesloten hoeft niet automatisch te betekenen dat dit ook op de schijf gebeurd is.
Als er bijvoorbeeld een fout optreedt in het programma, zodat BASIC met een foutmelding terugkeert in de directmode, worden in de 64 weliswaar alle bestanden automatisch afgesloten, maar niet in de diskdrive. Deze situatie is ronduit gevaarlijk, als er daarna opnieuw met de drive gewerkt wordt zou de diskette gedesorganiseerd kunnen raken!
In zo'n geval moet direkt het commando:

OPEN15,8,15,"I"

worden uitgevoerd. De precieze werking van dit disk-commando komt straks aan bod.

Het commando INPUT@1,A$ heeft in BASIC 2.0 een beperking; de aldus ingelezen string mag niet langer dan 80 tekens zijn. De maximale lengte van een stringvariabele is echter veel groter. Zo'n lange string is zonder problemen te prin-

ten, ook naar een bestand. Er kunnen dus langere strings in een bestand staan dan er met INPUT# kunnen worden ingelezen. Die lange strings kunnen echter wel gelezen worden met het GET# commando:

GET# filenummer, stringvariabelelijst

Voor iedere variabelenaam uit de lijst – meestal wordt er slechts één gebruikt – leest de GET# een enkel teken van disk. Alle tekens worden gelezen, ook de scheiders als CHR$(13) en de komma. Door nu met GET# de gegevens in te lezen en ze daarna weer tot langere strings samen te voegen kunnen ook de lange strings weer ingelezen worden.

Een waarschuwing: een teken met code 0 – CHR$(0) – wordt niet als zodanig ingelezen, maar als een lege string. GET#1,A$:print ASC(A$) kan de foutmelding 'ILLEGAL QUANTITY ERROR' opleveren. Er zijn meerdere mogelijkheden om dit te omzeilen, zie verder 6/1.2.3, voorbeelden.

Het filenummer, na PRINT# en INPUT# moet al bekend zijn uit een eerder OPEN commando, anders volgt de foutmelding:
'?FILE NOT OPEN'.

Een laatste commando dat bij bestanden gebruikt kan worden is:

CMD filenummer,tekst

Dit heeft tot gevolg dat alle uitvoer die normaal naar het beeldscherm gezonden wordt, nu naar het met filenummer aangegeven bestand gaat. Eventueel achter het filenummer opgegeven tekst wordt als eerst regel naar dat file gezonden. Met CMD kan bijvoorbeeld de listing van een programma in een bestand gezet worden, het zelfde formaat als die listing ook het beeldscherm verschijnt. Zo'n bestand zou dan later weer gebruikt kunnen worden door een ander programma. Na de laatste uitvoer die via CMD naar een bestand gestuurd is moet er altijd nog een lege regel geprint worden, om het bestand goed af te sluiten.

Disk-scommando's

Het is moelijk om de 1541 diskdrive ook rechtstreekse commando's te geven, een voorbeeld daarvan hebben we al gezien bij de noodmaatregel na een BASIC-fout:

OPEN15,8,15,"I"

Hier wordt een van de speciale kanalen geopend, nummer 15, wat het commando-kanaal van de diskdrive is. Via kanaal 15 kan nooit een bestand beschreven of gelezen worden, het kan alleen maar gebruikt worden om rechtstreekse opdrachten aan het Disk Operating System – kortweg DOS – te geven of om meldingen van de DOS terug te lezen. Het voorbeeld geeft een 'I' commando, wat de 1541 instrueert om zichzelf opnieuw te initialiseren. Na dit commando zijn alle eventueel nog openstaande bestanden afgesloten en is de BAM weer van de schijf gelezen.

Er zijn veel meer disk-commando's beschikbaar, maar een aantal daarvan zijn dermate gespecialiseerd dat ze pas later aan de orde zullen komen. Voor het moment zijn van belang:

NEW:disknaam,disk-id; waarmee een

diskette geformatteerd wordt. Als de disk-id wordt weggelaten zal de diskette niet geheel geformatteerd worden, maar worden alleen de directory, de BAM en de inhoud van de blokken gewist, wat veel sneller gaat dan een volledige formattering. Deze vorm kan gebruikt worden om een al geformatteerde disk geheel te wissen.

SCRATCH:filenaam; waarmee een bestand (of meerdere bestanden, als de speciale '*' en de '?' tekens in die filenaam gebruikt worden) wordt gewist. Na dit commando zullen de betreffende bestanden geheel verdwenen zijn, ook in de BAM worden de door die bestanden bezette blokken weer vrijgegeven.

INITIALIZE; waarmee de drive in de start-conditie wordt teruggezet, met dit commando zijn allerlei fout-condities op te heffen.

VALIDATE; om de interne organisatie van een diskette te herstellen. Het kan namelijk in het gebruik voorkomen dat door het herhaald SAVEn en SCRATCHen van bestanden er kleine groepjes lege blokken ontstaan, zo klein dat de drive ze niet meer gebruikt. Het VALIDATE-commando reorganiseert de diskette zodanig dat die lege stukjes weer gebruikt worden.
Een tweede belangrijke toepassing van dit codmmando is om niet geod afgesloten bestanden weer te verwijderen. Als er een fout wordt gemaakt tijdens het gebruik kan het voorkomen dat een bestand nooit echt afgesloten is, dit wordt op de directory-LIST getoond door een sterretje achter het type te zetten. In zo'n geval bestaat er een afwijking tussen de eigenlijke blok-indeling en de BAM van zo'n diskette, een gevaarlijke toestand. Als er dergelijke niet afgesloten bestanden op een disk voorkomen moeten deze nooit domweg geSCRATCHed worden maar moet die diskette worden geVALIDATE. Hierna zijn deze niet afgesloten bestanden uit de directory verwijderd en is de diskette weer veilig te gebruiken.

RENAME:nieuwnaam=oudenaam;
hiermee kan een bestand worden omgedoopt.

Al deze commando's mogen worden afgekort tot hun eerste letter.
De verdere disk-commando's komen later aan de orde.

Fout-uitlezing

Kanaal 51 kan ook worden gebruikt om meldingen van 1541 weer uit te lezen. Na iedere actie staat er in de diskdrive een melding klaar die aangeeft of er iets misgegaan is, en zo ja wat dan wel precies. In direct mode kan deze melding niet worden gelezen, maar in een BASIC-programma wel.
Deze disk-melding bestaat uit een viertal onderdelen, een numeriek foutnummer, een alfanumerieke foutboodschap en een tweetal numerieke waardes die de track- en sectornummers bevatten waar deze fout optrad. Door na iedere handeling dit kanaal 15 uit te lezen en het programma te laten controleren of het foutnummer ongelijk aan 0 is – een 0 geeft aan dat alles in orde is – kan een programmeur de status van de drive in de gaten houden. In de voorbeelden – 6/1.2.3 – wordt getoond hoe dit moet gebeuren.

6/1.2.3

Voorbeelden

Bij het programmeren zeggen wat goedgekozen voorbeelden meer dan duizend woorden, zeker als het ingewikkelde zaken als het programmeren van bestanden op de 1541 betreft.
Eerst een goede raad: gebruik een kladschijf bij het werken met deze voorbeelden en ook bij alle verdere experimenten op disk-gebied. Een foutje is zo gemaakt en kan, in het ergste geval, de hele inhoud van een diskette onbruikbaar maken.

Het eerste voorbeeld is een diskette-versie van het programmaatje van het allereerste cassette-voorbeeld.

```
2 REM NUMERIEKE VARIABELE IN
  BESTAND SCHRIJVEN
10 OPEN1,8,2,"TEST1,SEQUENTIAL,
   WRITE"
20 A=12345
30 PRINT#1,A
40 CLOSE1
```

Regel 10 is de regel waar het allemaal om draait, de rest is gesneden koek. In die regel 10 wordt een bestand met de naam TEST1 geopend, op de drive geopend, als bestand nummer 1, via kanaal 2. Het deel tussen aanhalingstekens geeft, behalve de naam, ook het type en de richting aan.
Na dat openen kan in regel 30 het bestandje beschreven worden, waarna het in regel 40 keurig wordt afgesloten.

Teruglezen gaat net zo makkelijk:

```
2 REM NUMERIEKE VARIABELE UIT
  BESTAND LEZEN
10 OPEN1,8,2,"TEST1,SEQUENTIAL,
   READ"
20 INPUT#1,A
30 PRINTA
40 CLOSE1
```

In regel 10 wordt TEST1 weer geopend, dit keer om te lezen, door "READ" als richting op te geven. Dan kan in regel 20 de waarde die in TEST1 staat worden uitgelezen. Na het afdrukken daarvan kun je het geheel weer netjes afsluiten met ;CLOSE1.

Bij deze twee voorbeelden is geen enkele fout-controle gebruikt, iets dat in het algemeen heel onverstandig is. De samenwerking tussen de 64 en de 1541 is namelijk ronduit slecht. Ook al ontdekt de diskdrive een fout dan nog zal de Commodore onverdroten doorgaan. Alle fout-detectie en fout-opvang moet door het programma worden afgehandeld, de BASIC vertolker trekt zich niets van disk-fouten aan.
Probeer maar eens wat er gebeurt als het eerste voorbeeld twee keer na elkaar wordt geRUNd. De tweede keer zal het

rode lampje op de drive alarm slaan – TEST1 bestaat immers al zodat het OPEN-commando tot een disk-fout leidt. Maar de 64 gaat rustig verder en meldt met misplaatste trots READY. Gelukkig is die fout-opvang niet al te ingewikkeld, als er tenminste wordt volstaan met de melding van de fout en er geen pogingen tot fout-opvang worden gedaan, zoals het volgende voorbeeld toont:

```
2 REM NUMERIEKE VARIABELE IN
  BESTAND SCHRIJVEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST1,S,W":GOSUB
   500
20 A=12345
30 PRINT#1,A
40 CLOSE 1
50 CLOSE 15
60 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Dit is een versie van het eerste programma, maar nu met fout-signalering. Bovendien is hier de meer gebruikelijke verkorte vorm van bestands-type en richting gebruikt.

Voor de fout-routine moet allereerst kanaal 15, het commando-kanaal, geopend worden in regel 5. Bestandsnaam, -type en richting ontbreken hierbij, immers, kanaal 15 is altijd het commando-kanaal. In regel 10 wordt dan, na het openen van het eigenlijke bestand, een sprong gemaakt naar de subroutine vanaf regel 500.

Dit is een standaard fout-detectie routine, die de meldingen uit het commando-kanaal leest.
Als het foutnummer gelijk aan 0 is, wat het geval zal zijn als alles goed gedaan is, wordt de subroutine weer verlaten. Als er wel een fout is opgetreden worden foutnummer, foutmelding en de track- en sector-nummers waar deze fout optrad afgedrukt samen met een veiligheids-waarschuwing, waarna het programmaatje afbreekt.
Die extra boodschap moet serieus genomen worden, wanneer een programma afbreekt worden de bestanden slechts wat BASIC betreft afgesloten. Op de diskdrive blijven ze geopend, hetgeen tot allerlei rampen kan leiden zodra de diskdrive weer aangesproken wordt. Tik in zo'n geval dadelijk OPEN15,8,15,"I" in, gevolgd door CLOSE15. Door op deze manier de drive opnieuw te initialiseren worden alle problemen vermeden.

Dat initialiseren kan natuurlijk ook in de fout-detectie routine zelf worden opgenomen. Dat brengt echter als nadeel met zich mee dat het opsporen van de oorzaak van de fout daarna vaak lastiger wordt.

Tot slot van het voorbeeldje moeten de beide bestanden, 1 en 15, weer gesloten worden. Ook daar is een volgorde voor: sluit kanaal 15 nooit af voor alle andere kanalen geCLOSEd zijn. Het sluiten van het commando-kanaal sluit namelijk automatisch alle andere bestanden.

Een beveiligde versie van het tweede voorbeeld, het lees-programmaatje, ziet er als volgt uit:

1.2 Diskdrive

```
2 REM NUMERIEKE VARIABELE UIT
  BESTAND LEZEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST1,S,R":GOSUB
   500
20 INPUT#1,A
30 PRINTA
40 CLOSE 1
50 CLOSE 15
60 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Verdere uitleg hierbij is overbodig.
De nu getoonde fout-controle routine zal verder bij alle voorbeelden gebruikt worden.

Er kunnen natuurlijk ook meerdere en verschillende soorten variabelen in bestanden worden geschreven.

```
2 REM DIVERSE VARIABELEN IN
  BESTAND SCHRIJVEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST2,S,W":GOSUB
   500
20 A$="DISKETTE-TEKST":A=12345:
   B$="ALWEER ALFANUMERIEK":
   B=54321
30 PRINT#1,A$
32 PRINT#1,A
34 PRINT#1,B$
36 PRINT#1,B
40 CLOSE 1
50 CLOSE 15
60 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Bij het teruglezen is het dan echter wel zaak om er voor te zorgen dat iedere waarde in een variabele van het juiste type terechtkomt.

```
2 REM DIVERSE VARIABELEN UIT
  BESTAND LEZEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST2,S,R":GOSUB
   500
20 INPUT#1,X$,X,Y$,Y
30 PRINTX$
32 PRINTX
34 PRINTY$
36 PRINTY
40 CLOSE 1
50 CLOSE 15
60 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Het inlezen van een tekst in een numerieke variabele gaat gegarandeerd fout, zoiets komt op een "?FILE DATA ERROR IN regelnr" te staan.
De andere kant op kan wel, als er een getal in een tekst-variabele wordt ingelezen gaat dat op zich prima. Eventueel kan zo'n getal in een tekst-variabele dan weer in een echte numerieke variabele geplaatst worden met de VAL-functie. Dit kan handig zijn als niet bekend is hoe een bestand precies in elkaar steekt, maar het is wel even oppassen geblazen.

Tot nog toe is iedere waarde of tekst met een eigen PRINT opdracht in een bestand gezet. Dat hoeft natuurlijk niet, anders zou het wegschrijven van grote hoeveelheden informatie een onmogelijk

karwei worden. Het volgende voorbeeld laat een andere benadering zien.

```
2 REM AANTAL VARIABELEN IN
  BESTAND SCHRIJVEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST3,S,W":GOSUB
   500
20 FOR N=1 TO 5
30 READ A$
40 PRINT#1,A$
50 NEXT N
60 CLOSE 1
70 CLOSE 15
80 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
1000 DATA ITEM NUMMER 1
1010 DATA ITEM NUMMER 2
1020 DATA ITEM NUMMER 3
1030 DATA ITEM NUMMER 4
1040 DATA ITEM NUMMER 5
```

Hier worden een vijftal teksten vanuit DATA-regels gelezen en daarna op een bestand gezet, waarbij slechts een enkele PRINT-opdracht gebruikt wordt. Het is daarbij echter wel zaak om de juiste vorm van PRINT te gebruiken.

Zo zou hetzelfde programma met in plaats van de huidige regel 40:

40 PRINT#1,A$;

moeilijkheden opleveren. In dat geval zouden namelijk alle teksten net zoals op het scherm achter elkaar, zonder scheiders, op het bestand verschijnen. Een lees-programma zou geen onderscheid kunnen maken tussen de vijf losse stukken en ze als een enkele tekst teruglezen.

40 PRINT#1,A$,

zou het nog bonter maken, want dan worden alle teksten op het bestand van elkaar gescheiden door een aantal spaties, net als op het scherm. Er zou nog steeds geen goede scheiding bestaan, terwijl het geheel aleen maar meer schijfruimte inneemt.

Door een PRINT-opdracht zonder verdere franje te gebruiken wordt er op het scherm een regel-einde gegenereerd, in feite een ASCII teken met code 13. Zo'n CHR$(13) verschijnt ook op een bestand en wordt bij het teruglezen gebruikt als teken dat de tekst die door een enkel INPUT commando gelezen moet worden ten einde is.

Als dit TEST3 bestand wordt teruggelezen is dat ook te zien. Het volgende voorbeeld zou dit moeten doen met gebruik van slechts een INPUT-commando.

```
2 REM AANTAL VARIABELEN UIT
  BESTAND LEZEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST3,S,R":GOSUB
   500
20 INPUT#1,X$
30 PRINTX$
40 GOTO 20
50 CLOSE 1
60 CLOSE 15
70 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

1.2 Diskdrive

In eerste instantie gaat het goed, maar na de vijfde regel "ziet" dit programma blijkbaar niet dat het bestand uitgeput is. De laatst gelezen tekst wordt steeds weer herhaald. Het gaat er dus om hoe dit bestands-einde tijdig ontdekt kan worden.
Daar bestaat een speciale systeem-variabele voor, ST. Na iedere I/O operatie wordt de I/O status in deze variabele gezet. ST kan vele zaken aangeven, waarbij de waarde 64 is een vlag voor het bestands-einde. Als het bestand wat zonet gelezen werd uitgeput is, dan zal er 64 bij de waarde van ST worden opgeteld.
Het volgende lees-programma maakt daar gebruik van, het leest TEST3 wel goed uit.

```
2 REM AANTAL VARIABELEN UIT
  BESTAND LEZEN, MET EINDE
  BESTAND TEST
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST3,S,R":GOSUB
   500
20 INPUT#1,X$
30 PRINTX$
40 IF NOT ST AND 64 THEN GOTO 20
50 CLOSE 1
60 CLOSE 15
70 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"

READY.
```

Alles draait om regel 40, waar wordt vastgesteld of er nog verder gelezen moet worden. Omdat ST nog andere condities kan aangeven en dus meer waardes dan alleen de 64 van bestands-einde kan bevatten, moet er een AND constructie worden gebruikt om deze andere waardes uit te filteren. De NOT draait dan het resultaat om, zodat er uiteindelijk staat:
ALS NIET BESTANDS-EINDE DAN GA NAAR LEZEN.
De truc uit regel 40 dient in ieder leesprogramma te worden toegepast om te voorkomen dat een bestand te ver gelezen wordt.

Een ander probleem bij het werken met bestanden wordt met het volgende voorbeeld gedemonstreerd.

```
2 REM EEN LANGE STRING IN
  BESTAND SCHRIJVEN
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST4,S,W":GOSUB
   500
20 FOR N=1 TO 5
30 READ A$
40 PRINT#1,A$
50 B$=B$+A$
60 NEXT N
70 PRINT#1,B$
80 CLOSE 1
90 CLOSE 15
100 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
1000 DATA EEN EERSTE REGEL
1010 DATA EN EEN TWEEDE
1020 DATA DE DERDE WORDT WEER
     WAT LANGER
1030 DATA DE VIERDE REGEL IS
     WEER KORTE
1040 DATA DE VIJFDE REGEL IS
     DE LAATSTE

READY.
```

Dit programmaatje schrijft een zestal

teksten in bestand TEST4, namelijk de vijf die in de DATA-regels staan en tot slot nog een zesde, B$, die wordt gemaakt door in regel 50 de andere vijf teksten achter elkaar te plakken.

Om eens te proberen om deze teksten terug te lezen gebruiken we het volgende programma, dat overigens volkomen gelijk is aan het voorlaatste voorbeeld.

```
2 REM AANTAL VARIABELEN UIT
  BESTAND LEZEN, MET EINDE
  BESTAND TEST
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST4,S,R":GOSUB
   500
20 INPUT#1,X$
30 PRINTX$
40 IF NOT ST AND 64 THEN GOTO 20
50 CLOSE 1
60 CLOSE 15
70 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

De eerste vijf teksten verschijnen zoals het hoort op het scherm. Maar in plaats van de zesde, lange tekst komt de foutmelding "?STRING TOO LONG ERROR IN 20" te voorschijn. Wat is hier nu weer aan de hand? Is INPUT# niet te gebruiken voor langere teksten?
Dat klopt, teksten langer dan 80 tekens zijn niet met het INPUT-commando in te lezen, hoewel een tekst-variabele op zich meer dan 200 tekens kan bevatten. Gelukkig brengt het GET commando uitkomst.

```
2 REM LANGE TEKST VARIABELEN UIT
  BESTAND LEZEN, MET EINDE
  BESTAND TEST
5 OPEN 15,8,15
10 OPEN1,8,2,"TEST4,S,R":GOSUB
   500
20 A$=""
30 GET#1,B$:IF B$="" THEN GOTO
   30
40 IF B$=CHR$(13) THEN GOTO 60
50 A$=A$+B$:GOTO 30
60 PRINT "GELEZEN REGEL:"
70 PRINT A$
80 IF NOT ST AND 64 THEN GOTO 20
90 CLOSE 1
100 CLOSE 15
110 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Het eenvoudige INPUT# uit de vorige programma's is vervangen door de constructie in de regels 20 tot en met 70. Eerst wordt de variabele A$, waarin de tekst opgebouwd gaat worden, leeg gemaakt. De eerste keer is dat niet echt nodig, maar bij het begin van het inlezen van de tweede tekst zal er al wat instaan, namelijk de eerste tekst.
In regel 30 leest GET# steeds 1 teken in, dat in B$ geplaatst wordt. Het kan gebeuren dat GET# een lege string oplevert, maar ook dat wordt in regel 30 ondervangen. In zo'n geval wordt meteen de volgende GET# uitgevoerd.
Als er wel iets ingelezen was, dan wordt in regel 40 gekeken of dat CHR$(13) was, de scheider tussen de diverse teksten. Zo ja, dan wordt de som van de ingelezen strings, A$ afgedrukt. In het andere geval wordt het teken in B$ bij de tekst in

1.2 Diskdrive

opbouw in A$ gevoegd en springt het programma terug naar het lezen in regel 30.

Deze constructie heeft voor- en nadelen. Want hoewel het verreweg de veiligste manier is om een bestand te lezen, waarbij er nooit fouten kunnen optreden, werkt het veel trager dan de INPUT# methode. Niet alleen omdat daar de tekst met een enkel commando werd ingelezen, maar voornamelijk, omdat de hulpstring B$ zovaak opnieuw ingelezen wordt. Dat leidt tot op de lange duur tot een 'garbage-collect', waarover in hoofdstuk 5 meer verteld wordt.

Tot slot van deze reeks voorbeelden nog een nuttig programma. Het komt vaak genoeg voor dat een bepaald bestand op een schijf staat waarvan niet meer bekend is wat de inhoud nu eigenlijk is. Goede raad is duur in zo'n geval, want zonder meer weggooien is toch wat al te cru – morgen kan blijken dat het een onvervangbaar programma geweest is – maar alles zonder meer bewaren kost alleen maar ruimte op de diskette. Het zou ideaal zijn als zo'n bestandje even doorgebladerd zou kunnen worden, om vast te stellen wat het nu eigenlijk is.
Dat kan, met het volgende programma. Het is rechtstreeks afgeleid van het vorige voorbeeld, met dien verstande dat zowel de bestandsnaam als het type via het toetsenbord ingevoerd mogen worden.

```
2 REM LEZEN WILLEKEURIG BESTAND
5 OPEN 15,8,15
10 PRINT "␣BESTANDEN BEKIJKEN"
20 INPUT "␣GEEF NAAM BESTAND";F$
30 INPUT "␣GEEF TYPE BESTAND";T$
40 OPEN1,8,2,F$+","+T$+",R":
   GOSUB 500
50 GET#1,B$:IF B$="" THEN GOTO
   50
60 PRINTB$;
70 IF NOT ST AND 64 THEN GOTO 50
80 CLOSE 1
90 CLOSE 15
100 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Dit keer worden de met GET ingelezen tekens niet eerst tot grotere teksten samengevoegd maar rechtstreeks op het scherm afgedrukt. Voor deze toepassing voldoet die simpeler methode uitstekend.

Let op, ook bestanden met het type PRG – programma's dus – kunnen met dit programma bekeken worden. Zo'n PRG-bestand kan namelijk gewoon worden geopend en gelezen, net als ieder ander bestand. Echter, alleen de leesbare tekst – in REM-regels en tekst-constanten – blijkt inderdaad ontcijferbaar. De BASIC-sleutelwoorden zijn getokenized (zie hoofdstuk 5/2.1) en daarom onleesbaar. Sterker nog, sommige van deze tokens hebben allerlei bij-effecten op het scherm. De kleuren kunnen opeens veranderen, of het scherm wordt opeens gewist.

Om die bij-effecten te ondervangen moeten alle niet afdrukbare tekens ook inderdaad niet worden afgedrukt. Er moet een ASCII-filter in het programma worden ingebouwd. Dat ziet er als volgt uit:

```
2 REM LEZEN WILLEKEURIG BESTAND,
  MET ASCII-FILTER
5 OPEN 15,8,15
10 PRINT "␣BESTANDEN BEKIJKEN"
```

1.2 Diskdrive

```
20 INPUT "[U]GEEF NAAM BESTAND";F$
30 INPUT "[U]GEEF TYPE BESTAND";T$
40 OPEN1,8,2,F$+",",+T$+",R":
   GOSUB 500:GOTO 60
50 IF ST AND 64 THEN GOTO 100
60 GET#1,B$:IF B$="" THEN GOTO
   60
70 IF ASC(B$)<32 OR ASC(B$)>127
   THEN GOTO 50
80 PRINTB$;
90 GOTO 50
100 CLOSE 1
110 CLOSE 15
120 END
500 REM LEES DISK-STATUS EN
    REAGEER
510 INPUT#15,FM,FM$,FT,FS
520 IF FM=0 THEN RETURN
530 PRINT FM;FM$,FT,FS:PRINT
540 PRINT "DISK-FOUT OPGETREDEN,
    SLUIT UW BESTANDENHANDMATIG
    AF"
```

Regel 70 zorgt ervoor dat al die niet afdrukbare tekens worden overgeslagen. Maar om deze regel in de logica van het programma te kunnen inpassen moest de einde-bestand test worden verplaatst. Bij ieder programma dat bestanden leest is de beste plek voor die test vlak voor de eigenlijke lees-opdracht, zodat er naar die test gesprongen kan worden als er mogelijk iets gelezen moet worden. Deze tweede versie werkt feilloos, zonder het scherm in de war te brengen.

6/1.2.4

Relatieve bestanden

Een van de betere eigenschappen van de 1541 is de mogelijkheid tot het aanleggen van relatieve bestanden. Dit zijn bestanden waarvan de diskdrive zelf bijhoudt waar bepaalde gegevens zich op de disk bevinden. Wanneer de gebruiker de drive kenbaar maakt dat hij gegeven nummer 17 uit een bepaald bestand wil hebben dan zal de drive ook precies dat gegeven naar de computer toezenden, het zoeken naar dit gegeven duurt echter minder dan 2 seconden! Op die manier is het mogelijk om de 1541, desnoods in combinatie met sequentiële bestanden, voor professionele en snelheidvereisende toepassingen te gebruiken.

Uit bovenstaande inleiding blijkt dat relatieve bestanden nóg een groot voordeel hebben: de gegevens staan op de schijf opgeslagen en niet in het geheugen van de computer. Het is dus mogelijk om met hele grote bestanden te werken, veel groter dan het geheugen van de computer zou kunnen bevatten. Ledenadministraties en dergelijke worden zo een fluitje van een cent.

Opbouw

Een relatief bestand bestaat uit een aantal gegevens, records genaamd. Deze records bestaan elk weer uit een aantal subgegevens, welke velden genoemd worden. Eén van de vereisten om nu een relatief bestand op te zetten is te weten wat de lengte van een record is. De velden zijn binnen een record in principe van lengte te variëren, dit komt in de praktijk echter nooit voor. Een voorbeeld van een record is bijvoorbeeld:
Veld 1: Naam (20 tekens)
Veld 2: Adres (20 tekens)
Veld 3: Postcode (7 tekens)
Veld 4: Woonplaats (20 tekens)
Veld 5: Telefoon (11 tekens)
De totale lengte van het record is dan 78 tekens. Tel daar 1 bij op voor een extra RETURN (CHR$(13)) dan komen we op 79 tekens uit en dit is een gunstige lengte. Waarom dit zo is wordt verderop in de tekst uitgelegd.

Nu de opbouw van een record bekend is kan het bestand 'aangemaakt' worden. Dit is het initialiseren van de file op een schijf, meestal wordt dan ook het maximale aantal records bepaald. Dit is niet noodzakelijk, er kunnen namelijk ten alle tijde records achter aan de file toegevoegd worden.
In het voorbeeld gaan we echter uit van een vaste grootte, te weten 100. Op die manier kunnen we 100 bekenden in ons bestand opslaan. Het openen van een relatieve file gaat als volgt:

OPEN 1,8,2,"REL BESTAND,L,"+
CHR$(recordlengte):OPEN 2,8,15
Er worden twee kanalen geopend, een voor de data en een voor de commando's. Let er op dat een groot bestand veel initialisatietijd nodig heeft. Soms wel 10 minuten!
Wanneer nu het honderdste record met een CHR$(255) beschreven wordt dan worden op die manier 100 records geïnitialiseerd. Het honderdste record wordt op de volgende manier bereikt:
PRINT#2,"P"+CHR$(2)+
CHR$(100)+CHR$(0)+CHR$(1)
Het schrijven van CHR$(255) naar dit record gaat op de gebruikelijke manier:
PRINT#1,CHR$(255)
Dit resulteert echter wel in een foutmelding, te weten RECORD NOT PRESENT. Daar we een bestand aan het initialiseren zijn is dit een vrij logische zaak, de foutmelding kan daarom ook genegeerd worden. De foutmelding wordt normaliter gebruikt om het fout positioneren in een reeds bestaand bestand te onderscheppen, voor het geval de gebruiker een niet bestaand record wil lezen.

De diverse parameters in het PRINT#2...–commando hebben de volgende betekenis:
P: Position. De drive wordt verzocht een bepaald record te localiseren
2: Dit is het kanaalnummer waarover geschreven wordt
100: Dit is het lowbyte van 100
0: Dit is het highbyte van 100
1: Dit is het eerste teken van het record dat gezocht wordt.
Om een record aan te duiden wordt gebruik gemaakt van een low- en een highbyte. Dit omdat een enkel byte slechts 256 waardes kan hebben. Twee bytes echter kunnen 256*256=65536 verschillende records aanwijzen, dit is dan ook het maximaal te definiëren aantal records.
Het opsplitsen in een low- en highbyte gaat als volgt:
HB=INT(RN/256):LB=RN–256*HB
waarin RN het gewenste recordnummer voorstelt. Let er op dat de volgorde is: lowbyte, highbyte en niet andersom! De laatste parameter, de 1, zoekt het eerste byte van het record op. Het is dus ook mogelijk om de drive midden in een record te positioneren, dit kan wel eens van pas komen.

INPUT vs. GET

Er is al even genoemd dat 79 een gunstige recordlengte is. Dit is te wijten aan de aard van het INPUT-commando. Indien er in een record geen scheidingstekens (komma, dubbele punt, puntkomma) voorkomen dan kan het record met het INPUT-commando teruggelezen worden, mits de lengte van het record niet langer dan 80 tekens is. Anders verschijnt de foutmelding STRING TOO LONG ERROR en wordt het programma afgebroken. Bevat een record wel scheidingstekens dan leest het INPUT-commando tot aan dat teken en wordt de rest overgeslagen.
Om deze problemen te vermijden is het ook mogelijk om het GET-commando te gebruiken. Dit commando leest wel scheidingstekens en kan ook alle 255 tekens van het record aan. Nadeel is echter dat het GET-commando veel langzamer is. Er moet dus afgewogen worden wat belangrijker is, meer mogelijkheden of snelheid.
Overigens zien de beide commando's er als volgt uit:
INPUT#1,G$
en

1.2 Diskdrive

G$="":FOR I=1 TO RL:GET#,A$:
G$=G$+A$:NEXT
Waarin RL voor recordlengte staat en de gegevens in G$ terechtkomen.

Subroutines

Om een programma ter verwerking van een relatief bestand te schrijven zijn er dus een aantal routines nodig:
- Aanmaken van een bestand (slechts één maal gebruiken)
- Openen van een bestaand bestand
- Lezen van een record
- Schrijven van een record

Uitgaande van bovenstaande beschrijving komen we tot de volgende subroutines:

```
1000 REM BESTAND AANMAKEN
1010 IF AR<1 OR AR>65535 THEN RETURN
1020 IF N$="" OR LEN(N$)>15 THEN RETURN
1030 IF RL<1 OR RL>254 THEN RETURN
1040 GOSUB 1400:REM OPENEN
1050 HB=INT(AR/256):LB=AR-256*HB
1060 PRINT#2,"P"+CHR$(2)+CHR$(LB)+CHR$(HB)+CHR$(1)
1070 PRINT#1,CHR$(255)
1080 CLOSE1:INPUT#2,EE,EE$,TT,SS:CLOSE2
1090 IF EE=0 OR EE=50 THEN PRINT"BESTAND GEOPEND":RETURN
1100 PRINT"FOUT:"EE;EE$;TT;SS:CLOSE1:CLOSE2:RETURN
```

In de regels 1010 t/m 1030 worden de diverse parameters gecontroleerd:
- is AR (aantal te definieren records) toelaatbaar
- is er een geldige naam voor het bestand (N$)
- is RL (recordlengte) toelaatbaar

regel 1040 opent het bestand en in 1050 en 1060 wordt de drive gepositioneerd. 1070 schrijft dan het initialisatie-byte 255 naar dit record.
Of dit allemaal goed is gegaan wordt in regel 1080 bepaald, het uitlezen van het commandokanaal gaat op de gebruikelijke manier. Eén foutmelding hoeven we ons echter niets van aan te trekken, dat is de eerder genoemde RECORD NOT PRESENT.

```
1400 OPEN 1,8,2,N$+",L,"+CHR$(RL+1)
1410 OPEN 2,8,15
1420 INPUT#2,EE,EE$,TT,SS
1430 IF EE THEN PRINT"FOUT:"EE;EE$;TT;SS:CLOSE1:CLOSE2:RETURN
1440 RETURN
```

Deze routine spreekt voor zich, het enige waar op gelet moet worden is dat er in deze routine reeds 1 (voor de extra RETURN) bij de recordlengte opgeteld wordt. Dit hoeft dus niet van te voren te gebeuren. In ons kennissenvoorbeeld zou dat betekenen dat RL gewoon op 78 gesteld kan worden.

```
1200 MD=0:GOTO 1220:REM RECORD LEZEN
1210 MD=1:REM RECORD SCHRIJVEN
1220 IF RN>AR THEN RETURN
1230 HB=INT(RN/256):LB=RN-256*HB
1240 PRINT#2,"P"+CHR$(2)+CHR$(LB)+CHR$(HB)+CHR$(1)
1250 IF MD THEN PRINT#1,GG$:RETURN
1260 IF LM=0 THEN INPUT#1,GG$:RETURN
1270 GG$="":FOR I=1 TO RL:GET#1,DU$:GG$=GG$+DU$:NEXT:RETURN
```

Bovenstaande routine combineert het lezen en schrijven, het enige wat namelijk verschillend is is het eigenlijk lees cq. schrijf-commando. 1200 is de aanroep voor lezen, 1210 die voor schrijven. Let er op dat de te schrijven gegevens in GG$ dienen te staan en ook weer in GG$ teruggelezen worden.
LM bepaalt de leesmode, een 0 roept het INPUT-commando aan, elke andere waarde de GET-routine. Het is verstandig om LM eenmalig (aan het begin van het programma bijvoorbeeld) te definiëren.

Het volgende programma voegt alle subroutines samen tot een geheel. Het hoofdprogramma loopt tot 900, daarna volgt enige data. Deze data wordt gewoonlijk door de gebruiker tijdens het programmaverloop ingevoerd, hier dient het enkel als voorbeeld.

1.2 Diskdrive

```
10 N$="TEST.REL":AR=10:RL=50:LM=0:REM LEESMODE=INPUT
20 GOSUB 1000:REM AANMAKEN
30 GOSUB 1400:IF EE THEN END:REM OPENEN
40 FOR I=1 TO 10:READ GG$:GG$=LEFT$(GG$,50):RN=I:GOSUB 1210:NEXT
50 FOR I=1 TO 10:RN=I:GOSUB1200:PRINT GG$:NEXT
60 INPUT"RECORDNR (0=STOP):";RN:IFRN<1ORRN>ARTHEN 80
70 GOSUB 1200:PRINT GG$:GOTO 60
80 CLOSE1:CLOSE2
893 END
900 DATA DIT IS REGEL 1
905 DATA EN NOG MEER TEKST
910 DATA WEKA UITGEVERIJ B.V.
915 DATA DONKER CURTIUSSTRAAT 7
920 DATA AMSTERDAM
925 DATA VAN BASIC TOT MACHINETAAL
930 DATA OP DE COMMODORE 64
935 DATA VIERDE AANVULLING
940 DATA DIT IS BIJNA DE LAATSTE REGEL
945 DATA EN DIT IS ECHT DE ALLERLAATSTE REGEL
1000 REM BESTAND AANMAKEN
1010 IF AR<1 OR AR>65535 THEN RETURN
1020 IF N$="" OR LEN(N$)>15 THEN RETURN
1030 IF RL<1 OR RL>254 THEN RETURN
1040 GOSUB 1400:REM OPENEN
1050 HB=INT(AR/256):LB=AR-256*HB
1060 PRINT#2,"P"+CHR$(2)+CHR$(LB)+CHR$(HB)+CHR$(1)
1070 PRINT#1,CHR$(255)
1080 CLOSE1:INPUT#2,EE,EE$,TT,SS:CLOSE2
1090 IF EE=0 OR EE=50 THEN PRINT"BESTAND GEOPEND":RETURN
1100 PRINT"FOUT:"EE;EE$;TT;SS:CLOSE1:CLOSE2:RETURN
1200 MD=0:GOTO 1220:REM RECORD LEZEN
1210 MD=1:REM RECORD SCHRIJVEN
1220 IF RN>AR THEN RETURN
1230 HB=INT(RN/256):LB=RN-256*HB
1240 PRINT#2,"P"+CHR$(2)+CHR$(LB)+CHR$(HB)+CHR$(1)
1250 IF MD THEN PRINT#1,GG$:RETURN
1260 IF LM=0 THEN INPUT#1,GG$:RETURN
1270 GG$="":FOR I=1 TO RL:GET#1,DU$:GG$=GG$+DU$:NEXT:RETURN
1400 OPEN 1,8,2,N$+",L,"+CHR$(RL+1)
1410 OPEN 2,8,15
1420 INPUT#2,EE,EE$,TT,SS
1430 IF EE THEN PRINT"FOUT:"EE;EE$;TT;SS:CLOSE1:CLOSE2:RETURN
1440 RETURN

READY.
```

Records met meer dan een veld

Wanneer een record uit diverse velden bestaat dan dienen deze aparte velden tot een geheel samengebracht te worden. Uitgaande van het kennissenbestand is de volgende constructie mogelijk:

GG$=NA$+AD$+PC$+WP$+TF$

Wanneer echter een naam niet precies 20 tekens heeft dan komt AD$ op de foute positie te liggen. In plaats van

WEKADonker Curtiusstraat 71051JLAmsterdam 020-867131

is het volgende eingelijk wat we willen:

WEKA Donker Curtiusstraat 7 1051 JLAmsterdam 020-867131,

met het begin van de velden op resp. positie 1, 21, 41, 48 en 68. Ze zijn dan ook weer eenvoudig terug te vinden met behulp van het MID$-commando:

NA$=MID$(GG$,1,20)
AD$=MID$(GG$,21,20/
PC$=MID$(GG$,41,7)
WP$=MID$(GG$,48,20)
TF$=MID$(GG$,68)

Het aanmaken van GG$ uit bovenstaande records gaat als volgt:

GG$=LEFT$(NA$+SP$,20)+LEFT$
AD$+SP$,20)+LEFT$(PC$+SP$,7)
LEFT$(WP$+SP$,20)+LEFT$
TF$+SP$,11)

waarin SP$ een string is van 20 spaties. In theorie moet SP$ evenlang zijn als het langste veld en uitsluitend uit spaties bestaan. Dus:

SP$="":FOR I=1 TO RL:SP$=SP$+' ":NEXT

Het volgende programma illustreert dit:

```
10 rl=78:for i=1 to 20:sp$=sp$+" ":next:lm=0:rem leesmode input
20 print"■■■■■■■■■■■■■■■■■Adressenbestand"
30 print"■■        1. Bestand aanmaken"
40 print"■         2. Bestand openen"
50 print"■         3. Records wijzigen"
60 print"■         4. Records afdrukken"
70 print"■         5. Einde"
100 get a$:if a$=""then 100
110 if a$<"1" or a$>"5" then 100
120 on val(a$) gosub 200,300,400,600
130 if a$<>"5" then 20
140 close1:close2:end
200 print"■■■■■■■■■■■■■■■Bestand aanmaken■■"
205 if op then print"
    Bestand open!":for t=1 to 2000:next:return
210 print"      Naam bestand";:input n$
220 if len(n$)<1 or len(n$)>15 then print"■";:goto210
230 print"■■";
240 print"      Maximum aantal records";:input ar$
250 if val(ar$)<1 or val(ar$)>65535 then print"■";:goto 240
260 ar=val(ar$)
270 open3,8,4,n$+"1,s,w":print#3,ar:close3:gosub1000
280 return
300 print"■■■■■■■■■■■■■■■Bestand openen■■"
310 if op then print"
    Bestand reeds open!":for t=1 to 2000:next:return
320 print"       Naam bestand  ";n$:print"■";:printtab(18);:input n$
```

1.2 Diskdrive

```
330 if len(n$)<1 or len(n$)>15 then print"■";:goto320
340 open 3,8,4,n$+"1,s,r":input#3,ar:close3
350 gosub 1400
360 if ee=0 then op=1
370 return
400 print"■■■■■■■■■■■■■■■■Records wijzigen■■"
410 if op=0 then print"
    Bestand niet open!":for t=1 to 2000:next:return
420 input"       Recordnummer";rn$:rn=val(rn$)
430 if rn<1 or rn>ar then print"■";:goto410
440 print"■■"
450 gosub 1200:rem lezen
460 na$=mid$(gg$,1,20)
461 ad$=mid$(gg$,21,20)
462 pc$=mid$(gg$,41,7)
463 wp$=mid$(gg$,48,20)
464 tf$=mid$(gg$,68)
470 print"Naam          ";na$
471 print"Adres         ";ad$
472 print"Postcode      ";pc$
473 print"Woonplaats    ";wp$
474 print"Telefoon      ";tf$
480 print"■■■■■"tab(10);:input na$
481 printtab(10);:input ad$
482 printtab(10);:input pc$
483 printtab(10);:input wp$
484 printtab(10);:input tf$
490 na$=left$(na$+sp$,20)
491 ad$=left$(ad$+sp$,20)
492 pc$=left$(pc$+sp$,7)
493 wp$=left$(wp$+sp$,20)
494 tf$=left$(tf$+sp$,11)
500 gg$=na$+ad$+pc$+wp$+tf$
510 gosub 1210
520 return
600 print"■■■■■■■■■■■■■■■■Records afdrukken■■"
610 if op=0 then print"
    Bestand niet open!":for t=1 to 2000:next:return
620 input"       Recordnummer";rn$:rn=val(rn$)
630 if rn<1 or rn>ar then print"■";:goto620
640 print"■■"
650 gosub 1200:rem lezen
660 na$=mid$(gg$,1,20)
661 ad$=mid$(gg$,21,20)
662 pc$=mid$(gg$,41,7)
663 wp$=mid$(gg$,48,20)
664 tf$=mid$(gg$,68)
670 print"Naam          ";na$
671 print"Adres         ";ad$
672 print"Woonplaats    ";wp$
```

1.2 Diskdrive

```
673 print"Postcode     ";pc$
674 print"Telefoon     ";tf$
680 print"■■      Druk op een toets"
690 get a$:if a$="" then 690
700 return
999 end
1000 rem bestand aanmaken
1010 if ar<1 or ar>65535 then return
1020 if n$="" or len(n$)>15 then return
1030 if r1<1 or r1>254 then return
1040 gosub 1400:rem openen
1050 hb=int(ar/256):lb=ar-256*hb
1060 print#2,"p"+chr$(2)+chr$(lb)+chr$(hb)+chr$(1)
1070 print#1,chr$(255)
1080 close1:input#2,ee,ee$,tt,ss:close2
1090 if ee=0 or ee=50 then print"bestand geopend":return
1100 print"fout:"ee;ee$;tt;ss:close1:close2:return
1200 md=0:goto 1220:rem record lezen
1210 md=1:rem record schrijven
1220 if rn>ar then return
1230 hb=int(rn/256):lb=rn-256*hb
1240 print#2,"p"+chr$(2)+chr$(lb)+chr$(hb)+chr$(1)
1250 if md then print#1,gg$:return
1260 if lm=0 then input#1,gg$:return
1270 gg$="":for i=1 to r1:get#1,du$:gg$=gg$+du$:next:return
1400 open 1,8,2,n$+",l,"+chr$(r1+1)
1410 open 2,8,15
1420 input#2,ee,ee$,tt,ss
1430 if ee then print"fout:"ee;ee$;tt;ss:close1:close2:return
1440 return

ready.
```

6/2

Communicatie

INHOUD

Een van de zaken die steeds meer computerhobbyisten interesseren is het communiceren met behulp van computers. Begrijpelijk, want het is een van de boeiendste toepassingen van de homecomputer. Databanken, Viditel, bulletin boards, allerlei zaken zijn met behulp van een 64 met een modem en wat software -- en natuurlijk een telefoon -- bereikbaar.
Hoe het een en ander echter technisch in elkaar steekt is voor de meeste mensen een groot raadsel. Interfaces, modems, RS232, voor verreweg de meeste gebruikers zijn het alleen maar kreten en kastjes. Dat geeft natuurlijk niets, zolang het allemaal maar goed werkt. Als er echter, om welke reden dan ook, een probleem ontstaat staan veel mensen met hun handen in het haar. Toch is het allemaal niet zo vreselijk ingewikkeld, zoals u in dit hoofdstuk zult zien.

6/2.1

Wat is computer-communicatie?

Laten we eerst eens bepalen wat die computercommunicatie nu precies inhoudt. Want daar kunnen heel wat onduidelijkheden over bestaan.
Zo zal bijvoorbeeld iedereen wel van mening zijn dat het gebruik van Viditel met een Commodore 64 een goed voorbeeld van computercommunicatie is. Maar datzelfde Viditel is ook zonder computer te gebruiken. Altans, zonder zichtbare computer, want zo'n speciaal op Viditel ingerichte televisie bevat in feite een klein Viditel-computertje. Zonder een volledig toetsenbord weliswaar, maar dat is niet van belang. Waar het wel om gaat is dat er allerlei informatie in een digitaal formaat wordt overgezonden, in het geval van Viditel via een telefoonlijn.

Om nog een voorbeeld te geven; het is tegenwoordig mogelijk om allerlei elektronische muziek-instrumenten zoals synthesizers en drum-machines met computers te besturen, via de zogenaamde midi-interface. Daarbij gaat alweer op dat er allerlei informatie in een digitaal formaat tussen de verschillende apparaten wordt uitgewisseld. In feite bevatten die drum-machines en synthesizers ook hele gespecialiseerde computertjes. Met andere woorden, ook zo'n midi-interface is een vorm van computercommunicatie.

6/2.2

Data-communicatie

Waar de meeste mensen aan denken als ze het over computercommunicatie hebben is het transporteren van voor mensen leesbare gegevens van de ene naar de andere computer. Viditel, allerlei andere data-banken en bulletin-boards zijn daar een uitstekend voorbeeld van.

Maar ook de programmeur die een terminal gebruikt die op een grote computer is aangesloten – waarbij de opdrachten op die terminal worden ingetikt om daarna verzonden te worden naar de centrale computer – maakt gebruik van een vergelijkbare vorm van computercommunicatie. Met een fraai woord heet die vorm 'datacommunicatie'.

Tot voor kort was datacommunicatie eigenlijk voorbehouden aan de technici die met de grote centrale computers werkten. Deze specialisten bouwden en bouwen hele netwerken op rond dergelijke mainframes, om de gebruikers in staat te stellen de machines zonder verdere problemen te gebruiken.
Feitelijk was datacommunicatie iets waar alleen de technici mee te maken hadden, voor de gebruikers was zo'n systeem tamelijk transparant. Met andere woorden, als het goed was merkten die gebruikers niet dat er een heel netwerk tussen hen en de eigenlijke computer geschakeld zat.

Met de opkomst van de homecomputer, waarbij de hobbyist het allemaal zelf moet doen en geen beroep kan doen op datacommunicatie-specialisten, is daar echter verandering in gekomen. De technische details zijn vooral voor die hobbyist van belang, want allerlei problemen in datacom zijn in feite terug te voeren op kleinigheden en dan ook simpel op te lossen. Als men maar weet hoe!

Dat hoe zullen we in komende afleveringen uit de doeken doen. Eerst geven we daartoe een algemeen overzicht over wat datacom nu precies inhoudt, de gespecialiseerde vormen – en de technische details – komen later aan bod.

Datacommunicatie wordt al lang op allerlei manieren gebruikt, bijvoorbeeld door reisbureaus en banken, maar ook door bevolkingsregisters in grotere steden.
Als u bijvoorbeeld bij een reisbureau een vlucht boekt zal men over het algemeen de door u gewenste reis meteen op een terminal intikken. Die terminal verzendt de informatie dan naar een centrale computer, waarin bijgehouden wordt welke plaatsen nog beschikbaar zijn. Als er nog ruimte is op de vlucht

2.2 Data-communicatie

die u gekozen hebt zal er een boodschap met die inhoud naar de terminal van het reisbureau teruggezonden worden. Mocht u definitief besluiten die vlucht te willen boeken wordt daarna via diezelfde terminal dat ook weer aan de centrale computer doorgegeven, die dan de lijsten met beschikbare plaatsen meteen bijwerkt.

Op die manier kunnen er in principe nooit meer stoelen worden verkocht dan er beschikbaar zijn, waarbij ieder reisbureau dat op zo'n computer aangesloten is de klant meteen kan bevestigen of de gewenste plaats al dan niet beschikbaar is.

Dat het een en ander in de praktijk nog wel eens mis loopt is dan ook geen computerfout, hoewel men gedupeerde klanten graag met dat kluitje in het riet stuurt. Overboeking is echter opzettelijk in het systeem ingebouwd, om te voorkomen dat in het geval van afzeggingen de luchtvaart maatschappij met lege stoelen blijft zitten. Die heren zien liever dat u door overboeking voor een dichte vliegtuigdeur blijft staan dan dat zij het risico lopen met onbezette plaatsen te vliegen.

Terminals

In dit voorbeeld is verschillende keren gesproken over het begrip terminal. Zo'n terminal is een apparaat waarmee we de eigenlijke computer opdrachten kunnen geven en de resultaten weer teruglezen. Kortom, het toetsenbord en beeldscherm tezamen vormen in het geval van een Commodore 64 een terminal.

Het verschil – op het eerste gezicht altans – zit hem in het feit dat deze terminal alleen maar met de 64 kan communiceren waarvan hij een integraal onderdeel vormt.

Maar ook dat is niet helemaal juist. Terminals bestaan namelijk in vele maten en soorten, vanaf primitieve teletypes – een soort telex-apparaat – tot en met machines voorzien van een eigen 'intelligentie'. Zo'n teletype kan namelijk alleen maar toetsaanslagen verzenden om daarna het antwoord van de eigenlijke computer weer te geven als lettertjes op papier. Met andere woorden, de snelheid wordt eigenlijk bepaald door de typesnelheid van de gebruiker.

In de praktijk bleek al snel dat een terminal met wat meer mogelijkheden veel prettiger werkte. Zo stellen sommige van die zogenaamde 'intelligente' terminals de gebruiker in staat om eerst een hele serie commando's voor de centrale computer in te tikken, om pas daarna de informatie in één keer te verzenden. Dat werkt dan veel en veel sneller en daardoor goedkoper. Want meestal moet men computertijd op zo'n centraal systeem per minuut betalen.

Als die centrale computer grote hoeveelheden uitvoer naar de terminal stuurt krijgen we een vergelijkbare situatie. De meeste communicatielijnen zijn namelijk veel en veel sneller dan een mens kan lezen, waardoor er veel tijd verspild wordt aan het wachten op die menselijke bediener. Vandaar dan ook dat een intelligente terminal vaak in staat is om binnenkomende berichten op te slaan, óf in het geheugen óf op een floppy disk.

2.2 Datacommunicatie

Zo'n intelligente terminal is tegenwoordig dan ook een computer op zich, compleet met diskdrives. Men kan lokaal – dus terwijl er geen verbinding bestaat met een centrale computer – allerlei zaken afhandelen, zoals tekstverwerking en dergelijke.

Het is echter ook zonder meer mogelijk om een home- of personal computer als intelligente terminal te programmeren. En daarmee komen we dan weer terug op de eerdere opmerking, dat het verschil tussen een 'echte' terminal en de in onze 64's ingebouwde terminals in het feit schuilt dat de ingebouwde terminal alleen maar met die Commodore 64 computer kan communiceren waarvan hij een integraal onderdeel vormt. Als we namelijk een stapje verder denken blijkt dat die gehele 64 als intelligente terminal beschouwd kan worden. Mits voorzien van de juiste programmering – en wat extra elektronica – is een 64 een heel aardige intelligente terminal.

Verschil

Het grote verschil met het rechtstreeks gebruiken van een computer of deze via een werkstation zoals een terminal bedienen zit hem niet alleen het feit dat zo'n centrale machine meestal door veel mensen tegelijk gebruikt wordt. Eigenlijk veel belangrijker voor dit verhaal is de afstand tussen de terminal en de 'host-computer'. Die afstand tussen terminal en eigenlijke computer brengt namelijk nogal wat technische problemen met zich mee.

We kunnen zo'n verbinding niet zien als wat extra lange kabels tussen de verschillende onderdelen van het systeem. Het is niet zo dat toetsenbord en beeldscherm zich aan de ene kant bevinden en de eigenlijke computer aan de andere kant.
In feite zijn terminal en computer twee volledig gescheiden eenheden, die helemaal op zichzelf staan. Een terminal kan ook zonder meer voor verschillende computers gebruikt worden, als zo'n computer maar dezelfde wijze van communicatie gebruikt als die terminal. Of als de terminal 'intelligent' genoeg is om voor die verschillende communicatie-wijzes geprogrammeerd te worden.

Communicatie-wijze

Laten we nog eens kijken naar het voorbeeld van daarnet, waarbij een reisbureau voor een klant een vlucht wil boeken. Daar komt namelijk heel wat berichtenverkeer bij kijken.

Nadat het verzoek om de informatie over die bepaalde vlucht namelijk ingetikt is moet dat verzoek bij de centrale computer terecht komen om daar verwerkt te worden. Met andere woorden, de ingetikte tekst moet op de een of andere manier verzonden worden.
Daar bestaan vele manieren voor, waarvan de meest gebruikelijke per telefoonlijn is. Verreweg de meeste datacommunicatie verloopt namelijk via normale telefoonlijnen.

Modulatie-demodulatie

Nu is zo'n telefoonlijn eigenlijk volkomen ongeschikt om digitale informatie over te verzenden. Een telefoonnet is bestemd om de menselijke stem mee te 'verzenden'. Met andere woorden, geluid – de stem – wordt omgezet in een signaal, dat aan de andere kant weer in geluid wordt vertaald.

2.2 Datacommunicatie

Daarvoor is ons telefoonnet ontworpen en dat kan het dan ook redelijk betrouwbaar en met weinig technische foefjes aan.
Zodra we echter iets anders dan geluids-signalen over een telefoonlijn willen verzenden komen we in de problemen. Ten eerste is dat telefoonnet daar simpelweg niet op gemaakt en ten tweede stelt de PTT nogal wat eisen aan apparatuur die op het telefoonnet mag worden aangesloten.

Om zo'n computer-signaal om te zetten naar een vorm die zonder al te veel problemen kan worden verzonden hebben we namelijk wat extra apparatuur nodig, een zogenaamd modem. Die naam is een samentrekking van de woorden modulator-demodulator en dat is dan ook precies wat zo'n kastje doet. Het uitgaande signaal wordt gemoduleerd – omgevormd – tot iets wat zich via een telefoonlijn laat versturen, het binnenkomende signaal wordt gedemoduleerd tot iets waar de computer of terminal weer raad mee weet.

In principe houdt dat moduleren en demoduleren in dat er op bit-niveau gewerkt wordt. Ieder teken dat verzonden wordt zal worden ontleed in de afzonderlijke bitjes – de ja of nee signaaltjes waar alles in onze computers om draait – om daarna per bitje verstuurd te worden.

Er zijn meerdere manieren waarop die modulatie-demodulatie kan gebeuren, welke dat zijn komen we later op terug. Voor het moment volstaat het om te zeggen dat beide modems – zowel de zendende als de ontvangende computer hebben er een nodig – dezelfde werkwijze moeten volgen.

Wel iets om nu al even op te letten is het feit dat de PTT – zoals al gezegd – hoge eisen stelt aan apparaten die aan het telefoonnet gekoppeld worden. Alleen door de PTT goedgekeurde apparaten mogen daarvoor gebruikt worden.

Dat houdt in dat in feite elk modem dat in Nederland te koop is een PTT-goedkeuringszegel zou moeten dragen, maar jammer genoeg is dat niet helemaal het geval. Het is namelijk wel verboden om niet-goedgekeurde modems te gebruiken, maar niet om ze te verhandelen. Oftewel, als een winkel u een niet-goedgekeurd modem verkoopt dan mag dat, maar als u datzelfde modem aansluit bent u wel in overtreding. Meestal heeft dat geen gevolgen, zolang het apparaat maar goed funktioneert en er geen telefoonmonteur over de vloer komt is er niets aan de hand.

Het kan echter ook anders gaan. Om maar een voorbeeld te geven, stel, u gebruikt een niet-goedgekeurd modem en op een gegeven moment gaat daar iets mee mis. Op de een of andere manier komt de netspanning – 220 volt – op de telefoonlijn te staan. Dat kan de nodige schade opleveren in de telefooncentrale, schade waarvoor u – omdat u een nietgoedgekeurd modem gebruikt – aansprakelijk bent. Zo'n ongelukje is gelukkig niet erg waarschijnlijk, ook de niet toegelaten modems zitten over het algemeen tamelijk degelijk in elkaar. Maar het zal je maar gebeuren . . .

Baud-rate
Zoals reeds gesteld dienen beide gesprekspartners in een computerverbin-

2.2 Datacommunicatie

ding het met elkaar eens te zijn over een aantal dingen. Zo dienen de beide modems het eens te zijn over de snelheid waarmee gecommuniceerd wordt. Daar zijn vele mogelijkheden voor, variërend van bijzonder traag tot supersnel.

Die snelheden worden uitgedrukt in baud, wat in feite staat voor kloksnelheid. De gegevens worden namelijk bitsgewijs verzonden, waarbij er per tik van die klok één bitje verstuurd wordt. Zo staat een snelheid van 300 baud – een heel gebruikelijke snelheid voor hobby-verkeer – in feite voor 300 bitjes per seconde.
Dat lijkt heel wat, maar voor computerbegrippen valt het nogal tegen. Het is niet sneller dan de snelheid waarmee een Commodore datarecorder gegevens opneemt.

Bovendien kan men bij communicatie niet zonder meer de baudsnelheid in een aantal tekens per seconde vertalen. Dat hangt namelijk weer af van allerlei andere factoren. Ruwweg komen die 300 baud overeen met een snelheid van zo'n dertig tekens per seconde. Met andere woorden, het verzenden van een vol scherm van 1 kB kost zo'n 30 seconden.

Gelukkig is het niet nodig dat zowel de zender als de ontvanger dezelfde baudrate gebruiken. Zo gebruikt Viditel twee verschillende baud-rates, waarbij de centrale computer zijn informatie met 1200 baud verzendt en de terminal slechts een snelheid van 75 baud gebruikt. Dat gaat prima, gezien het feit dat er aan de ene kant een relatief traag mens zit te typen die meestal langzamer zal zijn dan die 75 baud. De meeste informatie vloeit van de centrale computer naar de ontvanger, over het snelle 1200 baud-kanaal.

Woordlengte
Het feit dat we de snelheid in baud niet zonder meer kunnen omrekenen in een snelheid in tekens per sekonde ligt er onder meer aan dat zo'n teken niet altijd uit evenveel bits hoeft te bestaan. In een 64 is een teken altijd gelijk aan een byte, 8 bits dus, maar voor communicatie-doeleinden is dat meestal niet nodig.
Als we bijvoorbeeld alleen maar de beschikking willen hebben over de letters van het alfabet – zowel hoofdletters als kleine letters –, de cijfers en wat leestekens hebben we alles bij elkaar niet meer dan zo'n 72 tekens te verzenden. Als we daarbij dan ook nog rekening houden met wat speciale intern gebruikte codes kunnen we de hele communicatie afhandelen met een gereduceerde tekenset van – ruim gesteld – 128 verschillende codes.

En dat houdt dan in dat we in plaats van 8-bits codes 7-bits codes kunnen gaan verzenden, hetgeen een tijdwinst per teken oplevert van 12,5 procent. Veel communicatie-systemen maken daar dan ook gebruik van en verzenden inderdaad 7-bits woorden,in plaats van de 8-bits die de machines meestal intern gebruiken.

Als we afzien van die aparte hoofd- en kleine letters, zoals sommige van de grote wetenschappelijke computers doen, kunnen we alle nodige tekens zelfs in 6 bits uitdrukken, hetgeen vergeleken met 8 bits in theorie zelfs 25 pro-

2.2 Datacommunicatie

cent sneller gaat. Dit gaat echter wel ten koste van de leesbaarheid.

Start- en stopbits
Behalve de bitjes die het eigenlijke teken vormen moet er echter nog wel wat meer verzonden worden. Ieder teken, of dat nu uit 6, 7 of 8 bitjes bestaat, wordt ingeleid met een start-bit, een boodschap die zoveel betekent als 'opgepast, hier komt er weer een'.

Na elk teken worden er weer één of meer stopbits verzonden, hoeveel kan per verbinding verschillen.
Al die extra bitjes hebben weer een nadelige invloed op de totale snelheid van verzenden, waardoor de uiteindelijke snelheid meestal ruwweg berekend kan worden door de baudrate door 10 te delen.

Pariteit
Tussen het eigenlijke teken en de stopbits kan er nog een extra bitje verzonden worden, het pariteits-bit. Dit maakt geen deel uit van het eigenlijke teken, maar heeft daar wel alles mee te maken.
Met behulp van het pariteits-bit kan de ontvanger namelijk bepalen of het teken al dan niet verminkt is binnengekomen.

Dat verminken kan namelijk heel makkelijk gebeuren, een kraakje op de lijn, zoals die maar al te vaak voorkomen, kan al genoeg zijn. Als er door zo'n kraakje ook maar een honderdste seconde geen signaal ontvangen kan worden scheelt dat – bij een 300 baud verbinding – al 3 bits.

Met behulp van zo'n pariteits-bit kunnen veel – maar niet alle – fouten worden vastgesteld. Er zijn twee vormen van pariteit, even en oneven, en zoals altijd moeten beide computers het er over eens zijn welke gebruikt wordt. De zendende partij telt namelijk tijdens het verzenden hoeveel één-bits het teken telt, en verzendt dan tenslotte een pariteits-bit dat dit aantal even – of juist oneven – maakt. Omdat de ontvanger ook telt kan een afwijking hiervan worden vastgesteld, hetgeen op een transmissie-fout duidt.

Stel, we verzenden een letter in een 7-bits woord, dan kan dat er als volgt uitzien als we de start- en stop-bits even verwaarlozen:
0100110
Oftewel, drie keer een 1 en vier keer een nul.
Uitgaande van een even pariteit zal het pariteits-bit dan een 1 zijn, om het totaal aantal 1-bits op 4 – een even getal – te brengen.

Uitgaande van een oneven pariteit zal er aan de code 1100011 ook een 1 worden toegevoegd, immers, het aantal enen in het eigenlijke teken was 4 en dus niet oneven.

Deze pariteits-controle is weliswaar niet afdoende om alle mogelijke fouten te ontdekken, maar biedt wel een behoorlijke zekerheid tegen eventuele fouten. Op andere, betere systemen komen we later nog terug.

Desgewenst – bijvoorbeeld als er zo'n extra en betere controle gebruikt wordt – kan de pariteits-kontrole ook uitgezet worden. Dat scheelt dan weer een bitje per teken.

6/2.3

Interfaces

Inhoud

6/2.3.1

Inleiding

Er komt nog wel het een en ander bij kijken, voordat we computer-communicatie kunnen gaan bedrijven. We zijn er niet met alleen maar een modem. Om met onze Commodore 64 inderdaad 'online' te gaan hebben we nog wel het een en ander nodig. De standaard 64 is daar nog niet helemaal voor ingericht.

Weliswaar bevat de Kernal al de nodige machinetaal-routines waar een communicatie-programma omheen gebouwd kan worden, maar qua hardware ontbreekt er nog wel het een en ander. Wat allemaal, dat zullen we in dit gedeelte eens in vogelvlucht bekijken. De technische details komen in latere afleveringen aan de orde.

Om met het begin te beginnen, allereerst zal er, voor we ook maar iets kunnen doen, een interface nodig zijn. Gelukkig is die interface voor datacommunicatie internationaal gestandaardiseerd, onder de naam RS232.
Of, om precies te zijn, RS232C, terwijl ook de naam V24 nog wel eens gebruikt wordt voor dezelfde interface.
Die namen zijn afkomstig van de verschillende organisaties, die zo'n interface als standaard gekozen hebben. Zo staan de letters RS voor 'Recommendation Standard', oftwel aanbevolen stanaard. De V24 is in feite exact dezelfde standaard, maar dan door een andere organisatie benoemd.

Zo'n standaard RS232-aansluiting bezit onze Commodore 64 niet. Althans, niet zonder meer. Gelukkig is het echter heel simpel om met behulp van een speciale cartridge zo'n communicatie-interface alsnog op de user-port aan te sluiten.
Laat u niet in verwarring brengen door het feit dat Commodore zelf in sommige boeken beweert dat die RS232 wel ingebouwd zou zijn. Alleen maar een kabel en een beetje programmeren, zoals de Programmers Reference Guide stelt, is bij lange na niet genoeg om met datacommunicatie te beginnen. U heeft in bijna alle gevallen zo'n speciale RS232 cartridge – en een goed machinetaal-programma – nodig.

Dergelijke cartridges bestaan in meerdere maten en soorten, onder andere de VIC-1011A van Commodore zelf. Een andere cartridge, met wat meer mogelijkheden, wordt gemaakt door het Zweedse Handic.
Een nadeel echter van zo'n cartridge is wel dat we die user-poort meteen kwijt zijn. De cartridge bezet die poort, en daar mee uit. Tegelijkertijd andere zaken op de user-poort aansluiten – zoals een Centronics-interface voor een printer – gaat niet, hetgeen in sommige gevallen

wel heel erg lastig kan wezen. Naast de RS232-norm bestaan er ook nog allerlei andere standaard-interface's, zoals bijvoorbeeld de als opvolger van de RS232 bedoelde RS449. Deze laatste is echter totaal niet aangeslagen, zodat we mogen stellen dat de RS232 eigenlijk de enige communicatie-interface is waar we als hobbyisten mee te maken zullen krijgen.
In principe is zo'n interface niets anders als een aansluiting, die juist bedoeld is om allerlei problemen – die nu eenmaal de kop opsteken als men twee apparaten met elkaar verbinden wil – te vermijden. Door die standaardisatie zou men mogen aannemen dat twee met RS232-interfaces uitgeruste apparaten zonder moeite met elkaar gekoppeld kunnen worden; de techniek – hoe het precies allemaal in zijn werk gaat – zou eigenlijk voor de uiteindelijke gebruiker niet meer van belang moeten zijn.

Toch komt er van dit techniek nog heel wat om de hoek kijken. Want hoewel een RS232-verbinding in feite niets anders is dan een kabeltje tussen twee apparaten – natuurlijk met de definitie van welke signalen er allemaal gebruikt worden – is het een behoorlijk ingewikkelde materie. Men zou kunnen zeggen dat een RS232 verbinding in bepaalde opzichten niets anders is dan een verlengsnoer, maar dan wel een ingewikkeld verlengsnoer.
Gelukkig kan men in de meeste gevallen met vrij weinig technisch inzicht zo'n RS232-verbinding tussen twee apparaten leggen en gebruiken, maar dat gaat lang niet altijd op. Vandaar dat we de RS232 techniek toch tamelijk diepgaand zullen behandelen.
We doen dat echter wel in gedeeltes. Zo zult u eerst een simpele uitleg aantreffen, waarmee echter de meeste mensen al uit de voeten zullen kunnen komen. Want nogmaals, hoewel het ingewikkelde stof is, in de meeste gevallen zal men al die achterliggende kennis niet nodig hebben om aan het werk te kunnen. Pas als er problemen optreden, dan zal men dieper moeten gaan spitten.

6/2.3.2

RS232C, een overzicht

Zo'n RS232-verbinding bestaat in feite uit een hele serie draadjes, ieder met hun eigen, nauwkeurig omschreven functie. Behalve de eigenlijke data-verbindingen – twee stuks, eentje voor het verzenden en eentje voor het ontvangen – omvat de RS232-definitie nog allerlei andere lijnen.

Verreweg de meeste daarvan zijn echter totaal onbelangrijk voor onze toepassingen. Gelukkig maar, want de RS232 connector bezit maar liefst 25 pennen, waarvan er 22 in gebruik zijn!
Daaronder vinden we bijvoorbeeld lijnen zoals 'secondary channel', voor het geval dat we via een en dezelfde RS232 interface twee verbindingen tegelijkertijd zouden willen maken. Dit secondary channel wordt echter door de tegenwoordige modems – zeker bij de modellen die voor hobbyisten bestemd zijn – niet meer gebruikt.
Ook bevat de volledige RS232 meerdere verbindingen die gebruikt kunnen worden om de zendende en de ontvangende apparaten op elkaar af te stemmen.

Handshake

Dit op elkaar afstemmen wordt de 'handshake' genoemd, hetgeen het Engelse woord voor handenschudden is. In die RS232 hardware zijn daar meerdere mogelijkheden voor voorzien, die er allemaal kort gezegd op neerkomen dat de ontvanger een signaal op een bepaalde verbinding moet zetten alvorens de zender mag losbranden.

De reden voor die handshake is voor de hand liggend, immers, anders zouden er tekens verloren gaan als de ontvanger de zender niet zou kunnen bijbenen. Daarbij moeten we onderscheid maken tussen het eigenlijke ontvangen en wat er verder met die ontvangen gegevens gedaan wordt.
Dat zijn namelijk twee geheel van elkaar gescheiden zaken, een beetje zoals dat bij bijvoorbeeld een telex ook speelt. Zo'n telex namelijk ontvangt allerlei tekst en drukt die keurig af op een rol telexpapier, waarna het echter nog wel verder verwerkt moet worden. Dat telex-apparaat weet immers niet op wiens bureau een telexbericht omtrent een wanbetaler in Hongkong moet worden gelegd. Laat staan dat de machine die zaak verder zou kunnen afhandelen.
Eigenlijk kan zo'n telex alleen maar binnenkomende berichten 'bufferen', totdat er door een mens nadere aandacht aan besteed kan worden. De vewerking is geheel gescheiden van de ontvangst.

In het geval van RS232 communicatie op de Commodore 64 is er ook zo'n scheiding. De binnenkomende tekens worden opgeslagen in een interne buffer, tot ze

verder verwerkt kunnen worden. Het ontvangen en het verwerken wordt op die manier van elkaar losgekoppeld, waarbij het ontvangen de hoogste prioriteit heeft.
De in de ontvangst-buffer geplaatste tekens worden pas verder verwerkt als er tijd beschikbaar is. Daarbij kan het best gebeuren dat er, als er op bijvoorbeeld 1200 Baud ontvangen wordt, uiteindelijk slechts op de helft van die snelheid verwerkt kan worden, of zelfs nog langzamer.

Dat verschil in ontvangst-snelheid en verwerkings-snelheid wordt in eerste instantie door de buffer wel opgevangen, maar als die buffer volloopt moet de ontvanger de zender een seintje kunnen geven dat het nu wel eventjes genoeg geweest is. Stoppen dus, tot de verwerking het weer heeft kunnen bijbenen.
Pas als de buffer weer een stuk geleegd is zal de zender weer een verzoek om verdere informatie krijgen, en zo gaat dat de hele tijd maar door.

Vandaar dat iedere communicatie-verbinding een soort van handshake nodig heeft, om dat 'stop' en 'start' seintje te kunnen verwerken. Vroeger – de RS232-definitie is al behoorlijk lang geleden bedacht – gebruikte men daar aparte lijnen voor die alleen maar voor dat doel bestemd waren. En omdat er meerdere methoden in zwang waren zijn er ook meerdere lijnen in de definitie opgenomen.

Eenrichtings-verkeer

Daar komt dan nog bij dat de meeste communicatie in die tijd via een soort semi-eenrichtingsverkeer verliep. Want als we in dit voorbeeld spreken over het zendende en het ontvangende apparaat moeten we ons natuurlijk wel bedenken dat beide communicerende apparaten zowel kunnen zenden als ontvangen. Het gaat er alleen maar om wie er op een gegeven moment 'aan het woord' is, bij dat onderscheid tussen zender en ontvanger.

Maar aangezien het vroeger heel gebruikelijk was dat de verbinding maar in een richting tegelijk gebruikt kon worden – net zoals dat in de radio-telefonie gebeurde, waar men met het woord 'over' de communicatie-richting deed wisselen – moesten er wel aparte lijnen in de RS232 opgenomen worden waarmee de ontvanger de zender kon berichten dat er eventjes gewacht moest worden.

Tegenwoordig verloopt bijna alle data-communicatie via twee 'kanalen', zodat er gelijktijdig verzonden en ontvangen kan worden. Dat heeft er toe geleid dat de start- en stop-seintjes tegenwoordig vrijwel altijd als speciale codes in de data-stroom worden meegestuurd, zodat er geen aparte verbindingen meer voor nodig zijn.

Drie-draads

Dat alles houdt in dat de 25-polige RS232 eigenlijk een soort olifanten-geweer is geworden, veel te uitgebreid voor wat er tegenwoordig vereist wordt. In heel veel gevallen kan men met slechts drie van die tweeëntwintig draadjes de zaak keurig laten funktioneren, namelijk de verzend-lijn, de ontvangst-lijn en de gemeenschappelijke aarde van die twee.
In het Engels – want die taal zult u

regelmatig tegenkomen als u zich met datacom wilt bezighouden – heten ze respectievelijk Transmit Data, Receive Data en Signal Ground. Met dat drietal kunnen bijna alle verbindingen gelegd worden.

Veel moderne RS232-aansluitingen zijn dan ook aanzienlijk vereenvoudigd. Zo gebruiken de PTT-Viditel modems een 9-polige aansluiting, terwijl sommige fabrikanten van computers en modems met nog minder pennetjes toe kunnen. Deels zijn dergelijke aansluitingen geheel afwijkend van welke standaard dan ook, hetgeen de nodige problemen bij het aan elkaar knopen van de diverse apparaten kan opleveren, maar gelukkig is de standaard in de loop der jaren ook behoorlijk uitgebreid. Zo valt de 9-polige variant wel in de boeken me de officiële aanbevelingen terug te vinden.

Lastig

Toch valt het ergens wel te betreuren dat men de tot voor kort weid verbreide 25-polige D-submaniatuur aansluiting – om de officiële naam maar eens te gebruiken – gedeeltelijk verlaten heeft. Omdat namelijk alle apparaten die aansluiting bezaten was het meestal geen heksentoer om de gewenste verbindingen te leggen, hoewel er zeer zeker wel een zekere kennis voor nodig was. Meestal volstond men echter met het simpelweg doorverbindingen van alle 25 pennetjes, middels twee pluggen en een stuk lintkabel, en dat voldeed in bijna alle gevallen uitstekend. Tegenwoordig echter dient men rekening te houden met allerlei andere aansluitschema's, hetgeen vaak inhoudt dat er of een verloopkabel of soldeerwerk bij kijken komt.

Overigens bestaan er allerlei min of meer handige hulpjes voor gebruik bij het leggen en testen van RS232 verbindingen. Zo bestaan er bijvoorbeeld de 'breakout-box', wat vertaald zou kunnen worden met 'openmaak-doosjes'. Een breakout-box bezit aan beide kanten een 25-polige RS232 plug, die echter niet zonder meer met elkaar doorverbonden zijn. De twee maal tweeëntwintig aansluitingen eindigen in kleine stekkerbusjes, die middels de meegeleverde snoertjes kunnen worden doorverbonden op iedere gewenste manier. Bovendien is de breakout-box meestal voorzien van een hele batterij ledjes, lichtgevende diodes, die de 'status' van de verschillende lijnen aangeven. Het is een werkleijk ideaal instrument om allerlei RS232 problemen mee op te sporen en op te lossen, die breakoutbox.

Een professionele uitvoering ervan is echter behoorlijk prijzig, de kosten lopen in de honderden guldens. Gelukkig echter verschijnen er de laatste tijd ook allerlei simpeler en veel goedkopere hulpmiddeltjes op de markt, waarmee iemand met wat technische kennis al snel in staat is om een op het eerste gezicht 'problematische' verbinding te leggen.

Zender of ontvanger

Die mogelijke problemen kunnen bijvoorbeeld veroorzaakt worden doordat het niet altijd meteen duidelijk is welk apparaat de 'zender' is, en welk apparaat de 'ontvanger'. Want hoewel dat op het eerste gezicht glashelder lijkt wisselen die rollen natuurlijk de hele tijd om.

2.3 Interfaces

De RS232-interface is in eerste instantie bestemd om een computer en een modem aan elkaar te koppelen, maar wordt ook regelmatig toegepast om bijvoorbeeld twee computers rechtstreeks te laten communiceren.
Algemener gesteld, een RS232 verbinding loopt in principe tussen een computer met communicatie-controller en een randapparaat, zoals een modem of een printer. In zo'n opstelling zal het altijd duidelijk zijn welk van de beide apparaten als 'zender' fungeert, en welke de rol van 'ontvanger' speelt. Want hoewel die rollen voortdurend omdraaien zijn beide zijden van de verbinding toch vastgesteld in de definitie. Er wordt een draad gebruikt voor de communicatie van de computer naar het randapparaat, en de andere van de beide eigenlijke data-verbindingen verzorgt het verkeer in de omgekeerde richting.

Als er echter twee computers middels een RS232 gekoppeld moeten worden, bijvoorbeeld om twee systemen die verschillende formaten diskettes gebruiken toch informatie uit te laten wisselen, dan zullen beide computers het andere apparaat als randapparaat zien. Met andere woorden, beide computers zullen op dezelfde verbinding willen zenden en ontvangen.
In zo'n geval moeten die twee verbindingen gekruist worden, om toch tot resultaat te kunnen komen. Daar bestaan zelfs speciale hulpstukjes voor, een soort van stekker met aan beide zijden een RS232 plug. Intern worden alle pennen met elkaar doorverbonden, behalve die Transmit Data en Receive Data. Die beide lijnen worden in die stekker omgewisseld, zodat de TD-lijn aan de ene kant binnenkomt maar aan de andere kant opeens als RD-lijn verder gaat. Omgekeerd gebeurt natuurlijk precies hetzelfde met de RD-lijn. Op die manier denken beide computers met een modem te communiceren, terwijl ze feitelijk rechtstreeks met elkaar verbonden zijn. De simpele omwisseling van de zend- en ontvangst-lijnen lossen het probleem keurig op.

Zo'n stekker heet overigens een 'nullmodem', een term waarvoor geen goed Nederlands woord bestaat. Wie dat prefereert mag het echter ook wel een 'nepmodem' noemen, hoewel het heel wat voeten in de aarde zal hebben om een dergelijk apparaat bij een winkelier te bestellen. Het Engels is in computerland nu eenmaal behoorlijk ingeburgerd, en terecht eigenlijk. Waarom zouden allerlei technische termen op min of meer kromme manier vertaald moeten worden . . .

Voltage

In de inleiding is al ter sprake gekomen dat Commodore zelf beweert dat de 64 een ingebouwde RS232 zou bezitten, terwijl dat in de praktijk eigenlijk niet helemaal waar blijkt te zijn. Dat komt doordat die inderdaad ingebouwde RS232 op twee punten afwijkt van de officiële standaard.
Ten eerste hebben we gezien dat een RS232-aansluiting een 25-polige subminiatuur D-connector dient te zijn. De user-poort, die op de 64 als RS232-interffface geprogrammeerd kan worden, is echter een gewone rand-connector, een stukje van de printplaat dat uitsteekt en waar een soort stekkertje op geschoven kan worden. Er is dus op zijn minst een speciale kabel nodig om

die user-poort als RS232 te kunnen gebruiken.

Maar de in de 64 ingebouwde RS232 wijkt ook nog op een ander punt af van wat er voorgeschreven is. Alle signalen op een RS232-verbinding zijn namelijk binair, ze staan aan of uit. Dat aan en uit wordt ook in dit geval aangegeven door een tweetal spanningen. Zoals alle logische niveau's in onze 64 zijn ook op deze user-poort de spanningen 0 volt en +5 volt. In de RS232-standaard worden echter hele andere spanningen gebruikt om de logische niveau's aan te duiden. In de norm-omschrijving staat dat de logische 1 door een spanning tussen de -3 en -25 volt wordt aangegeven, terwijl een spanning tusse de +3 en +25 volt voor een 0 staat. Het gebied tussen -3 en +3 volt is 'niet gedefinieerd', dergelijke voltages hebben geen enkele betekenis.

Kortom, de TTL – die afkorting staat voor: transistor-transistor logica – spanningen die de Commodore levert voldoen niet aan de norm. De +5 volt voor een logische nul is nog wel acceptabel, maar de 0 volt – die voor een 1 staat – is officieel niet gedefinieerd. Zoals reeds gesteld, RS232 is een al vrij oude standaard, en in die tijd werden de tegenwoordig zo voor de hand liggende TTL-spanningen nog niet gebruikt.
Om een officiële RS232-aansluiting op een Commodore 64 te realiseren is dus wel degelijk een extra stukje hardware nodig, een cartridge namelijke die de spanningen omzet naar de vereiste RS232 niveau's. Zonder die omzetting zal blijken dat veel RS232-aansluitingen domweg niet functioneren, want hoewel sommige van de hedendaagse modems er wel op gebouwd zijn om met de TTL-spanningsniveau's te kunnen werken mag men daar niet klakkeloos van uit gaan.

Tenslotte

Hopelijk zal dit eerste, oppervlakkige overzicht over de RS232 mogelijkheden en problemen u niet meteen afschrikken. Nogmaals, in veel gevallen zal voor de gebruiker een RS232 verbinding alleen maar een kwestie zijn van inpluggen en aan de slag gaan.
Er kunnen echter ook allerlei problemen blijken te ontstaan, en in dat geval moet men de technische achtergronden, zoals we die in het volgende gedeelte gaan behandelen, wel kennen. Met een beetje kennis en inzicht kan men dan bijna ieder probleem wel zelf uit de wereld helpen.

Maar die achtergrond-kennis is ook nodig om allerlei eigenschappen van communicatie-programmatuur te kunnen doorgronden. Op het moment dat een onder Commodore-gebruikers weid verspreid programma als Vip-terminal de melding 'no carrier' op het scherm zet hebben we eigenlijk al rechtstreeks te maken met die RS232-techniek. Om zo'n melding goed te kunnen interpreteren is kennis omtrent die techniek onontbeerlijk.

6/2.3.3

RS232C, de techniek

Een RS232C-interface is een verzameling communicatie- en controle-signalen waardoor het mogelijk wordt om twee apparaten informatie uit te laten wisselen. Die twee apparaten kunnen bijvoorbeeld twee computers zijn, of een computer en een modem. Ook de aansluiting van een printer kan desgewenst via de RS232C-interface gerealiseerd worden.

Voor die RS232C maakt dat allemaal niets uit, het is alleen maar een verbinding tussen twee machines. Wat er precies met de informatie gebeuren moet maakt – op het niveau van het RS232C-'verlengsnoer' gezien – niets uit. De RS232C verzendt slechts seriële informatie van het ene punt naar het andere.

Richting

Er zijn een drietal verschillende manieren om die informatie te verzenden en te ontvangen, die fundamenteel verschillend zijn in het gebruik. Om een praktijkvoorbeeld te geven, als we via de RS232C een printer op de computer aan zouden sluiten, dan mogen we verwachten dat er slechts informatie van de computer naar de printer gestuurd zal worden. Het zal niet snel voorkomen dat de printer weer informatie naar de computer wil zenden.

Simplex

In zo'n geval is de 'simplex'-verbinding afdoende, de simpelse variant van de RS232C verbinding. Bij zo'n simplex-verbinding kan de informatie slechts van het ene naar het andere punt gestuurd worden, niet andersom. Met andere woorden, de computer kan tekst naar de printer zenden, de printer echter kan geen informatie terugsturen naar de computer. In zo'n geval moet er echter wel voor gezorgd worden dat de computer de af te drukken tekst niet sneller verstuurd dan de printer deze kan verwerken. Voor dat doel zijn er echter een aantal speciale controle-signalen in de RS232C definitie opgenomen.

Naast deze verouderde en bijna nooit meer gebruikte simplex-verbinding kennen we ook nog de duplex-verbindingen, waarbij beide apparaten informatie kunnen verzenden en ontvangen. Deze duplex-verbinding bestaat in een tweetal varianten.

Half-duplex

In het geval van een half-duplex verbinding is er slechts een enkel kanaal beschikbaar, waarvan beide apparaten afwisselend gebruik kunnen maken. Er kan slechts een enkel apparaat tegelijkertijd 'aan het woord' zijn.

Of het ene apparaat zendt – waarbij het andere moet ontvangen –, of het andere apparaat heeft de rol van zender.

Om van richting te kunnen wisselen moet de machine 'die de lijn heeft' deze eerst vrijgeven, net zoals dat in de radiotelefonie gebruikelijk is. Pas na dat elektronische equivalent van 'over' mag de andere kant informatie gaan verzenden.

In het geval van een modem als randapparaat moet er – in het geval dat de communicatie-richting gewijzigd wordt – wel het een en ander in dat modem omgeschakeld worden. Dat ene kanaal wordt immers of gebruikt om te ontvangen, of om te verzenden. Als een modem informatie verzendt moet deze eerst gemoduleerd worden, alvorens deze op de telefoonlijn gezet kan worden. Bij het ontvangen moet er echter juist gedemoduleerd worden.
Oftewel, afhankelijk van de richting moet of het ene of het andere stukje modem-elektronica ingeschakeld worden. Om dat te bereiken bevat de RS232C-interface twee speciale lijnen, die verderop besproken zullen worden.

Full-duplex

Bij de tweede duplex-variant zijn er twee kanalen beschikbaar, zodat beide apparaten tegelijkertijd kunnen zenden en ontvangen. Dit communiceren in beide richtingen tegelijkertijd biedt vele voordelen. Zo kan de samenwerking tussen de beide apparaten middels een full-duplex verbinding veel flexibeler geregeld worden. Dat gaat simpelweg doordat dat tweede kanaal gebruikt kan worden door de ontvanger om de zender een seintje te geven dat de informatie-stroom eventjes gestopt moet worden.

Van de drie hierboven besproken manieren waarop de RS232C-interface voor communicatie gebruikt kan worden is de laatste, de full-duplex, tegenwoordig bijna de standaard. Simplex is vrijwel uitgestorven, tenzij voor gebruik bij randapparaten als bijvoorbeeld een printer. Zo'n printer met RS232C-aansluiting zal naar alle waarschijnlijkheid de zendcircuits voor de RS232C communicatie missen, het apparaat kan alleen maar ontvangen. In dat geval is het gebruik van een simplex-verbinding natuurlijk heel voor de hand liggend.

Definitie

De communicatie-interface RS232C is in een drietal definities terug te vinden, namelijk de ISO 2110 voor de mechanische eigenschappen, de V28 norm voor de elektrische eigenschappen en de reeds eerder genoemde V24 – die gelijk is aan de RS232C-norm – voor de functionele eigenschappen. Van die drie is de V24-norm oftewel de RS232C-norm verreweg de belangrijkste, want daar staat precies omschreven welke signalen welke functies vervullen.

De standaard in de Commodore 64 ingebouwde RS232-interface voldoet noch aan de ISO 2110, noch aan de V28 en slechts ten dele aan de V24/RS232C norm.

ISO 2110

Volgens de ISO 2110 dient een RS232C aansluiting voorzien te zijn van een 25-polige sub-miniatuur D connector, een

2.3 Interfaces

platte stekker. De 64 echter heeft de RS232-signalen op de user-poort staan, een rand-contact.

V28

De elektrische aanbevelingen voor de RS232C zijn:
-3V tot -25V: een logische '1'
Control off
+3V tot +25V: een logische '0'
Control on
Hierbij zijn de spanningen gedefinieerd ten opzichte van de signaal-aarde. Het gebied tussen de −3V en de +3V is niet gedefinieerd, met andere woorden, de ontvangst-circuits mogen niet reageren op spanningen tussen de -3V en de +3V. Behalve de eigenlijke zend- en ontvangst-lijnen – waarop de communicatie plaatsvindt met '0'- en '1'-voltages kent de RS232C een groot aantal controle-lijnen, waarbij de logica omgekeerd is. Een logische '1'-spanning betekend als controle-signaal juist dat het signaal niet aanwezig is, en vice versa. Niet gebruikte controle-signalen dienen een spanning te voeren tussen de -3V en de -25V, control off-niveau dus.

Als we deze eisen vergelijken met de 64 RS232-interface blijkt dat de gebruikte spanningen op een ander niveau liggen. Als '1'- of 'control off'-niveau gebruikt de Commodore 0 volt, terwijl de '0'/ 'control on' +5 volt is.

Interface

Om aan de officiële RS232C-definitie te voldoen wat betreft de mechanisme en elektrische eisen is dan ook een extra interface noodzakelijk, die de spanning omzet naar voltages die wel aan de V28-norm voldoen. In bijna alle gevallen zal een dergelijke interface -12V en +12V signalen afleveren. Een dergelijke interface wordt in de user-poort gestoken en heeft als uitgang een officiële ISO 2110 aansluiting.

V24

In de V24/RS232C norm worden de diverse signalen besproken. In onderstaande tabel treft u hier een overzicht van aan, compleet met de in de ISO 2110 voorgeschreven pin-nummers.

2.3 Interfaces

Pen	Afkorting	Omschrijving
Aard-signalen		
1 7	GND GND	Dit is de afscherming van de kabel Gemeenschappelijke signaal retour-leiding
Communicatie hoofdkanaal		
2 3 4 5 6 8 20 21 23 22 11	TD RD RTS CTS DSR CD DTR DSQD DSRS RI STF	Transmit Data – de te verzenden data Receive Data – de ontvangen data Request To Send – Aanvraag om te zenden Clear to Send – Gereed om te zenden Data Set Ready – het modem is gereed Carrier Detect – verbindt modem met lijn Data Terminal Ready – terminal is gereed Data Signal Quality Detect – de bewaking van de signaal kwaliteit Data Signalling Rate Selector – automatische snelheidskeuze Ring Indicator – belsignaal detektie Select Transmit Frequency – omschakelen van zendfrequentie
Timing-signalen		
24 15 17	TSET 'DTE' TSET 'DCE' RSET 'DCE'	Transmit Signal Element Timing – externe zendklok 'DTE' Transmit Signal Element Timing – modem zendklok 'DCE' Receiver Signal Element Timing – modem ontvangklok

2.3 Interfaces

Timing-signalen		
24	TSET 'DTE'	Transmit Signal Element Timing – externe zendklok 'DTE'
15	TSET 'DCE'	Transmit Signal Element Timing – modem zendklok 'DCE'
17	RSET 'DCE'	Receiver Signal Element Timing – modem ontvangklok
Communicatie tweede kanaal		
14	TBCD	Transmitted Backward Channel Data – te verzenden data twee kanaal
16	RBCD	Received Backward Channel Data – ontvangen data tweede kanaal
19	TBCLS	Transmit Backward Channel Line Signal – RTS tweede kanaal
13	BCR	Backward Channel Ready – CTS tweede kanaal
12	BCCD	Backward Channel Carrier Detect – draaggolf tweede kanaal aanwezig
Test-signalen		
18	LL	Local Loop – lokale lustest
21	RL	Remote Loop back for point to point circuits – lustest op afstand
25	TI	Test Indicator – testindikatie

Van de in totaal 25 pennen die de aansluiting heeft zijn er 20 in gebruik voor de eigenlijke verbinding, terwijl er 2 – de nummers 9 en 10 – officieel gereserveerd zijn voor toekomstige uitbreidingen. Drie pennen, 11, 18 en 25, zijn niet toegekend.
De richting van de communicatie is ook van belang, want als er gesproken wordt over de TD-lijn – Transmit Data oftewel te verzenden gegevens – klopt die benaming natuurlijk maar aan een kant van de verbinding. Wat voor het ene apparaat verzenden is, is voor het andere ontvangen.
In de bovenstaande tabel moet men denken vanuit de computer, niet vanuit het modem of de printer. Op de TD-leiding zet de computer de te versturen bits, op de RD-lijn komen juist de door het modem teruggezonden gegevens binnen. Als men een RS232C moet bedraden is het altijd van belang om zich eventjes voor te houden aan welke kant van de interface men bezig is.

C64-RS232C signalen

Als we de bovenstaande tabel van RS232C-signalen volgens de officiële standaard eens vergelijken met de op de Commodore 64 beschikbare signalen valt op dat er nogal wat zaken ontbreken.

Pen	Afkorting	Omschrijving
Aard-signalen		
A	GND	Dit is de afscherming van de kabel
N	GND	Gemeenschappelijke signaal retour-leiding
Communicatie hoofdkanaal		
M	TD	Transmit Data – De te verzenden data
B/C	RD	Receive Data – De ontvangen data
D	RTS	Request To Send – Aanvraag om te zenden
K	CTS	Clear To Send – Gereed om te zenden
L	DSR	Data Set Ready – Het modem is gereed
H	CD	Carrier Detect – verbindt modem met lijn
E	DTR	Data Terminal Ready – Terminal is gereed
F	RI	Ring Indicator – Belsignaal detektie

2.3 Interfaces

In feite is de RS232C-interface van de 64 dus maar een gedeeltelijke uitvoering van wat de norm voorschrijft. Allerlei zaken, zoals het secondaire kanaal, de timing-signalen en de test-lijnen ontbreken geheel en al.

Maar ook de bovenstaande tabel geeft een te zonnig beeld van wat de Commodore allemaal in huis heeft aan mogelijkheden. Sommige van de lijnen zoals die in de tabel staan zijn weliswaar aanwezig – en wordt intern ook doorverbonden met de IC's die de RS232 afhandelen – maar worden in de Kernal verder niet ondersteund. Met andere woorden, het signaal is er wel, maar er wordt helemaal niets mee gedaan. Welke lijnen dat precies zijn zal echter later pas aan de orde komen.

Viditel-aansluiting

Behalve de officiële RS232C-aansluiting en de wat eigenaardige Commodore-aansluiting is er nog een soort RS232C-plug die weid verbreid is, namelijk de Viditel-plug. De PTT gebruikt weer een eigen plugje om de Viditel-modems aan een RS232C-verbinding te koppelen, een negenpolige aansluiting zoals we die wel kennen van joysticks en dergelijke. Voor de volledigheid geven we ook de pen-nummers en signalen uit die plug op.

Pen	Afkorting	Omschrijving
Aard-signaal		
5	GND	Gemeenschappelijke signaal retour-leiding
Communicatie		
3	TD	Transmit Data – de te verzenden data
4	RD	Receive Data – de ontvangen data
8	DSR	Data Set Ready – het modem is gereed
7	DTR	Data Terminal Ready – terminal is gereed

U ziet het, een hele beperkte set signalen. Maar toch meer dan genoeg om de vrij ingewikkelde Viditel-communcatie tot een goed einde te brengen.

Signalen

De twintig momenteel voor RS232C-communicatie gebruikte signalen vallen in een aantal groepen uiteen, namelijk aarde, primaire communicatie, secondaire communicatie (het tweede kanaal), timing-signalen en test-signalen. Daarvan zullen we in eerste instantie alleen de aarde- en primaire communicatie-leidingen bespreken, temeer daar de andere groepen op de Commodore-64 geheel afwezig zijn.

Afscherming

Op pen 1 – A op de user-poort – vinden we de afscherming, zoals die bijvoorbeeld ook in microfoon-snoeren gebruikt word. Als er langere afstanden moeten worden overbrugd met vrij

zwakke signalen – zoals bijvoorbeeld bij een microfoon-snoer – worden de signaal-aders meestal door een metalen mantel omgeven, die allerlei storingen afschermt. Deze techniek kan ook worden gebruikt bij RS232C kabels, in welk geval de mantel met deze pen doorverbonden moet worden. Meestal echter zullen RS232C kabels niet worden voorzien van een dergelijke afscherming; bij lintkabels is zoiets eigenlijk bijna onmogelijk. Als het echter zaak is om lange, storingsvrije verbindingen te leggen is het wel aan te raden om met afgeschermde kabels te werken.
Overigens maakt deze lijn officieel geen deel meer uit van de V24 standaard. In 1984 is de 'chassis-aarde' – zoals deze pen ook wel genoemd wordt – uit de standaard geschrapt. Er bestaan echter nog zeer veel computers en modems die deze lijn nog wel bezitten. Ook onze Commodore-machines.

Retour-leiding

Behalve de afscherming is er nog een tweede aard-leiding voorzien, op pen 7 (C-64; N). Dit is de gemeenschappelijke retour-lijn voor alle andere signalen, waarmee in feite de massa's van beide apparaten worden doorverbonden. Deze lijn moet altijd aangesloten worden.

Transmit Data

Op de TD-lijn – pen 2 oftewel M – wordt de informatie vanuit de computer verzonden naar het randapparaat. Altijd aansluiten dus.

Receive Data

Ook deze pennen – 3 of C en B – moeten altijd gebruikt worden, om informatie vanaf het rand-apparaat naar de computer te laten gaan. Slechts als men bijvoorbeeld een printer middels een RS232-verbinding wil aansluiten kan men deze lijn laten vervallen, daar de printer meestal niets terug te melden heeft. Echter, ook dat is niet altijd waar, afhankelijk van de toegepaste handshake.

Als er twee computers middels een RS232C aan elkaar worden gekoppeld moeten de TD en RD signalen ergens onderweg verwisseld worden, omdat anders beide machines op dezelfde ader willen verzenden. Het gevolg daarvan zou zijn dat de ontvangst-lijn RD aan beide zijden geen enkel signaal zou dragen. Dat kruisen kan zowel door in een van beide pluggen de beide draden te verwisselen als door het inzetten van een null-modem, een soort tussenstukje waarin die verwisseling plaatsvindt. De laatste oplossing is feitelijk de beste, daar men anders met een niet-standaard kabel zit. Vroeger of later zal men zich – na enkele uren vertwijfeld zoeken naar de 'fout', meestal opeens beseffen waarom een bepaalde verbinding niet wilde werken. Omdat men namelijk een 'computer-computer'-kabel, met omgewisselde RD en TD-lijnen, probeerde te gebruiken om een gewoon rand-apparaat aan te sluiten.

Request To Send en Clear To Send

Deze beide controle-lijnen, respectievelijk pen 4 (64: D) en pen 5 (K), hebben tezamen een schakel-functie bij half-duplex verbindingen. Zoals reeds even aangestipt is zal in het geval van een half-duplex verbinding of de modulator of de demodulator uit het modem op de lijn geschakeld moeten worden, afhankelijk van de richting waarin de communicatie verloopt.

2.3 Interfaces

Dat omschakelen doet het modem natuurlijk niet uit zichzelf; dat wordt vanuit de computer bestuurd. Als de RTS-lijn aktief gemaakt wordt – een spanning tussen de +3V en +25V dus – zal het modem de modulator inschakelen. Dat houdt meteen in dat de draaggolf op de lijn gezet wordt. Na een korte wachttijd, bestemd om de draaggolf stabiel te laten worden, zal het modem dan de CTS – Clear To Send oftewel klaar om te zenden – lijn aktiveren. Dit CTS-signaal is op zijn beurt voor de computer weer een teken dat er nu dan ook echt verzonden kan worden.

Het RTS-verzoek wordt dus door het modem gehonoreerd met een CTS-toestemming, nadat – bij half-duplex – de omschakeling naar moduleren heeft plaatsgevonden. Bij een full-duplex verbinding hebben we weliswaar met dezelfde volgorde te maken, maar daar is het CTS-signaal eigenlijk alleen maar een teken dat de draaggolf op de lijn staat en alles gereed is. Bij full-duplex kan er immers tegelijkertijd verzonden en ontvangen worden, de modulator en de demodulator uit het modem staan voortdurend ingechakeld, ieder op hun eigen kanaal.
Feitelijk is het RTS-CTS gebeuren bij een computer-computer verbinding overbodig. Vaak zal men de zaak dan ook kortsluiten, teneinde de beide computers te 'foppen'. Dit gaat namelijk heel simpel, door in beide pluggen de pennen 4 en 5 door te verbinden. Zodra een computer nu het RTS-signaal hoog maakt, zal datzelfde signaal via pen 5 weer onmiddellijk als CTS-signaal terugkomen. Dat kan in veel gevallen meteen twee draden tussen zender en ontvanger schelen.

Carrier Detect

Op pen 8 – op de user-poort uitgang H – vinden we het CD signaal. Dit signaal, dat ook wel DCD (Data Carrier Detect) of zelfs RLS (Received Line Signal) genoemd wordt, geeft aan dat er een draaggolf gedetecteerd is. Met andere woorden, er is contact met een modem aan de andere kant van de lijn. Dit signaal wordt door het modem opgewekt als teken aan de computer dat er tekens binnen kunnen gaan komen. Sommige communicatie-programma's gebruiken dit signaal, andere programma's negeren het volledig. Het kan zelfs gebeuren dat een programma het wel aan de gebruiker meldt dat er een carrier gedetecteerd wordt, maar daar verder niets mee doet. Een voorbeeld daarvan is Vip-Terminal, dat bij sommige opstellingen keurig 'No Carrier' meldt zodra er een draaggolf binnenkomt. Ondanks het feit dat er officieel bij het ontbreken van die carrier geen ontvangst mogelijk zou mogen zijn blijkt Vip-Terminal echter zonder problemen te kunnen werken na die melding.
Het CD-signaal wordt ook vaak 'voor de gek gehouden', door de pen intern in de plug door te verbinden met de signalen die we als volgende gaan bespreken.

Data Set Ready en Data Terminal Ready

Dit signalen-paar dient om aan te geven dat beide randapparaten aktief zijn. We vinden ze op de pennen 6 (L) voor de DSR respectievelijk 20 (E) voor het DTR-signaal. De namen Data Set en Data Terminal staan voor het randapparaat – de Data Set – en de eigenlijke computer, de Data Terminal.
In beide apparaten kan – afhankelijk alweer van de gebruikte communicatie-

software – door deze signalen getest worden of de tegenpartij wel aan staat. In feite is ook deze controle voor de typische hobby-opstelling, waarbij computer en modem naast elkaar staan, eigenlijk overbodig.

De DTR-lijn kan echter voor sommige modems ook een andere betekenis hebben. De meeste modems zullen zonder veel problemen zichzelf op de lijn schakelen, zodra dat mogelijk is. Er bestaan echter ook modems die de DTR-lijn niet zozeer als voorwaarde – is de computer wel klaar? – zien, maar als een commando. In dat geval heet dit signaal CDSL, hetgeen staat voor Connect Data Set to Line, in het Nederlands 'schakel modem op de telefoonlijn'. Bij dergelijke modems is dit signaal natuurlijk wel noodzakelijk.

In veel gevallen worden ook deze lijnen echter kortgesloten. Op zo'n manier kan men de kabel alweer vereenvoudigen, hetgeen in de meeste gevallen alleen maar gunstig is. Om ook de DTR- en DSR-signalen, alsmede het CD-signaal, te omzeilen dient men deze drie pennen onderling door te verbinden. In dat geval namelijk zal aan de zijde van de computer het DTR-signaal, zodra het door de computer wordt ingeschakeld, zowel de DSR als de CD optrekken. Daardoor 'meent' de computer dat er zowel een DSR – modem gereed – als CD – draaggolf aanwezig – signaal door het modem gegeven worden, zelfs al zou de andere kant van de kabel los liggen.

Anderzijds gebeurt er aan de kant van het modem iets vergelijkbaars; zodra het modem middels het DSR-signaal wil aangeven dat het aanstaat zal het dat signaal onmiddellijk terugkrijgen als DTR. De CD is natuurlijk niet van belang, dat signaal wordt door het modem zelf opgewekt. De reden echter om ook aan de modemzijde deze pen door te verbinden is simpel; want zo voorkomt men dat er kabels rondslingeren met een modem-kant en een computer-kant. Dat zou maar schier onoplosbare problemen kunnen veroorzaken, tot men opeens bedenkt dat de kabel gewoon andersom moest . . .

Ring Indicator

Op pen nummero 22 – F op de 64 – komen we het Ring Indicator signaal tegen. Met dit signaal kan het modem aangeven dat de telefoonlijn, waarop het aangesloten is, overgaat.
Desgewenst kan de computer dan 'besluiten' iets met die oproep te doen.
Op de Commodore echter is deze lijn op zich wel aanwezig op de user-poort, en intern zelfs doorverbonden met het verantwoordelijke IC, maar in de Kernal RS232C-routines wordt er verder niets mee gedaan. Wie er gebruik van maken wil zal daar zelf een stuk akelig lastig programmeerwerk voor moeten verzetten.

Op zich is dat echter niet noodzakelijk. Verreweg de meeste computers en programma's maken geen gebruik van dit eigenlijk verouderde controle-signaal. De tegenwoordige modems zijn namelijk mans genoeg om zelf de telefoon aan te nemen, als men daar tenminste behoefte aan heeft. De computer staat gewoon in een communicatie-programma te wachten tot het auto-answer modem de oproep heeft aangenomen en de verbinding gelegd is. Kortom, ook deze lijn is niet

van essentieel belang en kan op de 64 zelfs niet eens zonder meer gebruikt worden.

Dat waren ze dan, alle signalen in die uitgebreide RS232C-interface die voor Commodore-bezitters in eerste instantie van belang zijn. Alle andere lijnen kunnen op een standaard-Commodore niet eens aangesloten worden, laat staan dat ze enig gebruik hebben. De klok-signalen bijvoorbeeld spelen alleen maar een rol bij de zogenaamde synchrone transmissie, een transmissie-wijze die feitelijk nergens meer toegepast wordt. Bijna alle computer-communicatie verloopt tegenwoordig asynchroon, zeker als het om onze toepassingen gaat.
Ook het tweede kanaal, zoals dat nog in de RS232C-definitie voorkomt, is tegenwoordig zwaar achterhaald. We gebruiken bij full-duplex weliswaar twee kanalen, maar die werken op een heel andere manier. Naar het randapparaat toe verloopt dat gewoon via het hoofdkanaal, de TD en RD lijnen. Hoe het precies op de uiteindelijke telefoonlijn in zijn werk gaat zullen we later nog wel eens bekijken.
Maar met het RS232C backward-channel – zoals dat tweede kanaal in feite heet – heeft het allemaal niets meer te maken. Gelukkig maar, want dergelijke oplossingen waren meestal erg ingewikkeld in het gebruikt.

Het enige wat eigenlijk echt spijtig te noemen valt is het feit dat we ook de test-signalen ontberen op de Commodore versie van de RS2332C-interface. Maar in veel gevallen kunnen we daar nog wel wat aan doen, middels bijvoorbeeld schakelaars op het modem, zoals we in het hoofdstuk over modem-techniek nog zullen zien.
Kortom, in de praktijk hebben we alleen te maken met de volgende signalen: TD, RD, RTS, CD, DTR, DSR en Signal Ground. Van de vele pennetjes zijn er nu nog maar een achttal overgebleven. En daarvan kunnen we er in veel gevallen nog wel wat afsnoepen.

Kabels
De ideale RS232 kabel bevat in ons geval dus alles bij elkaar 8 aders. Met zo'n kabel maken we gebruik van alle mogelijkheden die RS232C ons op de C64 te bieden heeft. Maar mocht dat achttal toch nog teveel zijn, dan kunnen we dat aantal zonder veel problemen terugbrengen to slechts drie. Als we namelijk een beide zijden de RTS en de CTS doorverbinden, evenals de CD, DTR en DSR, dan hebben we nog maar drie adertjes nodig. We gebruiken dan alleen de beide datalijnen TD en RD, alsmede de gemeenschappelijke aardleiding van die twee. Met zo'n eenvoudig kabeltje zullen we heel vaak al uitstekende resultaten kunnen bereiken. Alle andere zaken worden dan via de software afgehandeld, de handshake wordt dan via een paar speciale codes tot stand gebracht. Er kleven echter wel wat nadelen aan zo'n benadering, want omdat we alle andere ingebouwde controle's te slim af zijn kunnen we er ook geen gebruik meer van maken. Het is dan aan ons om er voor te zorgen dat zowel het modem als de computer inderdaad aan staan, want de controlesignalen die daar gewoon voor gebruikt worden hebben we buiten spel gezet.
Echter, omdat de apparaten vrij eenvoudig zijn en bovendien keurig naast elkaar op tafel staan is dat best te doen.

6/2.4

Modems

Inhoud

6/2.4.1

Inleiding

Bij computer-communicatie gaat het er uiteindelijk om om contact te kunnen leggen met een andere computer en daar informatie mee uit te kunnen wisselen. Dat daar wel het een en ander bij komt kijken zal u zo langzaam maar zeker wel duidelijk geworden zijn.

Met de in het vorige hoofdstuk besproken interfaces alleen zijn we er nog niet helemaal, als we ook in staat willen zijn om via het telefoonnet verbindingen te leggen. Daar komt dan nog een modem – MOdulator/DEModulator – bij. Dat modem vertaalt uiteindelijk de via de RS232C door de computer verzonden informatie tot iets wat ook werkelijk per telefoonlijn verstuurd kan worden, en vertaald binnenkomende informatie weer terug naar iets wat daarna weer via de RS232 naar de computer kan.

Nu bestaan er ook voor modems een aantal standaarden, terwijl er een met de dag groeiend aantal merken en modellen op de markt is. Sommigen daarvan voldoen slechts aan een enkele standaard, andere types kunnen volgens meerdere standaarden werken en hebben in principe dan ook een groter communicatie-bereik dan hun simpeler neefjes. Dat communicatie-bereik moet u in dit geval overigens niet begrijpen als een te overbruggen afstand. Het betekent veeleer dat u met zo'n uitgebreider modem meer andere computer-systemen kunt bereiken.
Stel bijvoorbeeld dat u een al wat ouder type modem bezit, eentje die alleen met 300 baud kan werken, dan zult u daar nooit en te nimmer een Viditel-achtig systeem mee kunnen opbellen. Dat werkt namelijk met een dubbele baud-snelheid, waarbij voor het contact van de hoofdcomputer een snelheid van 1200 baud gebruikt wordt en de opbeller met een veel lagere snelheid – 75 baud – informatie verstuurt.

Kortom, het is in zo'n geval best een prettig idee om een modem te hebben staan dat zowel 300-300 baud als 1200-75 baud aankan.

Over dergelijke zaken zullen we het in dit gedeelte gaan hebben. Wat voor modemstandaarden bestaan er, wat kunnen ze en hoe belangrijk zijn ze eigenlijk. Kortom, onmisbaar als u de aanschaf van een modem overweegt, maar ook zeer zinvol als er al eentje hebt staan. Want in dat geval zult u hier kunnen lezen wat uw modem allemaal wel en niet kan.

6/2.5

Communicatie-programma's

Inhoud

6/2.5.1

Inleiding

Goed, u hebt een Commodore 64 of 128, u hebt een RS232C-interface en u hebt een eersteklas modem. Dan staat u nu niets meer in de weg om de zaak aan te sluiten en eens contact te zoeken met het een of andere bulletin-board, denkt u?
Mis. Er ontbreekt nog een heel belangrijke schakel in de keten, namelijk het programma dat uw computer tot een terminal omtovert. Net zoals dat het geval is bij bijvoorbeeld tekstverwerking is uw computer namelijk wel op zich prima geschikt voor data-communicatie, maar 'weet' de machine niet zonder meer hoe dat nu precies in zijn werk moet gaan. Er is nog een programma voor nodig.

In de Kernal zitten echter gelukkig al de nodige RS232C-routines, die het mogelijk maken om desgewenst zelfs in BASIC een klein terminal-programmaatje te schrijven. Maar in feite is dat echter geen goede oplossing, tenzij om er eens wat mee te experimenteren. BASIC is namelijk veel en veel te langzaam voor dat soort zaken.
Zeker bij communicatie-programma's is snelheid van het grootste belang, anders kunt u tot Sint-Juttemis wachten tot uw elektronische post eindelijk eens een keertje aangekomen is.
In dit hoofdstuk zullen we aan die communicatie-programma's de nodige aandacht besteden. Het zelf schrijven van een klein experimentje in BASIC maar ook het gebruik van allerlei weid verspreide programma's als Vip-Terminal zal aan de orde komen.

Maar als eerste onderdeel van dit hoofdstuk presenteren we, met enige gepaste trots, T64. T64 is een volwassen terminalprogramma, waarmee u vele andere computers kunt bereiken. Om het in te tikken is even werk, de listing beslaat de nodige pagina's, maar het is zeker de moeite waard. Bovendien bevat het programma enige truuks die ook een Vip-Terminal in de schaduw stellen.

6/2.5.2

T64

Communicatie-programmatuur voor de 64 en de 128 bestaat er in vele maten en soorten. Vanaf simpele BASIC-listinkjes tot en met hele lappen machinetaal vol met toeters en bellen, er is overvloed aan keus.
Toch willen we in 'Van BASIC tot Machinetaal' nog een nieuw programma aan u voorstellen, namelijk ons eigen T64. Hoewel T64 nauwelijks van extra's voorzien is blijkt het namelijk in de praktijk een van de prettigste terminal-emulator programma's die we kennen. Het programma is gedeeltelijk in BASIC geschreven en gedeeltelijk in machinetaal, waarbij het hele zuivere communicatie-gedeelte natuurlijk in ML is. Gezien de eisen die er aan de snelheid gsteld worden is dat noodzakelijk.

Aanmaken

Het intikken van de listing is een flinke klus, daar we het hele programma, inclusief het BASIC-gedeelte, in de DATA-regels hebben moeten verwerken van een laad-programma. Daar was jammer genoeg geen andere goede oplossing voor te vinden, gezien de uiterst getruukte struktuur ervan. Dat laad-programma is echter voorzien van een hele goede extra beveiliging; als u per ongeluk een foutje gemaakt heeft in een van de DATA's, dan zal de lader dat opmerken en u verzoeken om de betreffende regel te verbeteren. U hoeft dus niet – mocht er onverhoopt een foutje insluipen onder het intikken – al die regels met DATA door te gaan spitten op zoek naar een enkel omgedraaid cijfer.

Schrijf echter voor de veiligheid het produkt van uw noeste intik-arbeid wel weg naar cassette of diskette voordat u het RUN-commando geeft. Veiligheid is de moeder van de porceleinkast, en bovendien zult u eerst nog eventjes iets in de systeemvariabelen van uw machine moeten aanpassen.

Nadat u de lader veilig weggeschreven heeft en voordat u middels RUN uw exemplaar van T64 gaat aanmaken moet u namelijk eerst het BASIC start-adres een stuk hoger zetten, door nadat u de computer even uit- en weer aangezet heeft de volgende regel, gevolgd door een RETURN, in te geven:

POKE 44,25: POKE 6400,0: NEW

Nu de lader weer inladen en starten met RUN. Als alles goed gaat zult u na een tijdje instructies op uw scherm krijgen hoe u de zojuist aangemaakte copie van T64 veilig op tape of disk kunt zetten. Eventuele fouten in de DATA worden, zoals reeds gezegd, tijdens het laadproces ontdekt waarna u ze nog eventjes mag verbeteren.

Klaar

Als u eenmaal uw kopie weggeschreven hebt, kunt u voor de grap T64 eens LISTen. Juist ja, er verschijnt slechts een enkele korte BASIC-regel. Maar achter die regel zit meer dan u misschien wel denkt. Als u T64 namelijk gaat RUNnen, dan verschijnt er een keurig scherm met daarop een korte gebruiksaanwijzing, waar na u weer – zoals altijd – READY ziet verschijnen. Nu de LIST-opdracht intikken levert opeens een veel uitgebreidere BASIC-listing op, waarin u allerlei opties van T64 naar eigen wens kunt instellen.

Om T64 te kunnen gebruiken is het natuurlijk niet nodig om te begrijpen wat er zonet gebeurd is, maar het is toch wel aardig om te weten. De structuur van T64 is namelijk vrij ingewikkeld, het programma bestaat uit een stuk machinetaal en een stuk BASIC. Of liever gezegd, uit twee stukken BASIC, namelijk dat allereerste hele korte regeltje en het veel uitgebreider instel-programma dat u pas na het RUNnen van die ene regel te zien krijgt.
Het machinetaal-gedeelte nu zit tussen die twee BASIC-delen in, een wat ongebruikelijke plek. Als u T64 laadt, dan laadt u eigenlijk drie verschillende stukken die in een enkel programma verenigd zijn. Helemaal voorop staat dat ene BASIC-regeltje, onmiddellijk gevolgd door de machinetaal, die eigenlijk ook weer uit twee delen bestaat. Boven die ML komt weer een tweede BASIC-gedeelte, wat u echter bij een eerste keer LISTen niet te zien krijgt. Die allereerste BASIC-regel wordt namelijk afgesloten op zo'n manier dat de computer denkt dat het inderdaad het einde van het programma is.
De eerste keer dat u RUN intoetst wordt die regel uitgevoerd, die echter alleen maar onmiddellijk naar een heel klein ML-routinetje springt.

Die kleine ML-routine nu veranderd in feite alleen maar het BASIC start-adres, zodat het tweede BASIC-programma opeens tevoorschijn komt en geRUNd wordt. De hoofdmoot van de ML – het gedeelte dat de eigenlijke communicatie voor zijn rekening neemt – zit nu veilig onder de veranderde BASIC-start. Kortom, u kunt naar hartelust gaan uitbreiden in de BASIC, om bijvoorbeeld uw eigen standaard-instellingen op te nemen, zonder dat daardoor die machinetaal overschreven kan worden.

Vrij

Het grote voordeel van deze techniek is echter dat er nu geen gebruik gemaakt wordt van het geheugen bereik vanaf C000 (49152) tot en met CFFF (53247), waar bijna alle andere machinetaal-routines ondergebracht worden. Met andere woorden, u kunt T64 desgewenst combineren met uw eigen hulpprogramma's, zonder dat ze elkaar dwars zitten.

Mogelijkheden

In het tweede BASIC-programma treft u, behalve een aantal regels die het startscherm afdrukken, ook een hele serie programmaregels met daarop POKE-opdrachten aan. Sommige van deze regels zijn met behulp van een REM-commando buiten werking gesteld, andere regels niet.
Door nu bij de gewenste opties de REM's te verwijderen en op andere regels juist weer een REM toe te voegen kunt u uw eigen opties kiezen. U zou zelfs dit BASIC-programma kunnen uitbreiden met één of meer menu-schermen, waarop de gebruiker zijn of haar wensen duidelijk kan ma-

2.5 Communicatie-programma's

ken, waarna u de POKE's voor de gemaakte keuzes in het BASIC-gedeelte opneemt.

T64 heeft heel wat mogelijkheden te bieden, zoals bijvoorbeeld een ruime keuze aan communicatie-snelheden. U kunt kiezen uit 110, 300, 1200 of 2400 baud. Die 2400 baud insteling wordt weliswaar door meer terminal-programma's geboden maar blijkt dan in de praktijk al snel moeilijkheden op te leveren, aangezien de Kernal-routines bij deze snelheid niet goed meer functioneren. In T64 is hier rekening mee gehouden.

Verder kunt u kiezen uit een tweetal softwarematige protocollen, manieren dus waarmee er voor gezorgd kan worden dat de zender en de ontvanger elkaar kunnen bijbenen. Welke manier u moet kiezen hangt af van de mogelijkheden van het systeem waar u kontakt mee zoekt, meestal echter zal de standaard ENQ/ACK instelling het beste voldoen, mits de andere machine hier ook mee werkt.
Dit protocol is speciaal bedoeld voor gebruik bij de verzending van grote hoeveelheden gegevens naar de Commodore.
Het tweede protocol, CR/DC1, is bestemd voor transport vanaf de 64 of de 128 naar de andere computer, maar ook hier geldt dat die machine er wel mee moet kunnen werken.

Meer informatie over deze en andere protocollen zult u in een van de toekomstige afleveringen kunnen vinden.

Weer een andere POKE biedt u de mogelijkheid om de locale echo aan of juist uit te zetten. Als deze echo aanstaat, dan zullen alle ingetoetste tekens niet alleen verzonden worden maar ook meteen op het scherm verschijnen. In het andere geval worden alleen de tekens die via de RS232 ontvangen worden op het scherm gezet. Voor het moment moeten we volstaan met u aan te raden om deze echo aan te laten staan, tenzij u alle ingetikte tekens dubbel ziet verschijnen. In dat geval heeft u namelijk verbinding met een systeem dat zelf alle ontvangen tekens meteen weer terugzendt, hetgeen die verdubbeling ten gevolge heeft. Het doel overigens van dat meteen weer terugzenden is om heel zeker te kunnen zijn dat het teken dat u ingetikt heeft ook inderdaad als zodanig aan de andere kant gezien is.

Functie-toetsen
Binnen het BASIC-deel van T64 kunt u desgewenst een tekst toewijzen aan de variabelen F1$ tot en met F8$. Deze teksten worden dan, als u terwijl u in communicatie-mode bent de betreffende functie-toets indrukt, meteen op het scherm gezet. Stel dat u regelmatig een bulletin-board opbelt waarin u als gebruikersnaam 'bitboer' gebruikt, dan kunt u deze tektst onder een functie-toets opnemen.
De BASIC-regel:

```
F1$="bitboer"+CHR$(13)
```

is genoeg, als u nu de F1 indrukt verschijnt niet alleen de string bitboer op het scherm, maar deze wordt dan ook nog meteen verzonden, doordat u er CHR$(13) – de teken-code voor RETURN – aan vast geplakt heeft. Werkelijk ideaal, om op zo'n manier allerlei vaste teksten en veel gebruikte commando's met een enkele toetsdruk te kunnen verzenden.

Helemaal achterin het T64 BASIC-programma vindt u nog, achter een REM-

commando, de opdracht SYS 2072. Dit is het commando waarmee u de eigenlijke communicatie – de machinetaal dus – opstart. Door deze REM opdracht te verwijderen kan men T64 automatisch laten doorstarten, na de allereerste RUN.

Bewaren
Om de in de BASIC doorgevoerde veranderingen te kunnen bewaren op cassette of diskette kan men echter niet volstaan met het zonder meer wegschrijven van T64. Immers, de startwijzer van BASIC is verplaatst, zodat een dergelijke SAVE alleen dit stuk BASIC zou bewaren. We moeten echter de hele structuur, dus allebei de BASIC-programma's en de ML die daar weer tussen staat wegschrijven.
Dat gaat echter heel simpel, door eerst met:

POKE 43,1: POKE44,8

de start van de BASIC weer op zijn oude waarde terug te zetten. Een LIST-commando toont nu weer alleen het eenregelige voor-programmaatje.
Met het commando

SAVE "naam",1

wordt echter wel het gehele, gewijzigde programma in een keer naar cassette weggeschreven, zodat het voor de volgende sessie inclusief de veranderde instellingen kan worden teruggeladen. Diskdrive-bezitters gebruiken natuurlijk:

SAVE "naam",8

Commando's
Na de SYS 2072 opdracht, of die rechtstreeks via het toetsenbord gegeven is of vanuit het BASIC-programma, komen we in de eigenlijke communicatie-mode van T64 terecht. Op dat moment zal alles wat we intikken rechtstreeks via de user-poort verzonden worden. Daarbij is T64 zo vriendelijk om voor ons de omzetting van de interne Commodore-tekens – door mensen die zich vaker met communicatie tussen Commodore-computers en andere merken vaak terecht 'PETSCII' genaamd – naar echte ASCII te verzorgen. Oftewel, een hoofdletter A verschijnt ook aan de andere kant als een hoofdletter A.

Op die regel dat alles meteen verzonden wordt zijn echter een paar heel nuttige uitzonderingen. Zo zal de RUN/STOP toets door T64 worden gezien als teken dat men wil ophouden met communiceren en terug wil naar de gewone normale BASIC. Voor alle veiligheid verschijnt echter eerst nog de vraag:
Really? (y/n)
Door daarna een y in te drukken kunt u het communicatie-gedeelte werkelijk afbreken; u bevindt zich dan weer in BASIC met het BASIC-programma in het geheugen. Eventueel opnieuw opstarten, na desgewenst wat opties gewijzigd te hebben, kunt u weer verder met de bekende SYS-opdracht.

Bestanden verzenden
De andere twee uitzonderingen – toetscombinaties die dus niet meteen verzonden worden – worden gevormd door de Commodore-toets samen met de letter r en de Commodore-toets samen met de letter s. Met de eerste combinatie wordt het ontvangen van een tekstfile ingeleid, de tweede combinatie biedt ons de mogelijkheid om een bestand te verzenden.

In beide gevallen verschijnt de tekst:
dev:name"code

2.5 Communicatie-programma's

waarmee T64 ons vraagt om wat gegevens.

Met dev wordt het apparaatnummer bedoeld, waarvan het bestand gelezen moet worden of waarheen het juist weggeschreven zal worden. Voor het verzenden kunnen we kiezen uit de apparaten 1 en 8, dus de recorder of de diskdrive; voor het ontvangen komt daar bovendien nog de mogelijkheid 4, de printer dus, bij.
Na dat apparaat-nummer dient, gescheiden door een dubbele punt, de bestandsnaam te volgen. Deze moet aan alle gebruikelijke eisen voldoen. Voor diskgebruikers geldt dat hierbij ook het bestandstype, s of p, moet worden aangegeven.

Tenslotte kunnen we na die naam nog opgeven of het bestand al dan niet naar echte ASCII moet worden omgezet. Standaard wordt dat wel gedaan, zodat u met T64 bijvoorbeeld heel gemakkelijk tekstbestanden met niet-Commodore computers kunt uitwisselen. Mocht u echter een bestand zonder die omzetting willen verzenden of ontvangen, dan kunt u dat aangeven door achter de bestandsnaam:
"S
op te nemen.

Tijdens het invullen van de gegevens voor het zenden of ontvangen van checkt T64 meteen of de opgegeven waarden wel kloppen. Zo zal een apparaatnummer 3 meteen leiden tot de melding I/O ERROR...00.
Zeker in het begin kan men zich dan wel eventjes achter de oren zitten te krabben, met de vraag wat er eigenlijk dan wel mis ging. Maar na enige oefening werkt het echter heel gemakkelijk, zoals wat voorbeelden zullen verduidelijken.

Om bijvoorbeeld een bestand van het type p – een programma-bestand dus, zoals sommige tekstverwerkers die aanmaken om hun teksten in op te slaan – vanaf disk te verzenden zonder code-omzetting dient men na het Commodore-toets – t commando het volgende in te tikken:
8:tekst,p"C

Het ontvangen van een bestand op cassette, met codeomzetting:
1:test,s

Overigens kan men ten alle tijden een eenmaal begonnen zend- of ontvangopdracht afbreken door middel van de RUN/STOPtoets.

Veranderde toetsen

Een andere extra van T64 is dat het programma in de communicatie-mode een aantal weinig gebruikte toetsen nieuwe betekenissen toekent, die tijdens het communiceren met sommige systemen bijna onmisbaar zijn.

Welke dit zijn staat in de onderstaande tabel:

toets-combinatie	resultaat
shift + '+'	accolade-haakje openen
shift + '-'	accolade-haakje sluiten
Commodore + '-'	verticale balk
shift + ' '	tilde
shift + '*'	accent grave
pond-teken	backslash

Tenslotte

Hopelijk bent u net zo tevreden met T64 als wij dat zijn. Zelf hebben we het programma al langere tijd in gebruik, onder andere om allerlei tekst-bestanden met andere computers uit te kunnen wisselen, en dat gaat werkelijk uitstekend. De ingebouwde code-omzetting maakt T64 daar geknipt voor.

2.5 Communicatie-programma's

Maar ook voor allerlei andere toepasssingen zijn we steeds meer op T64 gaan vertrouwen. Vergeleken met andere programma's is T64 namelijk erg snel, als er bestanden die op disk of tape staan moeten worden verzonden of juist ontvangen. Waar bij andere programmatuur de snelheid van 1200 baud door de trage diskafhandeling – buffertje lezen, buffertje verzenden, enzovoorts – wel eens tot ruim onder de 600 baud terugloopt, daar blijft T64 uiterst rap. Natuurlijk moet ook T64 de bestanden per buffer lezen en schrijven, maar dat gaat niet gepaard met al te grote vertragingen.

Al met al een erg prettig programma voor allerlei communicatie-doeleinden, zonder al te veel toeters en bellen.

```
10 REM T64, WEKA'S TERMINAL-PROGRAMMA
20 REM
30 REM T64 BASICLADER
40 REM
50 IF PEEK(43)+256*PEEK(44)= 6401 THEN GOTO 70
60 PRINT "EERST BASIC-START ADRES AANPASSEN!": STOP
70 PRINT "■"
80 PRINT "■■■■■■■■T64, WEKA'S TERMINAL-PROGRAMMA"
90 PRINT "■■■■■■■■■■■■■■WORDT GEINSTALLEERD"
100 PRINT "■■■■■■■■■■■■■■EVEN GEDULD, AUB"
110 FOR I=2049 TO 5672 STEP 6
120 FOR J=0 TO 5
130 READ D
140 C=C+D*(J+1)
150 POKE I+J,D
160 NEXT J
170 READ CD: IF CD=C THEN C=0: GOTO 190
180 PRINT "FOUT IN DATAREGEL";I: STOP
190 NEXT I
200 PRINT "■■■■■■■■■■■■■■                    ■■■■■■■■■■■■■KLAAR■"
210 PRINT "■TIK NU IN:■"
220 PRINT "POKE43,1:POKE44,8:POKE45,34:POKE46,25"
230 PRINT "■DAARNA  T64 WEGSCHRIJVEN NAAR DISK MET:"
240 PRINT "SAVE ";CHR$(34);"T64";CHR$(34);",8"
250 PRINT "■VOOR CASSETTE WORDT DAT:"
260 PRINT "SAVE ";CHR$(34);"T64";CHR$(34);",1"
```

2.5 Communicatie-programma's

```
2049 DATA 13,8,10,0,158,50,1149
2055 DATA 48,54,51,58,138,0,1231
2061 DATA 0,0,169,19,160,17,1485
2067 DATA 133,43,132,44,96,76,1727
2073 DATA 168,8,62,10,65,10,795
2079 DATA 110,15,54,10,190,16,1388
2085 DATA 239,9,41,10,90,10,930
2091 DATA 169,10,38,112,255,255,3556
2097 DATA 0,255,255,5,6,13,1403
2103 DATA 17,222,160,0,96,48,1709
2109 DATA 24,12,6,3,0,238,1506
2115 DATA 160,16,56,108,198,130,2562
2121 DATA 0,0,0,246,160,0,1784
2127 DATA 0,0,0,0,0,0,0
2133 DATA 255,222,161,12,24,48,1638
2139 DATA 96,48,24,12,0,238,1740
2145 DATA 161,48,24,12,6,12,479
2151 DATA 24,48,0,254,161,96,2517
2157 DATA 48,24,0,0,0,0,96
2163 DATA 0,6,162,60,102,102,1860
2169 DATA 126,102,102,102,0,214,2328
2175 DATA 162,28,48,48,96,48,1322
2181 DATA 48,28,0,222,162,24,1946
2187 DATA 24,24,24,24,24,24,504
2193 DATA 24,230,162,56,12,12,1326
2199 DATA 6,12,12,56,0,238,1718
2205 DATA 162,114,214,156,0,0,1656
2211 DATA 0,0,0,0,0,32,192
2217 DATA 231,255,162,32,173,47,2502
2223 DATA 8,208,2,162,205,142,2955
2229 DATA 146,10,142,67,14,162,1902
2235 DATA 205,173,48,8,208,2,1779
2241 DATA 162,174,142,243,9,169,2967
2247 DATA 12,160,12,141,24,3,1070
2253 DATA 140,25,3,169,0,162,1847
2259 DATA 63,202,157,64,172,208,3302
2265 DATA 250,202,142,115,172,169,3414
2271 DATA 177,141,248,171,169,14,2816
2277 DATA 141,39,208,169,6,141,2395
2283 DATA 32,208,141,33,208,169,3057
2289 DATA 2,162,45,160,8,32,1333
2295 DATA 189,255,169,2,170,160,3024
2301 DATA 0,32,186,255,162,173,3490
2307 DATA 169,0,133,247,134,248,3714
2313 DATA 232,133,249,134,250,32,3223
2319 DATA 192,255,169,0,141,0,1914
2325 DATA 212,169,96,141,1,212,2679
2331 DATA 169,10,141,5,212,141,2538
2337 DATA 6,212,169,5,141,24,1806
2343 DATA 212,120,165,1,41,250,2656
2349 DATA 133,1,169,0,133,251,2813
2355 DATA 133,253,169,216,162,160,3780
2361 DATA 133,252,134,254,160,0,2855
2367 DATA 177,251,145,253,200,208,4374
2373 DATA 249,230,252,230,254,165,4645
2379 DATA 254,201,164,208,239,169,4189
2385 DATA 56,162,8,133,251,134,2995
2391 DATA 252,160,0,177,251,133,3333
2397 DATA 253,200,177,251,133,254,4377
2403 DATA 5,253,240,24,208,4,2391
2409 DATA 177,251,145,253,200,192,4278
2415 DATA 10,208,247,24,216,152,3255
2421 DATA 101,251,133,251,144,221,4052
2427 DATA 230,252,208,217,169,0,3071
2433 DATA 162,160,133,251,134,252,4067
2439 DATA 162,164,133,253,134,254,4095
2445 DATA 160,0,177,251,73,255,3590
2451 DATA 145,253,200,208,247,230,4698
2457 DATA 252,230,254,165,254,201,4610
2463 DATA 168,208,237,165,1,9,2014
2469 DATA 6,133,1,88,169,168,2480
2475 DATA 141,24,208,169,168,141,3175
2481 DATA 136,2,173,0,221,41,2010
2487 DATA 252,9,1,141,0,221,2163
2493 DATA 169,142,32,210,255,169,3678
2499 DATA 8,32,210,255,32,68,2290
2505 DATA 229,162,255,142,138,2,2588
2511 DATA 32,52,11,169,16,36,1141
2517 DATA 162,24,240,1,56,169,2228
2523 DATA 0,42,141,21,208,162,2603
2529 DATA 2,32,198,255,32,228,3208
2535 DATA 255,201,0,240,74,108,2635
2541 DATA 37,8,201,32,176,51,1970
2547 DATA 205,52,8,208,14,162,2207
2553 DATA 2,32,201,255,173,53,2872
2559 DATA 8,32,WAIT10,255,76,210,3362
2565 DATA 9,205,55,8,208,5,1686
2571 DATA 162,255,142,50,8,201,2544
2577 DATA 7,208,6,32,421,845
2583 DATA 76,54,10,201,8,208,23STOP6
2589 DATA 4,169,20,208,7,2,2475
2595 DATA 13,208,16,32,207,11,1706
2601 DATA 72,162,0,32,201,255,3059
2607 DATA 104,32,210,255,32,52,2290
2613 DATA 11,32,255,11,208,6,1960
2619 DATA 108,27,8,32,211,10,1429
2625 DATA 174,49,8,240,5,174,2325
2631 DATA 50,8,240,135,162,0,2136
2637 DATA 32,198,255,32,228,255,3991
2643 DATA 201,0,240,242,108,41,2675
2649 DATA 8,201,7,208,6,32,1485
2655 DATA 42,11,76,169,10,201,2224
2661 DATA 178,208,3,108,35,8,1258
2667 DATA 201,174,208,3,108,31,1911
2673 DATA 8,201,133,144,10,201,2641
2679 DATA 141,176,6,32,89,11,1150
2685 DATA 76,210,9,201,14,240,2837
2691 DATA 37,201,8,208,2,169,2319
2697 DATA 20,72,162,0,32,201,2016
2703 DATA 255,104,72,32,210,255,3387
2709 DATA 32,52,11,104,201,147,2472
2715 DATA 208,3,76,210,9,201,2533
2721 DATA 20,208,2,169,8,32,1350
2727 DATA 229,11,72,162,2,32,1317
2733 DATA 201,255,104,72,32,210,2731
2739 DATA 255,104,205,54,8,208,2582
2745 DATA 5,162,0,142,50,8,1195
2751 DATA 76,210,9,18,210,69,2059
```

2.5 Communicatie-programma's

```
2757 DATA 65,76,76,89,63,32,1308
2763 DATA 40,89,47,78,41,146,1752
2769 DATA 32,0,32,204,255,162,3191
2775 DATA 255,142,50,8,232,189,3015
2781 DATA 194,10,240,6,32,210,2378
2787 DATA 255,232,208,245,32,42,2735
2793 DATA 11,32,255,11,240,251,3590
2799 DATA 32,228,255,201,0,240,3497
2805 DATA 249,201,89,208,58,104,2664
2811 DATA 104,169,9,32,210,255,3177
2817 DATA 169,0,141,21,208,141,2562
2823 DATA 138,2,173,0,221,9,1820
2829 DATA 3,141,0,221,169,20,2134
2835 DATA 141,24,208,169,4,141,2355
2841 DATA 136,2,165,1,9,1,690
2847 DATA 133,1,32,68,229,120,2368
2853 DATA 32,21,253,88,96,162,2637
2859 DATA 32,142,4,212,232,142,3188
2865 DATA 4,212,96,56,32,240,2540
2871 DATA 255,138,24,105,5,10,1108
2877 DATA 10,10,141,1,208,152,2409
2883 DATA 56,233,40,176,1,152,2263
2889 DATA 24,105,3,10,10,10,393
2895 DATA 141,0,208,169,0,42,1693
2901 DATA 141,16,208,96,170,189,3165
2907 DATA 10,11,133,70,162,70,1941
2913 DATA 134,69,165,1,9,1,822
2919 DATA 133,1,32,231,176,165,3025
2925 DATA 1,41,254,133,1,169,2396
2931 DATA 0,160,3,145,71,173,2302
2937 DATA 42,3,172,43,3,141,1597
2943 DATA 156,11,140,157,11,169,2295
2949 DATA 151,160,11,141,42,3,1296
2955 DATA 140,43,3,96,177,179,2578
2961 DATA 181,183,178,180,182,184,3815
2967 DATA 165,153,240,3,76,62,1955
2973 DATA 241,160,0,177,71,170,2644
2979 DATA 200,177,71,133,251,200,3754
2985 DATA 177,71,133,252,200,138,3554
2991 DATA 209,71,240,14,177,71,2438
2997 DATA 168,177,251,170,200,152,3867
3003 DATA 160,3,145,71,138,96,2151
3009 DATA 173,156,11,172,157,11,2057
3015 DATA 141,42,3,140,43,3,1027
3021 DATA 176,204,201,65,144,17,2269
3027 DATA 201,91,176,3,73,128,2056
3033 DATA 96,201,97,144,6,201,2601
3039 DATA 123,176,2,73,32,96,1509
3045 DATA 10,176,11,74,201,65,2086
3051 DATA 144,17,201,91,176,13,2103
3057 DATA 144,9,74,201,65,144,2377
3063 DATA 4,201,91,144,2,9,1319
3069 DATA 32,96,120,169,127,141,2741
3075 DATA 0,220,32,188,246,88,3046
3081 DATA 76,225,255,72,138,72,2701
3087 DATA 152,72,169,127,141,13,2094
3093 DATA 221,172,13,221,48,15,1818
3099 DATA 32,188,246,32,255,11,2615
3105 DATA 208,78,32,252,10,120,2238
3111 DATA 76,86,254,152,172,1,2484
3117 DATA 221,45,161,2,170,41,1898
3123 DATA 1,240,40,173,0,221,2619
3129 DATA 41,251,5,181,141,0,1987
3135 DATA 221,173,161,2,141,13,1841
3141 DATA 221,138,41,18,240,13,1970
3147 DATA 41,2,240,6,32,125,1699
3153 DATA 12,76,88,12,32,178,1704
3159 DATA 12,32,187,238,76,113,2647
3165 DATA 12,138,41,2,240,6,1655
3171 DATA 32,125,12,76,113,12,1259
3177 DATA 138,41,16,240,3,32,1435
3183 DATA 178,12,173,161,2,141,2221
3189 DATA 13,221,104,168,104,170,2979
3195 DATA 104,64,198,168,240,5,2728
3201 DATA 152,74,102,170,96,169,2780
3207 DATA 0,141,15,221,173,1,2082
3213 DATA 221,74,176,4,169,7,1800
3219 DATA 133,170,165,170,42,74,2302
3225 DATA 172,155,2,145,247,200,3503
3231 DATA 140,155,2,173,161,2,1965
3237 DATA 9,16,41,253,141,13,1959
3243 DATA 221,9,128,141,161,2,2004
3249 DATA 96,173,149,2,141,6,1638
3255 DATA 221,173,150,2,141,7,1772
3261 DATA 221,169,17,141,15,221,2575
3267 DATA 169,18,77,161,2,141,1936
3273 DATA 161,2,173,153,2,141,2152
3279 DATA 6,221,173,154,2,141,2439
3285 DATA 7,221,162,10,134,168,2653
3291 DATA 96,0,162,1,142,21,1422
3297 DATA 208,202,142,24,212,32,2386
3303 DATA 52,11,173,220,12,240,2973
3309 DATA 1,96,169,19,141,220,2801
3315 DATA 12,133,158,32,23,240,2435
3321 DATA 172,158,2,204,157,2,2107
3327 DATA 208,251,216,165,162,105,3458
3333 DATA 18,168,173,15,221,74,2482
3339 DATA 176,245,196,162,208,246,4418
3345 DATA 96,173,220,12,240,15,2440
3351 DATA 169,0,141,220,12,32,1724
3357 DATA 204,255,169,17,133,158,2902
3363 DATA 32,23,240,162,5,142,2323
3369 DATA 24,212,76,52,11,32,1131
3375 DATA 255,11,208,8,104,104,2077
3381 DATA 169,0,133,144,56,96,2000
3387 DATA 32,228,255,201,0,240,3497
3393 DATA 236,24,96,13,18,68,1122
3399 DATA 69,86,58,78,65,77,1514
3405 DATA 69,34,67,79,68,69,1408
3411 DATA 32,0,32,204,255,162,3191
3417 DATA 0,134,183,232,134,186,3531
3423 DATA 169,14,133,184,141,51,2343
3429 DATA 8,162,0,160,175,134,2651
3435 DATA 187,132,188,32,42,11,1419
3441 DATA 162,0,240,4,32,210,2318
3447 DATA 255,232,189,68,13,208,2871
3453 DATA 247,165,212,9,1,133,2052
3459 DATA 212,32,46,13,201,49,1765
```

2.5 Communicatie-programma's

```
3465 DATA 240,8,201,52,240,4,2291
3471 DATA 201,56,208,13,32,210,2409
3477 DATA 255,165,215,56,233,48,2907
3483 DATA 133,186,32,46,13,201,2056
3489 DATA 58,208,36,32,210,255,3290
3495 DATA 32,46,13,201,34,240,2577
3501 DATA 30,201,13,240,61,201,2942
3507 DATA 20,208,10,166,183,240,3485
3513 DATA 237,202,134,183,76,164,3139
3519 DATA 13,166,183,157,0,175,2572
3525 DATA 232,208,243,201,34,208,3599
3531 DATA 29,32,210,255,32,46,2179
3537 DATA 13,32,210,255,165,215,3842
3543 DATA 41,127,201,65,208,5,2228
3549 DATA 169,0,141,51,8,32,1028
3555 DATA 46,13,201,13,208,249,3261.
3561 DATA 201,13,240,2,56,96,1811
3567 DATA 32,210,255,165,212,41,3183
3573 DATA 254,133,212,32,221,12,2461
3579 DATA 32,192,255,8,72,32,1765
3585 DATA 18,13,104,40,176,6,1432
3591 DATA 165,144,24,240,1,56,1826
3597 DATA 96,13,18,211,69,78,1833
3603 DATA 68,32,198,73,76,69,1812
3609 DATA 32,69,78,68,69,68,1429
3615 DATA 146,13,0,0,165,153,1915
3621 DATA 240,3,76,39,14,104,1324
3627 DATA 104,32,204,255,165,212,3897
3633 DATA 9,1,133,212,166,183,3186
3639 DATA 236,0,175,240,22,232,3223
3645 DATA 189,0,175,134,183,72,2597
3651 DATA 32,210,255,104,174,51,2809
3657 DATA 8,240,3,32,229,11,1836
3663 DATA 108,43,8,173,34,14,1164
3669 DATA 240,54,173,40,14,172,2129
3675 DATA 41,14,141,42,3,140,1515
3681 DATA 43,3,173,198,15,172,2467
3687 DATA 199,15,141,27,8,140,1640
3693 DATA 28,8,169,14,32,195,1937
3699 DATA 255,165,212,41,254,133,3453
3705 DATA 212,32,42,11,162,0,1256
3711 DATA 189,14,14,240,6,32,1441
3717 DATA 210,255,232,208,245,108,4121
3723 DATA 33,8,32,221,12,169,2103
3729 DATA 0,133,183,141,0,175,2429
3735 DATA 162,14,32,198,255,32,2545
3741 DATA 62,241,166,144,176,2,2510
3747 DATA 240,14,138,41,192,141,2652
3753 DATA 34,14,208,21,32,242,2382
3759 DATA 14,76,87,14,166,183,2411
3765 DATA 232,157,0,175,142,0,1956
3771 DATA 175,134,183,224,255,208,4411
3777 DATA 218,169,0,133,183,32,2195
3783 DATA 18,13,76,44,14,13,596
3789 DATA 18,201,47,207,32,197,2731
3795 DATA 210,210,207,210,46,46,2597
3801 DATA 46,0,0,146,13,0,695
3807 DATA 13,18,73,78,84,69,1414
3813 DATA 82,82,85,80,84,69,1655
3819 DATA 68,46,46,46,146,13,1290
3825 DATA 0,32,42,11,32,204,1618
3831 DATA 255,32,255,11,240,7,2370
3837 DATA 173,34,14,41,63,240,2202
3843 DATA 15,162,0,240,4,32,1511
3849 DATA 210,255,232,189,223,14,3371
3855 DATA 208,247,240,37,165,144,3259
3861 DATA 74,74,74,74,170,189,2724
3867 DATA 81,15,141,218,14,165,2466
3873 DATA 144,41,15,170,189,81,2382
3879 DATA 15,141,219,14,162,0,1820
3885 DATA 240,4,32,210,255,232,3851
3891 DATA 189,204,14,208,247,32,2898
3897 DATA 221,12,169,14,32,195,2138
3903 DATA 255,32,18,13,32,255,2115
3909 DATA 11,240,251,165,212,41,3210
3915 DATA 254,133,212,76,52,11,1786
3921 DATA 48,49,50,51,52,53,1078
3927 DATA 54,55,56,57,65,66,1281
3933 DATA 67,68,69,70,13,18,863
3939 DATA 211,69,78,68,32,198,2203
3945 DATA 73,76,69,146,0,32,1208
3951 DATA 204,255,162,0,134,185,2980
3957 DATA 240,4,32,210,255,232,3851
3963 DATA 189,97,15,208,247,32,2687
3969 DATA 85,13,144,6,32,242,2179
3975 DATA 14,108,33,8,169,0,1206
3981 DATA 141,34,14,141,0,175,1865
3987 DATA 133,183,173,42,3,172,2233
3993 DATA 43,3,141,40,14,140,1542
3999 DATA 41,14,169,35,160,14,1600
4005 DATA 141,42,3,140,43,3,1027
4011 DATA 173,27,8,172,28,8,1127
4017 DATA 141,198,15,140,199,15,2227
4023 DATA 173,29,8,172,30,8,1141
4029 DATA 141,27,8,140,28,8,967
4035 DATA 108,33,8,0,0,32,390
4041 DATA 255,11,240,251,169,1,2852
4047 DATA 141,34,14,76,226,15,1775
4053 DATA 205,52,8,208,8,174,2249
4059 DATA 48,8,240,3,76,223,2514
4065 DATA 15,174,34,14,240,16,1817
4071 DATA 32,25,16,32,50,16,604
4077 DATA 169,0,141,34,14,133,1596
4083 DATA 144,108,33,8,174,51,1667
4089 DATA 8,240,3,32,207,11,1726
4095 DATA 166,183,232,157,0,175,2906
4101 DATA 134,183,142,0,175,224,3145
4107 DATA 255,176,11,170,165,212,3417
4113 DATA 9,1,133,212,138,108,2596
4119 DATA 39,8,32,221,12,162,2067
4125 DATA 14,32,201,255,162,0,2511
4131 DATA 240,28,232,189,0,175,2798
4137 DATA 32,210,255,164,144,176,3649
4143 DATA 2,240,15,32,242,14,1949
4149 DATA 32,114,16,174,34,14,1258
4155 DATA 208,3,108,33,8,96,1286
4161 DATA 236,0,175,208,223,162,3680
4167 DATA 0,134,183,142,0,175,2435
```

2.5 Communicatie-programma's

```
4173 DATA 174,34,14,208,8,72,1588
4179 DATA 32,18,13,104,108,39,1297
4185 DATA 8,96,13,18,210,69,1775
4191 DATA 67,69,73,86,69,32,1305
4197 DATA 198,73,76,69,32,69,1422
4203 DATA 78,68,69,68,146,13,1501
4209 DATA 0,32,221,12,169,14,1704
4215 DATA 32,195,255,32,18,13,1483
4221 DATA 32,204,255,32,42,11,1609
4227 DATA 162,0,240,4,32,210,2318
4233 DATA 255,232,189,91,16,208,2978
4239 DATA 247,173,224,15,172,225,3535
4245 DATA 15,141,37,8,140,38,1368
4251 DATA 8,173,198,15,172,199,3062
4257 DATA 15,141,27,8,140,28,1278
4263 DATA 8,32,255,11,240,251,3587
4269 DATA 96,13,18,210,69,67,1763
4275 DATA 69,73,86,69,32,198,2097
4281 DATA 73,76,69,146,0,162,1988
4287 DATA 1,134,185,202,240,4,2856
4293 DATA 32,210,255,232,189,174,4134
4299 DATA 16,208,247,32,85,13,1804
4305 DATA 144,6,32,242,14,108,1938
4311 DATA 33,8,169,0,141,0,1261
4317 DATA 175,133,183,141,34,14,1808
4323 DATA 173,37,8,172,38,8,1197
4329 DATA 141,224,15,140,225,15,2409
4335 DATA 169,213,160,15,141,37,2062
4341 DATA 8,140,38,8,173,27,1461
4347 DATA 8,172,28,8,141,198,2361
4353 DATA 15,140,199,15,169,200,2997
4359 DATA 160,15,141,27,8,140,1601
4365 DATA 28,8,108,33,8,0,540
4371 DATA 49,17,10,0,143,32,1020
4377 DATA 87,69,75,65,32,84,1374
4383 DATA 69,82,77,73,78,65,1536
4389 DATA 76,32,80,82,79,71,1529
4395 DATA 82,65,77,77,65,0,1076
4401 DATA 55,17,20,0,143,0,864
4407 DATA 65,17,30,0,137,32,1066
4413 DATA 50,52,48,0,71,17,755
4419 DATA 40,0,143,0,98,17,1061
4425 DATA 50,0,143,32,80,82,1499
4431 DATA 73,78,84,32,75,79,1458
4437 DATA 80,32,83,85,66,82,1555
4443 DATA 79,85,84,73,78,69,1597
4449 DATA 0,125,17,60,0,143,1399
4455 DATA 32,45,45,45,45,45,932
4461 DATA 45,45,45,45,45,45,945
4467 DATA 45,45,45,45,45,45,945
4473 DATA 45,45,45,0,131,17,1027
4479 DATA 70,0,143,0,178,17,1491
4485 DATA 80,0,153,32,34,147,1719
4491 DATA 17,29,29,14,197,206,2439
4497 DATA 209,45,193,195,203,32,2865
4503 DATA 212,69,82,77,73,78,1737
4509 DATA 65,76,32,69,77,85,1484
4515 DATA 76,65,84,79,82,32,1376
4521 DATA 32,32,86,83,54,46,1232
4527 DATA 200,34,0,224,17,90,1789
4533 DATA 0,153,32,34,17,17,725
4539 DATA 34,59,166,54,41,59,1425
4545 DATA 34,68,79,79,82,32,1325
4551 DATA 197,46,196,46,32,200,2421
4557 DATA 73,76,68,69,82,73,1553
4563 DATA 78,71,32,40,195,41,1697
4569 DATA 32,49,57,56,53,34,994
4575 DATA 0,255,17,100,0,153,1879
4581 DATA 166,49,50,41,34,17,850
4587 DATA 17,18,32,32,32,32,629
4593 DATA 32,32,32,32,32,32,672
4599 DATA 32,32,32,32,32,32,672
4605 DATA 34,0,28,18,110,0,740
4611 DATA 153,166,49,50,41,34,1241
4617 DATA 18,32,32,32,192,178,2334
4623 DATA 192,176,192,174,176,32,2888
4629 DATA 32,32,32,32,32,34,684
4635 DATA 0,57,18,120,0,153,1566
4641 DATA 166,49,50,41,34,18,856
4647 DATA 32,32,221,32,221,32,2184
4653 DATA 221,32,32,221,32,32,1617
4659 DATA 32,221,32,32,34,0,868
4665 DATA 86,18,130,0,153,166,2273
4671 DATA 49,50,41,34,18,32,690
4677 DATA 192,219,192,221,32,171,3276
4683 DATA 192,174,221,32,221,192,3588
4689 DATA 219,192,32,34,0,115,1525
4695 DATA 18,140,0,153,166,49,2034
4701 DATA 50,41,34,18,32,32,658
4707 DATA 221,32,221,32,221,32,2373
4713 DATA 221,173,192,219,32,221,3505
4719 DATA 32,32,34,0,144,18,1026
4725 DATA 150,0,153,166,49,50,1818
4731 DATA 41,34,18,32,32,32,643
4737 DATA 32,221,32,173,192,189,3356
4743 DATA 32,32,221,32,32,32,1239
4749 DATA 32,34,0,173,18,160,1842
4755 DATA 0,153,166,49,50,41,1496
4761 DATA 34,18,32,32,32,32,646
4767 DATA 32,32,32,32,32,32,672
4773 DATA 32,32,32,32,32,32,672
4779 DATA 34,0,205,18,170,0,1571
4785 DATA 153,32,34,17,17,40,712
4791 DATA 79,80,71,69,76,69,1522
4797 DATA 84,58,32,18,32,32,720
4803 DATA 32,32,32,32,32,32,672
4809 DATA 32,32,34,0,243,18,1521
4815 DATA 180,0,153,32,34,32,1129
4821 DATA 32,32,32,32,32,32,672
4827 DATA 32,32,32,18,32,211,1690
4833 DATA 217,211,50,48,55,50,1556
4839 DATA 32,32,146,32,83,84,1581
4845 DATA 65,82,84,69,78,0,1147
4851 DATA 35,19,190,0,153,32,1600
4857 DATA 34,32,32,32,32,32,674
4863 DATA 32,32,32,32,32,18,588
4869 DATA 32,195,61,43,82,32,1379
4875 DATA 32,32,32,32,146,32,1242
```

2.5 Communicatie-programma's

```
4881 DATA 66,69,83,84,65,78,1582
4887 DATA 68,32,79,78,84,86,1617
4893 DATA 65,78,71,69,78,0,1100
4899 DATA 83,19,200,0,153,32,1678
4905 DATA 34,32,32,32,32,32,674
4911 DATA 32,32,32,32,32,18,588
4917 DATA 32,195,61,43,83,32,1384
4923 DATA 32,32,32,32,146,32,1242
4929 DATA 66,69,83,84,65,78,1582
4935 DATA 68,32,86,69,82,90,1616
4941 DATA 69,78,68,69,78,0,1095
4947 DATA 122,19,210,0,153,32,1747
4953 DATA 34,32,32,32,32,32,674
4959 DATA 32,32,32,32,32,18,588
4965 DATA 32,210,213,206,47,211,3416
4971 DATA 212,207,208,32,146,32,2300
4977 DATA 83,84,79,80,80,69,1622
4983 DATA 78,34,0,164,19,220,2217
4989 DATA 0,153,32,34,32,32,890
4995 DATA 32,32,32,32,32,32,672
5001 DATA 32,32,18,32,32,32,630
5007 DATA 32,32,32,32,32,32,672
5013 DATA 32,146,32,32,32,32,900
5019 DATA 32,32,32,32,32,32,672
5025 DATA 41,34,0,170,19,230,2264
5031 DATA 0,142,0,176,19,240,2523
5037 DATA 0,143,0,210,19,250,2721
5043 DATA 0,143,32,32,32,32,862
5049 DATA 32,80,76,65,65,84,1509
5055 DATA 83,32,79,70,32,86,1340
5061 DATA 69,82,87,73,74,68,1564
5067 DATA 69,82,32,82,69,77,1464
5073 DATA 0,244,19,4,1,143,1424
5079 DATA 32,32,32,32,32,79,954
5085 DATA 77,32,79,80,84,73,1556
5091 DATA 69,83,32,84,69,32,1204
5097 DATA 83,69,76,69,67,84,1564
5103 DATA 69,82,69,78,0,250,2252
5109 DATA 19,14,1,143,0,27,784
5115 DATA 20,24,1,143,32,80,1283
5121 DATA 79,75,69,50,48,57,1218
5127 DATA 51,44,51,53,58,32,986
5133 DATA 82,69,77,32,32,49,1033
5139 DATA 49,48,32,66,65,85,1340
5145 DATA 68,0,57,20,34,1,495
5151 DATA 151,50,48,57,51,44,1142
5157 DATA 51,56,58,32,32,32,817
5163 DATA 32,32,143,32,32,51,1119
5169 DATA 48,48,32,66,65,85,1339
5175 DATA 68,0,90,20,44,1,644
5181 DATA 143,32,80,79,75,69,1552
5187 DATA 50,48,57,51,44,52,1053
5193 DATA 48,58,32,82,69,77,1395
5199 DATA 32,49,50,48,48,32,904
5205 DATA 66,65,85,68,0,123,1461
5211 DATA 20,54,1,143,32,80,1343
5217 DATA 79,75,69,50,48,57,1218
5223 DATA 51,44,52,50,58,32,977
5229 DATA 82,69,77,32,50,52,1141
5235 DATA 48,48,32,66,65,85,1339
5241 DATA 68,0,129,20,64,1,861
5247 DATA 143,0,162,20,74,1,1085
5253 DATA 151,50,48,57,54,44,1157
5259 DATA 50,53,53,58,32,32,899
5265 DATA 32,32,32,143,32,69,1338
5271 DATA 78,81,47,65,67,75,1426
5277 DATA 32,65,65,78,0,198,1857
5283 DATA 20,84,1,143,32,80,1403
5289 DATA 79,75,69,50,48,57,1218
5295 DATA 54,44,48,58,32,32,870
5301 DATA 32,82,69,77,32,69,1285
5307 DATA 78,81,47,65,67,75,1426
5313 DATA 32,85,73,84,0,233,2155
5319 DATA 20,94,1,143,32,80,1423
5325 DATA 79,75,69,50,48,57,1218
5331 DATA 55,44,50,53,53,58,1118
5337 DATA 32,82,69,77,32,67,1273
5343 DATA 82,47,68,67,49,32,1085
5349 DATA 65,65,78,0,9,21,600
5355 DATA 104,1,151,50,48,57,1341
5361 DATA 55,44,48,58,32,32,871
5367 DATA 32,32,32,32,32,143,1338
5373 DATA 32,67,82,47,68,67,1342
5379 DATA 49,32,85,73,84,0,1080
5385 DATA 53,21,114,1,151,50,1496
5391 DATA 48,57,56,44,50,53,1074
5397 DATA 53,58,32,32,32,32,745
5403 DATA 32,143,32,68,67,49,1315
5409 DATA 45,67,78,84,82,32,1351
5415 DATA 73,78,73,84,73,65,1539
5421 DATA 76,73,83,69,82,69,1571
5427 DATA 78,0,88,21,124,1,1052
5433 DATA 151,50,48,57,53,44,1152
5439 DATA 50,53,53,58,32,32,899
5445 DATA 32,32,32,143,32,76,1380
5451 DATA 79,67,46,32,69,67,1226
5457 DATA 72,79,32,65,65,78,1379
5463 DATA 0,126,21,134,1,143,1714
5469 DATA 32,80,79,75,69,50,1374
5475 DATA 48,57,53,44,48,58,1085
5481 DATA 32,32,32,82,69,77,1327
5487 DATA 32,76,79,67,46,32,1111
5493 DATA 69,67,72,79,32,85,1405
5499 DATA 73,84,0,132,21,144,1738
5505 DATA 1,143,0,168,21,154,1988
5511 DATA 1,143,32,32,32,32,863
5517 DATA 32,68,69,70,73,78,1488
5523 DATA 73,69,69,82,32,70,1326
5529 DATA 85,78,67,84,73,69,1557
5535 DATA 45,84,79,69,84,83,1644
5541 DATA 69,78,0,174,21,164,2010
5547 DATA 1,143,0,211,21,174,2280
5553 DATA 1,70,49,36,178,34,1526
5559 DATA 76,79,71,73,78,32,1321
5565 DATA 46,46,46,46,46,44,954
5571 DATA 80,65,83,83,87,79,1700
5577 DATA 82,68,34,170,199,40,2235
5583 DATA 49,51,41,0,234,21,1570
```

2.5 Communicatie-programma's

```
5589 DATA 184,1,70,51,36,178,1848
5595 DATA 34,76,83,32,45,76,1244
5601 DATA 83,34,170,199,40,49,1951
5607 DATA 51,41,0,240,21,194,2362
5613 DATA 1,143,0,18,22,204,1693
5619 DATA 1,143,32,83,89,83,1658
5625 DATA 50,48,55,50,58,69,1215
5631 DATA 78,68,58,32,82,69,1340
5637 DATA 77,32,65,85,84,79,1570
5643 DATA 32,83,84,65,82,84,1624
5649 DATA 0,26,22,214,1,141,1825
5655 DATA 56,48,0,32,22,224,1734
5661 DATA 1,128,0,0,0,12,329
5667 DATA 11,5,12,8,8,1,135
READY.
```

6/3

Printers

Inhoud

6/3.1

Inleiding

Weinig computer-randapparaten zijn zo nuttig en tegelijkertijd zo complex als de printer. Ze bestaan in vele soorten en maten, om nog maar te zwijgen over de keuze in aansluitings-mogelijkheden. Het kopen – en het programmeren – van een goede printer is beslist geen sinecure!

In dit hoofdstuk zullen daarom de diverse mogelijkheden en onmogelijkheden van printers onder de loep genomen worden. Daartoe beginnen we met een inleiding, waarin allerlei algemene printer-zaken aan de orde zullen komen, zoals welke typen printers zijn er, wat kunnen ze etcetera. Helemaal aan het einde van deze inleiding vindt u overigens wat belangrijke raadgevingen die u beslist moet lezen voor u zich een printer gaat aanschaffen.

Daarna zullen we de basisbegrippen van het gebruik van een printer op de Commodore 64 eens op een rijtje zetten. Vooral voor BASIC-programmeurs razend interessant.

Dan komen er een aantal printers met naam en toenaam aan de orde, te beginnen met de diverse modellen van Commodore zelf. Dit gedeelte zal in de toekomst nog verder uitgebreid worden met produkten van andere leveranciers, interfaces etcetera. U kunt desgewenst uw speciale wensen op dit gebied via de lezers-contactkaart achterin het boek kenbaar maken.

Iedere computerbezitter heeft ergens in het achterhoofd toch wel het idee om zich ooit nog eens een printer aan te schaffen. Want met een printer gaat er een wereld aan nieuwe mogelijkheden open.

Zo wordt het zelf wat programmeren een stuk eenvoudiger als men eventjes een listing op de printer kan uitdraaien. Op het beeldscherm verliest men toch maar al te snel het overzicht, terwijl men op zo'n papieren uitdraai makkelijk eventjes wat aantekeningen kan maken.

Als men een diskdrive bezit is een printer ook een hele verbetering. Het afdrukken van de directory's van de diverse schijven biedt nu eenmaal de mogelijkheid om snel en simpel een bepaald programmaatje terug te vinden, zonder dat men disk na disk moet afzoeken. Want bij een diskdrive schijnen nu eenmaal onveranderlijk vele tientallen schijven te horen met daarop een ware schatkist aan programmatuur. Zoveel vaak dat men een computer nodig heeft om bij te houden wat waar staat.

Allerlei fraaie grafische programma's kunnen ook dankbaar gebruik maken van een printer, nu kan men de schermontwerpen ook nog eens 'op de printer dumpen', zoals dat genoemd wordt. Tenminste, als dat programma en die printer het met elkaar eens zijn over hoe dat moet gebeuren.

3.1 Inleiding

Een zelfgeschreven programma krijgt juist door die printer nog veel meer mogelijkheden. Allerlei fraaie overzichten en lijsten aanmaken wordt met een printer een fluitje van een cent. Alweer, als men tenminste weet hoe de printer in kwestie moet worden aangestuurd.

Over lijsten en overzichten gesproken, al die fraaie boekhoudprogramma's en andere administratieve toepassingen op onze computers kunnen eigenlijk ook alleen maar bij de gratie van een printer hun nuttige werk verrichten. Een proefbalans op het scherm is nu eenmaal een wat lastige zaak, terwijl de belasting-inspecteur wel zeker een papieren overzicht vereist.

Of, om een laatste toepassing te noemen waarbij printers een hoofdrol spelen, tekstverwerking. Met sommige tekstverwerkers kan men hele lappen tekst schrijven en die op allerlei manieren laten afdrukken. Brede letters, smalle letters, onderstrepen, cursief drukken, er zijn legio mogelijkheden. Met een goede tekstverwerker en een goede printer kan men bijna niet van zetwerk te onderscheiden teksten maken, compleet met bijvoorbeeld superen subscript. Tenminste, als die printer en die tekstverwerker dat op dezelfde manier doen.

Verwarring

En daar zit hem nu net de kneep. Want het wil maar al te vaak voorkomen dat een programma en een printer met op zich allebei uitstekende mogelijkheden elkaar niet goed begrijpen. Jammer genoeg bestaan er geen universele standaarden als het om printers gaat. Daardoor zullen sommige printers wel goed functioneren met een bepaald programma en anderen weer niet.

Om maar een eerste voorbeeld te geven, het kan best gebeuren dat een printer die op een Commodore 64 wordt aangesloten 'zomaar' vergeet om het papier steeds een regeltje op te voeren als u een programma wilt listen op papier. Uiteindelijk houdt u dan een vel papier in uw hand met daarop, in een enkele regel, die hele listing, waarbij alles over elkaar heen gedrukt is. Erg leesbaar is zo'n zwarte streep niet, overigens.

Nu is dit probleem wat hierboven omschreven werd eigenlijk heel eenvoudig op te lossen. Wat er mis ging was dat de printer aan het einde van iedere af te drukken regel een tweetal 'stuurcodes' verwachtte, eentje die de printkop weer naar het begin van de regel brengt en eentje die het papier een regel laat opvoeren.
Een Commodore 64 echter stuurt normaal gesproken maar één stuurcode, namelijk die 'wagen terug'-code, die de printkop weer vooraan de regel zet. Volgens het besturingssysteem van de 64 moet de printer dan zelf zo slim zijn om daar uit af te leiden dat er dan ook een regelopvoer nodig is. Een simpele begripsverwarring tussen de computer en de printer, meer niet.

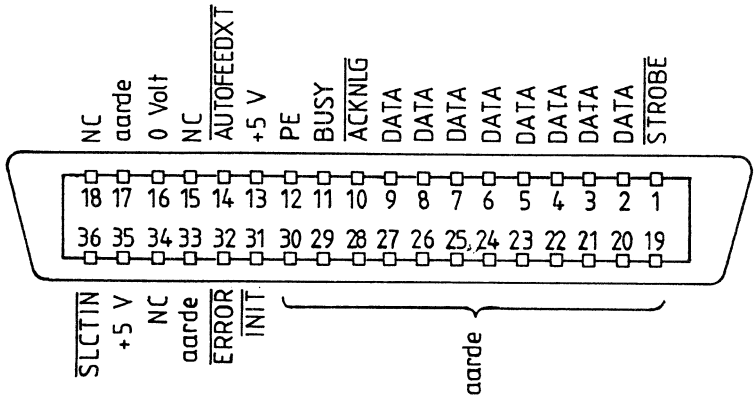
Aangezien dit specifieke probleem zich wel vaker voordoet zijn veel printers dan ook uitgerust met een klein schakelaartje, waarmee men kan instellen of er na een CR – zoals het wagen-terug (Carriage Return in het Engels) stuurteken officieel heet – al dan niet een automatische LF – Line Feed, regelop-

3.1 Inleiding

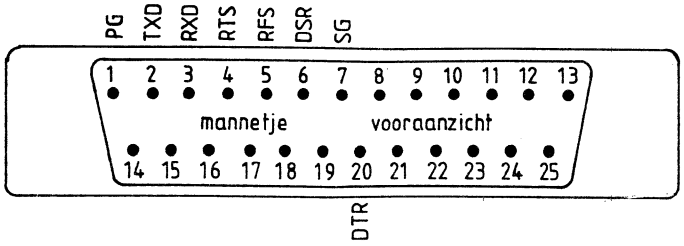
voer – gegenereerd moet worden. Simpelweg de schakelaar op de goede stand zetten en het probleem is de wereld uit.
Natuurlijk kan het de andere kant op ook mis gaan. Als de printer na iedere CR zelf een LF genereert maar de computer ook een LF stuurt krijgen we na iedere afgedrukte regel een extra witregel. Zonde van het papier en bovendien geen gezicht als men een wat officiële brief wil schrijven. Kortom, omzetten maar weer, die schakelaar.

Aansluitingen

Maar naast dergelijke 'kleine' probleempjes komt de toekomstige printerbezitter ook nog wel voor andere moeilijkheden te staan. Want ook de manier waarop een printer aan de computer aangesloten moet worden is niet echt gestandaardiseerd. Verreweg de meeste printers worden gebouwd om met de zogenaamde Centronics-aansluiting te kunnen werken. Deze is genoemd naar de fabrikant die deze 'interface' voor het eerst op de markt heeft gebracht.



Centronics-aansluiting



RS-232-aansluiting

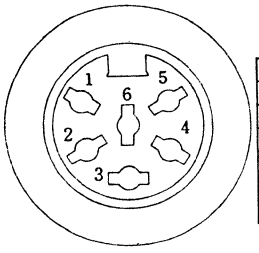
De Centronics-aansluiting is een parallelle verbinding, waarbij de informatie dus via meerdere draden tegelijkertijd wordt overgebracht. Maar ook de RS-232 aansluitingen komen nog wel eens op printers voor. Dat is dan de interface die tegenwoordig meestal voor modems wordt toegepast, een seriële verbinding.

Commodore zelf heeft een eigen interface ontwikkeld, de zogenaamde IEC-bus. Die interface wordt bij de 64 voor zowel de diskdrive 1541 als voor de printer gebruikt.

Dat betekent in de praktijk dat men nogal wat keuzemogelijkheden heeft als

3.1 Inleiding

het om een printer voor een 64 gaat. Ten eerste kan men voor een van de printers uit het aanbod van Commodore zelf kiezen, die standaard al van die eigen IEC-aansluiting voorzien zijn en dan ook zonder meer kunnen worden aangesloten.
Maar met de hoge vlucht van de 64 zijn ook andere fabrikanten gekomen met printers die rechtstreeks op die IEC-bus passen.



Pin No.	Signal
1	SERIAL SRQ
2	GND
3	SERIAL ATN
4	SERIAL CLK
5	SERIAL DATA
6	RES

In beide gevallen geldt echter jammer genoeg dat zo'n printer in feite alleen maar met een Commodore-computer gebruikt kan worden. Op andere merken computers vindt men die IEC-bus nu eenmaal niet, daar wordt bijna altijd een Centronics-bus gebruikt. Als men later ooit eens een andere computer aanschaft is die IEC-printer waardeloos. Bovendien is het aanbod in Centronics-printers veel en veel groter dan het aanbod aan specifieke IEC-printers.

Gelukkig zijn er ook wel mogelijkheden om een Centronics-type printer op de 64 aan te sluiten. Daarvoor bestaan er zelfs twee verschillende oplossingen.

Zo zijn er zogenaamde interfaces op de markt, die aan de ene kant op de Commodore-IEC bus passen en aan de andere kant op een Centronics-aansluiting. Zo'n interface zorgt ervoor dat de signalen van het ene type naar het andere worden omgezet. Sommige van die interfaces doen nog veel meer dan alleen maar omzetten, ze kunnen hele computersysteempjes herbergen die allerlei extra mogelijkheden bieden.

Goedkoper maar wat lastiger in het gebruik is een oplossing die de Centronics-printer via een speciale kabel en een stukje machinetaal-programma aan de user-poort knoopt. Het machinetaal-programmaatje zorgt er dan voor dat de print-opdrachten niet via de IEC-bus gaan, maar via de user-poort, in Centronics-formaat. Men moet dat stukje software echter wel steeds laden, terwijl er soms problemen zullen optreden als het printer-hulpprogramma qua geheugenbeslag in de knoop komt met andere handige hulpjes. Gelukkig echter bevatten sommige programma's zelf al zo'n printer-driver, zoals dat stukje software heet. De tekstverwerker Easyscript bijvoorbeeld kan zowel met standaard Commodore printers – via de IEC-bus – als met Centronics printers – via de user-poort – werken. In dat geval kan men met alleen een kabel volstaan.

Printers met een RS-232 aansluiting zijn tamelijk zeldzaam, zodat we er hier niet verder op ingaan. Eigenlijk worden dergelijke printers alleen in de wereld van de grote automatisering gebruikt. Wel aardig om te vermelden is dat Easyscript ook RS-232 printers zonder problemen kan aansturen, alweer via de user-poort. In dat geval heeft men echter wel een apart RS-232 tussenstuk nodig.

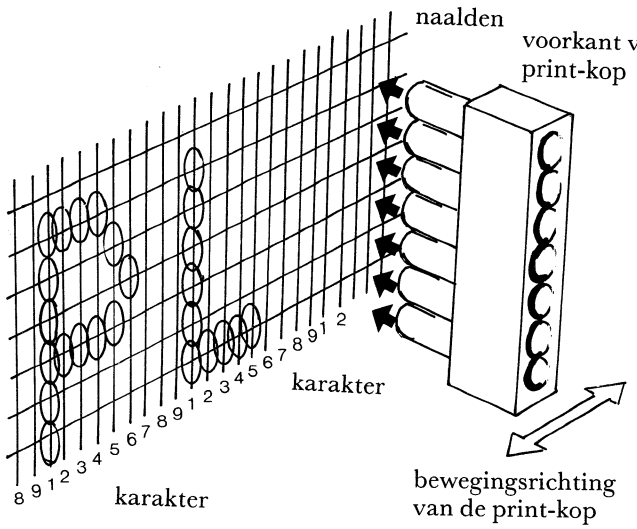
Printertypen

Met alleen de aansluitingen zijn we er echter nog lang niet. Er bestaan namelijk heel wat verschillende manieren om

3.1 Inleiding

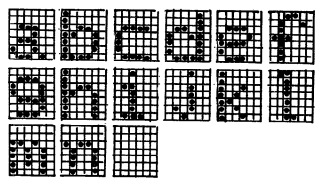
tekst op papier te krijgen, allemaal met hun eigen voor- en nadelen.

Matrixprinters
De matrixprinter is tegenwoordig het meest in gebruik. Bij dat printertype worden de tekens op het papier gevormd door patronen van losse puntjes. Bij de gewone matrixprinter werkt dat als volgt: in de printerkop zitten een aantal naaldjes, die ieder voor zich maar één enkel puntje tegelijkertijd kunnen afdrukken. Door nu die naaldjes in verschillende patronen 'af te vuren' kan zo'n matrixprinter zo ongeveer ieder gewenst teken op papier zetten.



Daardoor kan zo'n matrixprinter in principe de volledige Commodore-tekenset aan, inclusief de inverse tekens. Bij sommige modellen kan men desgewenst zelf tekens – soms slechts een enkele, soms de gehele tekenset – definiëren, zodat men dan werkelijk alles op papier kan zetten wat men maar wil, tot en met wiskundige formules met Griekse letters aan toe. Als men tenminste niet tegen een fors programmeerkarwei opziet, want zo eenvoudig als de printer-fabrikanten het voorspiegelen is het nu ook weer niet.

Veel matrixprinters kennen bovendien een grafische mode, een stand waarin de computer zelf de naalden stuk voor stuk kan besturen. Met de juiste programmatuur kan zo'n apparaat dan allerlei tekeningen en afbeeldingen afdrukken. Het zelf programmeren is echter alweer een behoorlijk ingewikkeld karwei.
De kwaliteit van een matrixprinter hangt van vele zaken af. Zo is het aantal naalden van belang voor zowel de snelheid als de kwaliteit van het schrift. Er bestaan bijvoorbeeld goedkope typen met slechts een enkele naald (unihammerprinters) die met die ene naald al die puntjes moeten maken. Dat werkt dan natuurlijk trager dan een type met meer naalden, die in een keer meerdere puntjes kan zetten. De uiteindelijke schriftkwaliteit wordt grotendeels bepaald door de grootte van de matrix, oftewel hoeveel puntjes er uiteindelijk worden gebruikt om een enkel teken te maken. Goedkopere printers werken met kleinere matrices, zo zal een simpeler printer bijvoorbeeld een patroon van 6 bij 7 puntjes per teken reserveren. Maar van die 6 puntjes die een teken dan maximaal breed zou mogen zijn worden er in de praktijk voor de letters etcetera maar 5 gebruikt, omdat anders de letters zonder tussenruimte geprint zouden worden.



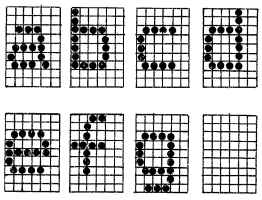
Voorbeeld letteropbouw (onderkast) van een simpele matrix-printer

Met andere woorden, de uiteindelijke letters moeten worden opgebouwd uit een patroon van 5 puntjes breed bij 7 puntjes

hoog. Dat levert een wel wat laag 'oplossend vermogen' op, het is lastig om de vormen van de verschillende tekens in zo'n klein matrix netjes weer te geven. Dat is dan ook wel te zien aan het schrift van zo'n goedkope printer, vooral de kleine letters zien er erg 'onrustig' uit.

Nadere bestudering leert al snel waardoor dat onrustige beeld veroorzaakt wordt; de letters staan namelijk allemaal op dezelfde hoogte. Normaal gesproken worden een aantal kleine letters – zoals de j en de g – deels onder de regel geschreven; met deze kleine matrix heeft men dat moeten laten vervallen. Daardoor lijkt het wel of deze letters hoofdletters zijn, hetgeen in de praktijk weliswaar wat onrustig oogt maar snel went. Voor officiële correspondentie echter zal zo'n goedkoop printertje zich mede door deze eigenaardigheid toch niet lenen.

Duurdere matrixprinters hanteren meer naalden – er zijn zelfs modellen met 24 naalden op de markt – en grotere matrices, waardoor het schriftbeeld veel fraaier kan worden.



Vaak ook zijn er meerdere letter-typen in deze duurdere printers ingebouwd, zoals bijvoorbeeld italics, dat cursieve letters toont.

Near Letter Quality

Een bijzonderheid van veel tegenwoordige kwaliteits-matrixprinters is de Near Letter Quality. Normaal gesproken drukken deze printers de tekens in één keer af, waarbij er dan ook duidelijk onderscheidbare afzonderlijke puntjes op het papier verschijnen. In de NLQ, zoals deze mode meestal genoemd wordt, wordt iedere regel in twee passages van de printkop op papier gezet. Tussen die eerste en die tweede printgang wordt het papier een heel klein stukje verschoven, zodat die tweede keer de openingen tussen de puntjes worden opgevuld.
Het uiteindelijke resultaat van deze operatie is een afdrukkwaliteit die met recht Near Leter Quality – Bijna Brief Kwaliteit – genoemd wordt. Slechts nauwkeurige bestudering toont dat de tekst door een matrix-printer is geproduceerd, op het eerste gezicht ziet een en ander er als schrijfmachinewerk uit.
De keerzijde van de medaille is echter dat de print-snelheid sterk terugloopt in NLQ. Een factor 2,5 à 3 is niet ongebruikelijk.

Thermisch

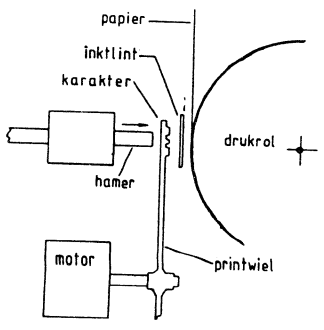
De werking van de matrixprinter kan op twee verschillende principes berusten. In beide gevallen ontstaat het uiteindelijke beeld op papier door puntjes op dat papier te zetten.
De aller-goedkoopste printertjes doen dat 'puntjes zetten' niet door middel van printnaalden, maar door hitte. Zo'n thermische printer bevat een printkop waar in plaats van naalden kleine weerstandjes zitten. Door een stroom door die weerstandjes te sturen worden ze heel snel heet, waardoor ze het speciale thermische papier laten verkleuren. Dat verhitten – en weer laten afkoelen – gaat zo snel in zijn werk dat er alleen maar kleine puntjes op het papier verschijnen, net als met een normale naaldprinter.

Het voordeel van zo'n thermische printer

3.1 Inleiding

ligt in het feit dat de bouw ervan veel eenvoudig kan zijn dan die van een naaldprinter. Want juist die printkop van een naaldprinter is een hoogstandje van fijn-mechanica.
Daardoor is een thermische printer goedkoop in aanschaf, hetgeen ze op het eerste gezicht – ondanks de wat slechtere kwaliteit van het schrift – heel aantrekkelijk maakt. Wie echter even doorrekent komt er al snel achter dat die lage aanschafprijs niet echt opweegt tegen de hoge kosten van het speciale thermische papier. Dat papier is bovendien nog slechts beperkt houdbaar ook, zowel voor als na het bedrukken.

Letterwiel-printers
Een volledig ander type printer is de letterwiel- of daisywheel-printer. Zo'n apparaat biedt een fraaier schrifttype dan de matrixprinter, omdat de letters niet uit losse puntjes opgebouwd worden maar – net als bij een schrijfmachine – in hun geheel worden afgedrukt. Daartoe gebruikt de daisywheel een letterwieltje, een cirkelvormig stukje plastic dat rondom van armpjes – spaken – voorzien is met op het einde van iedere spaak een bepaald teken. Door dat wieltje te draaien wordt het gewenste teken in de juiste positie gebracht, waarna een hamertje het door het lint heen op het papier afdrukt.



daisy-wheel printer.

Zoals gezegd, de kwaliteit van zo'n in zijn geheel afgedrukt teken is ten allen tijde beter dan die van een uit puntjes opgebouwde letter. Maar de daisywheel – het woord betekent margriet in het Engels en verwijst naar de vorm van zo'n letterwieltje – heeft ook nadelen.
Zo is een daisywheel minder flexibel dan een matrixprinter, het aantal tekens is beperkt tot wat er op een letterwieltje past. Meestal zijn dit er minder dan honderd, te weinig om de hele tekenset van een 64 onder te brengen.
Nu zijn die wieltjes wel verwisselbaar, zodat men voor bepaalde taken een speciaal wieltje kan inzetten. Er bestaan bijvoorbeeld wieltjes speciaal voor Nederland, met daarop de specifieke Nederlandse lettertekens. Het verwisselen van een wieltje is echter niet iets wat men onder het afdrukken door even snel doet. Afhankelijk van merk en model van de printer kan dat zelfs een tamelijk vervelend karweitje zijn.

De gemiddelde – betaalbare – daisywheel-printer is veel en veel langzamer dan een matrixprinter. Snelheden van 16 tot 18 tekens per seconde zijn heel gebruikelijk, terwijl een matrixprinter in dezelfde prijsklasse met gemak 100 tot 120 tekens per seconde doet.
Tenslotte moet gezegd worden dat de meeste daisywheel-printers zonder meer lawaaierig zijn. Matrixprinters zijn meestal niet echt prettig om te horen, met hun typisch snerpend geluid, maar letterwiel-printers 'hameren' hun teksten. In de meeste gevallen is dat bijzonder irritant. Toch is voor sommige toepassingen – bedrijfs-correspondentie bijvoorbeeld – de daisywheel onverslaanbaar. Met een goed lint en een

eersteklas letterwieltje levert zo'n apparaat nu eenmaal de best mogelijke kwaliteit af.

Plotters
Weer een heel andere klasse van apparaten is die van de zogenaamde plotter-printers. Dit zijn in eerste instantie tekenmachines, die onder computer-besturing in staat zijn hele complexe tekeningen op papier te zetten. Daartoe wordt gebruik gemaakt van kleine ballpoints of viltstiften, meestal in vier kleuren.
Om lijnen en figuren op papier te zetten zal de plotter zowel de pen als het papier heen en weer bewegen. De penhouder beweegt van links naar rechts, het papier van boven naar beneden. Beide bewegingen kunnen natuurlijk ook de andere kant op worden uitgevoerd, zodat een plotter-printer een volledige tekenautomaat is.

Behalve tekenen kan een plotter-printer ook afdrukken. Daartoe is een stukje extra intelligentie in zo'n apparaat ingebouwd, dat 'weet' hoe al die letters, cijfers en andere tekens er uit dienen te zien. Als er een programma op een plotter-printer moet worden gelist hoeft de computer alleen maar – net als bij een gewone printer – de eigenlijke tekst te versturen. De plotter-printer tekent dan vanzelf al die letters etcetera.

Dat tekenen gaat echter wel bijzonder traag. Althans, afgemeten aan de prestaties van een matrix-printer. Een plotter-printer in de betaalbare prijsklasse haalt niet meer dan 8 tot 10 tekens per seconde. Op zich, als we ons bedenken dat ieder tekentje echt getekend wordt, is dat nog niet zo slecht. Maar een wat langere listing afdrukken duurt wel erg lang. Een voordeel van de plotter-printer is echter wel weer de bestuurbaarheid. Op de meeste typen kan men vrij simpel zelf de teken-grootte kiezen, terwijl de penkleur wisselen ook al simpel is. Maar toch moet een plotter-printer niet als volwaardig alternatief voor een echte printer gezien worden. Slechts als men de plot-functie echt gebruikt valt zo'n apparaat te overwegen.

Schrijfmachines

Veel moderne electronische schrijfmachines – bijvoorbeeld die van het merk Brother – zijn in feite een soort van computers. Tussen het toetsenbord en de afdruk-eenheid bevindt zich een heel stuk digitale elektronica, vaak compleet met ingebouwd correctie-geheugen.
Daardoor zijn die machines vrij makkelijk om te bouwen tot computer-printers, hetgeen dan ook door allerlei bedrijfjes gedaan wordt. Sommige boeken en tijdschriften hebben zelfs zelfbouwschema's hiervoor geproduceerd.

Toch is deze op het eerste gezicht slimme oplossing niet echt aan te raden. Een schrijfmachine is nu eenmaal niet gebouwd om het computergeweld aan te kunnen. Een menselijke typist haalt nooit lang achter elkaar hoge aantallen aanslagen per minuut, een computer die een listing afdrukt wel. De kans dat de schrijfmachine daar op de lange duur onder bezwijkt – door overbelasting – is zeker met de goedkopere modellen dan ook niet uit te sluiten.
Als tweede printer is een schrijfmachine echter wel aan te raden. Dan heeft men bijvoorbeeld de beschikking over een snelle maar niet al te fraaie matrix-printer voor het zware werk, terwijl de

3.1 Inleiding

schrijfmachine kan worden ingezet voor correspondentie-doeleinden.

Inktjet-printers
Hadden alle tot nog toe besproken printer-typen gemeen dat het teken gevormd werd doordat de print-eenheid – al dan niet met lint ertussen – het papier werkelijk raakte, de inktjet-printer werkt volgens een ander principe. Hierbij wordt de inkt via een elektrostatisch systeem op het papier 'gespoten'. De druppeltjes inkt verlaten met hoge snelheid de 'lopen' van de printkop, die te vergelijken zijn met de naalden van een matrixprinter. Het voordeel van deze techniek is dat er minder mechanische delen nodig zijn, waardoor de printsnelheid hoger kan worden opgevoerd dan bij traditioneler systemen. Door de nadelen die er aan verbonden zijn is het systeem echter niet echt aangeslagen.

Zo blijken de inktjet-printers nogal gevoelig voor storingen, die vaak veroorzaakt worden door het indrogen van de inkt in de kop. Zeker als de printer niet dagelijks gebruikt wordt maakt dit het systeem lastig hanteerbaar. Bovendien moet er in een inktjet-printer speciaal papier gebruikt worden, dat de druppeltjes inkt snel opneemt. Deze papierkwaliteit – die wel wat lijkt op wat er in ouderwetse stencilmachines werd toegepast – leent zich niet voor representatieve doeleinden.
Al met al is de inktjet tot nog toe voor hobbyisten niet echt interessant.

Laserprinters
Deze nieuwste ontwikkeling op printergebied is momenteel nog veel en veel te kostbaar om bij een typische hobby-computer als de 64 te gebruiken. Volledigheidshalve willen we er toch even aandacht aan schenken.

De laserprinter koppelt namelijk een bijzonder grote flexibiliteit aan een zeer hoge snelheid. Dit komt doordat er een volledig ander proces is toegepast dan in de tot nog toe besproken printertypen.
Al die verschillende afdruk-methoden hadden namelijk gemeen dat het teken rechtstreeks op papier werd gezet, hierbij door naaldjes, letterwieltjes, hitte, pennen of inktdruppeltjes. De laserprinter lijkt veel meer op een fotokopieer-apparaat qua werking.
Zo'n fotokopieer-apparaat werkt meestal met een ronddraaiende trommel, waarop het te kopiëren beeld middels een lenzenstelsel wordt geprojecteerd. Die trommel wordt eerst van een statische elektrische lading voorzien, door de projectie wordt die uniforme lading veranderd in een patroon dat overeenstemt met het te kopiëren beeld. Met andere woorden, na opladen en projecteren 'staat' het origineel als een statisch ladingspatroon op de trommel.

Nu heeft statische elektriciteit een aantal heel bijzondere eigenschappen. Zo trekt een statisch geladen voorwerp voorwerpen met een tegengestelde lading aan; ook bij statische ladingen is er sprake van plus en min, positief en negatief. Door nu die trommel met dat statische ladingspatroon langs een bakje met poeder met een tegengestelde lading te laten roteren zal dat poeder zich op die plekken van de trommel die geladen zijn vastzetten.

3.1 Inleiding

Dat poeder – toner genaamd – kan daarna worden overgedragen op een blanco vel papier en daarop worden 'vastgebrand', door het vel door een soort oventje te leiden. Het resultaat is een haarscherpe fotokopie, zoals die uit iedere moderne kopieermachine komt rollen.

Terug naar de laserprinter. Ook daar treffen we zo'n trommel, tonerbakje, papier-overdracht en oventje in aan. Het verschil echter met de copieermachine is de wijze waarop het ladingspatroon op de trommel wordt aangebracht. Hier worden geen originelen geprojecteerd. Het patroon wordt namelijk op de trommel 'getekend' met behulp van een kleine ongevaarlijke laserstraal, onder besturing van de in de laserprinter ingebouwde computer. Dat proces heeft nauwelijks nog iets te maken met de mechanica van een 'gewone' printer, het 'puntjes-tekenen' verloopt bliksemsnel.

In feite zet die laser alleen maar puntjes op de trommel, net zoals dat bij iedere matrixprinter gebeurd. Maar door de veel hogere snelheden die daarbij bereikt kunnen worden kan dat puntenpatroon veel dichter worden, de gemiddelde laserprinter gebruikt maar liefst 300 puntjes per inch! Die hoge dichtheid leidt dan tot letters en andere tekens die wat kwaliteit betreft zeker net zo goed zijn als hetgeen een daisywheel produceert. Het komt zelfs in de buurt van de kwaliteit van fotozetwerk.

Maar, en dat is eigenlijk nog belangrijker, zo'n laserprinter kan door de ingebouwde intelligentie makkelijk vele soorten schrift produceren. Lettertypen, lettergroottes, lijnen en balken, het kan allemaal met de juiste stuurcodes in één keer op een pagina gezet worden. En snel, bliksemsnel. Een goede laserprinter haalt gemiddeld zo'n 8 tot 10 velletjes redelijk dicht bedruk A4 papier per minuut. Dat komt overeen met rond de 650 tekens per seconde, een snelheid die geen enkel ander printertype kan halen.

Spijtig genoeg zijn deze wondertjes der techniek voorlopig nog veel te duur voor de hobbyist, en dat zal nog wel even zo blijven ook. De gemiddelde prijs van de eenvoudiger typen schommelt momenteel – najaar 1986 – zo rond de *f* 8000,–. Maar wie weet, per slot van rekening zijn ook de matrixprinters de laatste jaren zo ongeveer in prijs gehalveerd.

Meer-kleuren printers

Sommige van de hierboven omschreven printer-typen zijn bovendien ook nog eens als meerkleuren-printer uitgevoerd. De plotter-printers zijn dat bijna allemaal, met in totaal vier kleuren pennen, maar ook de matrix-printers kunnen met meerkleuren-linten worden uitgevoerd. Bij de inktjet-printers is meerkleuren-druk zelfs alles behalve ongebruikelijk.

Alleen bij die laatste categorie van de inktjet-printers is het echt een handig extra, meerkleuren matrixprinters blijken in de alledaagse praktijk niet echt prettig in het gebruik. In verreweg de meeste gevallen zal men namelijk toch meestal zwart gebruiken, bijvoorbeeld om listings af te drukken. Dat betekent dat het – dure – meerkleurenlint meestal alleen wat het zwarte gedeelte

3.1 Inleiding

betreft versleten is als men het moet verwisselen, hetgeen tot hoge kosten per pagina leidt.

Bij die inktjet-printers kan men meestal de diverse inktkleuren per stuk vervangen, hetgeen tot een veel economischer gebruik leidt.

In de praktijk worden er maar weinig meerkleuren-printers verkocht. Hetgeen er weer toe leidt dat er maar weinig programma's op berekend zijn, zodat de meerkleuren-printer in de praktijk eerder als 'gimmick' gezien moet worden dan als een nuttige mogelijkheid.

Papier en papiertransport

Een printer moet in het gebruik nogal wat aankunnen. Aan de ene kant worden ze gebruikt om met zo min mogelijk moeite listings, overzichten en wat al niet zo snel mogelijk af te drukken, waarbij de kwaliteit van zowel het afdrukken als het papier er niet zo veel toe doen.
Aan de andere kant kunnen ze juist gebruikt worden om prestigieuze correspondentie te verzorgen, bij voorkeur op het eigen briefpapier.
Of wat te denken van etiketten voor het verenigingsblaadje?

Drie heel verschillende toepassingen, die ieder zo hun eisen stellen wat betreft papiersoort en de manier waarop de printer met dit papier kan omgaan.

Tractorfeed

Zo zal men om snel en simpel te kunnen afdrukken over het algemeen een printer met 'tractorfeed' of 'pinfeed' willen gebruiken. In dat geval kan de printer desgewenst een hele doos van 2000 vel papier geheel zelfstandig bedrukken, zonder dat er vellen ingezet hoeven te worden. In zo'n doos liggen de velletjes als een lange baan met scheurranden ertussen opgestapeld. Aan de beide zijkanten zitten, meestal weer met scheurrandjes, perfortieranden. Die geperforeerde randen worden door de printer gebruikt om het papier 'vast te pakken' en door het mechanisme te transporteren, zonder dat men daar verder naar om hoeft te kijken.

De geprinte vellen kunnen daarna simpel van elkaar gescheiden worden door middel van de scheurranden, ook de perforatieranden laten zich moeiteloos verwijderen. Daarbij blijft echter meestal wel te zien dat het papier in kwestie door een printer bedrukt is, en niet getypt. Er blijven sporen van die scheurranden zichtbaar.

Wie prijs stelt op nettere uitvoer kan papier met zogenaamde micro-perforatie kopen. Dat betekent niet dat de zijdelingse perforatiegaten opeens kleiner geworden zijn, gelukkig is daar een internationale standaard voor waardoor elk kettingformulier qua perforatie in iedere printer past.
Dat 'micro-perforatie' slaat echter op de scheurrandjes. Die zijn bij deze kwaliteit zodanig uitgevoerd dat na het afscheuren het bijna niet meer te zien is dat ze er ooit gezeten hebben.
Wie wil kan door allerlei bedrijven kettingformulieren met kant-en-klare teksten laten voorbedrukken. Eigen briefhoofd, nota's, het vormt allemaal geen probleem. Sommige fabrikanten leveren zelfs kettingformulieren met daarop een algemene nota voorgedrukt, waar-

bij men alleen de eigen firmanaam en verdere gegevens door de computer hoeft te laten invullen.

Friction-feed

Maar toch kan een kenner altijd zien of een bepaalde brief door een tractor-feed printer is afgedrukt of niet. Als men weet waar men op moet letten zijn de tekenen duidelijk. Vandaar dat vele printers ook met losse vellen kunnen werken, net als een typemachine. Het systeem lijkt daar ook sterk op, met alle nadelen vandien. Het goed recht inleggen van een vel kost tijd, en moet na ieder vel weer opnieuw gedaan worden.

Een manier om dit inleggen van vellen te omzeilen is het gebruiken van papier aan de rol, zoals dat bij bijvoorbeeld telex-machines gebruikelijk is. Weliswaar is dit niet bevorderlijk voor de kwaliteit van het printwerk, de scheurrand is altijd zichtbaar, maar aangezien sommige printers geen tractor-mechanisme hebben wordt er soms naar gegrepen als werkbesparende maatregel. Gelukkig hebben bijna alle matrix-printers, op de allergoedkoopste typen na, tractor-feed. Alleen de printer-plotters gebruiken over het algemeen papier op de rol, omdat men anders qua lengte van een tekening beperkt is tot het formaat van het vel.

Sheetfeeders

Bij de daisywheel-printers ligt de situatie weer heel anders. Deze apparaten zijn standaard bijna altijd alleen met friction-feed uitgerust, waarbij soms een tractor-feed als losse extra verkrijgbaar is.
Maar gezien het feit dat de daisywheel voornamelijk voor klasse-printwerk gebruikt wordt zal men over het algemeen toch losse vellen verwerken. Alleen het met de hand inleggen wordt wel heel tijdrovend.

Daar is dan weer een aparte oplossing voor bedacht, in de vorm van de sheetfeeder. Meestal wordt deze als losse extra geleverd, als een soort opzet-apparaat. In de sheetfeeder kan dan een pakje losse vellen gestoken worden die daarna automatisch, keurig één voor één, in de printer gedraaid worden. Ideaal om bijvoorbeeld op het eigen briefpapier te kunnen werken.
Een nog luxueuzer variant hierop is de sheetfeeder met meer dan één papiermagazijn. Dan kan men in het ene magazijn bijvoorbeeld het normale briefpapier laden, terwijl het andere magazijn het nota-papier bevat. Een simpel commando aan de printer vanuit bijvoorbeeld een tekstverwerker kiest dan de gewenste papiersoort voor een bepaald document.

Demonstreren

Uit het voorgaande blijkt wel dat het begrip 'printer' niet zo eenvoudig is. Om een printer voor uw doeleinden te vinden in een prijsklasse die voor u aanvaardbaar is kan een hele klus zijn. Temeer daar er vele, vele verschillende typen op de markt gebracht worden, ieder met de eigen mogelijkheden.

We willen u echter één ding van harte aanraden. Als u denkt de juiste printer gevonden te hebben voor uw doeleinden, laat u dan door de handelaar demonstreren dat zulks ook inderdaad zo is, zeker als u voor een niet Commodore-printer kiest. Anders loopt u de kans dat, eenmaal thuisgekomen, de

3.1 Inleiding

combinatie van printer, interface en software nu net niet blijkt te kunnen wat u wilde.
Laat u, als er geen mogelijkheid tot demonstratie is, anders op papier garanderen – bij voorkeur op de aankoopbon – dat uw gewenste toepassing inderdaad mogelijk is, onder bedinging van het recht van teruggave. Zeker als u van plan bent om het apparaat bij een bepaalde tekstverwerker te gebruiken is dat helemaal niet zo'n overdreven voorwaarde. Mocht de winkelier hier niet toe genegen zijn, bedenk u dan dat er meer computerhandelaren in Nederland zijn en ga elders uw aankopen doen. Anders loopt u het risico met de gebakken peren te zitten.

Uitpakken en aansluiten

Met zoveel soorten en typen printers is het lastig om algemene aanwijzingen te geven over het aansluiten ervan. Volg de aanwijzingen van de gebruiksaanwijzing is de beste raad die we u kunnen geven. Als de handelaar u het een en ander heeft gedemonstreerd zal het allemaal niet al te ingewikkeld zijn.

Waar u wel op moet letten is dat de meeste printers speciaal voor de verzending worden voorbereid, bijvoorbeeld door de printkop vast te schroeven. Probeer nooit te printen zonder dat deze 'packing-screws', zoals ze in het Engels heten, verwijderd zijn. Ook alle andere stukjes schuimplastic etcetera moeten voor alles verwijderd worden.
Bewaar deze verpakkingsmaterialen goed, als u uw printer om de een of andere reden moet verzenden of vervoeren is het namelijk raadzaam om ze weer aan te brengen. Anders kunnen de schokken van het transport uw kostbare bezit beschadigen.

De meeste printers kennen een zelftest, waarbij de printer zonder dat deze aan een computer aangesloten hoeft te zijn toch de tekenset kan afdrukken. Het verdient aanbeveling om, alvorens u de printer aansluit, deze zelftest uit te voeren, nadat u het lint geplaatst heeft. Op die manier weet u als er problemen zijn in ieder geval dat de printer op zich functioneert.

6/3.2

Basisbegrippen

6/3.2.1

Printerbesturing

Standaard Commodore-printers zijn, net zoals de diskdrive, voorzien van een eigen interne microcomputer. Dat houdt in dat er heel wat taken door de printer zelf afgehandeld kunnen worden, die dan niet de capaciteit van de computer belasten. Net als bij andere Commodore-randapparatuur moeten we echter wel duidelijk aangeven wat we willen van die in de printer ingebouwde computer.

Ook niet-standaard Commodore-printers hebben meestal de nodige ingebouwde mogelijkheden. Daarbij worden echter weer andere manieren gebruikt om aan te geven wat men van de printer wil.

Met andere woorden, er zijn twee fundamenteel verschillende manieren om een printer te besturen, die elkaar ten dele overlappen. Zo kennen sommige Commodore-printers beide manieren, zowel de specifieke Commodore-besturing als de meer algemene wijze die we op vele printer-typen aantreffen.

Stuurcodes

Iedere printer – Commodore en niet-Commodore – kent wel een aantal stuurcodes. In feite zijn dit gewone ASCII-tekens die echter niet als teken worden afgedrukt maar een bepaalde printer-functie aanroepen.

Uitgaande van de standaard 64 IEC-poort werkt de communicatie tussen computer en printer eigenlijk heel eenvoudig; de computer stuurt gewoon 8-bits tekens – die dus kunnen worden voorgesteld door een waarde tussen de 0 en de 255 – over. Dat dit serieel plaatsvindt – bitje voor bitje – maakt niets uit voor het uiteindelijk effect. Ook een Centronics-aansluiting – hoewel deze heel anders van de bouw is – doet uiteindelijk exact hetzelfde, er worden bytes verzonden van de computer naar de printer.

Die 256 verschillende codes kunnen twee verschillende soorten betekenissen hebben. Zo kan een getal simpelweg voor een bepaald teken staan – de code 65 is altijd een letter a, waarbij geldt dat dit bij een Commodore-printer zowel een grote als een kleine a zou kunnen zijn – maar ook voor een speciale actie van de printer.
Een voorbeeld daarvan is de code 10, die in de ASCII-tabel LF heet. Dat LF staat voor Line-Feed, Regel-Opvoer. Als we een CHR$(10) naar de printer sturen zal die printer reageren door het papier een regel op te voeren. Overigens zal ook de 64 zelf, als we een CHR$(10) op het scherm printen, op dezelfde manier reageren. Er is aan die ASCII-waarde 10 geen normaal afdrukbaar teken verbonden, het is slechts een stuurcode voor

zowel de printer als het beeldscherm.

In feite gaat dat voor alle ASCII-waarden onder de 32 op. Ze worden stuk voor stuk gebruikt om allerlei speciale functies af te handelen, die binnen de officiële ASCII-tabel dan ook gedefinieerd zijn.
Om nog een voorbeeld te geven, de meeste matrix-printers kennen een dubbelbreed schrift. Dat kan worden in- en uitgeschakeld met de ASCII-codes SO (Shift Out) en SI (Shift In), respectievelijk CHR$(14) en CHR$(15).

Nu is de ASCII-codetabel al jaren terug opgesteld, en de techniek heeft in die tijd niet stilgestaan. Vandaar ook dat er een aantal van die stuurcodes vandaag de dag eigenlijk niet meer gebruikt worden. Bovendien is de ASCII-set bedoeld om alle communicatie-problemen mee op te lossen, vandaar ook dat er een aantal stuur-tekens in terug te vinden zijn die voor de communicatie tussen computer en printer niet nodig zijn.
Sommige fabrikanten – onder andere Commodore – gebruiken die min of meer in onbruik geraakte tekens dan ook op niet-standaard ASCII wijze, iets om wel even rekening mee te houden.

Bovendien was de ASCII-tabel op een 7-bits code geënt, de hoogste waarde die gebruikt kon worden was 127. De tegenwoordige micro-computers en printers werken echter met 8-bits codes, zodat de waarden vanaf 128 tot en met 255 ook gebruikt kunnen worden. Ook in dit traject kunnen sommige waarden als stuurcodes gebruikt worden.
Een voorbeeld op de 64 is bijvoorbeeld de waarde CHR$(146), waarmee een invers te printen veld – waarbij dus juist de achtergrond wel en het eigenlijke teken niet geprint wordt, net als dat op het scherm gedaan wordt – moet worden afgesloten.

Al met al is het oppassen geblazen met deze stuurcodes, daar ze van printer tot printer kunnen verschillen in betekenis. Bovendien vormen ze één van de struikelblokken die men kan tegenkomen bij het gebruik van niet speciaal voor Commodore gebouwde printers, daar Commodore bepaalde stuurtekens op eigen wijze gebruikt.

Escape-series

De tegenwoordige printers hebben heel wat in hun mars. Zoveel zelfs dat ze in feite niet met alleen die ASCII controle-codes bestuurd kunnen worden. Dat zag men ten tijde van het opstellen van de ASCII-tabel al aankomen, vandaar dat er een speciaal stuurteken gereserveerd is om allerlei extra besturingen mee in te leiden. Dat teken is de escape, meestal afgekort tot ESC, CHR$(27). De escape – hetgeen ontsnappen betekent – wordt gebruikt om aan te geven dat er hierna geen af te drukken tekens zullen volgen maar een serie codes die tezamen een speciaal printer-commando vormen. Veel moderne printers herkennen tientallen verschillende escape-series, waarmee bijvoorbeeld de paginalengte van het papier ingesteld kan worden, of de linker- en rechterkantlijn. Ook allerlei grafische modes worden door escape-commando's aangezet.

Het probleem daarbij is echter dat sommige printerfabrikanten nogal eigen-

3.2 Basisbegrippen

zinnig hun eigen escape-sequences hebben gedefinieerd, hetgeen de programmeurs voor ernstige problemen stelt. Want voor zowel hobby- als professionele programmeurs wordt het wel erg lastig om bijvoorbeeld een fraaie screendump – waarbij de printer de inhoud van een grafisch schermbeeld moet afdrukken – te programmeren als men niet er van uit kan gaan dat daar vaste escape-commando's voor gebruikt kunnen worden. Gelukkig is er wel sprake van een zekere standaardisatie, zeker als we uitgaan van de printers van Commodore zelf of modellen die door andere fabrikanten speciaal voor de Commodore gebouwd zijn. Toch is het wel aan te bevelen om de handelaar te laten demonstreren dat uw grafische programma's het inderdaad doen met de printer die u wilt aanschaffen.

Secondaire adressen

Commodore-printers – en printers die door andere fabrikanten speciaal voor Commodore-computers gebouwd zijn – kennen bovendien nog een derde manier om de printer opdrachten te geven. Een methode overigens die alleen door Commodore wordt toegepast, andere computerfabrikanten hebben zich hier nooit aan gewaagd.

De communicatie tussen een Commodore-computer en een printer verloopt namelijk in principe altijd via een bestand, zoals straks uitgelegd zal worden. Deze bestanden zijn al eerder aan de orde gekomen, bij de bespreking van de externe gegevensopslag op bijvoorbeeld een diskdrive.
Bij het openen van een bestand kan de programmeur drie dingen opgeven, namelijk bestandsnummer, apparaat-nummer en het zogenaamde secondaire adres. Op de twee eerste hiervan zullen we later terugkomen, het secondaire adres is echter een onderdeel van de printerbesturing.
Kort gezegd komt het er op neer dat dit secondaire adres allerlei extra mogelijkheden van een Commodore-printer kan kiezen, afhankelijk van het type printer. Zo bestaan er Commodore-printers, bijvoorbeeld de MPS-802, die in staat zijn intern de af te drukken gegevens te formatteren tot keurige kolommen. Deze specifieke Commodore-mogelijkheid moet dan worden gekozen door een secondair adres op te geven bij het openen van een bestand. Bij de bespreking van de diverse typen printers zullen we hier, voor iedere printer apart, op terugkomen.

6/3.2.2

Printercommunicatie

De Commodore 64 kent een nogal bijzonder systeem om met zijn randapparaten – diskdrives, printers etcetera – te communiceren. Al die apparaten zijn namelijk op dezelfde bus aangesloten, de al besproken IEC-bus. Ook in een complexe opstelling van zeg een 64, twee 1541 disk-
drives en een tweetal printers – bijvoorbeeld een matrixprinter en een daisywheel – zijn alle vier de randapparaten via een en dezelfde kabel aan de computer gekoppeld. Ieder stukje informatie dat over de bus wordt uitgewisseld kan in principe door al die apparaten worden opgepikt.
Ook als er een interface gebruikt wordt, bijvoorbeeld van IEC naar Centronics, gaat dit op. Zelfs in het geval van een rechtstreeks op de user-poort aangesloten Centronics-kabel met een stukje extra machinetaal dat die informatie die voor de printer bestemd is naar de user-poort stuurt in plaats van via de IEC-bus blijft dit eigenlijk gelden.

De communicatie via die IEC-bus loopt altijd via een 'bestand'. Vanuit het standpunt van zowel de BASIC- als de machinetaal programmeur komt het naar de printer sturen van informatie er in feite op neer dat men een bestand naar die printer moet openen om daarna de af te drukken informatie via dat bestand te verzenden, net alsof men gegevens in een disk-bestand zou willen opslaan. Het verschil zit hem alleen maar in de wijze van 'opslaan', een diskette-bestand valt later weer terug te lezen in het computergeheugen, een papieren print-out niet. Tenzij men het opnieuw wil intikken, natuurlijk.

Om iets op de printer af te drukken moet er dus allereerst een 'bestand' worden geopend, met het normale OPEN-commando. Dat ziet er voor de printer als volgt uit:

OPEN filenummer, apparaatnummer, secondair adres

Dat filenummer mag vrij gekozen worden tussen de 1 en de 255, waarbij aangetekend moet worden dat filenummers boven de 128 een bijzonder effect sorteren. Normaal gesproken zal de 64 namelijk na iedere PRINT-opdracht naar een bestand een CR – wagen terug of Carriage Return – opdracht sturen, waarmee de printkop weer vóór aan de regel gepositioneerd wordt. De meeste printers – zeker die van Commodore zelf – hebben hier genoeg aan, als ze een CR stuurteken ontvangen voeren ze zelf een LF – Line Feed of regelopvoer – uit. Het uiteindelijk effect is dat de printkop niet alleen weer in de uitgangspositie geplaatst is maar dat er ook een blanco regel is voorgedraaid door het printer-

mechanisme.
Sommige printers werken echter anders en zullen bij het ontvangen van een CR inderdaad alleen maar de printkop naar links brengen. Een volgende regel wordt dan plompverloren over de vorige heengeprint. Vaak is dit op de printer zelf instelbaar, er bevinden zich dan ergens wat kleine schakelaartjes waarmee men onder andere kan kiezen of er bij ontvangst van een CR automatisch een LF gegenereerd moet worden.
Soms echter is dit niet het geval, en dan kan kiezen van een bestandsnummer van 128 of hoger uitkomst bieden. Het bedrijfssysteem van de 64 voegt namelijk, als het filenummer groter dan 127 is, zelf die LF aan iedere CR toe, zodat het probleem verholpen is.

Na het bestandsnummer volgt het apparaatnummer in het OPEN-commando. Met dat apparaatnummer geven we op welk randapparaat we bedoelen.
Voor printers zijn er twee mogelijkheden, 4 en 5. De meeste Commodoreprinters zijn door de fabriek ingesteld als apparaat 4, in de praktijk wordt apparaatnummer 5 vrijwel nooit gebruikt. Maar als men bijvoorbeeld een snelle matrixprinter gebruikt voor de 'ruwe' uitvoer met daarnaast een daisywheel voor de nette print-outs, dan kunnen beide printers tegelijkertijd worden aangesloten. Jammer genoeg houden echter lang niet alle printer-geörienteerde programma's zoals tekstverwerkers rekening met deze mogelijkheid.

Het secondaire adres wordt bij sommige printertypen gebruikt om extra wensen door te geven. Welke getallen hier kunnen worden gebruikt kan van model tot model verschillen, in de besprekingen van de diverse printers zal er verder op ingegaan worden.
Als het secondaire adres wordt weggelaten neemt de 64 aan dat de waarde 0 bedoeld is, die voor printers die er gebruik van maken inhoudt dat de gegevens exact zo moeten worden afgedrukt zoals ze worden ontvangen. Dat houdt ondermeer in dat de printer zijn werk in de gebruikelijke Commodore-tekenset zal verrichten, waarbij er alleen hoofdletters beschikbaar zijn. Vele printers en interfaces kunnen in de zogenaamde business-mode geschakeld worden – waarbij er weliswaar minder grafische tekens beschikbar zijn maar wel hoofd- en kleine letters – door als secondair adres de waarde 7 op te geven.

Na zo'n OPEN-opdracht is er een bestand beschikbaar dat aan de printer gekoppeld is. Alles wat we naar dat bestand sturen zal door de printer worden opgepikt en in de meeste gevallen ook meteen worden afgedrukt. Dat 'naar de printer sturen' gebeurt met de PRINT-opdracht:

PRINT#filenummer, variabele of constante

Alle in BASIC 2.0 PRINT statements toegestane constructies mogen gebruikt worden. Zowel constanten zoals "tekst" of 1234.56 als variabelen zoals A$, A en A% zijn toegestaan. Ook meerdere variabelen en/of constanten kunnen in een enkele PRINT# worden afgedrukt, bijvoorbeeld:

PRINT#, "Nu volgt een variabele";A;" met weer wat tekst erachter"

3.2 Basisbegrippen

Na het afdrukken van alle gegevens is het zaak om het gebruikte bestand weer af te sluiten, met:

CLOSE filenummer

Om later in het programma weer te gaan afdrukken kan heel simpel het te gebruiken bestand geopend worden. Het afsluiten is belangrijk omdat er op een 64 slechts 10 bestanden tegelijkertijd geopend kunnen zijn. Door steeds weer op tijd te sluiten zorgt een programmeur ervoor dat er altijd zoveel mogelijk bestanden beschikbaar blijven.

Luisteren

In de manier waarop de 64 en de randapparaten met elkaar communiceren staan de begrippen 'listening' en 'unlistening' – letterlijk vertaald 'luisteren' en 'onluisteren' – centraal. Doordat al die apparaten namelijk een en dezelfde bus gebruiken moet ten allen tijde duidelijk zijn welk apparaat door de computer wordt bedoeld. Het kan best gebeuren dat er tijdens de afloop van een programma tegelijkertijd bestanden op de diskdrive als op de printer geopend zijn. Als in zo'n geval er verwarring optreedt over welk apparaat met de verzonden informatie aan de slag moet zal er van alles mis kunnen gaan.

Dat wordt geregeld door aan de informatie die middels een PRINT# opdracht wordt verzonden een 'luister'-commando vooraf te laten gaan. In feite zou men kunnen stellen dat de 64 aangeeft welk randapparaat moet reageren – een ook een terugmelding van dat apparaat verwacht dat het paraat staat – vóór er informatie verzonden wordt. Na iedere PRINT# opdracht krijgt het apparaat weer de opdracht om op te houden met luisteren, de 'onluister'-opdracht.
Al deze extra communicatie speelt zich voor de BASIC-programmeur volstrekt onzichtbaar af, men hoeft er eigenlijk – op een uitzondering na – geen rekening mee te houden.

CMD

Die uitzondering is het CMD-commando, waarmee de computer opdracht gegeven kan worden om de uitvoer die normaal op het scherm zou verschijnen via een 'bestand' naar een randapparaat te versturen. De syntax luidt:

CMD filenummer, tekst

Het filenummer moet een bestandsnummer zijn dat voor die tijd al met een OPEN-commando als printer-bestand gedefinieerd is, de eventuele tekst-constante wordt als eerste afgedrukt. Na een CMD-commando zal alle uitvoer die normaal gesproken op het scherm zou verschijnen naar het door CMD actief gemaakte bestand gaan.

Echter, het CMD-commando wordt niet afgesloten door een 'onluister'-opdracht, in tegenstelling tot de PRINT#. Na een CMD blijft de printer alle informatie op de IEC-bus oppikken en afdrukken.
Vandaar dat we, als we klaar zijn met afdrukken via de CMD, de printer alsnog moeten instrueren om op te houden met luisteren. Dit wordt heel eenvoudig gedaan door een 'lege' PRINT# opdracht.

3.2 Basisbegrippen

Voor het programmeren is het CMD-commando niet erg handig, voor het maken van een listing van een programma op papier echter wel. Om een BASIC-programma in het geheugen van de 64 te laten afdrukken gebruikt men bijvoorbeeld de opdrachten

```
OPEN 4,4
CMD 4,"PROGRAMMANAAM"
LIST
PRINT# 4
CLOSE 4
```

Dat zou als uitvoer kunnen opleveren:

```
PROGRAMMANAAM

READY.

10 REM EEN PROGRAMMA ZONDER INHOUD
20 REM OM HET LISTEN TE DEMONSTRE-
   REN
30 END

READY.
```

Na het openen en het CMD-commando verschijnt echter alles wat anders op het scherm zou worden afgebeeld op de printer, ook de READY. na de 'programmanaam'. Het zou fraaier zijn om deze READY. te onderdrukken, zodat de naam en de eigenlijke listing niet door een extra tekst van elkaar worden gescheiden. Dat kan, het is zelfs vrij simpel. De 64 heeft namelijk de gewoonte om na iedere opdracht in de directe toestand – als we niet bezig zijn met een programma te laten runnen – de READY. af te drukken. Maar we kunnen desgewenst meerdere opdrachten in een enkele keer laten uitvoeren. Als we de commando's om een listing te maken op een regel samenvoegen zal die extra READY. onderdrukt worden.

```
OPEN 4,4:CMD 4, "PROGRAMMANAAM":LIST
PRINT# 4
CLOSE 4
```

Levert de volgende uitvoer op:

```
PROGRAMMANAAM
10 REM EEN PROGRAMMA ZONDER INHOUD
20 REM OM HET LISTEN TE DEMONSTRE-
   REN
30 END

READY.
```

Jammer genoeg kunnen we niet alle commando's op een enkele regel onderbrengen, aangezien de LIST-opdracht automatisch de uitvoering van de regel – of het programma, als we daar een LIST-commando in opnemen – beëindigt. Na die LIST moet de printer alsnog, in een nieuwe opdracht, verteld worden dat hij moet 'onluisteren'. De extra PRINT# en de CLOSE zouden echter wel weer samen op een regel kunnen worden geplaatst.

Buffer

Er is echter nog een reden om na een CMD-commando – en ook in andere gevallen – zo'n 'lege' PRINT# opdracht te geven. Het is namelijk niet zo dat de printer alles wat er maar toegezonden wordt ook onmiddellijk afdrukt. De volgende voorbeeldjes onderstrepen dit:

```
10 OPEN 4,4
20 PRINT#4,"DEZE  TEKST
   WORDT AFGEDRUKT"
30 CLOSE 4

READY.
```

Als dit programma wordt uitgevoerd verschijnt keurig de tekst die in regel 20 gePRINT wordt op de printer. Als we echter het programmaatje iets veranderen verschijnt er geen tekst op de printer:

```
10 OPEN 4,4
20 PRINT#4,"EN DEZE TEKST
   NIET!";
30 CLOSE 4
READY.
```

Pas als we daarna in de direct-mode intikken:

```
OPEN 4,4:PRINT# 4:CLOSE 4
```

verschijnt de in regel 20 gePRINTe tekst ook werkelijk op papier!

De reden hiervoor is echter niet zo ingewikkeld als men misschien wel denkt. Een printer heeft namelijk, net zoals een diskdrive, de beschikking over een interne buffer. Als er tekens naar de printer worden gezonden zullen die in eerste instantie in die buffer worden opgeslagen.
Pas als er een einde-regel code ontvangen wordt – met andere woorden, het CR-stuurteken – zal de inhoud van de buffer worden afgedrukt. Dat levert in de praktijk tijdwinst op, want anders zou in bepaalde programma's de printkop steeds weer voor een enkel tekentje in werking moeten komen. In het tweede voorbeeldje werd de PRINT# opdracht in regel 20 door een puntkomma afgesloten, hetgeen inhoudt dat er later nog meer op dezelfde regel zal worden afgedrukt. De tekst 'en deze tekst niet!' werd dan ook keurig in de printerbuffer opgeslagen. Maar omdat onmiddellijk daarna het programmaatje afgelopen was bleef die tekst daar ook en verscheen helemaal niet op papier.
Pas toen er alsnog, in de direct-mode, een PRINT-opdracht gegeven werd die wel met een CR werd afgesloten – geen punt-komma aan het einde dus – werd de buffer-inhoud alsnog afgedrukt. Ondanks het feit dat het bij de printer behorende bestand intussen afgesloten was bleek die buffer-inhoud gewoon bewaard te worden in de printer.

Printer-buffers zijn natuurlijk niet eindeloos groot. In de meeste gevallen zal een printer een buffertje bezitten dat niet veel groter is dan een enkele printregel.
As die buffer 'overloopt' wordt de inhoud automatisch afgedrukt, zodat er geen tekst verloren kan gaan doordat we meer tekens na elkaar printen – zonder een CR-code ertussen – dan er op een regel passen. Slechts aan het einde van een programma moeten we even opletten dat er geen tekens in de buffer achterblijven.

Behalve deze regel-buffer kan er nog een ander soort printer-buffer aanwezig zijn, die veel en veel groter is. Deze buffers – die in de printer ingebouwd kunnen zijn maar ook als losse kastjes verkocht worden – hebben een heel andere functie. In de meeste gevallen is de computer namelijk veel sneller dan een printer, zodat men soms lang moet zitten wachten tot de printer klaar is met bijvoorbeeld een listing. Al die tijd is de computer bezet. Zo'n extra buffer nu gedraagt zich naar de computer toe alsof het de printer zelf is, maar alles wat afgedrukt moet worden wordt echter in het buffergeheugen opgeslagen. Dat gaat natuurlijk sneller, zodat de computer al snel weer vrij is voor andere taken. De gebufferde tekst ondertussen wordt aan de andere kant van de buffer op de normale snelheid aan de eigenlijke printer doorgegeven, met een snelheid die de printer kan verwerken.

6/3.3

Matrixprinters

Inhoud

6/3.3.1

MPS 1000

De MPS 1000 is een zeer recente matrixprinter van Commodore. De printer is zodanig gebouwd dat niet alleen de Commodore 64 ondersteund wordt, maar ook de 128, de Amiga en alle IBM-PCs en compatibles. Dit maakt de machine uitermate geschikt voor mensen met een 64 en een Amiga, of bijvoorbeeld mensen met een 128 en een IBM-PC. Om dit te verwezenlijken heeft de MPS 1000 twee aansluitingen, een IEC-(seriële) bus aansluiting en een Centronics aansluiting. De Centronics aansluiting is een parallelle aansluiting die door de Amiga en de IBM PC gebruikt wordt. Het handboek van de MPS 1000 vermeldt dat beide aansluitpoorten NOOIT tegelijk gebruikt worden. Sluit dus niet twee computers tegelijk op de MPS 1000 aan!

Om zowel de 64 als de IBM PC te kunnen ondersteunen heeft de MPS twee modes waarin gewerkt kan worden: de Commodore mode en de IBM mode. In de Commodore mode is de 1000 bijna identiek aan de MPS 801, in de IBM mode is de MPS 1000 een EPSON LX-86, welke IBM 5152-Plus compatible is. In beide modes is een Near Letter Quality tekenset beschikbaar, dit maakt de printer zeer geschikt voor correspondentiedoeleinden. De 128 gebruikt de Commodore mode van de MPS 1000, als extra worden dan de Europese tekensets, zoals die op de 128 aanwezig zijn, ondersteund.

Uiteraard zullen we ons hier beperken tot de mogelijkheden die vanuit de 64 te gebruiken zijn, dit zijn er echter nogal wat. Dat komt omdat de IBM mode ook door de 64 gebruikt kan worden. Ook al heeft de 64 geen Centronics uitgang, de printer kan zo geprogrammeerd worden dat de IBM mode door de seriële ingang of door de Centronics ingang bestuurd wordt. Het omgekeerde, de Commodore mode via de Centronics ingang besturen, is echter niet mogelijk (maar ook niet noodzakelijk).

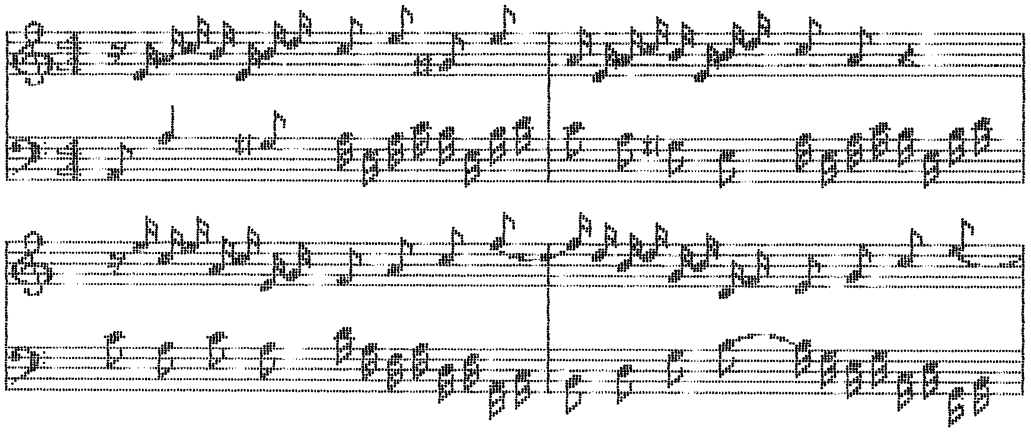
Dip-switches

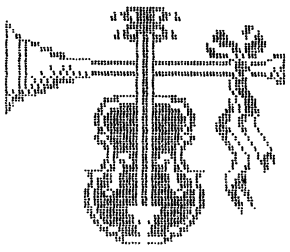
Zoals elke printer heeft de MPS 1000 ook een aantal dip-switches, deze dienen voor het gebruik in Commodore mode allemaal uit te staan. In de IBM mode moet switch 1-1 aan staan wanneer de Centronics aansluiting gebruikt wordt, wordt de IBM mode vanuit de 64 gebruikt (seriële bus) dan dient daarnaast ook switch 1-8 aangezet te worden, met eventueel daarbij (voor de auto-linefeed) switch 1-3.

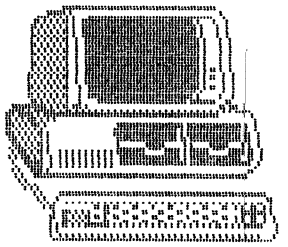
Commodore mode

Zoals gezegd gedraagt de MPS 1000 zich in de Commodore mode bijna als een MPS 801/1520. Dit maakt de printer geschikt om met allerlei commercieel verkrijgbare programma's te werken zoals tekstverwerkers en tekenprogramma's. De volgende twee plaatjes komen uit respectievelijk het printprogramma 'Printmaster' en het muziekprogramma 'The Music Shop'.

3.3 Matrixprinters







Echter ook minder ambitieuze uitvoer kan met de MPS 1000 verwezenlijkt worden. In onze eigen BASIC-programma's zullen we bijvoorbeeld regelmatig van tabellen gebruik willen maken. Ook programmalistings zijn de orde van de dag, evenals af en toe eens een verklarend verhaal bij de uitvoer.

De mogelijkheden van de Commodore mode zijn, vergeleken bij de IBM mode, ronduit beperkt. Hierop is één positieve uitzondering en dat is de mogelijkheid om getallen te formatteren. Hierop zullen we later uitgebreid terugkomen. Laten we eerst eens zien wat er aan effecten in de Commodore mode te verkrijgen is, gebruikmakend van *besturingscodes*.

Normale tekst

Wanneer de printer aangezet wordt zal deze normale tekst gaan printen zonder enige franje. Het volgende commando (geen regelnummers)

```
OPEN 4,4: PRINT#4,"DIT IS TEKST": CLOSE4
```

heeft dit als uitvoer:

```
DIT IS TEKST
```

Mocht u echter niet alle dip-switches uit hebben staan dan kan het resultaat anders zijn.

3.3 Matrixprinters

Brede tekst
Brede tekst wordt verkregen door gebruik te maken van besturingscode 14.
Deze kan als een CHR$ verstuurd worden, maar ook als control-teken binnen een string. Daar CHR$(14) overeenkomt met CTRL-N hebben de volgende twee commando's hetzelfde resultaat:

```
OPEN 4,4:PRINT#4,CHR$(14)"BREED": CLOSE 4
OPEN 4,4:PRINT#4,"[CTRL-N]BREED": CLOSE 4
```

Op de plaats van [CTRL-N] moet dan de CTRL-toets ingedrukt worden, tegelijkertijd met de letter N. Op het scherm verschijnt een geïnverteerde N. Het resultaat is in beide gevallen

```
BREED
```

Het nadeel van de breed-mode (enlarged of double-width in vaktermen) is dat deze aan blijft staan. Een volgend PRINT-commando heeft tot gevolg dat nieuwe tekst ook breed verschijnt! Dit kan natuurlijk nooit de bedoeling zijn, er is dan ook een besturingscode om het brede schrift weer uit te zetten: CHR$(15) of CTRL-O:

```
OPEN 4,4:PRINT#4,"BREED [CTRL-O] NORMAAL": CLOSE 4
```

Het gevolg is:

```
BREED NORMAAL
```

Near Letter Quality tekst
Het leukste van de MPS 1000 is natuurlijk de Near Letter Quality, ook wel NLQ genoemd. Deze wordt ingeschakeld door code 31, oftewel CHR$(31):

```
OPEN 4,4:PRINT#4,CHR$(31)"NLQ-TEKST": CLOSE 4
```

```
NLQ-TEKST
```

Ook de NLQ-tekst blijft ingeschakeld, totdat deze met CHR$(159) weer uitgeschakeld wordt:

```
OPEN 4,4:PRINT#4,"NOG NLQ" CHR$(159)" EN WEER NORMAAL": CLOSE 4
```

```
NOG NLQ EN WEER NORMAAL
```

Om bijvoorbeeld een listing in NLQ te krijgen zijn de volgende commando's nodig:

```
OPEN 4,4:CMD 4,CHR$(31): LIST
```

Opmerking: het is mogelijk om brede NLQ-letters te krijgen door zowel CHR$(14) als CHR$(31) naar de printer te sturen.

Geïnverteerde tekst
De MPS 1000 kan ook geïnverteerde tekst afdrukken, dit gaat met besturingscode 18 (CTRL-R oftewel RVS-ON):

```
OPEN 4,4:PRINT#4,"[RVS ON]GEINVERTEERD": CLOSE 4
```

```
GEINVERTEERD
```

De inverse-mode hoeft niet expliciet afgesloten te worden omdat dit automatisch aan het eind van een regel gebeurt (na een CR/LF om precies te zijn). Er is echter wel een aparte code (146, RVS-OFF) voor, deze moet gebruikt worden wanneer midden in een regel gestopt moet worden met de geïnverteerde letters:

```
OPEN 4,4:PRINT#4,"[RVS ON]AAN [RVS OFF] UIT":CLOSE 4
```

3.3 Matrixprinters

WEKA UIT

Kleine letters
In alle voorbeelden tot nu toe heeft de printer steeds in hoofdletters geantwoord. Er zijn echter twee manieren om de printer kleine letters te laten printen, de gebruikte tekenset komt dan overeen met de kleine/hoofdletters tekenset (set 2) op de Commodore 64.
Een van de manieren om de printer kleine letters te laten printen is met behulp van besturingscode 17 (CRSR DOWN). De tweede manier komen we later tegen. Om een tekst in hoofd- en kleine letters op de printer te krijgen wordt dus het volgende gedaan:

```
OPEN 4,4:PRINT#4,"WEKA U[CRSR
DOWN]UITGEVERIJ": CLOSE 4
```

De uitvoer is dan als volgt:

```
WEKA Uitgeverij
```

Ook de kleine letter mode wordt aan het einde van een regel automatisch uitgeschakeld. Dat is dus niet zo handig bijvoorbeeld bij het maken van een listing in kleine letters: alleen de eerste regel verschijnt dan in het klein! De tweede methode is echter permanenter van aard en komt zometeen aan bod.
Het uitschakelen van de kleine letter mode midden in een regel is ook mogelijk, en wel met besturingscode 145 (CRSR UP):

```
OPEN 4,4:PRINT#4,"[CRSR DOWN]
KLEIN [CRSR UP]GROOT":CLOSE 4
```

Dubbele letters (boldface)
Extra donkere letters worden verkregen door teksten tweemaal over elkaar heen te drukken. Dit is vrij bewerkelijk, maar het resultaat is wel mooi. Het volgende programma illustreert dit:

```
10 OPEN 4,4
20 PRINT#4,"BOLDFACE/NORMAAL
"CHR$(141);
30 PRINT#4,"BOLDFACE"
40 CLOSE 4
```

De uitvoer is dan:

```
BOLDFACE/NORMAAL
```

Het over elkaar PRINTen wordt gerealiseerd door een RETURN zondfer lijnfeed (code 141) naar de printer te sturen. Het gevolg is dat de printerkop wel naar het begin van de regel teruggaat, de papierrol schuift echter geen regel op. Het resultaat is dat tekst over elkaar heen gedrukt kan worden, met donkere letters als gevolg.

Tabulatie
Tabulatie is het verplaatsen van de printerkop naar een bepaalde kolom. Dit is uiteraard bij het maken van tabellen zeer handig. De MPS 1000 voert een tabulatie (kortweg tab) uit na het ontvangen van besturingscode 16 (CTRL-P). Daarna moet een tweecijferig getal volgen dat de postitie aangeeft waarnaartoe geTABt moet worden. Het volgende voorbeeld verduidelijkt een en ander:

```
OPEN 4,4:PRINT#4,"[CTRL-P]05PO-
SITIE 5[CTRL-P]25POSITIE
25":CLOSE4
```

klein GROOT

POSITIE 5

POSITIE 25

3.3 Matrixprinters

Het getal wat direct op de CTRL-P volgt geeft aan waar de printer verder PRINTen moet. Is dit getal kleiner dan 10 dan moet er een 0 voor gezet worden. Let er op dat het niet juist is om een CHR$ te sturen met de TAB-positie! Dit is dus fout:
OPEN 4,4:PRINT#4,"[CTRL-P]"CHR$ (10) "POSITIE 10":CLOSE 4
De fout ligt in het feit dat CHR$(10) niet gelijk is aan "10".

Bovenstaande maakt het ingewikkeld om naar een willekeurige (variabele) TAB-positie toe te springen. Immers, positie X wordt niet met een CHR$(X) bereikt maar met twee CHR$'s die uit X afgeleid moeten worden. De volgend subroutine doet dit:

```
1000 PRINT#4,CHR$(16)CHR$(X/10+48)CHR$(X-10*INT(X/10)+48);
1010 RETURN
```

Door nu de volgende regels toe te voegen kan de routine getest worden:

```
10 OPEN 4,4
20 INPUT X:IF X<1 OR X>80 THEN CLOSE 4:END
30 GOSUB 1000:PRINT#4,"POSITIE";X
40 GOTO 20
```

Secondary addresses
Tot zover het verhaal wat er in de tekstmode met normale besturingscodes bereikt kan worden. De MPS 1000 kent echter ook nog een aantal secondary addresses (SA's), getallen die extra in het OPEN-commando gedefinieerd worden. Dit OPEN-commando ziet er dan als volgt uit: OPEN 4,4,SA. Deze SA's bepalen een aantal extra mogelijkheden van de printer, zoals bijvoorbeeld het continu in kleine letters PRINTen. De MPS 1000 kent om precies te zijn 10 SA's, ze worden alle 10 hier behandeld.

SA=0, normale PRINTmode
Dit SA hebben we stilzwijgend al gebruikt, indien een OPEN-commando geen secondary adress meekrijgt dan neemt de printer aan dat dit 0 moet zijn. het gevolg is dat de printer precies zo reageert als hierboven besproken is.

SA=1, PRINT geformatteerde data en
SA=2, definieer formattering
Deze twee secondary adresses worden meestal tegelijk gebruikt en worden gebruikt om data netjes geformatteerd af te drukken. In feite komt het er op neer dat de printer met het PRINT USING commando gestuurd wordt, iets wat de 64-BASIC mode mist. Om de mogelijkheden van geformatteerde data te tonen kan het beste dit kleine programma geRUNd worden:

```
10 open 1,4,1:open 2,4,2:open 4,4,7
20 print#4,"Artikel          Code      Aantal      Prijs"
30 print#4,"──────────────────────────────────────────────────"
40 print#4
50 print#2,"aaaaaaaaaaaaaa  aaaaaa     999        Dfl. 9999.99"
60 readn$,c$,a,p
70 ifa<0thenclose4:close2:close1:end
80 print#1,n$"■"c$"■"a"■"p
90 goto60
100 data "Boek",100000,12,29.95
110 data "Drop",111111,5,.95
120 data "Fiets",222222,2,395
130 data-1,-1,-1,-1
ready.
```

3.3 Matrixprinters

De uitvoer ziet er dan als volgt uit:

```
Artikel      Code      Aantal  Prijs
-------------------------------------

Boek         100000        12    f.    29.95
Drop         111111         5    f.     0.95
Fiets        222222         2    f.   395.00
```

Een formateringsveld kan uit de volgende tekens opgebouwd zijn: 9,Z,$,S,.,-,A,spatie
Een 9 duidt aan dat op die plaats alleen cijfers worden afgedrukt, een Z duidt aan dat een inleidende 0 eventueel wordt afgedrukt. Het dollartekentje levert een $ op, dit kan fixed of flaoting zijn. Fixed betekent dat de plaats vast staat, floating zet de dollar direct voor het eerste cijfer. Het rijtje voorbeelden dat zometeen volgt maakt dit duidelijk.

De punt bepaalt de plaats van de decimale punt, het minnetje geeft aan of een negatief getal door een min-teken gevolgd wordt. Staat er een S in de definitie dan levert dit het teken van het getal op die plaats op, dit is een + of een –. De A geeft aan dat op die positie een willekeurige letter mag staan en de spatie tenslotte geldt als veld-scheider.

Een formatteringsstring kan tenslotte ook gewone tekens (cijfers, letters, grafisch) bevatten, deze dienen dan wel door het RVS-symbool voorafgegaan te worden.
Dientengevolge kunnen geïnverteerde tekens niet binnen een formatteringsstring gebruikt worden.

Formatteringsvoorbeelden

FORMAT	DATA	RESULTAAT
AAA	hallo	hal
$$$$$	43	$43
$9999	43	$ 43
999.9	120	120.0
999.9	12	12.0
ZZZ.9	12	012.0
999.9	1000	***.*
999.9	12.12	12.1
S999	145	+145
S999	−145	−145
9999−	145	145
9999−	−145	145−
ZZZZZ	12	00012
999.99	12.33	12.33
999.99	1.33	1.33
999.99	.33	0.33
[RVS ON]f99.99	12.3	f12.30
[RVS ON]£999.99	14.123	£ 14.12

In de laatste twee voorbeelden is het gebruik van vaste tekens (f en £) te zien. Deze worden dus door het RVS ON-symbool vooraf gegaan.
Denk er verder aan dat formateringsdefinities altijd via SA=2 gestuurd worden en de data zelf via SA=1.

Om van het ene veld naar het volgende veld te 'springen' moet besturingscode 29 (CURSOR RIGHT) gebruikt worden. Deze code geeft dus het einde van een dataveld aan. Een probleem doet zich voor wanneer een veld door spaties voorafgegaan wordt, deze worden door de printer namelijk 'gestript'.
GeSHIFTe spaties (code 160) worden echter ongemoeid gelaten, strings met 'leading spaces' moeten dus met geSHIFTe spaties beginnen.

3.3 Matrixprinters

SA=3, instellen aantal regels per pagina
De MPS 1000 is in staat om automatisch een aantal lege regels aan het eind van een bladzijde te PRINTen. Dit is vooral handig bij het maken van programmalistings, er wordt dan niet door de perforatie van het kettingformulier heen gePRINT.
Uiteraard dient er rekening met de bladzijlengte gehouden te worden, deze is 11 of 12 inch (66 of 72 regels). Welke lengte de printer aanhoudt is alleen te regelen met dipswitch 2-1, dit kan dus niet softwarematig ingesteld worden.
Zoals gezegd is het aantal regels dat per pagina gePRINT wordt dus wel in te stellen en wel via SA=3. Dit gaat dan als volgt:

```
OPEN 3,4,3:PRINT#3, CHR$(AR):
CLOSE 3
```

De variabele AR bevat dan bijvoorbeeld het aantal te PRINTen regels per bladzijde, bijvoorbeeld 56 of 60. De overgebleven regels laat de printer blank. De verstekwaarde (default) is 60, wordt er dus geen waarde opgegeven dan worden er automatisch 6 regels (bij 11-inch papier) wit gelaten.

Voordat de MPS 1000 echter rekening zal gaan houden met het aantal ingestelde regels dient de zgn. 'pagina mode' geactiveerd te worden. Dit gebeurt door het versturen van code 147 (SHIFT-CLR/HOME). Zonder het versturen van dit pagina-teken zal de printer dus normaal alle regels achter elkaar afdrukken! Het uitzetten van de paging mode gebeurt door het zenden van code 19, dit is hetzelfde als CLR/HOME zonder SHIFT.
Bovenstaand OPEN-commando wordt dus als volgt geactiveerd:

```
OPEN
4,4:PRINT#4,CHR$(147):CLOSE 4
```

Het uitzetten van de paging mode en dus het opheffen van het aantal regels per pagina gaat dan zo:

```
OPEN
4,4:PRINT#4,CHR$(19):CLOSE 4
```

SA=4, schakel foutmeldingen in en
SA=9, schakel foutmeldingen uit
Het nadeel van printers is dat ze meestal geen foutmeldingen kunnen geven. Toch kan het soms handig zijn om de printer te laten vertellen waarom bepaalde uitvoer er niet uitziet zoals die er uit hoort te zien. De MPS 1000 is in staat om foutmeldingen af te drukken die een handleiding kunnen zijn bij het zoeken van die fout in het desbetreffende programma.

Het inschakelen van de foutmeldingsmode gaat via SA 4, dus als volgt:

```
OPEN 4,4,4:PRINT#4:CLOSE 4
```

Het uitschakelen van de foutmeldingsmode gaat via SA 9, aldus:

```
OPEN 9,4,9:PRINT#9:CLOSE 9
```

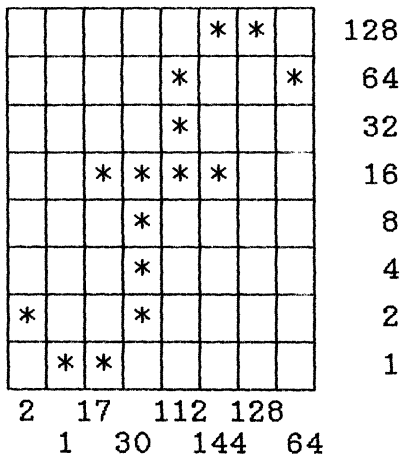
De MPS 1000 kent in totaal 6 foutmeldingen, deze hebben de volgende form:
PE:X
waarbij X een letter is die de volgende waardes kan hebben:
L – aantal regels per pagina (via SA=3) klopt niet; dit dient van 1 t/m 127 te lopen
C – fout commando; het secondary address wat gegeven is bestaat niet
M – foute data; een tekststring werd ontvangen terwijl de definitiestring (zie SA=2) een numerieke waarde aangeeft
E – foute exponent; de exponent van een getal dat via SA=1 geformatteerd gaat worden dient uit twee cijfers te bestaan en door een + of een – voorafgegaan te worden. Verder dient de absolute waarde van de mantisse (M) in het bereik 0‹=M‹10 te liggen. Voorbeelden van geldige data zijn: 1.23E+03, –0.14232E+47 en 9.999E–99

3.3 Matrixprinters

F – foute formatteringsdefinitie; de definitiestring bevat onbekende symbolen of bevat fouten in de syntaxis.
T – beëindigingsfout; voordat een nieuwe SA 'gebruikt' kan worden moet de printer eerst een CR (code 13), LF (code 10) of CR/LF-combinatie ontvangen hebben. Aan het einde van een PRINT-opdracht (zonder puntkomma aan het eind) gebeurt dit automatisch.

SA=5, definieer eigen teken
De MPS 1000 is in staat om de definitie van een eigen teken (8*8 puntjes) te ontvangen en deze ook af te drukken. Dit teken wordt met SA=5 gedefinieerd en met code 254 afgedrukt. De tekendefinitie dient met de hand uitgerekend te worden, dit gaat als volgt:
Neem een stuk ruitjespapier en baken een gebied van 8 blokjes hoog bij 8 blokjes breed af. Teken hier het te definiëren teken in door sommige blokjes zwart te maken en andere niet (zoals bij sprites). Geef de bovenste rij de waarde 128, de tweede rij de waarde 64 en de derde, vierde, vijfde, zesde, zevende en achtste rij respectievelijk de waardes 32, 16, 8, 4, 2 en 1. Door nu iedere kolom 'op te tellen' (een puntje levert de rijwaarde op) ontstaan er in totaal 8 getallen (er zijn immers 8 kolommen). Deze 8 getallen zijn nu de data die naar de printer toegestuurd moet worden. Kijk maar eens naar het volgende figuur en het programma wat daar op volgt, dit maakt een en ander een stuk duidelijker.

Het versturen van de data gaat dus via SA=5, het afdrukken van het teken gaat via CHR$(254). Per regel kan slechts één eigen teken gedefinieerd zijn, verschillende eigen tekens per regel moeten dus met 'overprinting' gerealiseerd worden.




```
Dit boek kost ƒ 99,95
```

```
1000 data 2,1,17,30,112,144,128,64
1010 for i=1 to 8
1020 read a:a$=a$+chr$(a)
1030 next
1040 open 5,4,5:print#5,a$:close5
1050 open 4,4,7
1060 print#4,"Dit boek kost "chr$(254)" 99,95"
1070 close 4
ready.
```

SA=6, instellen ruimte tussen regels
Secondary address 6 wordt gebruikt om aan te geven hoeveel wit er tussen elke regel gelaten moet worden of liever gezegd hoever het papier moet verschuiven om de volgende regel af te drukken. De waarde die verzonden wordt geeft het aantal stapjes aan, dit dient kleiner dan 128 te zijn.
Verder zitten er in 1 inch 216 stapjes zodat een waarde van 36 tot gevolg heeft dat er 6 regels per inch (verstekwaarde) afgedrukt worden. Wordt de waarde 72 gemaakt dan verschijnen er 216/72=3 regels per inch.
Het juiste commando om de stapgrootte

(SG) in te stellen is:
OPEN
6,4,6:PRINT 6,CHR$(SG):CLOSE 6

SA=7, PRINT in kleine letter mode
Dit SA wordt gebruikt indien de printer permanent in de kleine letter mode gezet moet worden. Om een listing in kleine letters te krijgen is dan het volgende commando nodig:
OPEN 4,4,7:CMD 4:LIST
gevolgd door
PRINT 4:CLOSE 4

SA=10, reset printer
Dit SA wordt gebruikt om de printer te resetten, de printer gedraagt zich dan precies als na het aanzetten:
OPEN 10,4,10:PRINT 10:CLOSE 10

Graphics

De MPS 1000 kent ook een grafische mode, dit is aan het begin van dit hoofdstuk al geïllustreerd met behulp van enige plaatjes, Zelf kunnen we ook plaatjes gaan maken, dit is echter wel vrij bewerkelijk. We zullen ons hier dan ook beperken tot de diverse commando's en een simpel voorbeeld.

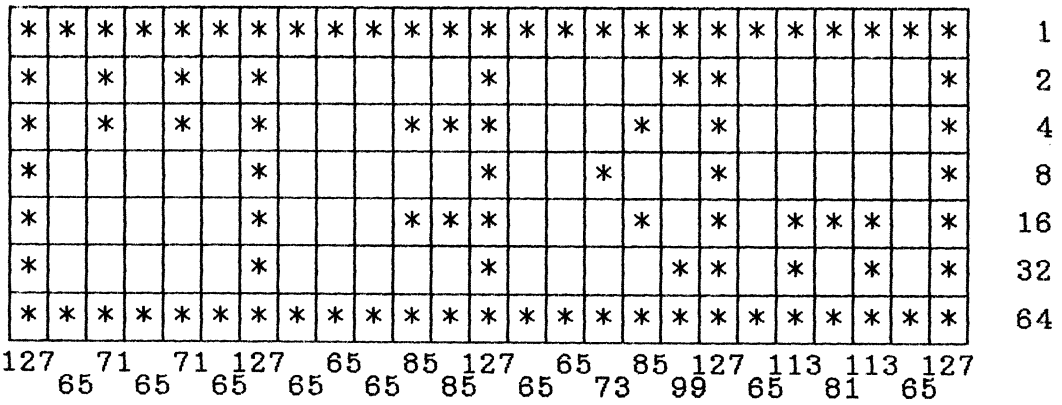
Opbouw
Een hires-plaatje op de MPS 1000 is opgebouwd uit lijnen van puntjes. Per beweging van de printkop kunnen 7 lijnen tegelijk getekend worden; dit is een niet zo handelbare waarde (denk aan hires screendumps), de eerste Commodore-printers werkten echter zo en de 1000 moet zich hier dus aan houden. In de IBM-mode kan de 1000 8 lijnen tegelijk tekenen, wanneer dus zelf geëxperimenteerd wordt is deze mode wellicht wat aantrekkelijker.
Horizontaal zijn er 480 puntjes te verdelen, de MPS 1000 kent echter ook een double density mode, hierin bevat een lijn 960 puntjes. We zullen ons verhaal beperken tot de single density mode, het gaat uiteraard ook op voor de double-mode.

Net zoals bij het zelf definieerbare teken van enige bladzijden terug dient de data van links naar rechts aangeboden te worden. De data is nu echter 7 bits groot omdat de printer dus maar 7 lijnen tegelijk tekent. Ook het te tekenen plaatje moet uitgerekend worden door de boven elkaar liggende puntjes bij elkaar 'op te tellen', de 'waarde' van de lijnen is nu echter anders: de bovenste lijn heeft de waarde 1, de onderste lijn heeft de waarde 64.
Verder dient elke gevonden waarde met 128 verhoogd te worden. Als voorbeeld zullen we het WEKA-logo 'uitrekenen' (zie blz.10).

Het is duidelijk dat grafiek op de MPS 1000 geduld vergt! We zullen echter even doorzetten en de data nu naar de printer versturen.

Wat is er nu precies gebeurd? Allereerst wordt de grafische data in een string, A$, opgeborgen. Deze string wordt, tezamen met een CHR(8), naar de printer gestuurd. CHR$(8) is de besturingscode die de grafische mode inschakelt; deze blijft net zolang ingeschakeld tot een teken met een waarde kleiner dan 128 ontvangen wordt. Het is dus noodzakelijk om de grafische bytes met 128 te verhogen. De CHR$(15) wordt gebruikt om een normale linefeed van een hele regel te genereren, wordt deze weggelaten dan schuift de printer slechts 7 puntjes (7/72 inch) op. Voor complete plaatjes is dit uiteraard zeer handig, omdat er anders stukken wit in verschijnen. Hier is het echter niet nodig, vandaar dus die CHR$(15).

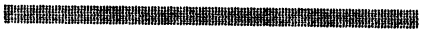
3.3 Matrixprinters




```
10 OPEN 4,4
20 FOR I=1 TO 25
30 READ A:A$=A$+CHR$(A+128)
40 NEXT
50 PRINT#4,CHR$(8);A$;CHR$(15)
60 CLOSE 4
70 DATA 127,65,71,65,71,65,127,65,65,65,85,85,127,65
80 DATA 65,73,85,99,127,65,113,81,113,65,127
READY.
```

Het is duidelijk, op een beetje geduld na is het kinderspel om de grafische mode van de MPS 1000 te gebruiken. Vooral de volgende extra commando's maken het de programmeur nóg iets eenvoudiger.

Extra commando's
De MPS 1000 kent nog een paar extra commando's om het werken met grafiek te veraangenamen, dit zijn:
CHR$(26)CHR$(N)CHR$(DATA);
– herhaal grafische data
Wanneer bovenstaand commando in een grafische string opgenomen wordt zal de data die door DATA aangeduid wordt N (0 t/m 255) maal herhaald worden; dit is zeer nuttig bij het maken van staafdiagrammen. Voorbeeld:
CHR$(8)CHR$(26)CHR$(13)
CHR$(255)
Het resultaat is:



CHR$(27)CHR$(16)CHR$
(HB)CHR$(LB);
– grafische tab
Dit commando wordt gebruikt om de printkop in puntjes te verschuiven en niet in posities, zoals bij het eerdere tab-commando. De waarde van de tab-positie

loopt van 0 t/m 480 en moet dus in twee bytes verzonden worden: HB en LB. De waarde van LB wordt gevonden door de tab-positie met 255 te AND-en, de waarde van HB is gelijk aan tab-positie gedeeld door 256. Voorbeeld:
CHR$(27)CHR$(16)CHR$(0)
CHR$(100)CHR$(8)CHR$255)
CHR$(15)
De uitvoer wordt dan:

(l)

CHR$(1) en CHR$(2) – bepaal density
Zoals gezegd heeft de MPS 1000 een single en een double density grafische mode, CHR$(2) (CTRL-B) schakelt de dubbele dichtheid in, CHR$(1) schakelt weer over naar enkele dichtheid.

IBM mode
In de IBM mode van de MPS 1000, gedraagt de printer zich als een IBM 5152-Plus of als een EPSON LX-86 printer. In deze mode zijn veel meer verschillende letter-types beschikbaar dan in de hierboven beschreven Commodore mode. In de IBM mode heeft u de beschikking over 10 letter-types: pica, elite, compressed, near letter quality, emphasized, double strike, elarged, underlined, superscript en subscript. Deze kunnen in combinatie met elkaar ook weer voor een nieuw letter-type zorgen, zoals enlarged near letter quality. Het complete totaal van verschillende letter-types wordt hierdoor vrij groot. Dit heeft tevens vaak tot gevolg, dat men geen overzicht heeft van alle mogelijkheden van deze printer. In deze sectie zal een overzicht worden gegeven van de 10 standaard letter-types, met hun besturingscodes. Ook zal van een aantal letter-types de combinaties worden beschreven.

De dip-switches
Zoals al eerder beschreven, moeten in de IBM mode de dip-switches 1-1, 1-3 en 1-8 aan staan. Als deze worden veranderd dient de printer uit te staan. Bij het aanzetten van de printer 'leest' de printer de stand van de dip-switches en zal zich gaan gedragen zoals is ingesteld.

Normale of tekst pica punten
Na het aanzetten van de printer zal deze normale tekst gaan printen. De volgende basic statements:

```
OPEN4,4:PRINT#4,

"NORMAAL OF PICA":CLOSE4
```

Geeft als uitvoer:

```
NORMAAL OF PICA
```

Elite printen
Het elite letter-type is iets kleiner dan het pica letter-type. Het resultaat hiervan is dat er meer tekens op een regel kunnen (12 tekens per inch, tegenover 10 tekens per inch in pica-mode) en dat de tekens iets zwarter zijn. De printer wordt in elite mode gezet door een ESCAPE':' naar de printer te sturen.

```
OPEN4,4:PRINT#4,CHR$(27)":";

:PRINT#4,"DIT IS ELITE":CLOSE4
```

Met als resultaat:

```
DIT IS ELITE
```

3.3 Matrixprinters

De elite-mode kan alleen worden uitgeschakeld, door een andere mode aan te zetten. Zoals bijvoorbeeld de compressed-mode.

Compressed of condensed printen
In de compressed-mode worden, zoals de naam al aangeeft, de tekens dicht op elkaar afgedrukt. De tekens staan hier nog dichter op elkaar dan in de elite mode (17.16 tekens per inch). Door de ASCII-Code 15 naar de printer te sturen komt deze in de compressed-mode.

```
OPEN4,4:PRINT#4,CHR$(15);

:PRINT#4,"DIT IS COMPRESSED OF

CONDENSED":CLOSE4
```

```
DIT IS COMPRESSED OF CONDENSED
```

De mode is uitschakelbaar door het AS CII 18 teken te versturen:

```
OPEN4,4:PRINT#4,

CHR$(18);:CLOSE4
```

Near letter quality printen
Near letter quality of NLQ is een veel gebruikt letter-type. Dit letter-type wordt veelal gebruikt als vervanging van het normale schrijfmachine letter-type, juist omdat NLQ, qua kwaliteit, zoveel op een echter letter lijkt. Uiteraard met dien verstande dat een schrijfmachine letter toch veel mooier is, maar dat men bij de conventionele schrijfmachine niet de gemakken heeft van de moderne tekstverwerker. De NLQ letter is even groot als de PICA letter (namelijk 10 tekens per inch). De near letter quality mode wordt aangezet door een ESCAPE 'x' 1 naar de printer te sturen en wordt uitgezet door een ESCAPE 'x' 0 naar de printer te sturen.

```
OPEN4,4:PRINT#4,CHR$(27)CHR$(120)
CHR$(1):PRINT#4,"NEAR LETTER
QUALITY TEKST"
PRINT#4,CHR$(27)CHR$(120)CHR$(0)
:CLOSE4
```

```
NEAR LETTER QUALITY TEKST
```

Dubbele letters
De MPS 1000 kent in eerste instantie vier verschillende letter-types. Deze letter-types kunnen op de normale manier afgedrukt worden, zoals hier boven is beschreven. Ze kunnen echter ook anders worden afegdrukt, door bijvoorbeeld de letters twee maal af te drukken. De letters voor de tweede maal afdrukken kan op een aantal verschillende manieren gebeuren: er exact bovenop, het papier iets verschoven of de print-kop iets verschoven. Al deze verschillende effecten worden hieronder beschreven.

Emphasized printen
In de emphasized mode wordt elk puntje van een teken, tweemaal afgedrukt, waarbij het tweede puntje iets links van het eerste puntje staat. Het gevolg hiervan is dat de letters iets breder en zwarter worden. De emphasized mode werkt alleen in de pica en near letter quality-modes. De emphasized mode wordt door middel van een ESCAPE 'E' aangezet en met een ESCAPE 'F' uitgezet.

```
OPEN4,4:PRINT#4,CHR$(27)"E";
:PRINT#4,"DIT IS PICA EMPHASIZED"
PRINT#4,CHR$(27)CHR$(120)CHR$(1)
:PRINT#4,"DIT NEAR LETTER QUALITY
EMPHASIZED"
PRINT#4,CHR$(27)CHR$(120)CHR$(0)
CHR$(27)"F":CLOSE4
```

```
DIT IS PICA EMPHASIZED

DIT NEAR LETTER QUALITY EMPHASIZED
```

3.3 Matrixprinters

Double-strike printen
In de double strike-mode wordt elk puntje van een teken, tweemaal afgedrukt. Het verschil met de emphasized mode is dat het tweede puntje niet links van het eerste komt, maar onder het eerste. Double strike kan met elk ander letter-type worden gecombineerd, maar niet met near letter quality. De double-strike mode wordt met een ESCAPE 'G' aangezet en met een ESCAPE 'H' uitgezet.

```
OPEN4,4:PRINT#4,CHR$(27)"G";
:PRINT#4,"DIT IS PICA
DOUBLE-STRIKE"
PRINT#4,CHR$(27)"H":CLOSE4
```

```
DIT IS PICA DOUBLE-STRIKE
```

Semi-double-strike printen
Met het semi-double-strike printen wordt elk puntje van een teken óók tweemaal afgedrukt. Het tweede puntje wordt exact boven het eerste afgedrukt. Dit is GEEN printer mode maar een software-matig letter type. Door eerst de gewenste regel af te drukken, het papier van de printer niet omhoog te draaien en dan dezelfde regel weer af te drukken is dit letter type te verkrijgen. Door een ESCAPE '5' ASCII 0 naar de printer te versturen, wordt de auto linefeed mode uitgeschakeld, dan 'scrollt' de printer dus niet meer. Met een ESCAPE '5' ASCII 1, wordt de auto linefeed mode weer aangezet, dan kan de printer dus wèl weer 'scrollen'.

```
OPEN4,4:PRINT#4,CHR$(27)"5"
CHR$(0);:PRINT#4,
"SEMI-DOUBLE-STRIKE"
PRINT#4,CHR$(27)"5"CHR$(1);
:PRINT#4,"SEMI-DOUBLE-STRIKE"
:CLOSE4
```

```
SEMI-DOUBLE-STRIKE
```

Enlarged printen
De enlarged mode geeft brede tekst. In plaats van één puntje worden twee puntjes afgedrukt. Dit gebeurt naast elkaar en wel zodanig dat het teken ongeveer twee maal zo breed wordt. Alle lettertypes kunnen verbreed worden, dus tijdens het printen van elk lettertype kan de enlarged mode worden aangezet. Dit gebeurt door ESCAPE 'W1' naar de printer te sturen. Met de ESCAPE 'W0' wordt de enlarged mode uitgeschakeld.
Een voorbeeld van verbrede pica en near letter quality tekst is hieronder te vinden. Andere voorbeelden zijn eenvoudig zelf te maken.

```
OPEN4,4:PRINT#4,CHR$(27)"W1";:
PRINT#4,"PICA, ENLARGED"
PRINT#4,CHR$(27)CHR$(120)CHR$(1);
:PRINT#4,"NEAR LETTER QUALITY,
ENLARGED"
PRINT#4,CHR$(27)"W0"CHR$(27)
CHR$(120)CHR$(0);:CLOSE4
```

```
PICA, ENLARGED
NEAR LETTER
QUALITY,
ENLARGED
```

Onderstrepen
Met behulp van de underline mode kunnen alle lettertypes onderstreept worden. Alle tekens en spaties worden in deze mode van een onderstreping voorzien. Deze mode wordt door een ESCAPE '-1' aangezet en door een ESCAPE '-0' uitgezet.

```
OPEN4,4:PRINT#4,CHR$(27)"-1";
PRINT#4,"DIT IS GEWONE TEKST
DIE ONDERSTREEPT IS"
PRINT#4,CHR$(27)"-0";:CLOSE4
```

DIT IS GEWONE TEKST

DIE ONDERSTREEPT IS

Superscript en subscript printen
Superscripts en subscripts worden gebruikt in wiskundige, scheikundig en andere wetenschappelijke notaties.
Superscripts worden ook gebruikt om een footnote aan te geven. De superscript mode wordt met een ESCAPE 'S1' aangezet, de subscript mode wordt met ESCAPE 'S0' aangezet. Beide worden ze met ESCAPE 'T' uitgezet. Hier onder volgt een stukje listing dat gebruik maakt van deze modes.

```
100 OPEN4,4
110 PRINT#4,"CO";
120 GOSUB 1000
130 PRINT#4,"2";
140 GOSUB 1050
145 GOSUB 2000
150 PRINT#4," (GAS)"
160 GOSUB 2050
200 CLOSE4
210 END
1000 PRINT#4,CHR$(27)"S"CHR$(1);
1010 RETURN
1050 PRINT#4,CHR$(27)"T";
2000 PRINT#4,CHR$(27)"S"CHR$(0);
2010 RETURN
2050 PRINT#4,CHR$(27)"T";
```

CO_2 (GAS)

Letter-type demonstratie programma

In het onderstaande programma worden, van een groot aantal letter-types en combinaties van letter-types, voorbeelden gegeven. Delen van dit programma zijn goed te gebruiken in eigen programmatuur, omdat dit programma bijzonder modulair en dus overzichtelijk is opgezet.

3.3 Matrixprinters

```
10 OPEN4,4
15 GOSUB 11000
20 PRINT#4,"PICA OF NORMAAL PRINTEN"
30 GOSUB 11100
35 PRINT#4,"ELITE PRINTEN, AANZETTEN ESCAPE :, ";
40 PRINT#4,"UITZETTEN: ANDER FONT AANZETTEN"
50 GOSUB 11200
52 PRINT#4,"COMPRESSED OF ";
55 PRINT#4,"CONDENSED PRINTEN, AANZETTEN ASCII 15, UITZETTEN ASCII 18"
60 GOSUB 11250
70 GOSUB 11300
75 PRINT#4,"NEAR LETTER QUALITY, ";
80 PRINT#4,"AANZETTEN ESCAPE X1, ";
90 PRINT#4,"UITZETTEN ESCAPE XO"
100 GOSUB 11350
110 GOSUB 11400
115 PRINT#4,"EMPHASIZED PRINTEN, ";
120 PRINT#4,"AANZETTEN ESCAPE E, ";
130 PRINT#4,"UITZETTEN ESCAPE F"
140 GOSUB 11450
150 GOSUB 11500
160 PRINT#4,"DOUBLE STRIKE PRINTEN, ";
170 PRINT#4,"AANZETTEN ESCAPE G, ";
180 PRINT#4,"UITZETTEN ESCAPE H"
190 GOSUB 11550
200 GOSUB 11600
210 PRINT#4,"ENLARGED PRINTEN"
220 PRINT#4,"AANZETTEN ESCAPE W1, ";
230 PRINT#4,"UITZETTEN ESCAPE WO"
240 GOSUB 11650
250 GOSUB 11700
260 PRINT#4,"UNDERLINE PRINTEN, ";
270 PRINT#4,"AANZETTEN ESCAPE -1, ";
280 PRINT#4,"UITZETTEN ESCAPE -O"
290 GOSUB 11750
300 GOSUB 11800
310 PRINT#4,"SUPERSCRIPT PRINTEN";
320 PRINT#4,"AANZETTEN ESCAPE S1, ";
330 PRINT#4,"UITZETTEN ESCAPE T"
340 GOSUB 11850
400 GOSUB 11900
410 PRINT#4,"SUBSCRIPT PRINTEN";
420 PRINT#4,"AANZETTEN ESCAPE SO, ";
430 PRINT#4,"UITZETTEN ESCAPE T"
440 GOSUB 11950
1000 PRINT#4
1010 PRINT#4
1020 PRINT#4,"PICA: ";:GOSUB15000
1030 GOSUB11100:PRINT#4,"ELITE: ";:GOSUB15000
1040 GOSUB11200:PRINT#4,"COMPRESSED: ";:GOSUB15000:GOSUB11250
1050 REM GOSUB11300:PRINT#4,"NEAR LETTER QUALITY: ";:GOSUB15000:GOSUB11350
1060 GOSUB11400:PRINT#4,"EMPHASIZED: ";:GOSUB15000:GOSUB11450
1070 GOSUB11500:PRINT#4,"DOUBLE STRIKE: ";:GOSUB15000;GOSUB11550
1080 REM GOSUB11600:PRINT#4,"ENLARGED: ";:GOSUB15000:GOSUB11650
1090 REM GOSUB11700:PRINT#4,"UNDERLINED: ";:GOSUB15000:GOSUB11750
1100 GOSUB12050:PRINT#4,"SEMI DOUBLE STRIKE: ";:GOSUB15000
1110 GOSUB12000:PRINT#4,"SEMI DOUBLE STRIKE: ";:GOSUB15000
```

3.3 Matrixprinters

```
1990 PRINT#4
2000 REM NEAR LETTER QUALITY ON
2010 GOSUB11300
2015 PRINT#4,"NEAR LETTER QUALITY"
2020 GOSUB11400:PRINT#4,"NEAR LETTER QUALITY, EMPHASIZED":GOSUB11450
2030 REM GOSUB11500:PRINT#4,"NEAR LETTER QUALITY, DOUBLE STRIKE":GOSUB11550
2040 GOSUB11600:PRINT#4,"NEAR LETTER QUALITY, ENLARGED":GOSUB11650
2050 GOSUB11700:PRINT#4,"NEAR LETTER QUALITY, UNDERLINED":GOSUB11750
2055 PRINT#4
2060 REM NEAR LETTER QUALITY, ENLARGED
2070 GOSUB11600
2080 PRINT#4,"NLQ, ENLARGED"
2100 REM GOSUB11500:PRINT#4,"NLQ, ENLARGED, DOUBLE STRIKE":GOSUB11550
2105 GOSUB11400:PRINT#4,"NLQ, ENLARGED, EMPHAZIZED":GOSUB11450
2110 GOSUB11700:PRINT#4,"NLQ, ENLARGED, UNDERLINED":GOSUB11750
2120 GOSUB11650
2130 GOSUB11350
2990 PRINT#4
3000 REM ELITE
3010 GOSUB 11100
3015 PRINT#4,"ELITE"
3020 REM GOSUB11400:PRINT#4,"ELITE, EMPHASIZED":GOSUB11450
3030 GOSUB11500:PRINT#4,"ELITE, DOUBLE STRIKE":GOSUB11550
3040 GOSUB11600:PRINT#4,"ELITE, ENLARGED":GOSUB11650
3050 GOSUB11700:PRINT#4,"ELITE, UNDERLINED":GOSUB11750
3060 PRINT#4
4000 REM COMPRESSED OR CONDENSED
4010 GOSUB 11200
4015 PRINT#4,"COMPRESSED OF CONDENSED"
4020 REM GOSUB11400:PRINT#4,"COMPRESSED, EMPHASIZED":GOSUB11450
4030 REM GOSUB11500:PRINT#4,"COMPRESSED, DOUBLE STRIKE":GOSUB11550
4040 GOSUB11600:PRINT#4,"COMPRESSED, ENLARGED":GOSUB11650
4050 GOSUB11700:PRINT#4,"COMPRESSED, UNDERLINED":GOSUB11750
4090 PRINT#4
7000 PRINT#4
7001 CLOSE4
7002 END
11000 REM RESET PRINTER
11010 PRINT#4,CHR$(27)"@"
11020 RETURN
11100 REM ELITE
11110 PRINT#4,CHR$(27)":";
11120 RETURN
11200 REM COMPRESSED OR CONDENSED ON
11210 PRINT#4,CHR$(15);
11220 RETURN
11250 REM COMPRESSED OR CONDENSED OFF
11260 PRINT#4,CHR$(18);
11270 RETURN
11300 REM NEAR LETTER QUALITY ON
11310 PRINT#4,CHR$(27)CHR$(120)CHR$(1);
11320 RETURN
11350 REM NEAR LETTER QUALITY OFF
```

3.3 Matrixprinters

```
11360 PRINT#4,CHR$(27)CHR$(120)CHR$(0);
11370 RETURN
11400 REM EMPHASIZED ON
11410 PRINT#4,CHR$(27)"E";
11420 RETURN
11450 REM EMPHASIZED OFF
11460 PRINT#4,CHR$(27)"F";
11470 RETURN
11500 REM DOUBLE STRIKE ON
11510 PRINT#4,CHR$(27)"G";
11520 RETURN
11550 REM DOUBLE STRIKE OFF
11560 PRINT#4,CHR$(27)"H";
11570 RETURN
11600 REM ENLARGED ON
11610 PRINT#4,CHR$(27)"W"CHR$(1);
11620 RETURN
11650 REM ENLARGED OFF
11660 PRINT#4,CHR$(27)"W"CHR$(0);
11670 RETURN
11700 REM UNDERLINE ON
11710 PRINT#4,CHR$(27)"-1";
11720 RETURN
11750 REM UNDERLINE OFF
11760 PRINT#4,CHR$(27)"-0";
11770 RETURN
11800 REM SUPERSCRIPT ON
11810 PRINT#4,CHR$(27)"S"CHR$(0);
11820 RETURN
11850 REM SUPERSCRIPT OFF
11860 PRINT#4,CHR$(27)"T";
11870 RETURN
11900 REM SUBSCRIPT ON
11910 PRINT#4,CHR$(27)"S"CHR$(1);
11920 RETURN
11950 REM SUBSCRIPT OFF
11960 PRINT#4,CHR$(27)"T";
11970 RETURN
12000 REM TURN ON AUTO LINEFEED
12010 PRINT#4,CHR$(27)"5"CHR$(1);
12020 RETURN
12050 REM TURN OFF AUTO LINEFEED
12060 PRINT#4,CHR$(27)"5"CHR$(0);
12070 RETURN
15000 REM VOORBEELD SCHEIKUNDIGE REAKTIE
15010 PRINT#4,"2 H";
15020 GOSUB11900:PRINT#4,"3";:GOSUB11950
15030 PRINT#4,"O";
15050 GOSUB11800:PRINT#4,"+";:GOSUB11850
15100 PRINT#4," + CO";
15110 GOSUB11900:PRINT#4,"3";:GOSUB11950
15200 GOSUB11800:PRINT#4,"2-";:GOSUB11850
15250 PRINT#4," -> 2 H";
```

3.3 Matrixprinters

```
15260 GOSUB11900:PRINT#4,"2";:GOSUB11950
15300 PRINT#4,"O + H";
15350 GOSUB11900:PRINT#4,"2";:GOSUB11950
15400 PRINT#4,"CO";
15450 GOSUB11900:PRINT#4,"3";:GOSUB11950
15500 PRINT#4," -> 3 H";
15600 GOSUB11900:PRINT#4,"2";:GOSUB11950
15700 PRINT#4,"O + CO";
15800 GOSUB11900:PRINT#4,"2";:GOSUB11950
15900 GOSUB11800:PRINT#4," (GAS)":GOSUB11850
15950 RETURN

READY.
```

3.3 Matrixprinters

PICA OF NORMAAL PRINTEN
ELITE PRINTEN, AANZETTEN ESCAPE :, UITZETTEN: ANDER FONT AANZETTEN
COMPRESSED OF CONDENSED PRINTEN, AANZETTEN ASCII 15, UITZETTEN ASCII 18
NEAR LETTER QUALITY, AANZETTEN ESCAPE X1, UITZETTEN ESCAPE X0
EMPHASIZED PRINTEN, AANZETTEN ESCAPE E, UITZETTEN ESCAPE F
DOUBLE STRIKE PRINTEN, AANZETTEN ESCAPE G, UITZETTEN ESCAPE H
ENLARGED PRINTEN
AANZETTEN ESCAPE W1 , UITZETTEN ESCAPE W0
UNDERLINE PRINTEN, AANZETTEN ESCAPE -1, UITZETTEN ESCAPE -0
SUPERSCRIPT PRINTENAANZETTEN ESCAPE S1, UITZETTEN ESCAPE T

SUBSCRIPT PRINTENAANZETTEN ESCAPE S0, UITZETTEN ESCAPE T

PICA: $2\ H_3O^+ + CO_3^{2-} \rightarrow 2\ H_2O + H_2CO_3 \rightarrow 3\ H_2O + CO_2$ (GAS)
ELITE: $2\ H_3O^+ + CO_3^{2-} \rightarrow 2\ H_2O + H_2CO_3 \rightarrow 3\ H_2O + CO_2$ (GAS)
COMPRESSED: $2\ H_3O^+ + CO_3^{2-} \rightarrow 2\ H_2O + H_2CO_3 \rightarrow 3\ H_2O + CO_2$ (GAS)
EMPHASIZED: $2\ H_3O^+ + CO_3^{2-} \rightarrow 2\ H_2O + H_2CO_3 \rightarrow 3\ H_2O + CO_2$ (GAS)
DOUBLE STRIKE: $2\ H_3O^+ + CO_3^{2-} \rightarrow 2\ H_2O + H_2CO_3 \rightarrow 3\ H_2O + CO_2$ (GAS)
SEMI DOUBLE STRIKE: $2\ H_3O^+ + CO_3^{2-} \rightarrow 2\ H_2O + H_2CO_3 \rightarrow 3\ H_2O + CO_2$ (GAS)

NEAR LETTER QUALITY
NEAR LETTER QUALITY, EMPHASIZED
NEAR LETTER QUALITY, ENLARGED
NEAR LETTER QUALITY, UNDERLINED

NLQ, ENLARGED
NLQ, ENLARGED, EMPHAZIZED
NLQ, ENLARGED, UNDERLINED

ELITE
ELITE, DOUBLE STRIKE
ELITE, ENLARGED
ELITE, UNDERLINED

COMPRESSED OF CONDENSED
COMPRESSED, ENLARGED
COMPRESSED, UNDERLINED

Pagina indeling in de IBM mode

De MPS 1000 heeft veel mooie commando's om de kantlijnen, de horizontale en verticale tabulator stops, de regelafstand, etc. in te stellen. Met deze commando's kunt u op vrij eenvoudige wijze de printer instellen, zonder dat binnen een bepaald programma allerlei acties ondernomen moeten worden om hetzelfde effect te verkrijgen. In een programma zijn al deze printer commando's natuurlijk te emuleren, maar met de MPS 1000 is dit allemaal niet meer nodig. De printer neemt veel moeizaam programmeerwerk uit handen. Ook bepaalde standaard software pakketten, die het niet zo nauw nemen met de pagina indeling van de printer, zijn met de MPS 1000 te verbeteren! Voordat de applicatie wordt opgestart, wordt eerst een BASIC programmaatje gedraaid, dat de printer goed instelt. Dan wordt de applicatie geladen en geRUNd, de printer zal er nu zelf voor zorgen dat de pagina indeling er correct uitziet (mits de applicatie die waardes van de printer NIET verandert). Hieronder zal beschreven worden, hoe de meest gebruikte pagina indeling op de printer is in te stellen.

Kantlijnen

Met behulp van een ESCAPE commando is de linker en rechterkantlijn van de MPS 1000 in te stellen. Het kantlijn-commando is ESCAPE 'X' gevolgd door het kolomnummer van de linker kantlijn en dat van de rechter kantlijn. Om bijvoorbeeld de linker kantlijn op 10 en de rechter kantlijn op 70 in te stellen, kan men het volgende BASIC programmaatje draaien.

```
10 OPEN4,4
20 PRINT#4,CHR$(27)"X"CHR$(10);
   CHR$(70);
30 CLOSE4

READY.
```

De maximale rechterkantlijn waarden voor de MPS 1000 zijn: 80 voor pica, 96 voor elite, 137 voor condensed of compressed, 80 voor near letter quality.

Perforatie skipping

Bij het uitdraaien van programmalistings op kettingformulieren is het vervelend, dat er altijd tekst op de perforatie komt te staan. De MPS 1000 heeft een ESCAPE code, waardoor listings of andere tekst niet meer boven op de perforatie wordt afgedrukt. Met een ESCAPE 'N' gevolgd door het aantal regels, dat door de MPS 1000 moet worden overgeslagen aan het einde van de pagina. Stel dat de printer 6 regels aan het einde van de pagina wit moet laten, dan moeten de volgende commando's gegeven worden.

```
OPEN4,4
PRINT#4,CHR$(27)"N";CHR$(6);
CLOSE4

READY.
```

Deze 6 lege regels komen aan het einde van het papier. Wilt u er 3 boven en 3 onder aan de pagina, draai dan het papier 3 regels door onder de perforatie.

Het grote programma, dat op een van de vorige bladzijden staat, is zo met behulp van de twee pagina indelingscommando's, op een handig leesbare manier uit te draaien:

1) open de printer.
2) zet de kantlijnen op de gewenste waarden.
3) laat 6 regels bij de perforatie wit.
4) zet de printer goed!
5) geef de standaard commando's om een basic programma te listen.

3.3 Matrixprinters

```
OPEN4,4
PRINT#4,CHR$(27)"N";CHR$(6);
PRINT#4,CHR$(27)"X"CHR$(10);
CHR$(70);
CMD4
LIST
PRINT#4
CLOSE#4

READY.
```

Ga naar het einde van de pagina
Er zijn twee methoden om naar het einde, of het begin van de volgende pagina te gaan. Ten eerste door bij te houden op welke regel het laatst is gePRINT en dan het resterende aantal regels, met behulp van carriage returns, over te slaan. Voor deze methode zal in het programma enige administratie (het nummer van de huidige regel, die gePRINT wordt) moeten worden bijgehouden. Ten tweede door de printer het form-feed commando te geven. Door een form-feed (ASCII 12) naar de printer te sturen, zal deze naar het begin van de volgende pagina gaan.

Papier lengte
De lengte van het papier is software-matig in te stellen. Dit is een handige faciliteit als er papier wordt gebruikt, dat een andere lengte heeft dan de standaard 11 of 12 inch. De lengte van het papier kan worden opgegeven in inches of in het aantal regels. Door middel van een ESCAPE 'C' nummer commando, waarin nummer het aantal regels is, wordt de papier lengte op het aantal regels ingesteld. Door een ESCAPE 'C' ASCII 0 nummer te versturen, wordt de papier lengte op het aantal inches ingesteld, wat door nummer wordt aangegeven.

Teken sets
De MPS 1000 heeft in de IBM mode twee tekensets, die bijna exact gelijk zijn. Alleen wat speciale tekens verschillen van elkaar. Met de ESCAPE '7' code wordt de eerste tekenset geselecteerd, dit is de standaard tekenset. Met ESCAPE '6' wordt de tweede tekenset geselecteerd.

Printer reset
De printer kan op twee verschillende manieren gereset worden. Ten eerste door de printer aan en uit te zetten. Ten tweede door een ESCAPE '@' naar de printer te versturen. De software-matige oplossing is handig binnen programma's om de printer dan vanuit een bekende status te programmeren.

Tabel voor IBM mode

Hieronder volgt een tabel, waarin de meest voorkomende codes voor de printer in de IBM mode zijn opgenomen.

Besturingscodes voor de IBM mode

Letter types

ESC ':'	Zet elite aan
ASCII 15	Zet condensed aan
ASCII 18	Zet condensed uit
ESC 'x1'	Zet near letter quality aan
ESC 'x' ASCII 1	Zet near letter quality aan
ESC 'x0'	Zet near letter quality aan
ESC 'x' ASCII 0	Zet near letter quality uit
ESC 'E'	Zet emphasized aan
ESC 'F'	Zet emphasized uit

3.3 Matrixprinters

ESC 'G'	Zet double-strike aan
ESC 'H'	Zet double-strike uit
ASCII 14	Zet een-regel enlarged aan
ASCII 15	Zet een-regel enlarged uit
ESC 'W' ASCII 1	Zet enlarged aan
ESC 'W1'	Zet enlarged aan
ESC 'W' ASCII 0	Zet enlarged uit
ESC 'W0'	Zet enlarged uit
ESC 'S' ASCII 1	Zet subscript aan
ESC 'S1'	Zet subscript aan
ESC 'S' ASCII 0	Zet superscript aan
ESC 'S0'	Zet superscript aan
ESC 'T'	Zet superscript of subscript uit
ESC '-' ASCII 1	Zet underline aan
ESC '-1'	Zet underline aan
ESC '-' ASCII 0	Zet underline uit
ESC '-0'	Zet underline uit

Tekenset selectie

ESC '7'	Selecteer teken set 1
ESC '6'	Selecteer teken set 2

Pagina indeling

ESC 'C' ASCII 0 n	Zet papier lengte op n inch
ESC 'C' ASCII n	Zet papier lengte op n regels
ESC 'N' ASCII n	Zet de perforatie skip op n regels
ESC 'O'	Zet de perforatie skip uit
ESC 'X' ASCII l r	Zet linker en rechter kantlijn

Papierdoorvoer

ASCII 10	Voer een line-feed uit
ASCII 13	Voer een carriage return uit
ESC '5' ASCII 1	Zet automatische line-feed aan
ESC '51'	Zet automatische line-feed aan
ESC '5' ASCII 0	Zet automatische line-feed uit
ESC '50'	Zet automatische line-feed uit
ASCII 12	Ga naar het einde van de pagina

Reset printer

ESC '@ '	Reset printer

6/4

De user port

Inhoud

6/4.1

Inleiding

Dit hoofdstuk zal de mogelijkheden van de user port behandelen. De user port is de uitbreidingspoort die zich (van achteren gezien) uiterst rechts aan de Commodore 64 bevindt. De user port wordt voor communicatiedoeleinden gebruikt, vandaar dat ze in dit deel behandeld wordt. Deze communicatie omvat niet alleen communicatie met andere computers, maar ook met randapparatuur en dergelijke. Dat 'en dergelijke' is misschien niet zo veelzeggend, vandaar dat een aantal van de mogelijkheden van de user port hieronder zijn opgesomd:
Directe communicatie computer-computer
Communicatie computer-computer via modem
Communicatie computer-diskdrive (SpeedDOS)
Communicatie computer-printer (Centronics Interface)
Besturing eprom-programmer
Besturing spraaksynthesizer
Procesbesturing (bijvoorbeeld doka, treinen, Fisher Techniek)
Besturing interfaces (bijvoorbeeld keyboards e.d.)

Dit lijstje is verre van compleet, zoals in de loop van dit hoofdtuk duidelijk zal worden.

Wellicht zal de vraag gerezen zijn hoe een 'user port' eigenlijk voorgesteld moet worden. De user port moet gezien worden als een connector met een aantal pennetjes en op die pennetjes kunnen elektrische spanningen aangebracht worden. Het zal bekend zijn dat computercommunicatie in het algemeen gebaseerd is op het overbrengen van spanningen: wel spanning is een logische 1, geen spanning is een logische 0.

De user port stelt ons in staat om zelf de gewenste spanningen (enen en nullen) aan de buitenwereld bekend te maken. Dit gebeurt simpelweg door enen en nullen naar de user port toe te schrijven. Omgekeerd kunnen ook van buiten af spanningen aan de user port aangeboden worden, deze kunnen we vinden door de user port uit te lezen. Een logische 1 komt dan overeen met wel spanning, een logische 0 betekent geen spanning.

Tot zover is het vrij eenvoudig te volgen. Er rijzen echter een paar vragen: Hoe groot zijn de genoemde spanningen en hoe lees ik ze af/schrijf ik ze weg?
Het antwoord op de eerste vraag is als volgt: spanningen onder de 0,8 volt zijn een logische 0 en spanningen boven de 2,4 volt zijn een logische 1. Deze spanningen zijn niet door Commodore zelf gekozen, het zijn de zogenaamde TTL-spanningsniveaus. TTL staat voor

4.1 Inleiding

transistor-transistor logic en is de benaming voor een fabricageproces van logische (digitale) schakelingen. Opdat alle digitale IC's van de zelfde familie (in dit geval de TTL-familie) met elkaar kunnen communiceren dienen ze gebruik te maken van dezelfde 1- en 0-definitie. Dit zijn dus respectievelijk spanningen boven de 2,4 volt (maar onder Vcc, meestal 5 volt) en spanningen beneden de 0,8 volt (maar boven Vss, meestal 0 volt).

Het antwoord op de tweede vraag is ook niet zo moeilijk, de user port is verbonden met een I/O-poort van één van de twee CIA's die zich in de Commodore 64 bevinden. CIA is de afkorting van Complex Interface Adapter, een chip die speciaal voor I/O (input/output) gebouwd is. Deze chip bevat, net zoals bijvoorbeeld de VIC en de SID, een aantal registers. Deze registers komen overeen met een aantal geheugenplaatsen van de Commodore 64, met als gevolg dat de CIA's gelezen en beschreven kunnen worden door in het geheugen te PEEKen of te POKEn.

Op het eerste gezicht zou verwacht kunnen worden dat de CIA slechts één register heeft, namelijk het register waar de data voor de I/O-poort in staat. Niets is echter minder waar, de CIA heeft, om precies te zijn, 16 registers. Genoeg dus om paragraaf 6/4.2 aan de CIA te weiden.

6/4.2

De 6526 Complex Interface Adapter(CIA)

Zoals in de vorige paragraaf is uitgelegd is de user port rond de 6526 Complex Interface Adapter opgebouwd. Deze chip bevat een aantal registers die bepaalde geheugenadressen beslaan. Voor CIA nummer 1 (er zijn er twee in de Commodore 64) zijn de geheugenplaatsen $DC00 t/m $DC0F (56320 t/m 56335) gereserveerd, CIA nummer 2, die de user port bestuurt, beslaat de adressen $DD00 t/m $DD0F (56578 t/m 56593). We zullen in deze paragraaf de gehele CIA bespreken omdat de user port ook andere dan alleen de I/O-lijnen van de CIA bevat.

Specificaties

Behalve een I/O-poort die direct, met de user port in verbinding staat bevat de CIA nog een I/O-poort. Elke CIA heeft dus twee I/O-poorten, genaamd Peripheral Data Register A (kortweg PA) en Peripheral Data Register B (PB). Van CIA # 1 worden de poorten gebruikt om het toetsenbord en de joysticks uit te lezen. Van CIA # 2 is PB voor de user port verantwoordelijk, PA bedient de seriële bus en de VIC-chip (bank-selectie). Verder zijn pen 2 en 3 van PA ook nog met de user port verbonden.

Het is belangrijk om te onthouden dat we in dit verhaal alleen over CIA # 2 zullen spreken. Wanneer dus PB genoemd wordt, bedoelen we eigenlijk PB van CIA # 2.

Naast twee I/O-poorten bevat de CIA ook nog twee Data Direction Registers, DDRA en DDRB. De Data Direction Registers, in het nederlands data richtings registers, bepalen of een I/O-lijn een ingangs- of een uitgangslijn is. Dit is namelijk zeer belangrijk! Op een uitgangslijn zal namelijk vanuit de computer een spanning naar buiten gestuurd worden. Wanneer de lijn echter een ingangslijn moet zijn zal de computer geen spanning willen uitzenden, maar juist één willen ontvangen. Om de I/O-poort dus niet op te blazen moeten alle ingangs- en uitgangslijnen specifiek gedefinieerd worden!

Dit definiëren van ingangs- en uitgangslijnen gebeurt nu in het data richtings register. Staat in een DDR-bit een 1 dan is de bijbehorende I/O-lijn een uitgang, is dit DDR-bit een 0 dan is de bijbehorende I/O-lijn een ingang. Het volgende 'ezelsbruggetje' is hierbij van toepassing: 0/1 = I/O. Willen we bijvoorbeeld de lijnen 0 t/m 5 van de user port als uitgang definiëren en de lijnen 6 en 7 als ingang dan dient het DDR (DDRB in dit geval, want de user port werkt met poort B) de waarde 00111111 (63) te bevatten. De enen definiëren een uitgang, de nullen definiëren een ingang.

4.2 De 6526 CIA

Dat de 6526 een zeer uitgebreide I/O-chip is blijkt uit het feit dat in deze chip nog een aantal andere functies zijn geïntegreerd: twee 16-bits timers, een time-of-day klok met bijbehorend alram, een 8-bits schuifregister en twee handshakelijnen. Al deze functies zullen één voor één besproken worden.

Timer A en Timer B
De 6526 bezit twee 16-bits timers, genaamd timer A en timer B. Timers worden gebruikt om tijdsvertragingen te genereren, pulsen met variabele lengte te maken en variabele frequentiepatronen te produceren. Het meest gebruikt echter is de mogelijkheid om eens in de zoveel tijd (aan het eind van een ingesteld interval) een signaal of interrupt te genereren. Een interrupt is een signaal voor de processor dat er voor zorgt dat de processor zijn huidige werk even staakt om een zogenaamde interruptroutine uit te gaan voeren. Aan het eind van die routine vervolgt de processor de taak waar hij oorspronkelijk mee bezig was.

De 6526 kan, zoals we later zullen zien, 5 verschillende interrupts genereren. Of liever gezegd, bij vijf verschillende gebeurtenissen; de interrupts manifesteren zich namelijk als één en hetzelfde interruptsignaal. Er zijn echter 5 verschillende oorzaken. Twee van die oorzaken zijn het aflopen van timer A en het aflopen van timer B, de overige drie komen, zoals gezegd, later aan bod. De twee timer-interrupts zijn dus het gevolg van het 'aflopen' van de bijbehorende timer. Beide timers kunnen een startwaarde gegeven worden. Deze waarde is 16 bits breed. Zodra de 'start'-opdracht gegeven wordt begint de gewenste teller af te tellen. dit gebeurt met de snelheid van de klok: 1 miljoen pulsjes per seconde, of te wel 1 MHz (Megahertz).
Het maximale tijdsinterval dat met een timer te maken is is dus 2 ↑ 16/1000000 oftewel 0.065536 seconde, dit is gelijk aan ca. 66 ms (miliseconde). Niet erg lang zou je denken, echter voor computertermen vrij lang. De beide timers kunnen echter ook tot één 32-bits timer gecombineerd worden zodat het langste interval 65536 maal zo groot wordt. De maximale tijdsvertraging is dan gelijk aan 4295 seconden, oftewel 1 uur, 11 minuten en 35 seconden!

De timers worden ingesteld door hun startwaarde in de overeenkomstige timerregisters te schrijven. Deze registers kunnen ten alle tijde uitgelezen worden om te kijken hoe ver de timer 'gevorderd' is. Verder zijn er een aantal sturingsbits die de precieze werking van beide timers definiëren, deze bits zullen bij de registermap uitgelegd worden.

Overigens is timer A van CIA # actief in het dagelijkse 64-leven: de interruptroutine, welke er onder andere voor zorgt dat het toetsenbord wordt uitgelezen en dat de cursor knippert, wordt 'gestart' door het aflopen van timer A. Deze timer wordt met de waarde 16409 geladen zodat bij een frequentie van 1 Megahertz (de klokfrequentie van de 64) de interruptroutine om de 16 ms (dus 60 keer per seconde) wordt aangeroepen. Door het highbyte van deze waarde te veranderen zal de cursor sneller of minder snel gaan knipperen.
Probeer het volgende commando maar eens:

OKE 56325,20
Het commando
POKE 56325,58
brengt alles weer in de originele staat.

4.2 De 6526 CIA

Het heeft geen zin om het lowbyte van deze waarde aan te passen, het effect hiervan is namelijk minimaal: het lowbyte heeft een maximale afwijking van 256/1000000 seconde tot gevolg: 0.000256 seconde om precies te zijn!

Het kan wel eens nodig zijn om de timers juist niet op de klokfrequentie van het systeem te laten aftellen, maar op de frequentie van een externe pulsbron. Dit hoeft niet perse een kristaloscillator te zijn, het mag ook een tel-mechanisme van bijvoorbeeld een lichtgevoelig oog zijn. Op die manier kan de 6526 bijvoorbeeld als frequentieteller gebruikt worden of als meetapparaat voor tijdsvertagingen. Dit zijn echter nogal geavanceerde toepassingen die we hier voorlopig buiten beschouwing zullen laten. Wel is het dus een veelgebruikte techniek om timer B op de afloop-pulsjes van timer A af te laten lopen. Op die manier wordt een 32-bits timer uit twee 16-bits timers geconstrueerd. Timer B telt dan namelijk eens in (bijvoorbeeld) de 65536 klokpulsen af (telkens wanneer timer A de waarde 0 bereikt).

De TOD-klok en het alarm

De 6526 bevat ook een 24-uurs time-of-day klok, kortweg TOD. TOD is een klok die gebruik maakt van de externe wisselspanningsbron om bij de tijd te blijven. Zoals wellicht bekend is is dit in Nederland 220 volt, 50 Hz. Bij het transformeren naar een voor de computer meer geschikt niveau blijft de 50 Hz-informatie bewaard. Deze 50 Hz leveren 50 pulsjes per seconde en deze kunnen dan weer gebruikt worden om de TOD aan de gang te houden. De werkelijke waarde van de netfrequentie wil wel eens van 50 Hz afwijken, over een etmaal genomen is deze echter vrij constant. Dit blijkt bijvoorbeeld uit het feit dat de meeste elektronische wekkers en wekkerradio's ook van deze 50 Hz gebruik maken.

De werkelijke tijd wordt in 4 registers bewaard; voor de uren, minuten en seconden elk een register, maar ook de tienden van seconden worden door TOD bijgehouden. Deze zijn in het vierde register af te lezen. Verder is TOD een AM/PM-klok, dit houdt in dat 20.00 er als 8.00 uitziet, een extra bit (het hoogste bit van het uren-register) houdt de AM/PM-stand bij.

Omdat de Amerikaanse netfrequentie anders is dan de Europese zal bij het in gebruik stellen van TOD de tijd nogal traag verlopen: de 6526 zal op een 60 Hz-bron ingesteld zijn wat inhoudt dat er 60 pulsjes moeten komen voordat er 1 seconde versprongen wordt. In werkelijkheid zal er dan al 1.2 seconde verlopen zijn! Hiertoe is het noodzakelijk om de 6526 apart te laten weten of hij op een 60 Hz- of op een 50 Hz-bron aangesloten staat. Dit gebeurt middels een bit in een van de registers.

Het alarm kan ingesteld worden op een willekeurige tijd en zal er voor zorgen dat op die tijd een interrupt gegenereerd wordt. Het is de taak van de programmeur om er voor te zorgen dat deze interrupt als alarm-interrupt wordt herkend en dat de overeenkomstige actie genomen wordt. De alarm-registers zijn dezelfde registers als de TOD-registers, vandaar dat het instellen een apart truukje vereist waar we later op terug zullen komen. De alarm-functie is eigenlijk alleen van belang wanneer we de 6526 in een wekker willen gebruiken.

Het 8-bits schuifregister
Schuifregisters worden gebruikt om seriële data naar paralelle data om te zetten en omgekeerd. Een schuifregister kan 8 bits data achter elkaar over een lijn naar buiten sturen, maar ook 8 bits data serieel ontvangen en in een 8-bits register op slaan. Dit schuifregister kan gebruikt worden in bijvoorbeeld RS232-communicatie. De 64 maakt hier echter geen gebruik van.

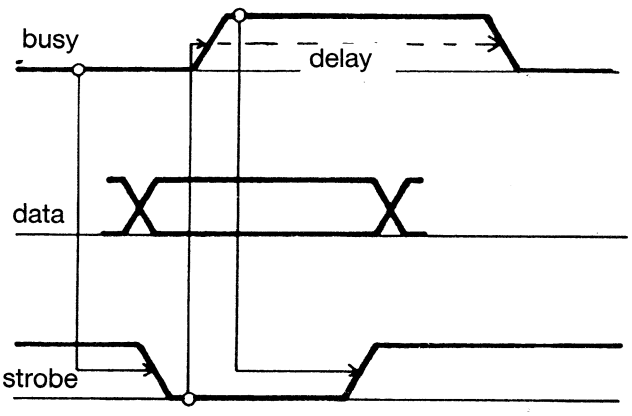
In de input-mode wordt de data van de SP-lijn overgenomen op de opgaande flank van de CNT-pen van de chip. Na acht van deze pulsen is het register vol en wordt een interrupt gegenereerd. In de output-mode wordt de data bit voor bit verzonden in een tempo aangegeven door timer A. Deze data wordt via de SP-lijn naar buiten gestuurd. Wanneer alle 8 bits verzonden zijn wordt eveneens een interrupt gegenereerd, ten teken dat de volgende data verzonden kan worden.

Overigens zijn de SP en CNT-lijnen van beide CIA's op de user port beschikbaar, de 64 zelf maakt er geen gebruik van.

De handshakesignalen
Handshake is een computerterm die duidt op communicatie tussen twee apparaten. Bijvoorbeeld een printer en een computer of twee computers. Handshaking houdt in dat de beide apparaten het er altijd over eens zijn wie wat aan het doen is. Dit is niet altijd het geval, kijk maar naar het RS232 communicatieprotocol.

Bij handshaking komen naast de gebruikte datalijnen altijd nog eens twee lijnen kijken: de zogenaamde handshakelijnen. Veel gebruikte namen zijn strobe en busy, of data-valid en acknowledge. In principe komt het hier op neer:
Het verzendende apparaat 'kijkt' of de ontvanger bezig is (busy/acknowledge). Zo ja dan wordt er gewacht, zo nee dan wordt er data aangeboden, tezamen met een strobe of data-valid signaal. Dit ten teken dat er geldige data op de datalijnen aanwezig is. De ontvanger zal nu met een busy/acknowledge signaal aangeven dat de data ontvangen en overgenomen is. Tot die tijd echter zal de zender wachten en overgenomen is. Tot die tijd echter zal de zender wachten en de data, tezamen met het strobe-signaal, blijven aanbieden. Het busy/acknowledge-signaal zal op een gegeven moment aangeven dat de data door de ontvanger is overgenomen, dit is dan weer een teken voor de zender dat de data en het strobe-signaal weggehaald kunnen worden. Totdat de ontvanger zijn busy-signaal heeft weggehaald zal de zender dan wachten met het verzenden van volgende data.

In een timing-diagram ziet bovenstaande uiteenzetting er als volgt uit:



Figuur 6/4.2-1: Handshake-timing

4.2 De 6526 CIA

De 6526 is op het handshake-protocol voorbereid door twee extra lijnen, FLAG en PC (weer andere namen dus). De lijn PC komt overeen met data valid, wanneer data naar poort B geschreven wordt zal hier automatisch een puls van 1 klokpuls lengte verschijnen die aangeeft dat er data op poort B staat. FLAG is een ingang die laag gemaakt dient te worden wanneer de data overgenomen is (busy/ acknowledge), de 6526 zal dan een interrupt genereren en het FLAG-interrupt bit zetten (zie verder).

In Centronics-interfaces wordt ook gebruik gemaakt van het handshake-protocol, de diverse populaire versies die er voor de 64 bestaan maken echter geen gebruik van de PC-lijn. Dit omdat de puls die hier verschijnt te kort is. De STROBE/-puls (zoals de officiële Centronics-benaming luidt) dient actief te blijven tot een BUSY is gedetecteerd. Vandaar dat de STROBE/-puls softwarematig op de PA2-lijn (die ook op de user-port aanwezig is) wordt opgewekt door deze laag te maken totdat er een acknowledge-signaal op FLAG binnenkomt.

De interrupts

Zoals eerder gemeld is de 6526 in staat om bij vijf verschillende gelegenheden interrupts te genereren. Deze manifesteren zich allemaal op de zelfde interruptlijn, er moet dus een methode zijn om uit te vinden welke gebeurtenis de interrupt veroorzaakt heeft. In computertermen heet dit polling, de 6502-serie maakt gebruik van polled interrupts. Dit houdt in dat de processor zelf na moet gaan welk randapparaat (of welke chip) de interrupt heeft gegenereerd. Zoiets hebben we al eerder gezien bij de raster-interrupts van de VIC-chip.

Om elke interrupt te identificeren bezit de 6526 een register waarin de interrupts bijgehouden worden. Vijf bits in dit register komen overeen met de vijf interruptgevende gebeurtenissen: timer A, timer B, alarm, seriële poort en FLAG. Deze bits kunnen gelezen worden, een 1 betekent dat de interrupt van de betreffende gebeurtenis komt. Een extra bit wordt gebruikt om aan te geven dat een willekeurige interrupt heeft plaatsgevonden, dit heeft tot voordeel dat in systemen met meerdere chips niet alle afzonderlijke bits bekeken hoeven te worden maar dat met een enkele BIT-instructie de interruptgevende 6526 gelokaliseerd kan worden.

Het kan wel eens nuttig zijn om bepaalde gebeurtenissen niet een interrupt te laten genereren. In de eerder genoemde softwarematige Centronics interfaces wordt FLAG wel gebruikt, dit genereert echter geen interrupt. Dit is mogelijk door bepaalde interrupts te maskeren en daar mee uit te schakelen. Het overeenkomstige bit wordt dan wel gezet, er wordt echter geen interrupt gegenereerd en ook het algemene chip-interrupt-bit wordt niet gezet. Wel kan constant gekeken worden of het gebeurtenis-interruptbit gezet wordt, op die manier kan dus gewacht worden op bijvoorbeeld het FLAG-bit en dus op het not busy-signaal. Dit maskeren gebeurt door middel van het beschrijven van het interrupt-register. Op de plaats waar normaal het chip-interrupt-bit staat dient dan een 1 gezet te worden om interrupts wel toe te staan. Verder worden enen gezet op de posities van de individuele interruptbits als ze aangepast moeten worden, een 0 laat de status ongewijzigd. Meer details hieromtrent volgen in het register-

overzicht.

6526 registermap

Registers 0 t/m 3: PA, PB, DDRA en DDRB

De registers 0 t/m 3 bevatten respectievelijk de dataregisters PA en PB en de datarichtingsregisters DDRA en DDRB. De dataregisters worden gebruikt om data in te lezen of naar buiten te sturen, afhankelijk van de overeenkomstige datarichtingsbits. Een 0 definieert een lijn als ingang, een 1 definieert een lijn als uitgang:

```
LDA #%11111111
STA DDRA
LDA #%00000000
STA DDRB
```

Deze instructies definiëren alle lijnen van PA als uitgang en alle lijnen van PB als ingang. De volgende instructies

```
STA PA
LDX PB
LDY PA
```

zetten de inhoud van de accu op poort A, zetten de inhoud van poort B in het X-register en laten aan het Y-register weten wat de laatste waarde was van de naar PA geschreven data. Het is dus mogelijk om een uitgangslijn te LEZEN, het resultaat is de laatst weggeschreven data.

Registers 4 t/m 7: TALO, TAHI, TBLO, TBHI

De registers 4 t/m 7 besturen de timers. Beide timers zijn 16 bits breed en omvatten elk dan ook twee registers. De timers kunnen ten alle tijde beschreven en gelezen worden, het schrijven stelt ze in en het lezen levert de actuele waarde op. De timer-functies worden door de control-registers bepaald, deze worden verderop besproken.

Registers 8 t/m 11: TOD10, TODSEC, TODMIN, TODHR

De registers 8 t/m 11 worden gebruikt voor de TOD-klok alsmede het alarm. Om het alarm te bereiken wordt gebruik gemaakt van het control-register, zie verderop. In de alarm-mode zijn de registers write-only, een ingesteld alarm kan dus niet teruggelezen worden. In de TOD-mode zijn de registers read/write, een read levert de actuele inhoud, een write stelt de tijd in. Bij deze twee punten dienen de volgende zaken in acht genomen te worden:

TOD-read:

Bij het lezen van de uren worden de minuten, seconden en tienden in hun registers 'geklokt', dit om een goede uitlezing te garanderen. Intern loopt TOD wel door, dit is echter niet meer te zien. Dit stoppen van de uitvoer is noodzakelijk om op bijvoorbeeld 7:30:44:90 uur ook de juiste tijd te krijgen: bij het lezen van de tienden zijn deze wellicht al op 0 gesprongen zodat het resultaat 7:30:44:00 is, bijna een hele seconde fout! Na het lezen van de tienden echter gaat de uitvoer weer verder en kunnen de nieuwe waardes opnieuw gelezen worden. Hier dient rekening mee gehouden te worden wanneer niet de hele klok uitgelezen wordt: een read van de uren staakt de uitvoer! Altijd dus het tiendenregister lezen na afloop.

De waardes die uit de registers rollen zijn niet binair, zoals we gewend zijn, maar BCD-gecodeerd. Dit staat voor binary coded decimal, een code die iets meer op onze eigen notatirwijze lijkt. Daar waar bijvoorbeeld de waarde 15 er in hex als OF (0000 1111) uitziet ziet ze er in BCD als 15 (0001 0101) uit. De bovenste 4 bits worden gebruikt om de tientallen te representeren (0 t/m 9, dus 0000 t/m 1001), de on-

4.2 De 6526 CIA

derste 4 bits stellen de eenheden voor (ook 0 t/m 9, 0000 t/m 1001). Elk decimale cijfer wordt dus in vier bits gecodeerd zoals we dat gewend zijn. Het gevolg is dat bijvoorbeeld het getal 47 er als 0100 0111 uitziet, in hex is dit (u raadt het al) gelijk aan 47.
BCD is niet eenvoudiger om mee te rekenen, echter des te eenvoudiger om te representeren. Tenslotte, elk halve byte (nibble) stelt een decimaal getal voor, we hoeven geen hexadecimaal naar decimaal conversie uit te voeren!

Uitgaande van het bovenstaande blijken de registers, bij nadere inspectie, als volgt opgebouwd te zijn:
TOD10 (8): bits 0 t/m 3 bevatten de tienden
TODSEC (9): bits 0 t/m 3 bevatten de eenheden, bits 4 t/m 6 bevatten de tientallen
TODMIN (10): bits 0 t/m 3 bevatten de eenheden, bits 4 t/m 6 bevatten de tientallen
TODHR (11): bits 0 t/m 3 bevatten de eenheden, bit 4 bevat de tientallen, bit 7 bevat het AM/PM-bit (1=PM).
Let er op dat de seconden- en minutentientallen slechts 3 bits nodig hebben omdat de waardes van 0 t/m 5 lopen. De niet gebruikte bits zijn echter gelijk aan 0. Hetzelfde geldt voor bits 5 en 6 van register 11.
TOD-write
Bij het beschrijven van de registers dienen de waardes in BCD aangeleverd te worden, de BCD-waardes worden met behulp van de volgende formule gevonden:
BCD=INT(DEC/10)*16+
DEC-10*INT(DEC/10)
Hierbij stelt DEC een willekeurig decimaal getal van 0 t/m 99 voor.
Verder dient rekening gehouden te worden met het feit dat bij het beschrijven van het urenregister TOD stopt. Dit om de juiste tijd in te kunnen stellen. Bij het beschrijven van het tiendenregister (8) begint TOD weer te lopen zodat de juiste tijd altijd gewaarborgd is.

Register 12: SDR
Het serial data register (SDR) bevat de data die over de seriële lijn verzonden moet worden of de data die over de seriële lijn ontvangen is. Zodra dit register beschreven wordt begint de datatransmissie, tenminste wanneer timer A in de continue mode (zie verder) aan het tellen is.

Register 13: ICR
Het interrupt control register (ICR) wordt gebruikt om de diverse interrupts te identificeren en te maskeren. De opbouw is als volgt:
Bit 0: timer A interrupt bit
Bit 1: timer B interrupt bit
Bit 2: alarm interrupt bit
Bit 3: serial port interrupt bit
Bit 4: FLAG interrupt bit
Bit 7: IR of S/C

Bits 0 t/m 4 worden gezet wanneer de bijbehorende gebeurtenis een interrupt heeft gegenereerd. Bit 7 wordt dan ook gezet wanneer die interrupt niet gemaskeerd was. Het maskeren gebeurt door ditzelfde register te beschrijven en gebruik te maken van bit 7 dat dan een Set/Clear functie vervult. Een 1 maskeert (uitschakelen) de interrupts waarvan de bits gezet zijn (de overigen worden niet gewijzigd), een 0 staat de interrupts waarvan de bits gezet zijn weer toe. Een voorbeeld verduidelijkt een en ander wellicht. De opgave is om de interrupt van timer A te maskeren en die van alarm toe te staan:

4.2 De 6526 CIA

LDA # %10000001; bit 7 =1: Set bit 0
STAICR

LDA # %0000010; bit 7=0: Clear bit 2
STA ICR

Registers 14 en 15: CRA en CRB
De control registers A en B worden voor diverse zaken gebruikt. Hier volgt een opsomming per bit voor CRA:
bit 0: 1/0 = start/stop timer A
bit 1: 1/0 = PB6 on/off;
dit bit wordt gebruikt om de pulsjes van timer A naar PB6 door te lussen.
bit 2: 1/0 = toggle/pulse;
dit bit bepaalt of eventuele uitvoer aan PB6 van TA een pulstrein is (elke afloop genereert een puls van 1 klokpuls lengte) of dat op het aflopen van TA de uitgang PB6 telkens wisselt (1 wordt 0, 0 wordt 1). In dat geval is de gemeten frequentie twee maal zo laag!
bit 3: 1/0 = one shot/continuous;
dit bit bepaalt of na het aflopen van timer A opnieuw gestart wordt met aftellen of niet. In beide gevallen worden TALO en TAHI wel weer met de initiële waardes geladen.
bit 4: 1 = force load;
dit bit wordt 1 gemaakt om timer A te initialiseren. Dit kan gebeuren tijdens het lopen en stilstaan van TA.
bit 5: 1/0 = CNT/ø2;
dit bit bepaalt of TA afloopt op de systeemklok (ø2) of op de externe CNT pulsingang.
bit 6: 1/0 = output/input SP;
bit 7: 1/0 = 50 Hz/60 Hz;
maak dit bit dus altijd gelijk aan 1!

Voor CRB gelden de volgende definities:
bit 0: 1/0 = start/stop timer B
bit 1: 1/0 = PB7 on/off;
dit bit wordt gebruikt om de pulsjes van tim B naar PB7 door te lussen.
bit 2: 1/0 = toggle/pulse;
dit bit bepaalt of eventuele uitvoer aan PB7 van TB een pulstrein is (elke afloop genereert een puls van 1 klokpuls lengte) of dat op het aflopen van TB de uitgang PB7 telkens wisselt (1 wordt 0, 0 wordt 1). In dat geval is de gemeten frequentie twee maal zo laag!
bit 3: 1/0 = one shot/continuous;
dit bit bepaalt of na het aflopen van timer B opnieuw gestart wordt met aftelen of niet. In beide gevallen worden TBLO en TBHI wel weer met de intiële waardes geladen.
bit 4: 1 = force load;
dit bit wordt 1 gemaakt om timer B te initialiseren. Dit kan gebeuren tijdens het lopen en stilstaan van TB.
bit 5 en 6:
Deze bepalen tezamen waar TB door geklokt wordt. De volgende combinaties zijn mogelijk:

bit 6	bit 5	gevolg
0	0	TB klokt op ø2
0	1	TB klokt op CNT
1	0	TB klokt op aflopen TA
1	1	TB klokt op aflopen TA, CNT moet hoog zijn

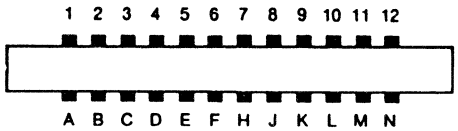
bit7: 1/0 = alarm/TOD;
Dit bit moet een 1 zijn wanneer het alarm ingesteld wordt. Is dit bit een 0 dan wordt automatisch de tijd ingesteld.

6/4.3

Opbouw van de user port

In de vorige paragrafen hebben we ontdekt dat de user port rond de 6526 CIA is opgebouwd. De vraag rijst nu welke pennen van de CIA nu wel en welke pennen nu niet op de user port beschikbaar zijn.

Om deze vraag te beantwoorden dient de volgende figuur bekeken te worden:



Zo is de benaming van de user port pennen wanneer van achteren in de poort gekeken wordt. De pennen hebben de volgende betekenis:

1 - massa
2 - +5, maximaal 100 mA belasting
3 - reset
4 - CNT1: CNT-ingang van CIA # (zie 6/4.2)
5 - SP1: seriële in/uitgang van CIA #1 (zie 6/4.2)
6 - CNT2: CNT-ingang van CIA #2 (zie 6/4.2)
7 - SP2: seriële in/uitgang van CIA #1 (zie 6/4.2)
8 - PC2: handshakelijn van CIA #2 (zie 6/4.2)
9 - SERIAL ATN: als op de seriële bus. Is PA3 van CIA #2
10 - 9V +fase, 50 mA
11 - 9V -fase
12 - massa

A - massa
B - FLAG2: FLAG-ingang van CIA #2 (zie 6/4.2)
C - PB0 van CIA #2
D - PB1 van CIA #2
E - PB2 van CIA #2
F - PB3 van CIA #2
H - PB4 van CIA #2
J - PB5 van CIA #2
K - PB6 van CIA #2
L - PB7 van CIA #2
M - PA2 van CIA #2
N - massa

Poort B van CIA #2 kan gevonden worden op geheugenplaats $DD01 (56577). Het bijbehorende datarichtingsregister bevindt zich op $DD03 (56579). Voorbeeldprogramma's zullen in paragraaf 6/4.4 verschijnen.

6/4.4

TOD-programmering

We hebben het al even over de in de Commodore 64 ingebouwde klok gehad, maar het praktische gebruik ervan is nog niet aan de orde geweest.
Even voor alle duidelijkheid, het gaat hier om de Time Of Dayklok, die een onderdeel vormt van de 6526 CIA-chip. Deze interface-adapter, waarvan de C64 er een tweetal bezit, omvat namelijk naast allerlei andere handige hardware een volledige klok. Nu is dit niet de enige klok die de 64 kent, natuurlijk. Want ook onder BASIC is er een klok beschikbaar, die door de interrupt-routine "bij de tijd" gehouden wordt.

Daarnaast zijn er nog wel andere plekken in de computer-architektuur aan te wijzen waar de een of andere klok loopt al was het maar de klokfrequentie die de hele machine uiteindelijk bestuurt. Voorts, om nog maar een voorbeeld te geven, bevinden zich in de CIA-chip naast deze klok ook nog een tweetal timers, die voor allerlei toepassingen gebruikt worden. Aan klokken en andere tijds-mechanismen geen gebrek!

Waar het om gaat bij de CIA TOD-klok, is dat deze een zogenaamde "real-time"-klok is. Dat wil zeggen dat deze klok, ongeacht wat er in de computer aan de hand is, op tijd blijft lopen. En dat is iets dat de normale systeem-klok (TI$) niet klaarspeelt. Die moet stilgezet worden als er bijvoorbeeld een cassette gelezen of beschreven moet worden. Op dergelijke momenten moet de normale interrupt – die dat klokje op gang houdt – nu eenmaal worden afgeschakeld.

De TOD-klok – zoals deze meestal genoemd wordt – is echter geheel onafhankelijk van de interrupt-routine en kan dan ook voor allerlei toepassingen worden ingezet. De tijd wordt geheel automatisch binnen die CIA-chip bijgehouden, de computer – noch de programmeur – heeft er verder omkijken naar.

Alarm
Behalve alleen maar de tijd bijhouden kan de TOD-klok echter ook nog als wekker fungeren; er is een alarm-functie ingebouwd. Als de kloktijd het van tevoren ingestelde ogenblik bereikt, zal er een processor-interrupt gegenereerd worden, waarna er bijvoorbeeld een alarm gegeven kan worden. Ideaal om te voorkomen dat het tijdens het programmeren weer veel en veel te laat wordt!

24-uur formaat
Tenslotte is de TOD-klok wat makkelijker in het gebruik dan de meeste computertijdmetingen. Maar al te vaak zal een computer intern met decimale tijden werken.

4.4 TOD-programmering

Dat houdt bijvoorbeeld in dat een uur niet uit 60 minuten van ieder 60 seconden bestaat, maar uit 100 decimale minuten die weer in 100 decimale seconden onderverdeeld zijn. Niet echt vreselijk handig, aangezien wij mensen gewend zijn aan ons eigen – eigenlijk onlogische – tijdssysteem.

De Time Of Day klok echter houdt de tijd keurig bij, zoals we dat zelf ook doen: in normale uren, minuten en seconden.
Alleen de tiende-seconden – de kleinste tijdseenheid die de TOD klok bijhouden kan – zijn decimaal, maar dat is gebruikelijk. De uren lopen gelukkig ook netjes van 0 tot en met 12, zoals gebruikelijk op onze wijzerplaten.

Al dat fraais heeft echter één enkel nadeel: het wordt op geen enkele manier door de computer ondersteund. De TOD-klok is weliswaar in de 64 aanwezig – in tweevoud maar liefst: er zijn immers twee van de Complex Interface Adapter-chips in gebruik – maar vanuit BASIC valt er niet zo veel mee te beginnen. Zeker de alarmfunctie, die immers via een interrupt werkt, is jammer genoeg niet binnen een puur BASIC-programma te gebruiken.
En om er nu een fraai stuk machinetaal voor te schrijven is ook niet ieders werk; daar zitten toch nog wel wat haken en ogen aan vast.

Het TOD klok-programma
Om de bovengenoemde reden, is er een speciaal programma ontwikkeld, waarmee die TOD-klok ook vanuit BASIC gebruikt kan worden. Met dit stuurprogramma kan de tijd worden ingesteld en een alarm worden gekozen, dat daarna ook hoorbaar zal afgaan. Bovendien zal er desgewenst een klein klokje in de rechterbovenhoek van het scherm gezet kunnen worden, dat keurig – tot op tienden van seconden nauwkeurig – de tijd bijhoudt.

Om het WEKA TOD-programma te kunnen gebruiken is geen enkele kennis van machinetaal vereist; het volstaat om met een aantal SYS-commando's de verschillende functies te besturen. Door de integratie in WEKA-BASIC echter wordt het besturen nog eenvoudiger. Kortom, als u vrienden en kennissen wilt gaan verbazen met de computer-als-eierwekker, met fraaie geluidseffecten en een animatie van een eitje dat uit de pan sprint op het scherm, dan staat niets u meer in de weg.

Nuttiger toepassingen, waarvan er heel wat te bedenken vallen, mogen natuurlijk ook.

Overigens is het hier gepresenteerde TOD-programma niet alleen nuttig als programma op zich, het is zeer zeker ook een voorbeeld van hoe een beetje gehaaide programmeur gebruik kan maken van sommige van de 64-bouwstenen. Juist omdat bij de Commodore alle input/output via het normale geheugen verloopt, is het vrij gemakkelijk om rechtstreeks met de verschillende chips te "communiceren".
Daardoor ken men, met wat kennis van de opbouw van de computer en van de werking van de diverse I.C.'s, tamelijk simpel allerlei verrassende effecten bereiken.

Zoals we al weten, maakt de TOD-klok deel uit van de 6526 CIA-chips. Deze CIA's nu – Complex Interface Adapters – worden bestuurd door het schrijven van data in de CIA-registers.
In de C64 zijn twee CIA's ondergebracht; één ervan – die we voor het TOD-

4.4. TOD-programmering

programma gebruiken – bevindt zich op adres $DD00. Vanaf dit adres – 56576 in decimaal – vinden we de diverse registers in het geheugenbereik van de 6510-processor.

Voor de TOD-klok op deze CIA zijn de volgende registers beschikbaar:
- vier tijdsregisters:
  - o $DD0B – uren registers;
  - o $DD0A – minuten register;
  - o $DD09 – seconden register;
  - o $DD08 – tiende seconden register.

- twee controle registers:
  - o $DD0E – keuze register 50/60 Hz;
  - o $DD0F – keuze register klok-tijd / alarm-tijd.

- een interrupt-register:
  - o $DD0D – alarm interrupt.

Lezen en schrijven
Het lezen en schrijven van deze registers dient in een bepaalde volgorde plaats te vinden, aangezien er binnen de CIA bepaalde voorzieningen zijn getroffen om bijvoorbeeld te voorkomen dat de tijd tijdens het uitlezen zou veranderen. Om bijvoorbeeld de tijd op een exact te bepalen moment te kunnen starten, wordt de klok gestopt zodra het uur-register beschreven wordt, om pas weer te gaan lopen nadat het tiende seconde-register beschreven is. Met andere woorden, als men zou volstaan met het invullen van uren, minuten en seconden – hetgeen in feite voor veel toepassingen al nauwkeurig genoeg is – zal de klok niet gaan lopen.

Bij het uitlezen gaat iets soortgelijks op; ook dan wordt de klok – althans in de registers zoals die gelezen worden – even gestopt zodra het uur-register gelezen wordt. Pas nadat ook de tiende-seconden gelezen zijn, zal de klok weer doorlopen in de registers. Intern wordt de tijd natuurlijk wel goed bijgehouden, alleen de registers zoals die gelezen worden zijn eventjes "bevroren".

De reden daarvoor is simpel, want als die registers niet bevroren – *gelatcht* zoals het officieel heet – zouden zijn, zou onder het lezen opeens een van de registers van waarde kunnen veranderen.

Het onderstaande voorbeeld laat dit zien.
Stel, dat de vier registers op een gegeven moment de tijd 9:59:59.9 bevatten, dus een tiende seconde voor tien uur. Als op dat ogenblik de tijd wordt uitgelezen, kan tussen het lezen van de verschillende registers nu net weer een tiende seconde verlopen zijn. Zodra dat het geval is, zullen alle registers van inhoud veranderen. In een tabelletje ziet dat er als volgt uit:

Actie	TOD-inhoud	Gelezen tijd
Lees uren	9:59:59:9	9:
Verhoog tijd	10:00:00:0	9:
Lees minuten	10:00:00:0	9:00
Lees seconden	10:00:00:0	9:00:00
Lees tiende secs.	10:00:00:0	9:00:00:0

Oftewel, de gelezen tijd zou in dat geval bijna een heel uur verschillen met de werkelijke tijd. Vandaar dat de registertijd wordt stilgezet tijdens het lezen ervan, als men tenminste aan de "goede kant" – het uur-register als eerste dus – begint.

AM/PM
De TOD-klok hanteert intern een uurwaarde die vanaf 0 tot en met 12 loopt, mat daarbij een aanduiding of het een ochtend- of een middag-tijd is. Dit van

4.4 TOD-programmering

oorsprong Engelse systeem van tijdsregistratie is langzaam maar zeker ook in Nederland ingeburgerd geraakt, met de komst van de digitale horloges.
AM staat voor "voor de middag", PM betekent na het middaguur. Door bij de PM-tijden 12 op te tellen, krijgen we de tijd in het 24-uurs systeem.
Zo is 9 uur AM gewoon 9 uur 's ochtends, maar 9 uur PM is 9 uur 's avonds oftewel 21.00 uur.

Of een bepaalde tijd nu als AM of PM moet worden geïnterpreteerd, wordt aangegeven met het zevende bit van het uurregister..

Alarm stellen
Om een alarm-tijd in te stellen worden dezelfde registers gebruikt, nadat men de CIA heeft "verteld" dat de nu volgende schrijf-operaties niet zozeer de TOD-tijdsregisters maar juist de alarm-registers betreffen.
Om dat te kunnen doen, is er een vlag-bit voorzien in register $DC0F, bit 7. Als dit bit een 0 is, dan zal de CIA schrijf-opdrachten opvatten als het instellen van de TOD-tijd; als we echter dit bitje eerst op 1 zetten, dan stellen dezelfde schrijf-opdrachten de alarm-tijd in.

Het alarm valt echter niet meer terug te lezen, ook als het vlagbit op 1 staat zullen lees-operaties de gewone TOD-tijd opleveren. Kortom, als men die alarm-tijd mogelijk later nog eens nodig heeft, dan moet die ergens in een viertal "schaduw-registers" worden opgeslagen.

BCD-waarden
De TOD-klok werkt – in tegenstelling tot wat er in feite gebruikelijk is in de Commodore 64 – met de zogenaamde BCD-notatie. Deze afkorting staat voor Binary Coded Decimal, hetgeen betekent dat de verschillende getallen niet zomaal als binaire getallen beschouwd kunnen worden. Een getal in BCD-notatie is in feite een soort middenweg tussen ons decimale stelsel en de binaire computer-logica.

Waar het op neer komt is dat voor ieder decimale eenheid – eenheden, tientallen enzovoorts – een groep van vier bits gebruikt worden, waarin het decimale getal binair opgeslagen wordt. Een byte zal in BCD-notatie getallen tussen de 0 en de 99 kunnen bevatten, meer dan genoeg voor een klok dus.
Het grote voordeel hiervan is, dat de te gebruiken logica bij het omzetten van zo'n BCD-getal naar informatie voor bijvoorbeeld een cijfer -display, zoals die bijvoorbeeld in moderne wekkers gebruikt worden, heel beperkt blijft. Iedere groep van vier bits bestuurt een enkel cijfer.

Maar ook voor toepassingen waarbij er geen speciale cijfer-display's worden gebruikt, is BCD soms heel handig. Zo zal in het TOD-programma blijken dat het omzetten van toetsenbord-ingaves naar BCD-getallen en dan weer terug naar scherm-gegevens heel snel kan gaan.

Het programma
Het TOD-programma is zo geschreven dat het direct in WEKA-BASIC opgenomen kan worden. Hiervoor wordt verwezen naar deel 9, hoofdstuk 5.3. De assemblerlisting zal echter in dit hoofdstuk besproken worden om de TOD-programmering geheel te verduidelijken.

De listing valt in twee delen uiteen: interruptroutine en zet-routine. De interruptroutine wordt door de normale 64-

4.4. TOD-programmering

interrupt zo'n 60 maal per seconde aangeroepen en zorg ervoor dat het klokje op het scherm verschijnt en dat het alarm afgaat. Deze beide mogelijkheden zijn afzonderlijk instelbaar: er kan een klok zonder alarm verschijnen, maar ook kan het alram ingeschakeld zijn zonder dat de klok zichtbaar is. De zet-routine wordt gebruikt om de klok en het alarm gelijk te zetten.

Hoe een en ander te werk gaat is te zien in de volgende listing.

```
 10 033C           ALMBIT      =  128
 15 033C           TIMEBIT     =  64
 20 033C           CHRGET      =  $0073
 25 033C           ICR2        =  $DD0D
 30 033C           SID         =  54272
 35 033C           CIA2        =  $DD00
 45 033C           RAM         =  $C48E
 50 033C           ROM         =  $C487
 65 033C           GETBYT      =  $B79E
 70 033C           KOMMA       =  $AEFD
 75 033C           ILLQUAN     =  $B248
 99 C81D           *=$C81D
100 C81D F011      ALARM       BEQ AOFF          ;GEEN PARAMS
110 C81F C991                  CMP #145          ; ON-TOKEN
120 C821 F01F                  BEQ AON
130 C823 38                    SEC               ;CARRY BETEKENT ALM
140 C824 4C7DC8    LOO         JMP SETTIME       ;SET ALARM
149 C827           !
150 C827 F010      TIME        BEQ TOFF
160 C829 C991                  CMP #145
170 C82B F01E                  BEQ TON
180 C82D 18                    CLC               ;C=0, DUS TIME
190 C82E 90F4                  BCC LOO
199 C830           !
200 C830 208EC4    AOFF        JSR RAM
210 C833 2022AB                JSR ALMUIT
220 C836 4C87C4                JMP ROM
290 C839           !
300 C839 208EC4    TOFF        JSR RAM
310 C83C 2045AB                JSR TIMEUIT
320 C83F 4C87C4                JMP ROM
390 C842           !
400 C842 208EC4    AON         JSR RAM
410 C845 204EAB                JSR ALMAAN
420 C848 4C87C4                JMP ROM
490 C84B           !
500 C84B 208EC4    TON         JSR RAM
510 C84E 2080AB                JSR TIMEAAN
520 C851 4C87C4                JMP ROM
590 C854           !
```

4.4 TOD-programmering

```
 900 C854 00      TIMEMODE    BYT 0
 910 C855 00      TEMP3       BYT 0
 990 C856         !
1000 C856 A501    NEWIRQ      LDA 1
1010 C858 8D55C8              STA TEMP3
1020 C85B 208EC4              JSR RAM
1030 C85E 2013AA              JSR DONIRQ
1040 C861 2087C4              JSR ROM
1050 C864 A9C8                LDA #>VERDER
1060 C866 48                  PHA
1070 C867 A971                LDA #<VERDER
1080 C869 48                  PHA
1090 C86A 48                  PHA
1100 C86B 48                  PHA
1110 C86C 48                  PHA
1120 C86D 48                  PHA
1130 C86E 4C0000  EXIT        JMP 0
1140 C871 78      VERDER      SEI
1150 C872 AD55C8              LDA TEMP3
1160 C875 8501                STA 1
1170 C877 68                  PLA
1180 C878 A8                  TAY
1190 C879 68                  PLA
1200 C87A AA                  TAX
1210 C87B 68                  PLA
1220 C87C 40                  RTI
1230 C87D         !
2000 C87D AD0FDD  SETTIME     LDA CIA2+15         ;CRB
2010 C880 297F                AND #%01111111      ;TIME
2015 C882 08                  PHP
2020 C883 9002                BCC L05             ;HOUDEN ZO
2030 C885 0980                ORA #%10000000      ;ALARM
2040 C887 8D0FDD  L05         STA CIA2+15
2050 C88A 209EB7              JSR GETBYT          ;UREN
2060 C88D E018                CPX #24             ;>24
2070 C88F B02B                BCS ILL             ;FOUT
2080 C891 208EC4              JSR RAM             ;UREN INSTELLEN
2090 C894 BD0AAB              LDA UURTAB,X
2095 C897 28                  PLP
2100 C898 9002                BCC L10
2105 C89A 0980                ORA #128
2110 C89C 8D0BDD  L10         STA CIA2+11
2115 C89F 2087C4              JSR ROM
2120 C8A2 20BFC8              JSR HAAL            ;MINUTEN
2130 C8A5 8E0ADD              STX CIA2+10
2140 C8A8 20BFC8              JSR HAAL            ;SECONDEN
2150 C8AB 8E09DD              STX CIA2+9
2160 C8AE A900                LDA #0              ;TIENDEN
2170 C8B0 8D08DD              STA CIA2+8
```

4.4. TOD-programmering

```
 2180 C8B3 AD0FDD            LDA CIA2+15
 2190 C8B6 297F              AND #%01111111
 2200 C8B8 8D0FDD            STA CIA2+15
 2210 C8BB 60                RTS                 ;KLAAR
 2220 C8BC                   !
 2230 C8BC 4C48B2 ILL        JMP ILLQUAN
 2290 C8BF                   !
 2300 C8BF A93A   HAAL       LDA #':
 2310 C8C1 20FFAE            JSR KOMMA+2         ;CHECK OP ':'
 2320 C8C4 209EB7            JSR GETBYT          ;LEES
 2330 C8C7 E03C              CPX #60             ;>60
 2340 C8C9 B0F1              BCS ILL             ;DAN FOUT
 2350 C8CB 208EC4            JSR RAM
 2360 C8CE 20DFAA            JSR NAARBCD         ;ANDERS CONVERTEREN
 2370 C8D1 4C87C4            JMP ROM
 2390 C8D4                   !
 9999 AA13                   *=$AA13
10000 AA13 2C54C8 DONIRQ     BIT TIMEMODE
10010 AA16 102A              BPL L03             ;ALARM=UIT
10020 AA18 AD0DDD            LDA ICR2
10030 AA1B 2964              AND #00000100       ;ALM-BIT
10040 AA1D F023              BEQ L03             ;GEEN ALM-IRQ
10050 AA1F A90F   TOON       LDA #15
10060 AA21 8D18D4            STA SID+24          ;VOLUME
10070 AA24 A9F8              LDA #248
10080 AA26 8D06D4            STA SID+6           ;SUST/REL
10090 AA29 A932              LDA #50
10100 AA2B 8D01D4            STA SID+1           ;FREQ
10110 AA2E A921              LDA #33
10120 AA30 8D04D4            STA SID+4           ;GATE ZAAGTAND
10130 AA33 A200              LDX #0
10140 AA35 A000              LDY #0
10150 AA37 88     L04        DEY                 ;WACHTLUS
10160 AA38 D0FD              BNE L04
10170 AA3A CA                DEX
10180 AA3B D0FA              BNE L04
10190 AA3D A920              LDA #32             ;GATE UIT
10200 AA3F 8D04D4            STA SID+4
10290 AA42                   !
10300 AA42 2C54C8 L03        BIT TIMEMODE
10310 AA45 507B              BVC NODISP          ;GEEN DISPLAY
10320 AA47 A5D6              LDA 214             ;CURSOR OP 0,0
10330 AA49 05D3              ORA 211
10340 AA4B D005              BNE GEENHOME        ;NEE
10350 AA4D E6D6              INC 214             ;JA, DAN 1 REGEL
10360 AA4F 206CE5            JSR 58732           ;OMLAAG
10370 AA52                   !
10380 AA52 A92E   GEENHOME   LDA #'.              ;PUNT
10390 AA54 8D2404            STA 1060
10400 AA57 A93A              LDA #':
```

4.4 TOD-programmering

```
10410 AA59 8D2104           STA 1057
10420 AA5C 8D1E04           STA 1054
10430 AA5F A900             LDA #0          ;KLEUR STATUSREGEL
10440 AA61 A225             LDX #37
10450 AA63 9D00D8 LO        STA 55296,X
10460 AA66 CA               DEX
10470 AA67 E01B             CPX #27
10480 AA69 D0F8             BNE LO
10490 AA6B        !
10500 AA6B AD0BDD UREN      LDA CIA2+11     ;EERST DE UREN
10510 AA6E C992             CMP #146        ;12 AM
10520 AA70 D004             BNE L4
10530 AA72 A912             LDA #18         ;DAN 12:00
10540 AA74 D01C             BNE L1
10550 AA76 AD0BDD L4        LDA CIA2+11
10560 AA79 100B             BPL L3          ;<13:00
10570 AA7B 38               SEC
10580 AA7C E96E             SBC #110        ;ANDERS -110
10590 AA7E C91A             CMP #26         ;20/21 UUR
10600 AA80 F01C             BEQ L2
10610 AA82 C91B             CMP #27
10620 AA84 F018             BEQ L2
10630 AA86 C912   L3        CMP #18         ;12 PM
10640 AA88 D008             BNE L1
10650 AA8A A900             LDA #0          ;DAN 00:00
10660 AA8C 8D0BDD           STA CIA2+11     ;OPBERGEN
10670 AA8F 8D08DD           STA CIA2+8      ;TOD STARTEN
10680 AA92 20C3AA L1        JSR NAARHEX     ;EN PRINTEN
10690 AA95 8C1D04           STY 1053
10700 AA98 8E1C04           STX 1052
10710 AA9B 4CA2AA           JMP MINUTEN
10720 AA9E 6905   L2        ADC #5
10730 AAA0 D0F0             BNE L1
10740 AAA2        !
10750 AAA2 AD0ADD MINUTEN   LDA CIA2+10
10760 AAA5 20C3AA           JSR NAARHEX
10770 AAA8 8C2004           STY 1056
10780 AAAB 8E1F04           STX 1055
10790 AAAE        !
10800 AAAE AD09DD SECONDEN  LDA CIA2+9
10810 AAB1 20C3AA           JSR NAARHEX
10820 AAB4 8C2304           STY 1059
10830 AAB7 8E2204           STX 1058
10840 AABA        !
10850 AABA AD08DD TIENDEN   LDA CIA2+8
10860 AABD 0930             ORA #$30
10870 AABF 8D2504           STA 1061
10880 AAC2        !
10890 AAC2 60     NODISP    RTS
10900 AAC3        !
```

4.4. TOD-programmering

```
11000 AAC3 48      NAARHEX     PHA
11010 AAC4 290F                AND #15             ;BITS 0-3
11020 AAC6 20D4AA              JSR VERTAAL         ;NAAR ASCII
11030 AAC9 A8                  TAY
11040 AACA 68                  PLA                 ;BITS 4-7
11050 AACB 4A                  LSR A
11060 AACC 4A                  LSR A
11070 AACD 4A                  LSR A
11080 AACE 4A                  LSR A
11090 AACF 20D4AA              JSR VERTAAL         ;IDEM
11100 AAD2 AA                  TAX
11110 AAD3 60                  RTS                 ;KLAAR
11130 AAD4 C90A    VERTAAL     CMP #10             ;>10
11140 AAD6 9004                BCC GETAL           ;NEE
 11150 AAD8 69F6                ADC #246            ;JA, DAN -10 (A/F=1/6)
11160 AADA D002                BNE VOLG            ;ONCONDITIONEEL
11170 AADC 6930    GETAL       ADC #$30            ;GETAL:+48
11180 AADE 60      VOLG        RTS
11190 AADF         !
12000 AADF 8A      NAARBCD     TXA
12010 AAE0 A0FF                LDY #255
12020 AAE2 C8      LO6         INY                 ;0
12030 AAE3 38                  SEC                 ;TELKENS -10
12040 AAE4 E90A                SBC #10
12050 AAE6 10FA                BPL LO6             ;NOG NIET <0
12060 AAE8 98                  TYA                 ;Y BEVAT INT(X/10)
12070 AAE9 0A                  ASL A
12080 AAEA 0A                  ASL A
12090 AAEB 0A                  ASL A
12100 AAEC 0A                  ASL A               ;A*16
12110 AAED 8D08AB              STA TEMP1
12120 AAF0 98                  TYA                 ;INT(X/10)
12130 AAF1 0A                  ASL A               ;*2
12140 AAF2 8D09AB              STA TEMP2
12150 AAF5 0A                  ASL A
12160 AAF6 0A                  ASL A               ;*8
12170 AAF7 18                  CLC
12180 AAF8 6D09AB              ADC TEMP2           ;+*2=*10
12190 AAFB 8D09AB              STA TEMP2           ;INT(X/10)*10
12200 AAFE 8A                  TXA
12210 AAFF 38                  SEC
12220 AB00 ED09AB              SBC TEMP2           ;X-INT(X/10)*10
12230 AB03 0D08AB              ORA TEMP1           ;+X AND 15
12240 AB06 AA                  TAX
12250 AB07 60                  RTS                 ;KLAAR
12290 AB08         !
12300 AB08 00      TEMP1       BYT 0
12310 AB09 00      TEMP2       BYT 0
12500 AB0A 000102  UURTAB      BYT 0,1,2,3,4,5,6,7,8,9,16,17,18
12510 AB17 818283              BYT 129,130,131,132,133,134,135,136,
                                   137,144,145
```

4.4 TOD-programmering

```
12590 AB22          !
13200 AB22 2C54C8 ALMUIT    BIT TIMEMODE        ;CHECK STATUS
13210 AB25 101D             BPL KLAAR           ;AL UIT
13220 AB27 A904             LDA #%00000100      ;CLEAR IRQ
13230 AB29 8D0DDD           STA ICR2
13240 AB2C A980             LDA #ALMBIT         ;UITZETTEN
13250 AB2E 4D54C8 LO1       EOR TIMEMODE
13260 AB31 8D54C8           STA TIMEMODE
13265 AB34 D00E             BNE KLAAR           ;ANDERE IRQ ACTIEF
13270 AB36 78               SEI                 ;INT UIT
13280 AB37 AD6FC8           LDA EXIT+1          ;AANPASSEN
13290 AB3A 8D1403           STA $314
13300 AB3D AD70C8           LDA EXIT+2
13310 AB40 8D1503           STA $315
13320 AB43 58               CLI
13330 AB44 60       KLAAR   RTS
13390 AB45          !
13400 AB45 2C54C8 TIMEUIT   BIT TIMEMODE
13410 AB48 50FA             BVC KLAAR           ;AL UIT
13420 AB4A A940             LDA #TIMEBIT
13430 AB4C D0E0             BNE LO1             ;ONCONDITIONEEL
13490 AB4E          !
13500 AB4E 207300 ALMAAN    JSR CHRGET          ;ON OVERSLAAN
13510 AB51 2C54C8           BIT TIMEMODE        ;CHECK STATUS
13520 AB54 30EE             BMI KLAAR           ;AL AAN
13530 AB56 A984             LDA #%10000100      ;SET IRQ
13540 AB58 8D0DDD           STA ICR2
13550 AB5B A980             LDA #ALMBIT         ;AANZETTEN
13560 AB5D 4D54C8 LO2       EOR TIMEMODE
13570 AB60 8D54C8           STA TIMEMODE
13580 AB63 C9C0             CMP #%11000000      ;TWEE IRQ'S
13590 AB65 F0DD             BEQ KLAAR           ;DAN NIET OMLEGGEN
13600 AB67 AD1403           LDA $314            ;ANDERS WEL
13610 AB6A 8D6FC8           STA EXIT+1
13620 AB6D AD1503           LDA $315
13630 AB70 8D70C8           STA EXIT+2
13640 AB73 78               SEI
13650 AB74 A956             LDA #<NEWIRQ
13660 AB76 8D1403           STA $314
13670 AB79 A9C8             LDA #>NEWIRQ
13680 AB7B 8D1503           STA $315
13690 AB7E 58               CLI
13700 AB7F 60               RTS
13710 AB80          !
13800 AB80 207300 TIMEAAN   JSR CHRGET
13810 AB83 2C54C8           BIT TIMEMODE
13820 AB86 70BC             BVS KLAAR           ;AL GEZET
13830 AB88 AD0EDD           LDA CIA2+14
```

4.4. TOD-programmering

```
13840 AB8B 0980                 ORA #128            ;50 HZ
13850 AB8D 8D0EDD               STA CIA2+14
13860 AB90 A940                 LDA #TIMEBIT
13870 AB92 D0C9                 BNE L02
13890 AB94          !
```

4.4 TOD-programmering

Werking
Van bovenstaande listing zijn de regels tot 2000 niet van belang voor het verhaal m.b.t. de TOD-programmering, vandaar dat die hier buitenwege gelaten worden.

regels 2000 t/m 2210: SETTIME
De routine SETTIME wordt gebruikt om de tijd en/of het alarm in te stellen, afhankelijk van de waarde van de carry-vlag: is deze gezet dan wordt het alarm ingesteld, is de carry-vlag gewist dan wordt de tijd ingesteld. Het instellen van tijd of alarm wordt bepaald door bit 7 in CIA-register 15. Is dit bit een 1 dan kan het alarm ingesteld worden, is dit bit niet gezet dan neemt de CIA aan dat de tijd ingesteld moet worden. Het instellen van dit bit gebeurt in de eerste 6 regels, allereerst wordt het bit gewist (2010), is echter de carry gezet dan wordt dit bit weer op 1 gezet (2030).

Het instellen van de uren gebeurt via een uren-tabel, dit voorkomt een hoop ingewikkelde rekenarij. Het gebruik van een tabel is noodzakelijk omdat de CIA niet met de uren 0 t/m 23 werkt, echter met BCD-uren. Verder is van de middaguren (PM) het hoogste bit ook nog eens gezet. Al met al reden om toevlucht tot een tabel te nemen.
Een klein probleempje doet zich voor bij de stand 12:00 t/m 12:59 PM (12:00 t/m 12:59). De CIA ziet dit als 12:00 AM en dient dus met hex $12 (18) gevuld te worden. Dit gebeurt ook. Voor het alarm is het echter noodzakelijk om de CIA met 12:00 PM ($92) te vullen! Waarom is niet geheel duidelijk, in de praktijk blijkt het echter te kloppen. Vandaar dat in regel 2105 het hoogste bit alsnog gezet wordt indien we met het alarm bezig zijn. Dit was, zoals bekend, aan het carry-bit te zien.

Na het instellen van de uren wordt het tijd om de minuten en seconden in te stellen, dit is een vrij eenvuodige gebeurtenis die via de subroutine HAAL (2300 t/m 2370) opgelost wordt. HAAL checkt op een ':' en leest dan een byte-waarde die van 0 t/m 59 mag lopen. De routine NAARBCD vertaalt een getal in het X-register naar zijn BCD-equivalent, dit komt ook in X terecht. De routine NAARBCD staat in de regels 12000 t/m 12250 (zie verder). Na het opbergen van zowel de minuten en de seconden is SETTIME ten einde en kan via een RTS (2210) teruggesprongen worden.

regels 12000 t/m 12250: NAARBCD
De routine NAARBCD maakt gebruik van de volgende formule:
BCD=16*(X DIV 10)+(X MOD 10)
Hierbij staat X DIV 10 voor INT(X/10) en X MOD 10 voor de rest van de deling X/10, oftwel X-10*INT(X/10).
Deze sommetjes zijn uiteraard ook in machinetaal uitvoerbaar, het delen door 10 wordt bewerkstelligd door telkens 10 van X af te trekken en het aantal malen dat deze aftrekking een positief resultaat oplevert Y met één te verhogen. Het resultaat is dat INT(X/10) in Y terecht komt (regels 12000 t/m 12050). Het vermenigvuldigen met 16 gebeurt door Y in de accu te zetten en deze 4 maal naar rechts te schuiven (4 keer *2 = *16). Het resultaat kan nu tijdelijk opgeslagen worden (12110).

Het resultaat (Y) dient nu met 10 vermenigvuldigd te worden, dit is mogelijk door het eerst met 8 te vermenigvuldigen en dan het dubbele er bij op te tellen. Dit gebeurt in de regels 12120 t/m 12190: Y wordt (via de accu) met twee vermenigvuldigd en in TEMP2 opgeslagen, daarna

4.4. TOD-programmering

wordt A nog eens twee maal met twee vermenigvuldigd. Het gevolg is dat A nu 8*Y bevat. Door TEMP2 (2*Y) dan nog bij de accu op te tellen is het resultaat gelijk aan 10*Y oftewel 10*INT(X/10).

Dit laatste resultaat wordt in 12200 t/m 12220 van X afgetrokken, waarna alleen nog X DIV 10 (TEMP1) hierbij opgeteld hoeft te worden (12230). Dit resultaat wordt dat naar X overgebracht waarop de routine ten einde is.

regels 10000 t/m 12890: NEWIRQ
Dit is de routine die aan de originele interruptroutine 'geplakt' wordt. Het doel is om telkens de tijd op het scherm te zetten en om te kijken of er een alarm afgaat. De routine wordt alleen aangeroepen indien de interruptroutine is omgeleid.
Allereerst moet er gekeken worden of de interrupt van het alarm afkomstig is. Is dit het geval dan wordt de TOON-routine (10050 t/m 10200) uitgevoerd. Deze routine laat een piep via de SID horen. Of de interrupt van het alarm komt is te zien aan bit 2 van het interrupt control register; is dit bit gezet dan is er een alarm-interrupt opgetreden. Tevens moet worden gecheckt of het alarm aanstaat, dit gebeurt in 10000. De variabele TIMEMODE bevat twee bits die aanduiden of het alarm en/of het time-display aanstaan. Bit 7 bepaalt de alarm-mode, bit 6 bepaalt de time-mode. De BIT-instructie zet bit 7 van TIMEMODE in de N-vlag, vandaar de BPL in 10010. Deze wordt alleen uitgevoerd indien bit 7 niet gezet is, d.w.z. wanneer het alarm uit staat. Ditzelfde geldt voor het time-display, is dit geactiveerd dan is bit 6 in TIMEMODE gezet. De BIT-instructie in 10300 zal dan de V-vlag zetten zodat de BVC niet wordt uitgevoerd, is het display niet geactiveerd dan wordt de BVC wel uitgevoerd.

Het neezetten van de tijd wordt voorafgegaan door het checken van de cursorpositie, is deze gelijk aan de home-positie dan wordt de cursor één regel verlaagd. Dit is erg praktisch omdat de tijd rechtsboven in het scherm verschijnt en dus met LIST-opdrachten e.d. interfereert.

Het omrekenen van CIA-uren naar BCD-uren gebeurt niet via een tabel maar met pure rekenarij. Dit heeft de regels 10510 t/m 10650 tot gevolg. Opletten geblazen is het weer bij 12.00 AM, dit is gelijk aan 0:00, vandaar de regels 10650 t/m 10680; de TOD wordt hier op 00:00:00 gezet.
De routine NAARHEX wordt gebruikt om een BCD-getal naar twee hex-ASCII waardes om te zetten, het highnibble (de bovenste vier bits) komen in X terecht, het lownibble (de onderste vier bits komt in Y terecht. Deze kunnen zo gePRINT of gePOKEt worden, in onze routine gebeurt het laatste. De tienden van seconden kunnen alleen de waardes 0 t/m 9 aannemen, vandaar dat het volstaat om in 10860 deze waarde met $30 te OR-ren. Het resultaat is dan een ASCII-code die van 48 t/m 57 ('0' t/m '9') loopt.

regels 11000 t/m 11180: NAARHEX
De routine NAARHEX neemt de onderste vier bits van de accu en maakt hier een ASCII-waarde 48 t/m 57 ('0' t/m '9') of 1 t/m 6 ('A' t/m 'F') van. Dit werkt uiteraard alleen voor screen-POKEs omdat de normale ASCII-waardes van 'A' t/m 'F' van 65 t/m 70 lopen.
Het vertalen gebeurt door het nibble via VERTAAL eerst op een getal (0 t/m 9) te checken, komt deze check uit dan wordt er slechts 48 opgeteld (11170). Komt de check niet uit dan wordt er 9 afgetrokken

4.4 TOD-programmering

(oftewel 247 bij opgeteld, de carry is namelijk gezet, regel 11150) zodat het resultaat een getal van 1 t/m 6 is.

regels 13200 e.v.
Deze regels worden gebruikt om de interruptroutine aan of uit te zetten, er wordt echter wel gekeken of de interruptroutine al aktief is. Is dit namelijk het geval dan dient de routine niet nog een keer omgelegd te worden. Tevens wrden in TIME-MODE de juiste bits aan- en uitgezet zodat de interruptroutine weet welke routines wel en niet uitgevoerd moeten worden.

Gebruik
De routines kunnen het best vanauit WEKA-BASIC gebruikt worden, het is echter mogelijk om bovenstaande routines ook via SYS-commando's aan te roepen. De aanroepadressen zijn dan:
AOFF ($C830/51248): zet het alarm uit
TOFF ($C839/51257): zet het time-display uit
AON ($C842/51266): zet het alarm aan
TON ($C84B/51275): zet het time-display aan
($C823/51235): zet het alarm
($C82D/51245): zet de tijd

De laatste twee routines hebben uiteraard parameters, de correcte aanroep is dan ook
SYS (51235) 20:00:00
om het alarm op 8 uur 's avonds te zetten en
SYS (51245) 9:45:00
om de tijd op kwart voor 10 te zetten. Vergeet de haakjes om de SYS-adressen niet! Verder dienen de ROM- en RAM-routines aanwezig te zijn, indien er dus geen gebruik van WEKA-BASIC gemaakt wordt moeten de volgende regels toegevoegd worden:

```
19999 C487          *=$C487
20000 C487 A501     ROM           LDA 1
20010 C489 0901                   ORA #1
20020 C48B 8501                   STA 1
20030 C48D 60                     RTS
20040 C48E          !
20100 C48E A501     RAM           LDA 1
20110 C490 29FE                   AND #254
20120 C492 8501                   STA 1
20130 C494 60                     RTS
20140 C495          !
```

Eigenaardigheden
Tijdens het ontwerpen en gebruiken van het TOD-programma zijn er een aantal onverklaarbare zaken op komen duiken, het auteursteam houdt zich dan ook aanbevolen voor oplossingen van lezers met betrekking tot de volgende zaken:
- Het eerste alarm dat wordt ingesteld en ook wordt bereikt geeft geen interrupt. Het is dus noodzakelijk om eenmalig een dummy-alarm te forceren.
- Soms wordt een uitgevoerd alarm na een minuut herhaald.
  Oorzaak is waarschijnlijk een fout in de CIA, omdat een minuut later de CIA-registers een fractie van een seconde dezelfde stand hebben als de minuut daarvoor.
- Eerder is al genoemd en opgelost het probleem dat 12:00 voor het alarm als 12:00 PM ingesteld moet worden. Dit blijft een vreemde zaak. Suggesties zijn welkom.

Op bovenstaande punten na echter is de CIA een leuke chip om mee te stoeien, hopelijk draagt bovenstaand programma bij tot meer inzicht in deze bouwsteen.
Voorbeelden en de BASIC-lader kunnen gevonden worden in deel 9, hoofdstuk 5.3.

6/4.5

Keyboard-interface

Vriend en vijand zijn het over op zijn minst één eigenschap van de Commodore 64 eens: zijn muzikale capaciteiten zijn onovertroffen. Tenminste, voor een huiscomputer van f. 500,-- dan. Het nadeel is echter dat de muziek die er in zit er zo moeilijk weer uit te halen is. Daartoe zijn inmiddels tientallen programma's verschenen, de één nog mooier dan de ander. Een nadeel van al deze programma's is echter dat de muziek die er mee gespeeld kan worden meestal via het toetsenbord ingegeven moet worden, hetzij van te voren, hezij in 'real time'. En daar ligt nu juist het probleem, een schrijfmachinetoetsenbord leent zich natuurlijk niet best voor het spelen van grote klassiekers. Vandaar dat de volgende stap, logischerwijs, een volwaardig keyboard is dat op de user-port van de 64 aangesloten kan worden. Ook deze keyboards zijn commercieel verkrijgbaar, de prijs wil echter meestal aan de hoge kant zijn en de bijgeleverde software is meestal ook niet alles. Vandaar dat we in deze aanvulling de interface bespreken die nodig is om een 3 1/2-oktaaf toetsenbord op de Commodore 64 aan te sluiten. Tevens wordt er een stuk software gepersenteerd dat het mogelijk maakt om onder interruptsturing te spelen, zodat de computer vrij blijft voor andere zaken. In de software zitten tevens definities voor 13 verschillende muziekinstrumenten, dit aantal kan naar believen uitgebreid worden. Ook het toetsenbord kan vergroot worden, de interface verzorgt de uitlezing van 64 toetsen zodat er ruim 5 oktaven omvat kunnen worden (1 oktaaf is 12 halve tonen).

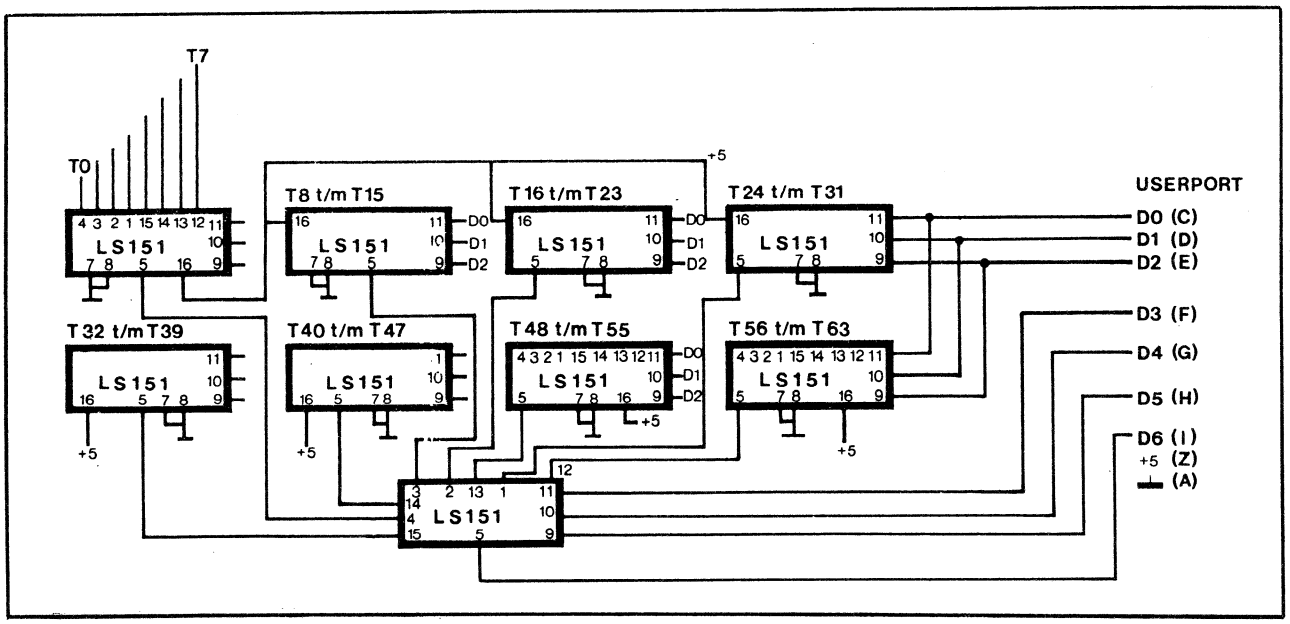
De interface

De interface staat met 7 lijnen in verbinding met de userport, 6 uitgaande en 1 ingaande. De 6 uitgaande lijnen worden gebruikt om een multiplexer aan te sturen (deze heeft dan een keuze uit 2^6= 64 mogelijkheden), via de ingaande lijn wordt de waarde van de geselecteerde toets gelezen. Omdat er uiteraard geen 64→1 multiplexers bestaan is hier gebruik gemaakt van negen 8→1 multiplexers. Vier 16→1 multiplexers kunnen ook gebruikt worden, een blik in het IC-Handboek leert echter dat deze 4 IC's dan meer stroom nodig hebben dan de 64 kan leveren, zodoende dàt er hier voor negen 74 LS 151-IC's gekozen is. Deze hebben elk 6mA nodig.

Uitgaande van deze gegevens kan het schema van figuur 6/4.5-1 getekend worden.

In de figuur is eenvoudig te zien hoe de 9 multiplexers via een buffertje op de 64 aangesloten kunnen worden. De buffer is nodig omdat de user-port slechts 1 TTL-load kan aansturen en verder is een buffer aanbevelenswaardig in verband met kortsluiting.

6/4.5 Keyboard-interface



Figuur 6/4.5-1: Schema van de keybord-interface.

Wordt er nu een waarde tussen 0 en 63 op de 6 datalijnen D0 t/m D5 'gePOKEt' dan wordt één van de 64 ingangen van de multiplexers geselecteerd. D0 t/m D2 selecteren op de 'onderste' 8 multiplexers gelijktijdig 8 lijnen, D3 t/m D5 verzorgen dan weer een selectie uit één van deze acht.

Omdat een TTL-ingang op '1' staat als deze in de lucht hangt is het het eenvoudigst om iedere ingangslijn via een schakelaar (het keyboard) met de 0 te verbinden . Is een toets niet ingedrukt dan komt dat overeen met een logische 1, is de toets wel ingedrukt dan komt dit overeen met een logische 0.

Er zijn legio mogelijkheden om aan een geschikt toetsenbord te komen. De meest gebruikelijke is om er een te kopen. Zelf heb ik echter gebruik gemaakt van een afdankertje. Dit was een oude elektrische piano waar meer kapot dan heel aan was, het keyboard gedeelte (het enige wat nodig is) werkte echter (op een enkele toets na) perfekt. Het strekt wellicht tot aanbeveling om hier en daar eens rond te kijken naar zoiets.

De software

Daar de software toch niet zo recht-toe recht-aan is als men zou vermoeden wordt de globale werking van de assenblerversie aan de hand van de WEKA-BASIS-versie uitgelegd.
De BASIC-versie maakt gebruik van een machinetaal-routine, die het toetsenbord van links naar rechts uitleest (scant) en de aangeslagen toetsen (met een maximum van 3) in een buffer opslaat. Deze buffer begint op geheugenplaats 828, gekenmerkt door de variabele T. Het aantal ingedrukte toetsen (1 t/m 3) wordt in geheugenplaats 2 opgeslagen.

Hier van uitgaande kunnen we listing 6/4.5-1 gaan bekijken. De regels 5 t/m 50 kunnen dan meteen overgeslagen worden, dit zijn wat initialisaties voor de sound-

4.5 Keyboard-interface

chip (de SID) en de variabelen. De variabelen A en F worden gebruikt ter bepaling van de frequentie, hiervoor wordt verwezen naar deel 111.
Het programma is gebaseerd op het volgende principe:
- Toetsenbord uitlezen;
- indien 0 toetsen, dan stemmen uit;
- spelende tonen prolongeren in zelfde stem;
- nieuwe tonen in vrije stemmen opbergen.

Vooral punt 3 is van belang, daar het niet zomaar mogelijk is om toets 1 in stem 1, toets 2 in stem 2 en toets 3 in stem 3 op te bergen. Dit heeft te maken met het feit dat een toon een bepaalde 'omhullende' heeft die de amplitude (volume) van die toon bepaalt. Het zomaar verschuiven van 'lopende' noten heeft onderbrekingen van deze amplitude-sequence tot gevolg wat resulteert in lelijke klikken.
Ook punt 4 is niet vanzelfsprekend, het zoeken van een vrije stem moet telkens ergens anders gebeuren, anders zou een 'loopje over het klavier' resulteren in een (enkele) noot in stem 1, gevolgd door weer een enkele noot in stem 1. Indien de omhullende op nagalm is ingesteld (bijvoorbeeld bij viool) dan zou dit effect op die manier verloren gaan. In dit programma is gekozen voor 'zoeken vanaf stem 1' de ne keer en 'zoeken vanaf stem 3' de andere keer.

Regel 60 bepaalt of er toetsen zijn ingedrukt. Is dit niet het geval dan worden alle stemmen die nog niet leeg waren (S(K) = 255 betekent stem is leeg) leeg gemaakt door hun waarde in de array S op 255 te zetten en door de gespeelde noot in de array N op 0 te zetten (0 betekent niet toegekend aan een of andere stem). Het commando GATEK zorgt er voor dat de noot ook daadwerkelijk niet verder gespeeld wordt.

Zijn alle stemmen leeg gemaakt dan kan er weer naar regel 60 gesprongen worden (er hoeft immers niets meer te gebeuren).

Zijn er echter wel toetsen ingedrukt, dan moeten de toetsen die al ingedrukt waren en nu zijn losgelaten uitgezet worden. In de array S staat welke toets (noot) bij welke stem hoort. Vandaar dat bij elke ingedrukte toets (PEEK (T+J) gezocht wordt of deze zich in een stem (S(I)) bevindt. Zo ja, dan wordt B ongelijk 0 gemaakt. Blijkt aan het eind dat B echter 0 is dan is de toets (noot) niet geprolongeerd en kan ook deze uitgezet worden (eind regel 80).

Aangekomen bij regel 100 zijn dan eventuele vrije stemmen uitgezet en kunnen nieuwe noten toegevoegd worden. Blijkt een toets al ingedrukt te zijn (regel 110: N(TP)<>0) dan wordt via 140 de (eventuele) volgende toets bekeken. Is N(TP) echter wel gelijk aan 0 (de toets is dan 'nieuw') dan kan er naar een lege stem gezocht worden. De variabele L, welke tussen 0 en 1 wisselt via L=1-L, bepaalt of er van onder naar boven of van boven naar onder gezocht wordt. Regel 120 kan nu bepalen welke stem vrij is (S(J)=255) en deze vrij stem van nootwaarde (TP) voorzien. Ook N(TP) krijgt een waarde, namelijk de waarde van de stem waar hij aan toegekend is. FREQJ, A*F ↑ TP en GATEJON zorgen voor het instellen van de frequentie en het aanzetten van de stem.
Het commando J=V aan het einde van regel 120 zorgt ervoor dat de FOR-NEXT-lus 120/130 zonder kleerscheuren verlaten kan worden. Is er namelijk een vrije stem gevonden dan hoeft (en mag!) er niet

verder gezocht worden.

De machinetaalversie is identiek aan de BASIC-versie, op de volgende punten na:
Het aantal ingedrukte toetsen staat in TOETSEN, niet in 2;
de ML-versie wordt automatisch (interrupt) uitgevoerd;
er is een functie-uitlezing toegevoegd;
er is oktaaf-verschuiving mogelijk;
de lees-routine (SYS Z) is de routine in 110 t/m 320.

Om de analogie met de BASIC-versie enigszins te verduidelijken is gebruik gemaakt van labels die naar regelnummers verwijzen (bijvoorbeeld R60). Verder is er commentaar toegevoegd zodat de listing redelijk te begrijpen moet zijn. We zullen hier dan alleen even bij de routine FUNCTIES stilstaan, welke een aantal ingedrukte toetsen (in dit geval vaste schakelaars) vertaalt naar een instrument-definitie.
Deze routine (regel 6000 t/m 6290) kijkt naar CONTROL welke door de routine INSTRUMENT is ingevuld. INSTRUMENT kijkt naar de lijnen 44 t/m 51 en bergt deze 8 lijnen exact in CONTROL op. Het is nu mogelijk om elke lijn een instrument te laten kiezen (mooi, maar dit vreet lijnen en schakelaars), het is ook mogelijk om van deze lijnen er 5 (zoals ik heb gedaan) te gebruiken om 32 instrumenten te kunnen definiëren. Bijgeleverd zijn, zoals gezegd, 13 instrumenten, er kunnen dus zonder problemen 18 toegevoegd worden (regels 11000 e.v.). Eén definitie bepaalt het 'lege' instrument, oftewel geen instrument (regel 11000).

De definities bestaan allen uit 4 bytes. De eerste twee bytes bepalen de omhullende (AD en SR), het derde byte bepaalt de golfvorm (16=driehoek, 32=zaagtand, 64=plus en 128=ruis). Is de gekozen golfvorm puls dan bevinden zich in de onderste 4 bits van het derde byte de bits 8 t/m 11 van de pulsbreedtedefinitie. Bits 0 t/m 7 bevinden zich dan in het vierde byte.
Om bijvoorbeeld een instrument met de volgende specificaties te kiezen ziet de definitie er als volgt uit:

Specificatie :Attack=5, decay=12, sustain=14, release=7
puls=driehoek, pulsbreedte=1024 (25%)
Definitie :Byte1=A*16+D=92
Byte2=S*16+7=32
Byte3=64+(1024/256)=68
Byte4=1024 AND 255=0
Dus :92,231,68,0

Mogelijke uitbreidingen
Allereerst is het uiteraard mogelijk om de SID-chip compleet uit te buiten, dit valt echter buiten het bestek van dit hoofdstuk. Verder kunnen er eventueel meer toetsen aangesloten worden door gebruik te maken van meer multiplexers en 7 uitgangen. Hier wordt de dradenconcentratie echter veel hoger, 51 toetsen met 7 multiplexers verbinden is al een hele klus.
Een laatste (nuttige) uitbreiding is het uitbreiden van de 64 met een tweede SID-chip. Er kunnen dan zes stemmen (in plaats van drie nu) polifoon gespeeld worden en het bestaande programma hoeft daartoe nauwelijks uitgebreid te worden.
Er zijn diverse manieren om een tweede SID aan de Commodore 64 te hangen, wellicht dat we een van deze manieren in een latere aanvulling behandelen.

Nog een laatste opmerking: voor de niet-assembler bezitters is er een BASIC-lader (listing 6/4.5) bijgevoegd.

4.5 Keyboard-interface

```
5 IFA=0THENA=1:LOAD"KEYB.M B",8,1
10 SIDINIT:VOLUME15
20 ADSR1,1,0,15,10:WAVE1S:GATE1
30 ADSR2,1,0,15,10:WAVE2S:GATE2
40 ADSR3,1,0,15,10:WAVE3S:GATE3
50 A=2227:F=2↑(1/12):Z=49152:T=827:G=255:DIMN(63)
60 SYSZ:MT=PEEK(2):IFMT<>0THEN65
62 FORK=1TO3
63 IFS(K)<>GTHENN(S(K))=0:S(K)=G:GATEK
64 NEXT:GOTO60
65 FORI=1TO3
70 IFS(I)=GTHEN90
80 B=0:FORJ=1TOMT:B=B+(S(I)=PEEK(T+J)):NEXT:IFB=0THENN(S(I))=0:S(I)=G:GATEI
90 NEXT
100 FORI=1TOMT:TP=PEEK(T+I)
110 IFN(TP)<>.THEN140
111 L=1-L:W=1:X=3:Y=1:V=4:IFLTHENW=3:X=1:Y=-1:V=0
120 FORJ=WTOXSTEPY:IFS(J)=GTHENS(J)=TP:N(TP)=J:FREQJ,A*F↑TP:GATEJON:J=V
130 NEXT
140 NEXT
170 GOTO60

READY.
```

```
 10 033C           STEMMEN     =  3
 15 033C           BUFFER      =  828
 20 033C           DDRB        =  56579        ;VOOR TOETSEN
 25 033C           POORTB      =  56577        ;RICHTING POORT B
 30 033C           SID         =  54272
 99 C000           *=49152                     ;SOUND-CHIP
100 C000 2091C1                 JSR INIT
101 C003 78                     SEI             ;INT. DISABLE
102 C004 A910                   LDA #<R60       ;INT. OMLEGGEN
103 C006 8D1403                 STA $314        ;NAAR R60
104 C009 A9C0                   LDA #>R60
105 C00B 8D1503                 STA $315
106 C00E 58                     CLI
107 C00F 60                     RTS
110 C010 A002     R60           LDY #STEMMEN-1
115 C012 A9FF                   LDA #255        ;BUFFER LEEG
120 C014 993C03   LO            STA BUFFER,Y
130 C017 88                     DEY
140 C018 10FA                   BPL LO
150 C01A A93F                   LDA #63         ;ONDERSTE 6
160 C01C 8D03DD                 STA DDRB        ;LIJNEN OUTPUT
165 C01F 20F9C0                 JSR INSTRUMENT  ;INSTELLEN
170 C022 A200                   LDX #0
180 C024 A000                   LDY #0
190 C026 8C01DD   LUS           STY POORTB      ;INGANGSDEF.
200 C029 AD01DD                 LDA POORTB
210 C02C 2940                   AND #%01000000  ;INGANG LEZEN
230 C02E F009                   BEQ VERDERO     ;NIET INGEDRUKT
240 C030 98                     TYA             ;WEL
250 C031 9D3C03                 STA BUFFER,X    ;OPBERGEN
260 C034 E8                     INX
270 C035 E003                   CPX #STEMMEN    ;BUFFER VOL"?
280 C037 F005                   BEQ KLAAR       ;JA
290 C039 C8       VERDERO       INY             ;NEE
300 C03A C02C                   CPY #44         ;ALLE TOETSEN"?
310 C03C D0E8                   BNE LUS         ;NEE
320 C03E 8E07C2   KLAAR         STX TOETSEN     ;JA
```

4.5 Keyboard-interface

```
330 C041 E000
340 C043 D01D
350 C045 A003
360 C047 B901C2 R63
370 C04A C9FF
380 C04C F00E
390 C04E AA
400 C04F A900
410 C051 9D56C2
420 C054 A9FF
430 C056 9901C2
440 C059 2027C1
450 C05C 88       R64
460 C05D D0E8
470 C05F 4CE3C0
480 C062 A003     R65
490 C064 B901C2 R70
500 C067 C9FF
510 C069 F02C
520 C06B A900
530 C06D 8D05C2
540 C070 AE07C2
550 C073 B901C2 L1
560 C076 DD3B03
570 C079 D003
580 C07B EE05C2
590 C07E CA       V1
600 C07F D0F2
610 C081 AD05C2
620 C084 D011
630 C086 B901C2
640 C089 AA
650 C08A A900
660 C08C 9D56C2
670 C08F A9FF
680 C091 9901C2
690 C094 2027C1
700 C097 88       R90
710 C098 D0CA
720 C09A AE07C2
730 C09D 8E05C2 R100
740 C0A0 BD3B03
750 C0A3 A8
760 C0A4 B956C2
770 C0A7 D037
780 C0A9 8C06C2
790 C0AC EE08C2
800 C0AF AD08C2
810 C0B2 2901
820 C0B4 F014
850 C0B6 A003
860 C0B8 B901C2 R120
870 C0BB C9FF
880 C0BD D005
890 C0BF 20E6C0
900 C0C2 A001
910 C0C4 88       R130
920 C0C5 D0F1
940 C0C7 4CDDC0
970 C0CA A001     VERSIE2
980 C0CC, B901C2 R120BIS
990 C0CF C9FF
1000 C0D1 D005
     CPX #0                ;NIETS INGEDRUKT
     BNE R65               ;WEL
     LDY #STEMMEN
     LDA STEM-1,Y
     CMP #255              ;VOORGAANDE NOOT
     BEQ R64               ;LEEG
     TAX                   ;NIET LEEG, DAN
     LDA #0                ;UITZETTEN
     STA NOOT,X
     LDA #255
     STA STEM-1,Y
     JSR GATEUIT           ;NR IN Y
     DEY
     BNE R63               ;ALLES AFGEHANDELD,
     JMP WEG               ;DAN KLAAR
     LDY #STEMMEN
     LDA STEM-1,Y
     CMP #255              ;ONGEBRUIKTE STEM
     BEQ R90               ;JA
     LDA #0                ;NEE, DAN TOETS
     STA TEMP              ;VINDEN
     LDX TOETSEN
     LDA STEM-1,Y
     CMP BUFFER-1,X
     BNE V1
     INC TEMP              ;GEVONDEN
     DEX
     BNE L1
     LDA TEMP              ;TOETS GEHANDHAAFD
     BNE R90               ;JA
     LDA STEM-1,Y          ;NEE, DAN
     TAX                   ;UITZETTEN
     LDA #0
     STA NOOT,X
     LDA #255
     STA STEM-1,Y
     JSR GATEUIT
     DEY
     BNE R70
     LDX TOETSEN           ;AANTAL
     STX TEMP
     LDA BUFFER-1,X        ;TOETS
     TAY                   ;AANZETTEN
     LDA NOOT,Y
     BNE R140
     STY TEMP2
     INC KEUZE             ;STEM KIEZEN
     LDA KEUZE             ;(1-> OF 3<-)
     AND #1
     BEQ VERSIE2           ;OM EN OM
     LDY #STEMMEN
     LDA STEM-1,Y
     CMP #255              ;LEEG"?
     BNE R130
     JSR DOR120            ;DAN GEBRUIKEM
     LDY #1
     DEY
     BNE R120
     JMP VERDER
     LDY #1                ;IDEM, MAAR
     LDA STEM-1,Y          ;DAN ANDERSOM
     CMP #255
     BNE R130BIS
```

4.5 Keyboard-interface

```
1010 C0D3 20E6C0            JSR DOR120          ;GEBRUIKEN
1020 C0D6 A003              LDY #STEMMEN
1030 C0D8 C8      R130BIS   INY
1040 C0D9 C004              CPY #STEMMEN+1
1050 C0DB D0EF              BNE R120BIS
1070 C0DD AE05C2  VERDER    LDX TEMP            ;KLAAR
1080 C0E0 CA      R140      DEX
1090 C0E1 D0BA              BNE R100
1100 C0E3 4CB0C1  WEG       JMP FUNCTIES        ;GELUID
1200 C0E6 AD06C2  DOR120    LDA TEMP2
1210 C0E9 9901C2            STA STEM-1,Y        ;NOOT
1220 C0EC AA                TAX                 ;OPBERGEN IN STEM
1230 C0ED 98                TYA                 ;STEM OPBERGEN
1240 C0EE 9D56C2            STA NOOT,X          ;IN NOOT
1250 C0F1 8A                TXA
1260 C0F2 203CC1            JSR FREQ            ;S IN Y, N IN A
1270 C0F5 2015C1            JSR GATEON          ;AANZETTEN
1290 C0F8 60                RTS
1300 C0F9 A02C    INSTRUMENT LDY #44            ;V.A 44
1310 C0FB A900              LDA #0
1320 C0FD 8D0DC2            STA CONTROL         ;OPBERGEN
1330 C100 8C01DD  INSLUS    STY POORTB
1340 C103 AD01DD            LDA POORTB
1350 C106 2940              AND #%01000000
1360 C108 18                CLC
1370 C109 F001              BEQ LINS1
1380 C10B 38                SEC
1390 C10C 6E0DC2  LINS1     ROR CONTROL
1400 C10F C8                INY
1410 C110 C034              CPY #52             ;LAATSTE INGANG
1420 C112 D0EC              BNE INSLUS
1430 C114 60                RTS
2000 C115 B90EC2  GATEON    LDA WAVES-1,Y
2010 C118 0901              ORA #1              ;AAN
2020 C11A 990EC2            STA WAVES-1,Y
2030 C11D 48                PHA
2040 C11E B938C1            LDA TAFELVAN7-1,Y   ;*7
2050 C121 AA                TAX
2060 C122 68                PLA
2070 C123 9D04D4            STA SID+4,X         ;OPBERGEN
2080 C126 60                RTS
3000 C127 B90EC2  GATEUIT   LDA WAVES-1,Y
3010 C12A 29FE              AND #254            ;UIT
3020 C12C 990EC2            STA WAVES-1,Y
3030 C12F 48                PHA
3040 C130 B938C1            LDA TAFELVAN7-1,Y
3050 C133 AA                TAX
3060 C134 68                PLA
3070 C135 9D04D4            STA SID+4,X         ;OPBERGEN
3080 C138 60                RTS
3200 C139 00070E  TAFELVAN7 BYT 0,7,14
4000 C13C 0A      FREQ      ASL A               ;NOOT * 2 (!)
4005 C13D AA                TAX
4010 C13E AD0AC2            LDA STARTOKTAAF     ;(0-3)
4020 C141 8D09C2            STA OKTAAF
4030 C144 8A                TXA
4040 C145 38                SEC
4050 C146 EE09C2  LFREQ1    INC OKTAAF
4060 C149 E918              SBC #24             ;OKTAAF ZOEKEN
4070 C14B B0F9              BCS LFREQ1          ;(N=*2, ZIE 4000)
4080 C14D 6918              ADC #24             ;GEVONDEN
4090 C14F AA                TAX
4100 C150 BD79C1            LDA TONEN,X         ;BASISNOOT
```

4.5 Keyboard-interface

```
4110 C153 8D0BC2              STA FREQUENTIE              ;OPBERGEN
4120 C156 BD7AC1              LDA TONEN+1,X               ;HI-BYTE
4130 C159 8D0CC2              STA FREQUENTIE+1
4140 C15C AE09C2              LDX OKTAAF                  ;MAAL 2↑OKTAAF
4150 C15F 0E0BC2 LFREQ2       ASL FREQUENTIE              ;*2
4160 C162 2E0CC2              ROL FREQUENTIE+1            ;IDEM
4170 C165 CA                  DEX
4180 C166 D0F7                BNE LFREQ2                  ;NIET KLAAR
4190 C168 B938C1              LDA TAFELVAN7-1,Y           ;KLAAR
4200 C16B AA                  TAX
4210 C16C AD0BC2              LDA FREQUENTIE
4220 C16F 9D00D4              STA SID,X                   ;IN SID
4230 C172 AD0CC2              LDA FREQUENTIE+1
4240 C175 9D01D4              STA SID+1,X
4250 C178 60                  RTS
4260 C179 730189 TONEN        WOR 371,393,417,442,468,496
4270 C185 0D022C              WOR 525,556,590,625,662,701
4280 C191                     ↑BASISNOTEN↑
5000 C191 A02B    INIT        LDY #43                     ;NOTENARRAY OP 0
5010 C193 A900                LDA #0
5020 C195 9956C2 LINIT        STA NOOT,Y
5030 C198 88                  DEY
5040 C199 10FA                BPL LINIT
5050 C19B A9FF                LDA #255                    ;STEMMEN LEEG
5060 C19D A002                LDY #STEMMEN-1
5070 C19F 9902C2 LINIT2       STA STEM,Y
5080 C1A2 88                  DEY
5090 C1A3 10FA                BPL LINIT2
5100 C1A5 A90F                LDA #15                     ;VOLUME OP 15
5110 C1A7 8D18D4              STA 54296
5120 C1AA A901                LDA #1                      ;STARTOKTAAF
5130 C1AC 8D0AC2              STA STARTOKTAAF
5140 C1AF 60                  RTS
6000 C1B0 AD0DC2 FUNCTIES     LDA CONTROL
6010 C1B3 291F                AND #31
6020 C1B5 CD0EC2              CMP CONTROLOUD
6030 C1B8 F045                BEQ KLAARINST               ;NIETS VERANDERD
6040 C1BA 8D0EC2              STA CONTROLOUD              ;WEL
6050 C1BD 491F                EOR #31
6060 C1BF 291F                AND #31
6070 C1C1 0A                  ASL A                       ;MAAL 2
6080 C1C2 0A                  ASL A
6090 C1C3 A8                  TAY
6100 C1C4 B912C2              LDA DEFINITIES,Y            ;LEZEN
6110 C1C7 8D05D4              STA SID+5                   ;OPBERGEN
6120 C1CA 8D0CD4              STA SID+12
6130 C1CD 8D13D4              STA SID+19
6140 C1D0 B913C2              LDA DEFINITIES+1,Y
6150 C1D3 8D06D4              STA SID+6
6160 C1D6 8D0DD4              STA SID+13
6170 C1D9 8D14D4              STA SID+20
6180 C1DC B914C2              LDA DEFINITIES+2,Y
6190 C1DF 8D03D4              STA SID+3
6200 C1E2 8D0AD4              STA SID+10
6210 C1E5 8D11D4              STA SID+17
6220 C1E8 29F0                AND #240
6230 C1EA 8D0FC2              STA WAVES
6240 C1ED 8D10C2              STA WAVES+1
6250 C1F0 8D11C2              STA WAVES+2
6260 C1F3 B915C2              LDA DEFINITIES+3,Y
6270 C1F6 8D02D4              STA SID+2
6280 C1F9 8D09D4              STA SID+9
```

4.5 Keyboard-interface

```
 6290 C1FC 8D10D4                STA SID+16
 6300 C1FF 4C31EA KLAARINST      JMP $EA31          ;NORMALE IRQ
10000 C202 000000 STEM           BYT 0,0,0
10010 C205 00     TEMP           BYT 0
10015 C206 00     TEMP2          BYT 0
10020 C207 00     TOETSEN        BYT 0              ;AANTAL
10030 C208 00     KEUZE          BYT 0
10040 C209 00     OKTAAF         BYT 0
10050 C20A 00     STARTOKTAA*    BYT 0
10060 C20B 0000   FREQUENTIE     BYT 0,0
10070 C20D 00     CONTROL        BYT 0
10080 C20E 00     CONTROLOUD     BYT 0
10090 C20F 202020 WAVES          BYT 32,32,32
11000 C212 000000 DEFINITIES     BYT 0,0,0,0
11010 C216 6A4010 FLUIT          BYT 106,64,16,0
11020 C21A 4C7048 KLARINET       BYT 76,112,72,0
11030 C21E 040020 PIZZICATO      BYT 4,0,32,0
11040 C222 60904E HOBO           BYT 96,144,78,192
11050 C226 78A920 VIOOL          BYT 120,169,32,0
11060 C22A 2AA420 TROMPET        BYT 42,164,32,0
11070 C22E 00F010 ORGEL          BYT 0,240,16,0
11080 C232 2A0044 PIANO          BYT 42,0,68,0
11090 C236 090010 XYLOFOON       BYT 9,0,16,0
11100 C23A 090041 BELLEN         BYT 9,0,65,154
11110 C23E A0F020 SYNTH1         BYT 160,240,32,0
11120 C242 49704E BASSOON        BYT 73,112,78,20
11130 C246 000000                BYT 0,0,0,0
11140 C24A 000000                BYT 0,0,0,0
11150 C24E 000000                BYT 0,0,0,0
11160 C252 090055 HARP           BYT 9,0,85,1
15000 C256 00     NOOT           BYT 0
```

```
C242 BASSOON        C23A BELLEN         033C BUFFER         C20D CONTROL
C20E CONTROLOUD     DD03 DDRB           C212 DEFINITIES     C0E6 DOR120
C216 FLUIT          C13C FREQ           C20B FREQUENTIE     C1B0 FUNCTIES
C115 GATEON         C127 GATEUIT        C252 HARP           C222 HOBO
C191 INIT           C100 INSLUS         C0F9 INSTRUMENT     C208 KEUZE
C03E KLAAR          C1FF KLAARINST      C21A KLARINET       C014 LO
C073 L1             C146 LFREQ1         C15F LFREQ2         C195 LINIT
C19F LINIT2         C10C LINS1          C026 LUS            C256 NOOT
C209 OKTAAF         C22E ORGEL          C232 PIANO          C21E PIZZICATO
DD01 POORTB         C09D R100           C0B8 R120           C0CC R120BIS
C0C4 R130           C0D8 R130BIS        C0E0 R140           C010 R60
C047 R63            C05C R64            C062 R65            C064 R70
C097 R90            D400 SID            C20A STARTOKTAA*    C202 STEM
0003 STEMMEN        C23E SYNTH1         C139 TAFELVAN7      C205 TEMP
C206 TEMP2          C207 TOETSEN        C179 TONEN          C22A TROMPET
C07E V1             C0DD VERDER         C039 VERDER0        C0CA VERSIE2
C226 VIOOL          C20F WAVES          C0E3 WEG            C236 XYLOFOON

READY.
```

4.5 Keyboard-interface

```
1000 DATA32,145,193,120,169,16,141,20,3,169,192,141,21,3,88
1010 DATA96,160,2,169,255,153,60,3,136,16,250,169,63,141,3
1020 DATA221,32,249,192,162,0,160,0,140,1,221,173,1,221,41
1030 DATA64,240,9,152,157,60,3,232,224,3,240,5,200,192,44
1040 DATA208,232,142,7,194,224,0,208,29,160,3,185,1,194,201
1050 DATA255,240,14,170,169,0,157,86,194,169,255,153,1,194,32
1060 DATA39,193,136,208,232,76,227,192,160,3,185,1,194,201,255
1070 DATA240,44,169,0,141,5,194,174,7,194,185,1,194,221,59
1080 DATA3,208,3,238,5,194,202,208,242,173,5,194,208,17,185
1090 DATA1,194,170,169,0,157,86,194,169,255,153,1,194,32,39
1100 DATA193,136,208,202,174,7,194,142,5,194,189,59,3,168,185
1110 DATA86,194,208,55,140,6,194,238,8,194,173,8,194,41,1
1120 DATA240,20,160,3,185,1,194,201,255,208,5,32,230,192,160
1130 DATA1,136,208,241,76,221,192,160,1,185,1,194,201,255,208
1140 DATA5,32,230,192,160,3,200,192,4,208,239,174,5,194,202
1150 DATA208,186,76,176,193,173,6,194,153,1,194,170,152,157,86
1160 DATA194,138,32,60,193,32,21,193,96,160,44,169,0,141,13
1170 DATA194,140,1,221,173,1,221,41,64,24,240,1,56,110,13
1180 DATA194,200,192,52,208,236,96,185,14,194,9,1,153,14,194
1190 DATA72,185,56,193,170,104,157,4,212,96,185,14,194,41,254
1200 DATA153,14,194,72,185,56,193,170,104,157,4,212,96,0,7
1210 DATA14,10,170,173,10,194,141,9,194,138,56,238,9,194,233
1220 DATA24,176,249,105,24,170,189,121,193,141,11,194,189,122,193
1230 DATA141,12,194,174,9,194,14,11,194,46,12,194,202,208,247
1240 DATA185,56,193,170,173,11,194,157,0,212,173,12,194,157,1
1250 DATA212,96,115,1,137,1,161,1,186,1,212,1,240,1,13
1260 DATA2,44,2,78,2,113,2,150,2,189,2,160,43,169,0
1270 DATA153,86,194,136,16,250,169,255,160,2,153,2,194,136,16
1280 DATA250,169,15,141,24,212,169,1,141,10,194,96,173,13,194
1290 DATA41,31,205,14,194,240,69,141,14,194,73,31,41,31,10
1300 DATA10,168,185,18,194,141,5,212,141,12,212,141,19,212,185
1310 DATA19,194,141,6,212,141,13,212,141,20,212,185,20,194,141
1320 DATA3,212,141,10,212,141,17,212,41,240,141,15,194,141,16
1330 DATA194,141,17,194,185,21,194,141,2,212,141,9,212,141,16
1340 DATA212,76,49,234,0,0,0,0,0,0,0,0,0,0,0
1350 DATA0,0,32,32,32,0,0,0,0,106,64,16,0,76,112
1360 DATA72,0,4,0,32,0,96,144,78,192,120,169,32,0,42
1370 DATA164,32,0,0,240,16,0,42,0,68,0,9,0,16,0
1380 DATA9,0,65,154,160,240,32,0,73,112,78,20,0,0,0
1390 DATA0,0,0,0,0,0,0,0,0,9,0,85,1,0
1400 FORI=49152TO49750:READA:S=S+A:POKEI,A:NEXT
1410 IFS<>65608THENPRINT"FOUT IN DATA!":END
1420 SYS49152:PRINT"SPELEN MAAR!"

READY.
```

6/5

De expansion port

Inhoud

6/5.1

Inleiding

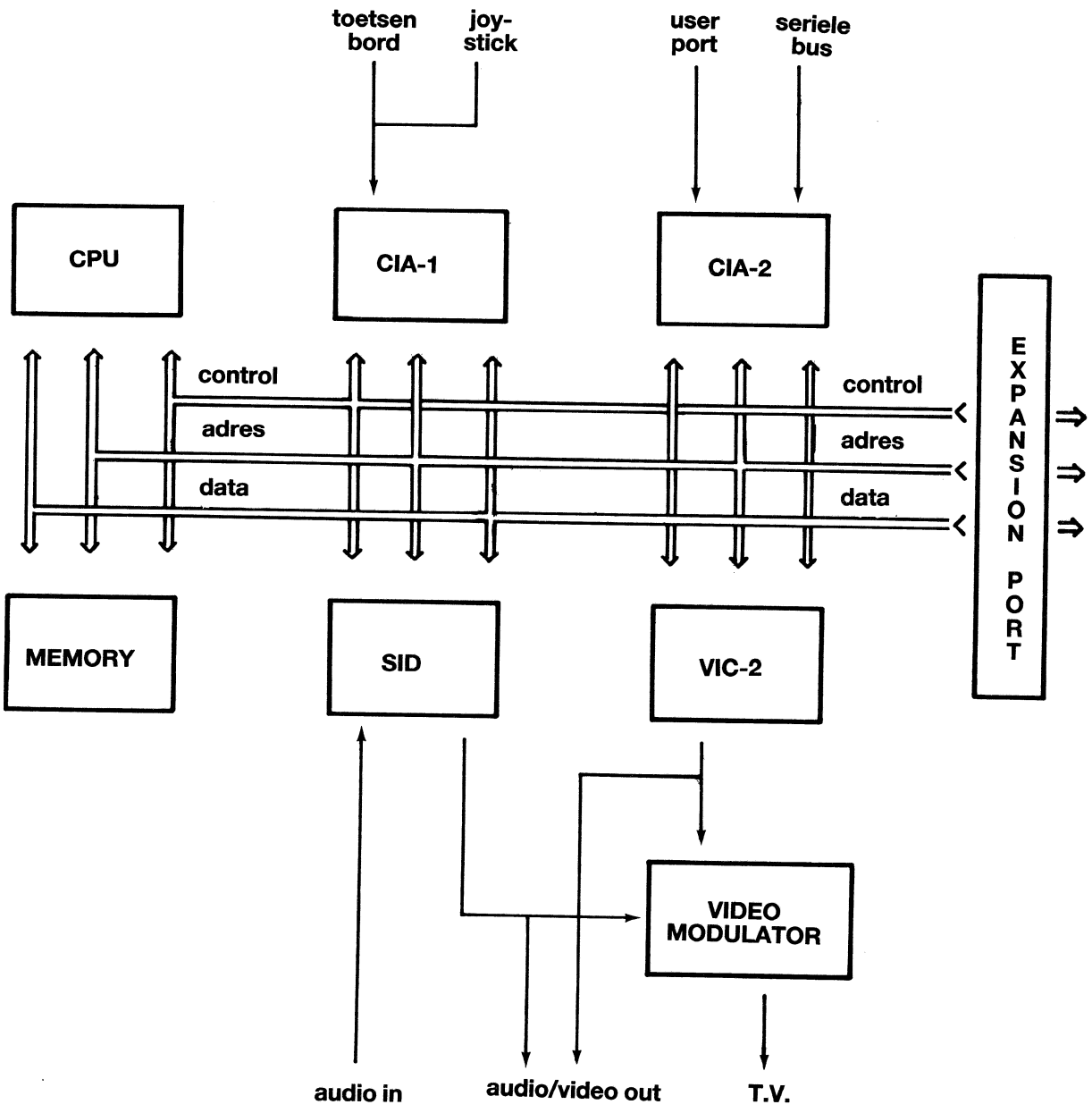
In dit hoofdstuk zullen de mogelijkheden van de expansion port worden behandeld. De expansion port is de uitbreidingspoort, die links achter aan de Commodore 64 zit.
De expansion port kan voor vele doeleinden gebruikt worden. Meestal wordt deze uitbreidingspoort gebruikt om de zogenaamde 'cartridges' (eprom-kaartjes) in te steken. Deze cartridges bevatten bijna altijd een programma. Zo'n programma kan bijvoorbeeld een spelletje zijn, maar kan ook een nieuwe kernal bevatten, waarmee efficienter gebruik kan worden gemaakt van de diskdrive.

De cartridge mag dan wel de meest gebruikte uitbreiding voor deze poort zijn, de enige uitbreiding is dit zeker niet. Er bestaan voor de Commodore 64 bijvoorbeeld uitbreidings kaarten, die van de Commodore een CP/M machine maken. Ook bestaat er een uitbreidings kaart met een 65000 microprocessor. Deze kaart laat de Commodore circa 4 maal zo snel lopen. Een laatste voorbeeld is een 80-kolomskaart. Met behulp van zo'n uitbreidingskaart is het mogelijk om op de Commodore 64 met een 80-koloms tekstverwerker te werken.

De expansion port is de enige poort die direct in contact staat met de data-, adres- en controlbus van de microprocessor. Dit impliceert direct dat deze poort de meeste mogelijkheden biedt om uitbreidingen aan de computer aan te brengen. De hierboven beschreven voorbeelden illustreren dit. Deze uitbreidingen zijn niet of nauwelijks via de andere poorten te realiseren.

Voordat met de verdere beschrijving van de expansion poort wordt verder gegaan, is het op zijn plaats om in het kort iets over de structuur van deze computer te zeggen. De Commodore 64 is opgebouwd uit een aantal primaire bouwstenen. Deze bouwstenen worden met elkaar verbonden door middel van drie bussen: de control bus, de data bus en de adres bus. Met deze bussen kunnen deze bouwstenen met elkaar communiceren (Zie figuur 6/5.1-1). De expansion poort is rechtstreeks met deze drie bussen verbonden, waaruit nu de relevantie van de poort blijkt. Via deze poort kunnen de krachtigste uitbreidingen aan de Commodore 64 worden verbonden.

5.1 Inleiding



Figuur 6/5.1-1: Blokschema van de Commodore 64.

5.1 Inleiding

Toelichting
In het voorgaande figuur zijn 8 blokken en drie bussen te onderscheiden:
1) CPU, dit is de microprocessor. De microprocessor is de rekenaar en bestuurder van de machine. De CPU leest instructies uit het geheugen en voert deze uit. Al naar gelang deze instructies zal de CPU data over de data-bus sturen of ontvangen. Data komt altijd ergens vandaan of gaat altijd ergens naar toe. Dit wordt gedaan door op de adres-bus een adres aan te geven, zodat een van de 7 andere blokken de data zal ontvangen of versturen.
2) MEMORY, dit is het geheugen. In het geheugen staat het programma dat uitgevoerd wordt. Echter ook de kernal en de BASIC interpreter zijn in het geheugen aanwezig. Naast deze programma's staat ook al de data, die bij deze programma's hoort, in het geheugen.
3) CIA-1, dit is een input/output chip, die het toetsenbord en de joystick poorten uitleest.
4) CIA-2, deze input/ouput chip zorgt voor de communicatie via de seriële bus en is voor ongeveer de helft vrij te programmeren (zie hoofdstuk 6/4).
5) SID, dit is de audio processor, deze chip genereert al de geluiden die de computer voort brengt.
6) VIC-2, dit is de video processor. Deze chip creëert het beeld. De beeldinformatie haalt deze chip autonoom uit het geheugen. De CPU is dus niet de enige die in het geheugen kan lezen.
7) VIDEO-MODULATOR, dit is een stukje elektronica, dat van de audio- en videosignalen een T.V. signaal maakt.
8) EXPANSION PORT, dit is de uitbreidingspoort, waarlangs al het inwendige van de computer te bereiken is.
DB) Over de data-bus wordt alle data getransporteerd. De data kan in twee richtingen over de bus worden verplaatst. Vanuit de processor gezien, kan de data van de processor afgaan (WRITE).
AB) Via de adres-bus worden de adressen die bij de data horen overgezonden. De CPU kan adressen creëeren evenals de VIC-2 (om beeld informatie uit het geheugen te halen), maar ook kunnen via de EXPANSION PORT adressen op deze bus 'gezet' worden.
CB) De control-bus zorgt er onder andere voor dat de communicatie tussen de verschillende onderdelen goed verloopt, hier zal later nog op worden teruggekomen.

6/5.2

Geheugenorganisatie

De geheugenorganisatie van de Commodore 64 is direct afhankelijk van de expansion port. Door middel van twee input lijnen van de expansion port in combinatie met twee output lijnen van de microprocessor, kan de Commodore 64 over 8 verschillende geheugen indelingen beschikken. Tevens zijn er twee I/O blokken, die binnen de computer niet gebruikt worden en die via de expansion port gebruikt kunnen worden.

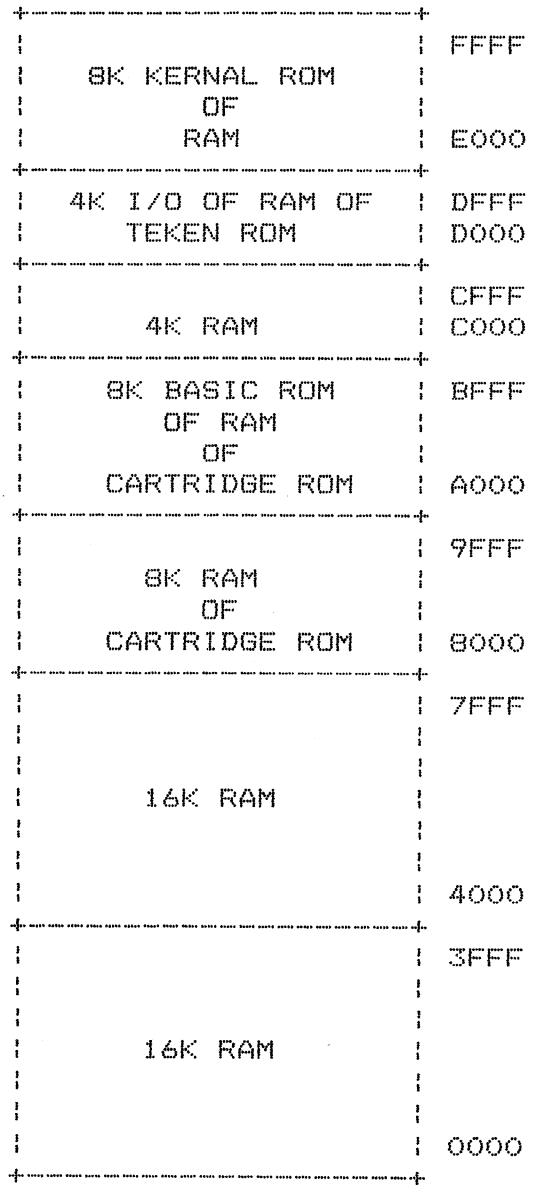
Geheugen indeling
Ten eerste zijn er de twee signalen die van de processor af komen, deze worden HIRAM en LORAM genoemd. Deze signalen zijn twee van de zes input/output lijnen die de 6510 microprocessor bezit. Door naar geheugenadres 1 te schrijven kunnen de waarden van LORAM en HIRAM veranderd worden. De twee laagst waardige bits zijn hiervoor gereserveerd. LORAM zit op bit 0 van adres 1 en HIRAM zit op bit 1 van adres 1. De overige vier signalen worden gebruikt voor de selectie van I/O versus ROM, en voor de communicatie met de cassatte recorder. Ten tweede zijn er de twee signalen die van de expansion port afkomen, deze worden GAME en EXROM genoemd. Deze signalen zijn niet softwarematig te beïnvloeden, maar enkel hardwarematig met behulp van een uitbreiding aan de expansion port (via een hardwarematige truuk zijn deze signalen wel softwarematig te beïnvloeden, hier zal verder niet op worden ingegaan). Eerst zal de algemene geheugen indeling van de computer worden gegeven. Daarna de 8 verschillende geheugen indelingen zullen hieronder besproken worden.

I/O indeling
De Commodore 64 maakt gebruik van een I/O mechanisme dat memory mapped I/O heet. Dit wil zoveel zeggen dat de I/O devices via een geheugenadres worden aangesproken. De devices zijn hier de chips, die in het schema van 6/5.1 naar voren komen. In de I/O struktuur komen behalve de twee CIA's de VIC-II en de SID chip nog twee open I/O 'slots' voor. Deze twee I/O 'slots' kunnen door middel van de expansion port gebruikt worden. Zie 6/5.2.3 voor een overzicht van de I/O geheugen indeling.

6/5.2.1
Algemene geheugenindeling

De algemene geheugenindeling laat in een oog opslag zien waar de verschillende RAM, ROM en I/O blokken zitten.

5.2 Geheugenorganisatie

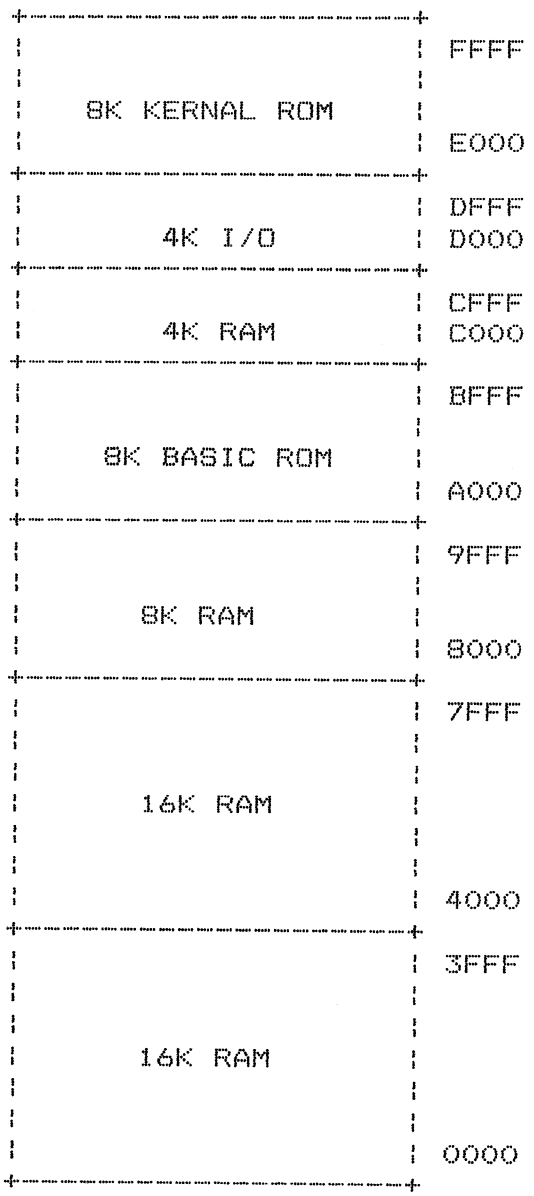


Figuur 6/5.2.1-1: Algemene geheugenindeling.

6/5.2.2 Geheugenindelingen

De standaard geheugeninstelling heeft de BASIC ROM aanstaan en heeft 38 Kbytes vrij gebruikers geheugen. De waarden van de vier geheugen indelings signalen zijn:

LORAM = 1
HIRAM = 1
GAME = 1
EXROM = 1

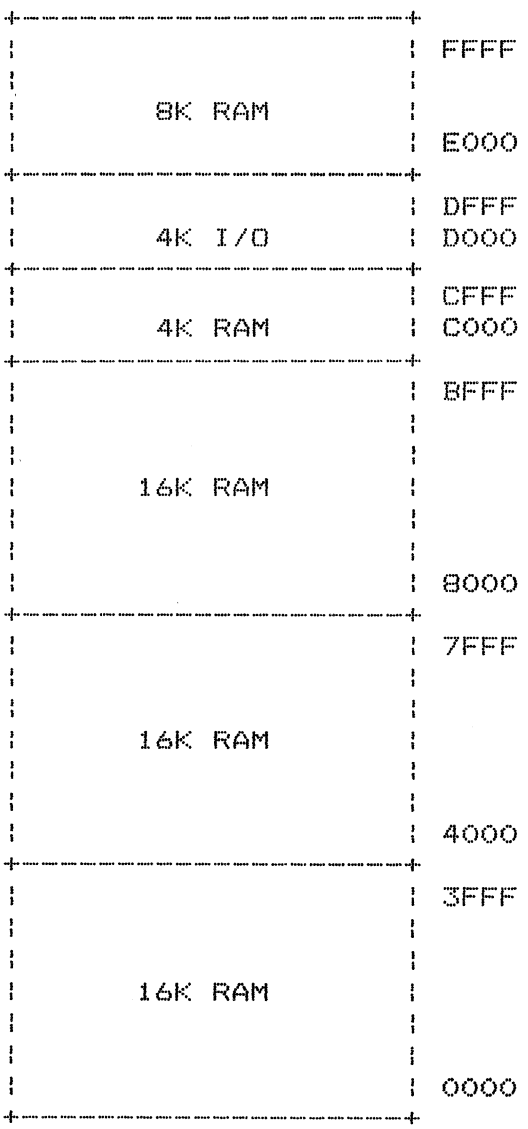


Figuur 6/5.2.2-1: Standaard geheugenindeling.

De volgende geheugenindeling biedt 60 kbyte RAM, met 4K aan I/O, de gebruiker zal zelf de I/O routines moeten schrijven; de kernal is namelijk uitgeschakeld.

5.2 Geheugenorganisatie

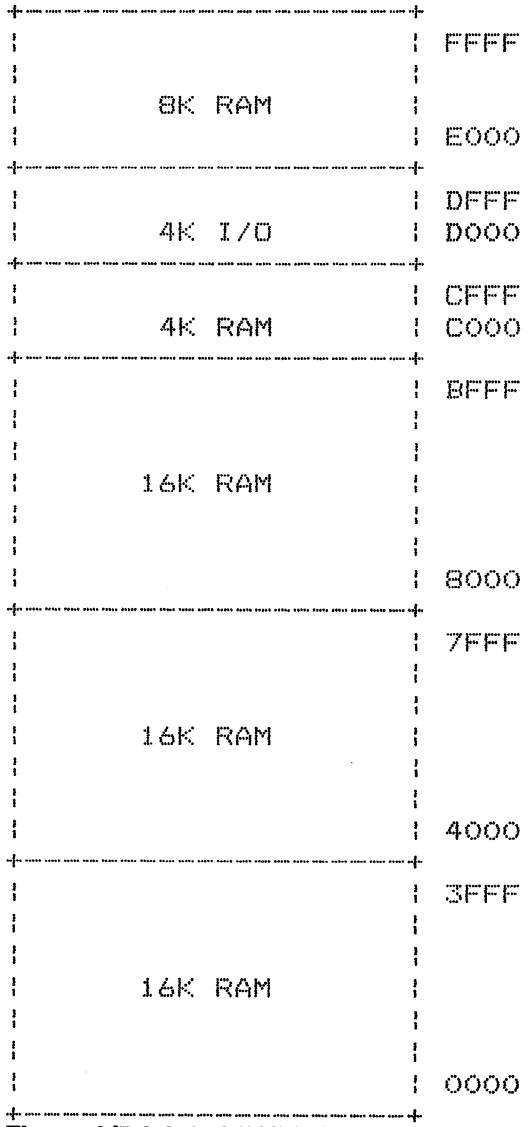
LORAM = 1 of 1 of 1
HIRAM = 0 0 0
GAME = 1 1 0
EXROM = 0 1 0



Figuur 6/5.2.2-2: 52K RAM, 4K I/O met kernal.

De volgende geheugenindeling wordt gebruikt voor de zo genoemde 'softload' languages, waarbij 52K RAM vrij is, het I/O blok beschikbaar is en de I/O-driver routines in de kernal normaal werken.

LORAM = 0 of 0
HIRAM = 1 1
GAME = 1 1
EXROM = 0 1

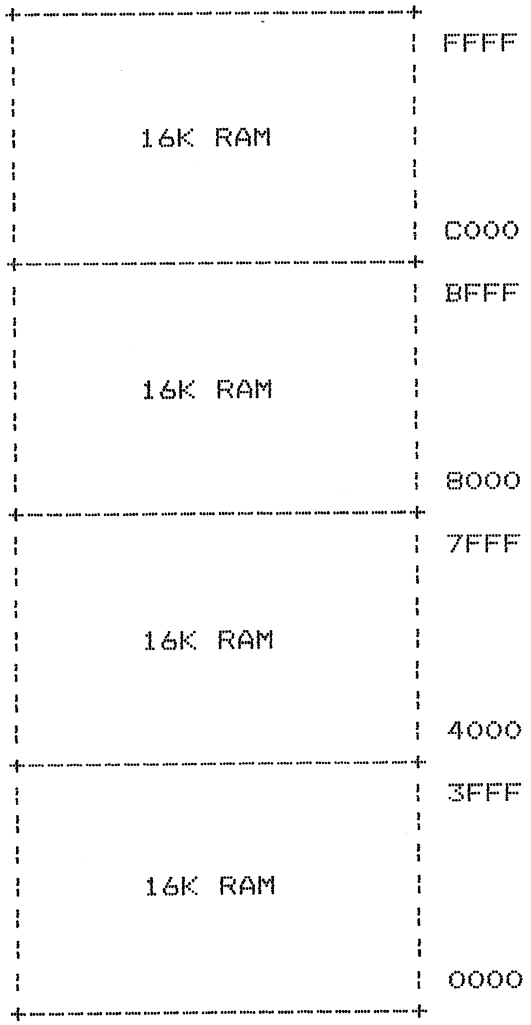


Figuur 6/5.2.2-3: 64K RAM.

Met de volgende indeling heeft de gebruiker de beschikking over het gehele 64Kbyte RAM gebied van de Commodore 64. Voor I/O operaties moet het I/O blok aangezet worden.
LORAM = 0 of 0 of 0
HIRAM = 0 0 0

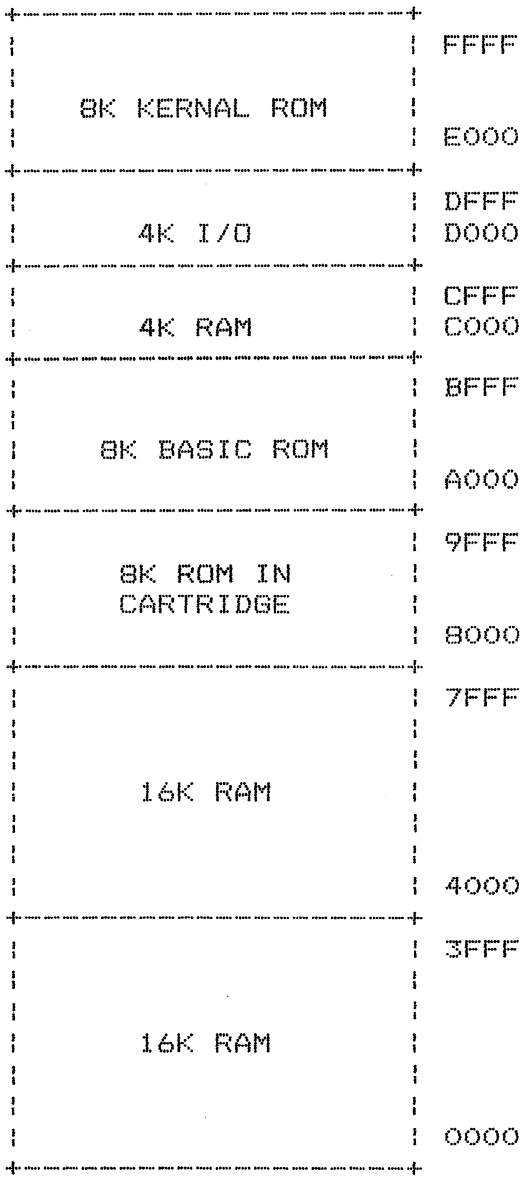
5.2 Geheugenorganisatie

GAME = 1 1 0
EXROM = 0 1 0



Figuur 6/5.2.2-4: 64K RAM.

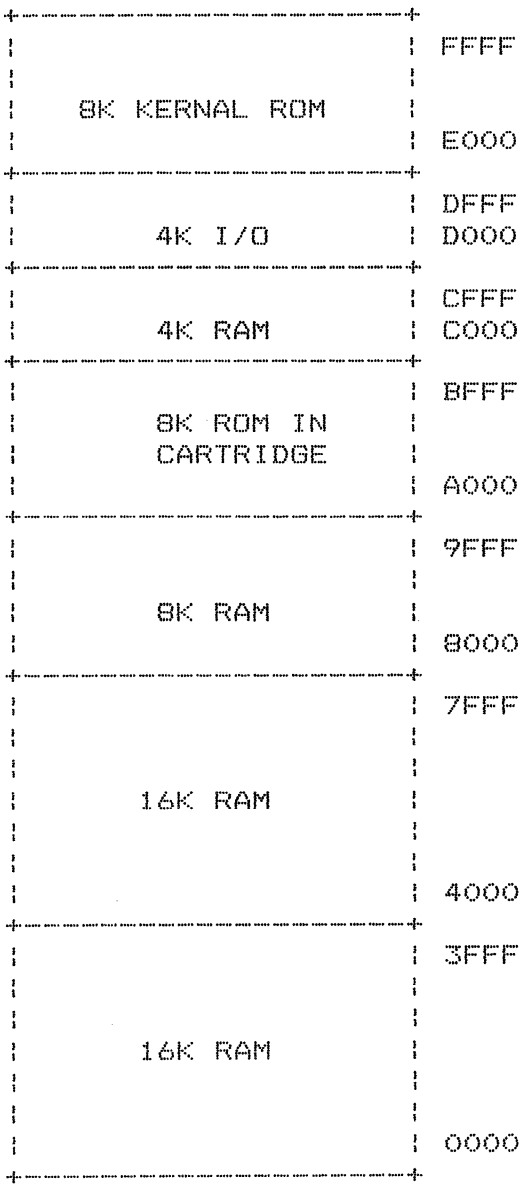
De standaard configuratie voor een BASIC systeem met een cartridge ROM uitbreiding wordt als volgt verkregen:
LORAM = 1
HIRAM = 1
GAME = 1
EXROM = 0



Figuur 6/5.2.2-5: Standaard geheugenindeling met BASIC uitbreiding.

In de onderstaande indeling is de BASIC ROM vervangen door een 8K cartridge ROM
LORAM = 0
HIRAM = 1
GAME = 0
EXROM = 0

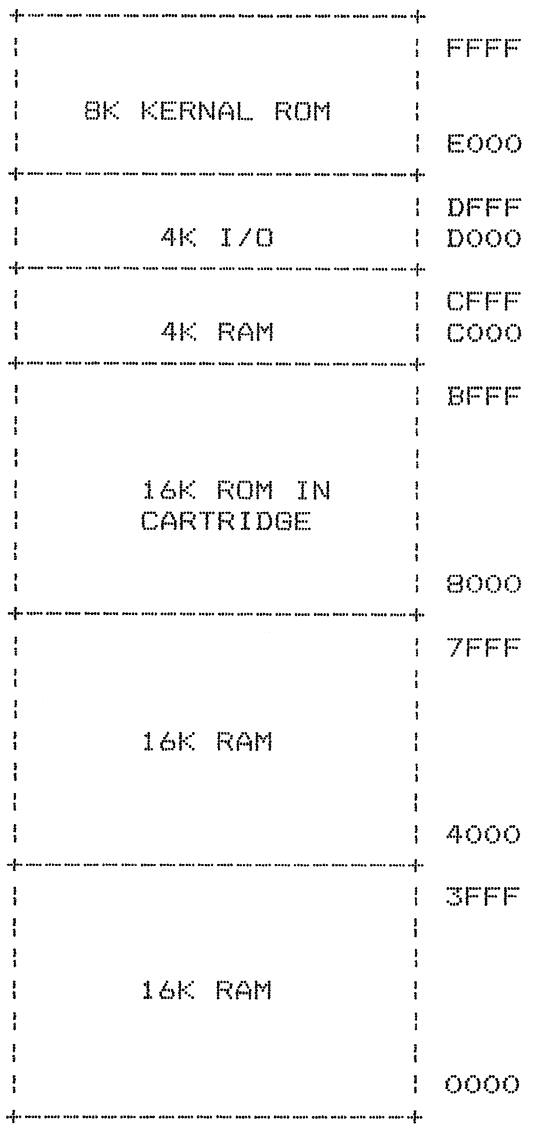
5.2 Geheugenorganisatie



Figuur 6/5.2.2-6: Cartridge ROM in plaats van BASIC ROM.

Hieronder staat de geheugen indeling met een 16Kbyte cartridge ROM uitbreiding, dit is voor applicaties die geen basic nodig hebben, zoals tekstverwerkers, andere talen (Simon's BASIC), etc.

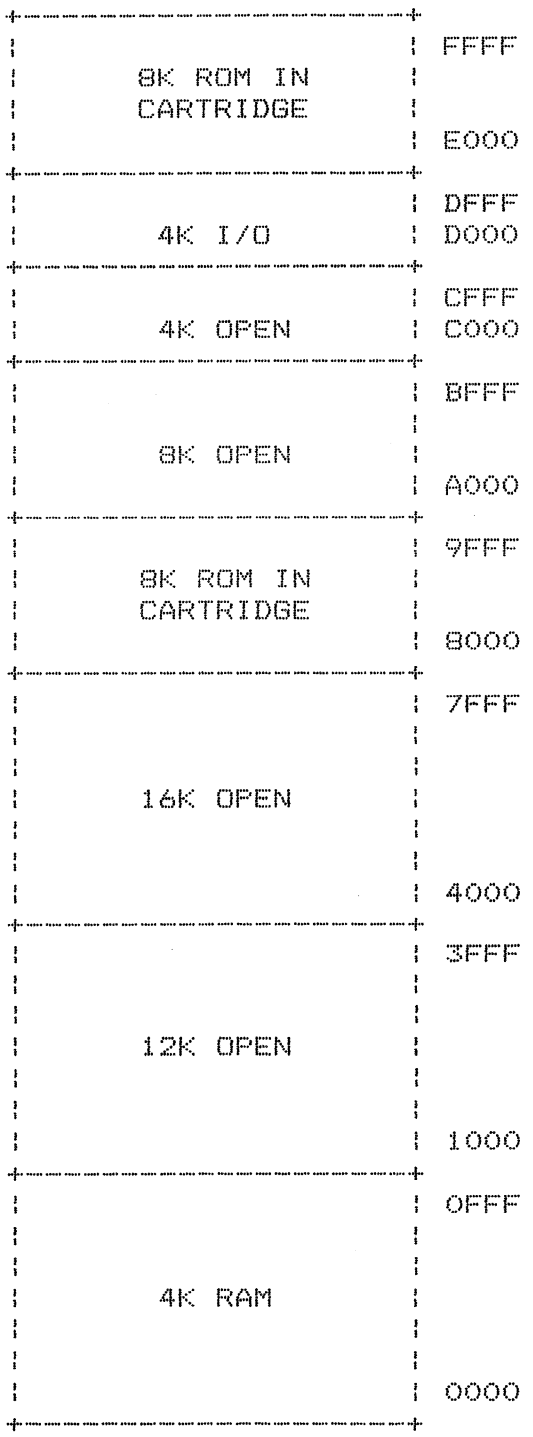
LORAM = 1
HIRAM = 1
GAME = 0
EXROM = 0



Figuur 6/5.2.2-7: 32K RAM, 16K cartridge ROM, 4K I/O met kernal.

In de laatste geheugenindeling is de kernal vervangen door een cartridge ROM en is ook de BASIC ROM door een cartridge ROM vervangen.

5.2 Geheugenorganisatie



Figuur 6/5.2.2-8: Cartridge ROM in plaats van basic ROM en cartrdige ROM voor kerval.

LORAM	= 0	of 0	of 1	of 1
HIRAM	= 0	1	0	1
GAME	= 0	0	0	0
EXROM	= 1	1	1	1

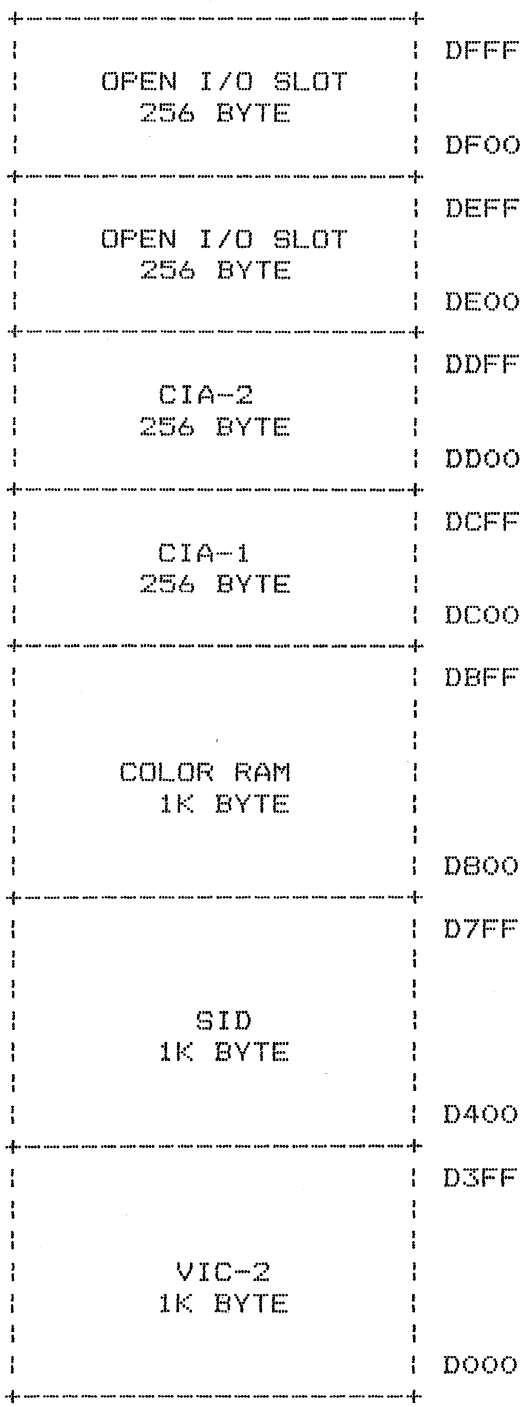
6/5.2.3 I/O geheugenindeling

De I/O is een belangrijk onderdeel van elke computer. Zonder I/O kan een computer namelijk niet communiceren met de 'buitenwereld'. De I/O is op verschillende machines op verschillende manieren gerealiseerd. In die realisaties zijn twee basis principes te onderscheiden: ten eerste de memory mapped I/O en ten tweede separate I/O space.

Het memory mapped I/O principe gaat er van uit dat de I/O devices als chips in het geheugen gemapped zijn. Dat wil zeggen dat door te lezen en te schrijven op een geheugen adres waar I/O zit, rechtstreeks data van de microprocessor naar het I/O device of vice verse gaat. Voor deze lees en schrijf acties kunnen meestal de normale geheugen toegangs instructies gebruikt worden. Wel wordt er een bepaald geheugen gedeelte 'opgeofferd' voor de I/O.

Het separate I/O space principe gaat er van uit dat de I/O devices niet in het geheugen gemapped zijn. De I/O staat in een aparte I/O ruimte. Hier valt een analogie te trekken tussen de aparte I/O ruimte en de aparte register ruimte waar iedereen wel mee bekend zal zijn. Registers zijn over het algemeen niet te lezen of te schrijven door in het geheugen te lezen of schrijven. Bij het separate I/O space principe is het ook niet mogelijk om data via een geheugen adres naar of van de I/O devices te laten gaan. Voor deze vorm van I/O zijn aparte instructies nodig om in de I/O ruimte te kunnen lezen en te schrij-

5.2 Geheugenorganisatie



Figuur 6/5.2.3-1: I/O geheugen indeling.

ven. Bij deze vorm van I/O wordt geen gedeelte van het geheugen 'opgeofferd'.

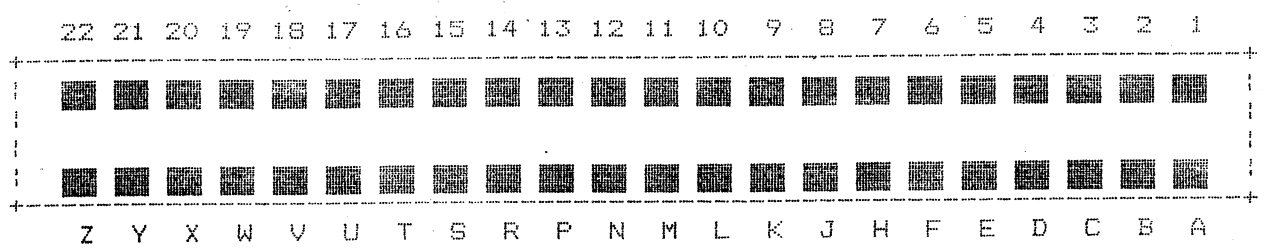
De Commodore 64, maakt gebruik van het eerste principe, de memory mapped I/O. Alle I/O is gegroepeerd en zit in een geheugen blok van 4Kbyte. De indeling van de ruimte is in figuur 6/5.2.3-1 zichtbaar gemaakt, hierin zijn alle I/O chips in opgenomen. Tevens is er ruimte opengelaten, die voor uitbreidingen gebruikt kunnen worden. Deze uitbreiding kunnen in de expansion port gestoken worden en zijn dan via de onderstaande geheugen adressen te bereiken.

6/5.3

Opbouw van de expansion port

De expansion port is opgebouwd uit een 44-pins vrouwelijke 'edge' connector. In deze connector kunnen dubbelzijdige printjes gestoken worden, die voor een bepaalde uitbreiding zorgen. In deze paragraaf zal beschreven worden, hoe een dubbelzijdige print met de connector verbonden dient te worden. Daartoe wordt de pin beschrijving van de connector gegeven.

De connector is als volgt gedefinieerd:



Figuur 6/5.3-1: Expansion port connector definitie.

Let er bij de bovenstaande figuur op, dat de letters wel in alfabetische volgorde staan, maar dat er 'gaten' in het alfabet zitten. De signalen hebben de volgende betekenis:

1 - GND : massa
2 - +5 VDC : maximale belasting
3 - +5 VDC : van 450 mA.
4 - $\overline{\text{IRQ}}$: Interrupt request aan de microprocessor (laag actief)
5 - R/$\overline{\text{W}}$: Read/Write
6 - DOTCLK: 8.18 Mhz video dot clock
7 - $\overline{\text{I/O1}}$: I/O blok 1, tussen DE00-DEFF, ongebufferde I/O (zie 6/5.2.3)
8 - $\overline{\text{GAME}}$: laag actieve TTL input (zie 6/5.2)
q - $\overline{\text{EXROM}}$: laag actieve TTL input (zie 6/5.2)
10 - $\overline{\text{I/O2}}$: I/O blok 2, tussen DF00-DFFF, gebufferde TTL output (zie 6/5.2.3)
11 - $\overline{\text{ROML}}$: 8K gedecodeerde RAM/ROM, blok 8000-9FFF, laag actief, gebufferde TTL output.
12 - BA : Bus available signaal van de VIC-II, ongebufferd.
13 - $\overline{\text{DMA}}$: Direct memory access request, laag actief, TTL input

5.3 Opbouw expansion port

14 - D7 : Data bus bit 7, ongebufferd, max 1 TTL load
15 - D6 : Data bus bit 6, ongebufferd, max 1 TTL load
16 - D5 : Data bus bit 5, ongebufferd, max 1 TTL load
17 - D4 : Data bus bit 4, ongebufferd, max 1 TTL load
18 - D3 : Data bus bit 3, ongebufferd, max 1 TTL load
19 - D2 : Data bus bit 2, ongebufferd, max 1 TTL load
20 - D1 : Data bus bit 1, ongebufferd, max 1 TTL load
21 - D0 : Data bus bit 0, ongebufferd, max 1 TTL load
22 - GND : massa
A - GND : massa
B - $\overline{\text{ROMH}}$: 8K gedecodeerde RAM/ROM, blok E000-FFFF, laag actief, gebufferde TTL output
C - $\overline{\text{RESET}}$: microprocessor reset, laag actief, gebufferde TTL output, ongebufferd in
D - $\overline{\text{NMI}}$: microprocessor Non Maskable Interrupt, laag actief, gebufferd TTL out, ongebufferd in
E - Ø2 : Fase 2 van de systeem klok
F - A15 : Adres bus bit 15, ongebufferd, 1 TTL load maximaal
H - A14 : Adres bus bit 14, ongebufferd, 1 TTL load maximaal
J - A13 : Adres bus bit 13, ongebufferd, 1 TTL load maximaal
K - A12 : Adres bus bit 12, ongebufferd, 1 TTL load maximaal
L - A10 : Adres bus bit 11, ongebufferd, 1 TTL load maximaal
M - A9 : Adres bus bit 10, ongebufferd, 1 TTL load maximaal
N - A8 : Adres bus bit 9, ongebufferd, 1 TTL load maximaal
P - A7 : Adres bus bit 8, ongebufferd, 1 TTL load maximaal
R - A6 : Adres bus bit 7, ongebufferd, 1 TTL load maximaal
S - A5 : Adres bus bit 6, ongebufferd, 1 TTL load maximaal
T - A4 : Adres bus bit 5, ongebufferd, 1 TTL load maximaal
U - A3 : Adres bus bit 4, ongebufferd, 1 TTL load maximaal
V - A2 : Adres bus bit 3, ongebufferd, 1 TTL load maximaal
W - A1 : Adres bus bit 2, ongebufferd, 1 TTL load maximaal
X - A0 : Adres bus bit 1, ongebufferd, 1 TTL load maximaal
Y - A15 : Adres bus bit 0, ongebufferd, 1 TTL load maximaal
Z - GND : massa

7

BASIC

Inhoud

[1)] Dit hoofdstuk heeft een eigen inhoudsopgave.

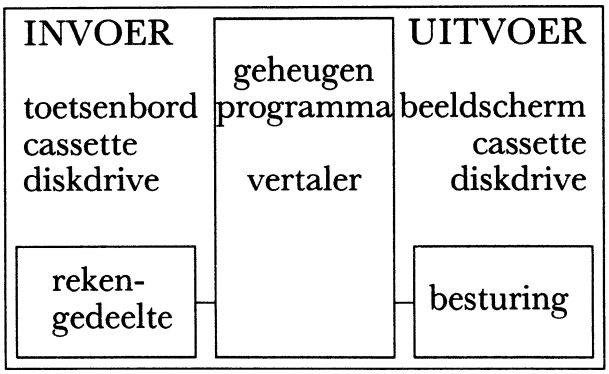
U begint nu aan het gedeelte getiteld „BASIC" van het boek „van BASIC tot Machinetaal". Uiteraard leert u in dit boek de Commodore versie van BASIC. Andere machines werken met een ander 'dialect'. Door de gehele tekst van dit gedeelte loopt de volgende 'rode draad':

1) Welk doel wilt u bereiken?
2) Wat is daarvoor noodzakelijk?
3) Welke middelen heeft u tot uw beschikking?
4) Hoe bereikt u het doel zo snel mogelijk?

In eerste instantie is het te bereiken doel gemakkelijk te omschrijven: „U wilt dat de computer voor u doet wat u hem opdraagt". Een zeker begrip van opbouw en werking van de computer is daarvoor noodzakelijk; dit vormt tevens het begin van het verhaal.

Een computer is een gegevens verwerkend apparaat, dat werkt op het lichtnet. Geen enkel apparaat werkt geheel uit zichzelf, een computer dus ook niet. Wil hij iets doen dan zal de gebruiker hem van opdrachten moeten voorzien.
Het volgend schema geeft globaal de opbouw van een computer weer; althans vanuit een BASIC standpunt gezien (zie hoofdstuk 4 'CBM 64, DE OPBOUW').

Het hart van de computer (onder uw toetsenbord ingebouwd) bestaat uit het rekengedeelte (ALU = De Engelse afkorting voor Arithmetic and Logic Unit). Hoe dit gedeelte precies werkt daar gaan we niet op in, maar erg veel kan het niet. De handelingen die het verricht doet het echter snel en bovendien foutloos.



Figuur 7-1: Schematische opbouw van de CBM 64

Een tweede zeer belangrijk component is de besturings-eenheid. Dit gedeelte is vergelijkbaar met de verkeerstoren van een grote luchthaven, alleen worden er geen vliegtuigen op de juiste wijze gedirigeerd, maar elektrische stroompjes. De besturing zorgt ervoor dat alle stroompjes van de juisten plaatsen worden gehaald en naar de correcte plekken worden gestuurd zonder dat er dingen verkeerd gaan.
Eigenlijk vormen ALU en besturing een geheel en spreken we over de CPU (Central Processing Unit – cerntrale verwerkings eenheid).

Het woord stroompje zegt ook al iets over de manier waarop u met de computer kunt communiceren. Stroom is er namelijk wel of niet; meer mogelijkheden bestaan niet. De mensen hebben hier een betekenis aan gegeven, er wordt gesproken over 1 en 0. Gaat u dus instructies opschrijven voor de computer dan doet u dit in de vorm van reeksen enen en nullen, want dit is de enige taal die de computer verstaat. Voor ons mensen is een lijst instructies van bijvoorbeeld acht pagina's geheel volge-

schreven met enen en nullen volledig onbegrijpelijk. Door de fabrikant is daarom in de computer ingebouwd de zogenaamde vertaler (interpreter). Deze verwerkt een op het Engels lijkende opdrachtenserie en kan instructies in BASIC (Beginners All Symbolic Instruction Code, 1964 in Amerika voor het eerst uitgebracht) omzetten in voor de machine begrijpelijke codes.

Instructies voor de CPU en resultaten van bewerkingen worden opgeslagen in het geheugen. Dit is opgebouwd uit twee delen, namelijk het ROM (Read Only Memory - Alleen Lees Geheugen) gedeelte, dat onuitwisbaar is en voor u slechts toegankelijk om uit te gebruiken, maar niet om in te veranderen en het RAM geheugen (Random Access Memory - Willekeurig Toegankelijk Geheugen). Dit is het geheugengedeelte waarin u kunt wijzigen en wat onder andere wordt gebruikt om programma's en gegevens op te slaan zolang de machine op het lichtnet is aangesloten. Het compacte geheel dat de computer vormt moet u voorzien van gegevens (de invoer) en dit kunt u doen via het toetsenbord, of u kunt de gegevens hebben opgeslagen op een cassetteband of een magneetschijf (floppy disc).
U kunt eigenlijk elk medium gebruiken dat de codes 1 en 0 kan voorstellen.

Als de computer zijn werk heeft gedaan is het bijzonder prettig als het resultaat niet voor u verborgen blijft, er is daarom gezorgd voor uitvoer (output). Dit kan gebeuren naar een beeldscherm, (televisietoestel of monitor), naar cassetteband of floppy-disc, naar printer of modem enz. In dit verhaal gaan we er in eerste instantie van uit dat u een eenvoudig systeem heeft, bestaande uit toetsenbord (incl. computer), televisiescherm en cassetterecorder. Later gaan we op eventuele extra apparatuur in.

VOORWOORD BIJ DE PROGRAMMALISTINGS

In dit naslagwerk worden alle programmalistings precies zo gepubliceerd als ze er op het scherm uitzien. Dit betekent dat in de listings bepaalde besturingstekens voor kunnen komen. Deze tekens hebben de volgende betekenis.

De beide sets hebben betrekking op de twee karakter-sets die zich in de Commodore-64 bevinden. Is een listing in hoofdletters geschreven, dan geldt set 1. Is de listing in kleine letters geschreven dan is set 2 van toepassing.

Functie	SET1	SET2
Cursor links		
Cursor rechts		
Cursor laag		
Cursor hoog		
Home		
Clear		
Functietoets 1		
Functietoets 2		
Functietoets 3		
Functietoets 4		
Functietoets 5		
Functietoets 6		
Functietoets 7		
Functietoets 8		
Zwart (CTRL-1)		
Wit (CTRL-2)		
Rood (CTRL-3)		
Cyaan (CTRL-4)		
Paars (CTRL-5)		
Groen (CTRL-6)		
Blauw (CTRL-7)		
Geel (CTRL-8)		
Oranje (Commodore-1)		
Bruin (Commodore-2)		
Lichtrood (Commodore-3)		
Grijs 1 (Commodore-4)		
Grijs 2 (Commodore-5)		
Lichtgroen (Commodore-6)		
Lichtblauw (Commodore-7)		
Grijs 3 (Commodore-8)		
Reverse aan (CTRL-9)		
Reverse uit (CTRL-0)		

7/1

Belangrijke afspraken

Inhoud

1) toegestaan in BASIC zijn alle letters, alle cijfers en alle leestekens, hoewel diverse leestekens een speciale betekenis hebben, waarover later meer. Het cijfer 0 wordt in BASIC geschreven als Ø, de letter x stelt niet tevens het maal-teken voor, maar alleen de letter, het maalteken is het sterretje '*'.

2) BASIC gebruikt geen decimale komma, maar een decimale punt.

3) De 0 voor een decimale punt mogen we weglaten.

4) Rekenkundige bewerkingen worden aangegeven door de volgende tekens:
machtsverheffen ↑ - 2↑4
vermenigvuldigen * - 3*6
delen / - 6/2
aftrekken - - 8-3
optellen + - 4+7
De volgorde van de rekenkundige bewerkingen is zoals boven, tenzij er ronde haken zijn toegepast. Dan wordt de bewerking binnen de haken als eerste uitgevoerd. Voor worteltrekken is geen apart symbool, dit is een vorm van machtsverheffen, bv. 3↑1/3 is de derdemachtswortel van 3.
Een handig ezelsbruggetje om de volgorde van de rekenkundige bewerkingen te onthouden is het zinnetje „Meneer Van Dale At Oliebollen"

5) Bij de syntax (officiële schrijfwijze) voorbeelden gelden de volgende regels: BASIC keywords staan in hoofdletters. Tussen aanhalingstekens vermelde gegevens dient u zelf in te tikken, alsmede de aanhalingstekens zelf. Gegevens tussen vierkante teksthaken [] zijn niet verplicht, van gegevens tussen gewone haakjes mag u er zoveel u nodig heeft op een regel plaatsen. Tussen '<' en '>' geplaatste gegevens dient u zelf in te voeren. Het teken '/' geeft aan dat u moet kiezen tussen twee mogelijkheden. Tekst tussen vierkante teksthaken die is onderstreept moet letterlijk gebruikt worden.
Om dit duidelijk te maken:
GA NAAR <bestemming>[s'morgens]
PER FIETS/PER TREIN<vertrektijd>
zou kunnen worden:
GA NAAR Amsterdam PER TREIN 10.00

7/1.1

Het toetsenbord

De meeste toetsen vindt u ook op een schrijfmachine, de werking van deze toetsen is hetzelfde - normaal krijgt u kleine letters, met SHIFT hoofdletters. Als er twee tekens op een toets staan: normaal het onderste, met SHIFT het bovenste.
Welke toetsen komen niet op een normaal schrijfmachine toetsenbord voor en wat doen ze?

De bijzondere toetsen

COMMODORE toets.
Samen met SHIFT (hoofdlettertoets) ingedrukt zorgt deze voor overgang van hoofdlettermode (uitgangssituatie bij aanzetten van computer) naar kleine lettermode, hierbij heeft u hoofd- en kleine letters tot Uw beschikking zoals bij een schrijfmachine.

Figuur 7/1.1-1
Het toetsenbord van de CBM 64

Samen met cijfertoetsen 1 t/m 8 zorgt deze voor 8 van de zestien mogelijke kleuren van de op het scherm afgebeelde tekens. Bij de 'GRAFISCHE TEKENS' leert u de derde functie van de Commodoretoets kennen.

CRSR toetsen.
Met behulp van deze cursorbesturingstoetsen stuurt u de CURSOR (het knipperende blokje dat aangeeft waar de volgende letter wordt afgebeeld) over het scherm. Door de toetsen enkel te bedienen brengt u de cursor naar beneden of naar rechts, samen met SHIFT gaat de cursor naar boven of naar links.

RETURN.
Beschouw deze toets als de 'invoer'-toets. Als op RETURN wordt gedrukt betekent dit zoveel als: „Ik ben klaar, computer nu ben jij aan de beurt", of „OVER" bij radioverkeer.

RESTORE.
Deze wordt tezamen met de STOP-toets gebruikt om programma's te onderbreken en de computer in de beginsituatie terug te brengen, zonder dat een eventueel aanwezig programma verloren gaat.

INST/DEL.
Indrukken van deze toets zonder meer verwijdert het teken links van de CUR-

1.1 Het toetsenbord

SOR waarna de cursor 1 positie naar links gaat. Indrukken tesamen met SHIFT maakt een positie ruimte op de plaats van de CURSOR; de overige tekst schuift 1 positie naar rechts.

CLR/HOME.
Indrukken zonder meer brengt de CURSOR naar de linker bovenhoek van het scherm, de home- of uitgangspositie. Indrukken tesamen met SHIFT maakt het scherm schoon. Alleen het scherm wordt gewist, in het geheugen aanwezige instructies, waarover later, worden niet verwijderd.

De PIJL naar LINKS.
Wordt in sommige programma's gebruikt voor het uitvoeren van bepaalde instructies.

CTRL.
Deze toets tesamen met de cijfertoetsen 1 t/m 8 geeft 8 van de zestien tekstkleuren, namelijk diegenen die vermeld staan voorop de cijfertoetsen.
Tesamen met 9 zorgt hij ervoor dat de tekens inverse, oftewel donker op lichte ondergrond worden afgedrukt. Tezamen met 0 wordt dit effect weer teniet gedaan.
Bij het indrukken tijdens de uitvoering van het LIST-commando (zie 7.1.4) wordt uitvoering hiervan vertraagd.

RUN/STOP.
Indrukken van deze toets onderbreekt het programma (werkt niet altijd). Tezamen met SHIFT zorgt deze toets ervoor dat het eerstvolgende programma van de cassetterecorder geladen wordt. Het ingeladen programma zal gelijk starten (vandaar RUN)

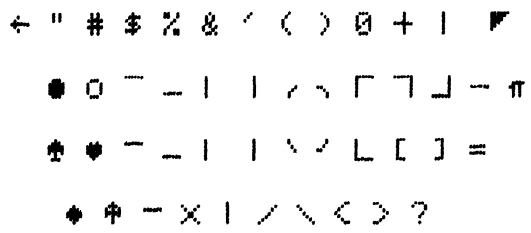
SHIFT/LOCK.
Vergrendeltoets voor hoofdletters. Erg gevaarlijke toets. Als u rare tekens krijgt op het scherm kan het zijn dat deze toets is ingedrukt. Uitschakelen door nogmaals indrukken.

F1 t/m F8.
Zijn 8 zogenaamde functietoetsen (even nummers krijgt u door de toetsen tesamen met SHIFT in te drukken). Aan deze toetsen kunt u een bepaalde betekenis geven, hierover later meer.

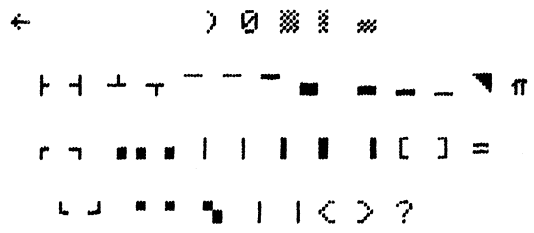
GRAFISCHE TEKENS.
Op de voorzijde van de meeste toetsen staan twee grafische tekens, als u in de hoofdlettermode bent krijgt u door het indrukken van de betreffende toets met SHIFT het rechter teken dat op de toets staat en door het indrukken van de toets tesamen met de Commodoretoets het linker grafische teken.

U heeft dus eigenlijk de beschikking over diverse toetsenborden, namelijk:
1) Normaal
2) Met SHIFT
3) Met COMMODORE
en twee tekensets, namelijk hoofdletter mode en kleine letter mode, met behulp van SHIFT en COMMODORE schakelt u tussen de twee tekensets.

1.1 Het toetsenbord



Figuur 7/1.1-2: De tekens in hoofdlettermode met SHIFT.



Figuur 7/1.1-3: De tekens in hoofdlettermode met COMMODORE-toets.

7/1.2

Actie

U kunt even de mogelijke kleuren van de computer bekijken door hem aan te zetten en nauwkeurig de volgende regels in te tikken, waarbij u na elke regel op RETURN moet drukken.
Bekommer u op dit moment niet om de betekenis van de regels, die komt later aan de orde.

```
10 X=INT(16*RND(1))
20 POKE53281,X
30 POKE53280,15-X
40 FOR T=1TO500:NEXTT
50 GOTO10
```

Is alles ingetikt, dan tikt u letterlijk RUN en drukt daarna op RETURN. Maakte u bij het intikken een fout dan zal de computer dit bij de uitvoering melden met de mededeling '?SYNTAX ERROR IN', gevolgd door een regelnummer, dat wil zeggen dat u een fout tegen de regels van de taal BASIC heeft gemaakt. U kunt de fout corrigeren door eerst in te tikken LIST (uiteraard gevolgd door RETURN). Het programma wordt dan weer op het scherm afgedrukt en door met de cursor-besturingstoetsen en de toets INST/DEL te werken kunt u de fouten verbeteren. Of u tikt de regel helemaal opnieuw in. Vergeet niet na uw correcties op RETURN te drukken. Is het programma gecorrigeerd, dan tikt u weer letterlijk het woord RUN in en drukt daarna op RETURN.

Wilt u discoverlichting, onderbreek het programma dan met behulp van de STOP toets en tik in 40 gevolgd door RETURN. Na RUN flitst het u voor de ogen.

Bent u uitgekeken, zet dan de computer uit en weer aan - om zo uit het programma te stappen en het geheugen te wissen - want u gaat nu echt starten met „computeren".

7/2

Rekenen in de directe mode en de eerste BASIC programma's

Inhoud

7/2.1

Rekenen met de computer in de directe mode

We gaan nu de computer als het ware gebruiken als een (weliswaar dure) rekenmachine en vragen niet meer van hem. Wanneer u intikt: 5*4 en daarna RETURN, gebeurt er ogenschijnlijk niets. Dit klopt, want u heeft aan de computer geen instructie gegeven om het resultaat van de uitgevoerde bewerking aan u te tonen.
PRINT is het eerste (en een van de meest gebruikte) BASIC keyword (sleutelwoord) dat u nodig heeft. PRINT zorgt ervoor dat het resultaat van een bewerking op het scherm wordt afgedrukt.

Probeer nu eens de volgende oefeningen, vul ze aan met zelfgemaakte sommen waarvan de uitkomsten u bekend zijn. Belangrijk is dat u bij elke opgave voorspelt wat u op het scherm te zien krijgt. Vergeet niet: oefening baart kunst.

INTIKKEN	OP SCHERM	OPMERKING
1) PRINT 15	.....	
2) PRINT 3+5	.....	
3) PRINT 3*5	.....	
4) PTINT 3*5	.....	PTINT is geen BASIC!
5) PRINT"TEKST"	.....	
6) PRINT"TEKSTEN"	.....	alles tussen de aanhalingstekens wordt letterlijk afgedrukt.
7) PRINT 2+6	.....	PRINT is bekend, T 2 is een toegestane naam voor een variabele (zie 7.3.2) er is geen waarde toegekend, dus =0
8) PRINT 4/0	.....	computer kan ook niet door 0 delen!
9) PRINT"4/0"	.....	zie voorbeeld 6
10) PRINT 2*3,3*5,4*6,5*7	.....	het scherm is verdeeld in 4 kolommen, na ',' komt de volgende uitkomst 1 kolom verder.
11) PRINT(4+5)*3	.....	tussen haakjes wordt eerst uitgevoerd
12) PRINT 4+5*3	.....	Volgorde bewerkingen!
13) PRINT"2+3=";2+3	.....	';' drukt antwoord direct na tekst!

U heeft nu kennis gemaakt met het eerste BASIC commando, namelijk 'PRINT' maar u heeft nog geen programma geschreven, want de computer heeft alle opdrachten vergeten nadat ze waren uitgevoerd.

7/2.2

Een BASIC programma

Tot nu toe voerde de computer uw opdrachten direct uit, maar u wilt hem voorzien van instructies die later pas mogen worden uitgevoerd. U wilt een programma maken, dat wil zeggen een reeks instructies die de computer kan interpreteren en uitvoeren. Hoe nu te handelen? De computer hanteert de volgende methode: als een regel begint met een regelnummer (van 0 tot 63999) dan wordt deze regel in het geheugen opgeslagen en na het commando RUN (letterlijk intikken, niet de RUN/STOP toets) pas uitgevoerd.

Alle aanwezige regels worden dan uitgevoerd, in oplopende volgorde van regelnummer, tenzij er speciale instructies aanwezig zijn om deze volgorde te veranderen, waarover later. U doet er verstandig aan als u een programma gaat schrijven om ruimte tussen de regelnummers te houden. U kunt later regels willen tussenvoegen en het is verstandig daar nu al rekening mee te houden.

Twee belangrijke BASIC commando's
NEW: verwijdert alle aanwezige BASIC programmaregels uit het geheugen.
LIST: Drukt de in het geheugen aanwezige BASIC programma regels in oplopende volgorde af op het scherm.

Oefenen met programmaregels.
Probeer zoveel mogelijk routine op te doen door te oefenen. Vergeet niet na elke regel op RETURN te drukken ten teken dat u met het intikken van een regel klaar bent en de computer aan de beurt is. Tik nu in: NEW (hiermee verwijdert u eventueel aanwezige BASIC programmaregels).

```
10 PRINT "IK GEBRUIK MIJN
   GEHEUGEN"
20 PRINT "ALS U REGELNUMMERS
   GEBRUIKT"
30 POKE53280,15-X
40 FOR T=1TO500:NEXTT
50 GOTO10
```

```
IK GEBRUIK MIJN GEHEUGEN
ALS U REGELNUMMERS GEBRUIKT
```

Figuur 7/2.2-1: Zo ziet de uitvoer eruit

U ziet wat er op het scherm wordt afgedrukt, wilt u dit nog meer keren afdrukken dan kan dit door wederom RUN te geven, uiteraard gevolgd door RETURN.

U kunt controleren wat er in het geheugen aanwezig is door: LIST in te tikken; dit verzorgt een opgave van de aanwezige

2.2 Een BASIC programma

programmaregels. De syntax (officiële schrijfwijze)is:
<LIST>.

Uw tweede werkende, korte, BASIC programma is tot stand gekomen.

7/2.3

Uitbreiden, tussenvoegen, verwijderen en overschrijven

Tik nu in:
5 REM PROGRAMMEREN
30 PRINT"EN WACHT MET UITVOEREN"
40 ?"TOT U EEN BEVEL GEEFT"
Na LIST ziet u het volgende:

```
10 PRINT "IK GEBRUIK MIJN
   GEHEUGEN"
20 PRINT "ALS U REGELNUMMERS
   GEBRUIKT"
30 PRINT"EN WACHT MET UITVOEREN"
40 PRINT"TOT U EEN BEVEL GEEFT"
```

```
IK GEBRUIK MIJN GEHEUGEN
ALS U REGELNUMMERS GEBRUIKT
EN WACHT MET UITVOEREN
TOT U EEN BEVEL GEEFT
```

Figuur 7/2.3-1: Het resultaat

U ziet dat met regel 5 niets gedaan wordt, maar hij wordt blijkbaar wel in het geheugen bewaard. Het keyword (sleutelwoord) REM veroorzaakt dit (afkorting voor REMARK - Engels voor opmerking). Alles achter het REM commando wordt bij de uitvoering van het programma genegeerd, maar wel in het geheugen opgeslagen.

Bij de LISTING van het programma ziet u dat het vraagteken vervangen is door PRINT. Het vraagteken is een toegestane afkorting. Voor de meeste BASIC keywords zijn afkortingen, u kunt ze vinden in bijlage 1.

Bij regel 40 ziet u dat het sluitende stel aanhalingstekens mag worden weggelaten. Dit mag alleen aan het einde van een programmaregel, dus voor het geven van RETURN. Advies: niet doen, het staat slordig en met zoeken van fouten is het lastig.

Probeer nu:
40 en daarna LIST
En u ziet dat regel 40 is verwijderd. Door het intikken van een regelnummer gevolgd door RETURN wordt een eventuele programmaregel met dit nummer verwijderd.
Tik nu in:

```
15 PRINT"VOORAAN DE REGEL"
25 PRINT"ERG GOED"
35 PRINT"TOT U BEVEELT"
40 PRINT"OM HET UIT TE VOEREN"
50 PRINT"DAN DOE IK DAT
   GEHOORZAAM"
60 END
```

Uitbreiden, tussenvoegen, verwijderen en overschrijven

```
IK GEBRUIK MIJN GEHEUGEN
VOORAAN DE REGEL
ALS U REGELNUMMERS GEBRUIKT
ERG GOED
EN WACHT MET UITVOEREN
TOT U BEVEELT
OM HET UIT TE VOEREN
DAN DOE IK DAT GEHOORZAAM
```

Figuur 7/2.3-2: De uitvoer

U maakt hier kennis met de volgende verschijnselen:
1) De laatst ingevoerde versie van een regel met een bepaald regelnummer is de geldende.
2) U kunt regels tussenvoegen en toevoegen naar eigen believen, in de LISTING sorteert de computer zelf naar oplopende volgorde.

Maak het scherm nu schoon door middel van CLR/SHIFT en probeer de volgende commando's:

LIST 30
LIST 20-40
LIST -40
LIST 20-

U ziet dat u de beschikking heeft over 5 versies van het commando LIST zodat u exact dat stuk van het programma kunt opvragen dat u nodig heeft om te controleren, te verbeteren of te bekijken.
LIST[[<eerste regel>]-[<laatste regel>]]

Het verbeteren of veranderen van een regel:
Breng via INST/DEL en de cursorbesturing op de juiste plaatsen de nodige correcties aan en druk op RETURN. U hoeft niet met de cursor naar het einde van de regel, de gehele regel wordt na RETURN in het geheugen opgenomen.

7/2.4

Wat gebeurt er als u de computer uitzet?

Heel eenvoudig, als u niets doet bent u uw programma kwijt. Dit wilt u waarschijnlijk graag voorkomen.

Leg een cassette in de cassetterecorder en geef het commando SAVE "NAAM" gevolgd door RETURN. (NAAM is een door u gekozen naam voor het programma, maximaal mag deze naam 16 tekens lang zijn.) De computer meldt zich met PRESS RECORD & PLAY ON TAPE. Zodra u dit doet wordt het scherm blank en wordt het programma op de cassetteband opgeslagen. Als dit gebeurd is verschijnt de mededeling SAVING NAAM gevolgd door READY op het scherm.

Bent u er nu zeker van dat het programma goed op de band staat? Nee, maar u kunt dit controleren door terug te spoelen en in te tikken VERIFY "NAAM" gevolgd door RETURN. U krijgt het verzoek PRESS PLAY ON TAPE. Het scherm wordt weer blank, hierna volgt SEARCHING FOR NAAM; FOUND NAAM; het scherm wordt weer blank en nu controleert de computer of het in zijn geheugen aanwezige programma identiek is met het programma op de cassetteband. Indien dit het geval is volgt de mededeling VERIFYING; O.K., zoniet dan volgt een ?VERIFY ERROR en moet u de SAVE procedure herhalen.

Als uw programma correct op de band staat kunt u nu rustig de computer uitzetten.

Wilt u uw programma later weer gebruiken, spoel dan de cassette terug en tik in LOAD"NAAM"gevolgd door RETURN. U krijgt te zien PRESS PLAY ON TAPE, daarna wordt het scherm blank. Zodra een programma gevonden is, volgt SEARCHING FOR NAAM; FOUND 'naam van gevonden programma'. Is dit niet het gezochte programma dan wordt dit niet geladen maar gaat de cassetterecorder verder met zoeken. Is het juiste programma gevonden dan wordt het scherm blank en wordt het programma ingeladen. Zodra de laadprocedure voltooid is - dit ziet u door LOADING; READY - kunt u weer met het programma werken, want het is weer in het geheugen aanwezig. Treedt er bij het laden een fout op dan volgt '? LOAD ERROR'

U heeft nu de volgende BASIC commando's en keywords geleerd:
END - om aan te geven dat het programma ten einde is.
LIST - om het aanwezige programma op te vragen. (5 versies)
LOAD - om een op cassette opgeslagen programma in te laden.
NEW - om een in het geheugen aanwezig BASIC programma te verwijderen.

2.4 Wat gebeurt er als u de computer uitzet?

PRINT - om uitkomsten van bewerkingen en/of teksten op het scherm af te drukken.
REM - om toelichting in het programma op te nemen die bij de uitvoering genegeerd wordt.
RUN - om het aanwezige programma te laten uitvoeren.
SAVE - om een programma op cassetteband op te slaan voor later gebruik.
VERIFY - om te controleren of het programma dat u 'geSAVEd' hebt correct op de cassetteband staat.

Verder heeft u kennis gemaakt met regelnummers en het overschrijven, tussenvoegen, verwijderen en veranderen van programmaregels.

7/3

Raadsels op uw scherm, variabelen en schermindeling

Inhoud

7/3.1

Raadsels op uw scherm

Bij het intikken van de diverse oefeningen (als u tenminste zo enthousiast was) heeft u hoogstwaarschijnlijk af en toe verbaasd gekeken wat er nu weer op het scherm gebeurde. U wist al dat er twee tekensets waren en de nodige grafische tekens, maar na het intikken van een stel aanhalingstekens gebeurden er bij het gebruiken van cursortoetsen en andere bijzondere toetsen heel vreemde dingen. U kreeg vreemde inverse tekens (dat wil zeggen omgekeerd, dus in plaats van een licht teken op een donkere ondergrond juist het omgekeerde) en de toetsen deden niet wat u wilde. De verklaring: door het intikken van het eerste van een stel aanhalingstekens heeft u de computer in de zogenaamde 'quote mode' gebracht, de bijzondere toetsen werken dan niet meer rechtstreeks maar geven een invers teken. Later bij de uitvoering van het programma worden deze tekens niet afgedrukt, maar worden ze opgevat als instructies aan de computer en als zodanig uitgevoerd. Een lijst van de bijzondere tekens die u in de 'quote mode' krijgt, met hun betekenis vindt u in bijlage 2.

Omdat niets meer normaal werkt kunt u in de quote mode niet gemakkelijk fouten corrigeren, want u kunt de cursor niet sturen. U lost het probleem op door op RETURN te drukken, of door een tweede stel aanhalingstekens in te tikken. Beide oplossingen hebben hun voor- en nadelen, maar bij beide moet u nadenken.
Er kan nog iets geks gebeuren als u met het toetsenbord bezig bent, namelijk als u via INSERT een aantal plaatsen ruimte heeft gemaakt, heeft u uw computer in de 'insert mode' gebracht. Alle controletoetsen hebben hetzelfde effect als in de quote mode behalve INST en DEL. DEL geeft een inverse T en INST geeft normale spaties. U komt uit de 'insert mode' door RETURN of SHIFT/RETURN, terwijl het probleem ook over is als er evenveel tekens zijn ingetikt als er spaties zijn gemaakt.

Dus elk inverse teken via 'quote-' of 'insert mode' opgenomen in een programma wordt bij uitvoering hiervan opgevat als een instructie die de computer dan gehoorzaam uitvoert.

7/3.2

Variabelen, stringvariabelen en geheugenplaatsen

Het is u al bekend dat u programmaregels in het geheugen kunt opslaan. Hoe dit exact gebeurt is op dit moment niet zo belangrijk, wel belangrijk is dat u zich een bepaalde voorstelling maakt van het computergeheugen. U kunt zich dat waarschijnlijk het eenvoudigste voorstellen als een verzameling van veel (65536) postbussen. Het hangt van u af wat u er in stopt, tenminste in het RAM gedeelte.
U zou natuurlijk kunnen spreken over nummer 1885 is zo groot, of in nummer 2459 staat het teken M. Deze nummers noemen we in het vervolg 'adressen'. Het is echter mogelijk om de in het programma gebruikte 'postbussen' niet exact te specificeren, door hun adres, maar ze aan te duiden door een naam. Stelt u zich maar voor dat er in de computer een minimannetje een etiket op de te gebruiken postbus plakt en daar de naam waaronder de inhoud voortaan bekend is op schrijft. Omdat de inhoud van de postbus kan variëren maar het etiket blijft zitten spreken we voortaan over variabelen. Er blijken dan twee soorten variabelen te zijn, we moeten ons dan ook aan regels houden voor de naamgeving van die variabelen. We spreken over getalvariabelen - hierin kunnen alleen positieve en negatieve gehele en gebroken getallen worden opgeborgen - en over stringvariabelen, waarin alle tekens kunnen worden opgeslagen. We maken onderscheid tussen getal- en stringvariabelen door de naam van een stringvariabele te laten eindigen met het '$' teken. Verder zijn we behoorlijk vrij in de naamgeving van variabelen, het eerste teken mag echter nooit een cijfer zijn. Ook moeten we onthouden dat de computer alles prima vindt, maar alleen naar de eerste twee tekens kijkt.

AP is dus toegestaan, maar APPELBOOM ook. De computer ziet echter geen verschil tussen deze twee. Er zijn nog een paar beperkingen; u mag ST niet gebruiken evenmin als TI en TI$ (hierover later meer). Verder mogen gereserveerde BASIC keywords en commando's niet in de namen van variabelen voorkomen. Na RUN krijgt u dan '?SYNTAX ERROR IN ...'

Variabelen, stringvariabelen en geheugenplaatsen

```
STAM=10

?SYNTAX  ERROR
READY.
TIEN$="ROOD"

?ILLEGAL QUANTITY  ERROR
READY.
FORT$="BLAUW"

?SYNTAX  ERROR
READY.
DANDY$="MOOI"

?SYNTAX  ERROR
READY.
```

Figuur 7/3.2-1: Voorbeelden van niet correcte namen van variabelen

In het voorbeeld zitten de niet toegestane namen ST, TI en de keywords FOR en AND opgenomen en dit geeft nu eenmaal problemen.

Door gebruik te maken van het toekenningssymbool '=' kunt u de variabelen een waarde geven. Probeer maar het volgende programma. Tracht ook hier weer te voorspellen wat er op het beeldscherm te zien komt.

```
5 REM VOORBEELD 2
10 A=4
20 B=5:C=6
30 D=3*4
40 A$="PIET"
50 C$="ERMAN"
60 D$="123"
70 F$="PRINT"
80 PRINTA
90 PRINTB;C
100 PRINTA,B,C
110 PRINTD
120 PRINTA$
130 PRINTA$;C$
140 PRINTA$,C$
150 PRINTA$,,C$
160 PRINTD$
170 PRINTF$
180 END
READY.
```

Toelichting:
regel 5: naam van het programma in REM statement
regel 10: variabele A krijgt de waarde 4
regel 20: variabele B krijgt de waarde 5 en C de waarde 6. U ziet dat u op een regel meer instructies mag plaatsen, als ze maar gescheiden worden door een ':'.
regel 30: D krijgt een waarde gelijk aan het antwoord van de gegeven berekening.
regel 40: A$ krijgt de inhoud PIET
regel 50: C$ krijgt de inhoud ERMAN
regel 60: D$ krijgt de inhoud 123. Dit verschilt wezenlijk van het getal 123, want met D$ kunt u niet rekenen.
regel 70: F$ krijgt de inhoud PRINT. Tussen de aanhalingstekens mag wel een keyword staan.
regel 80: de waarde van A wordt afgedrukt. Let op dat er eerst een spatie wordt afgedrukt, deze is bestemd voor het afdrukken van een eventueel '-' teken, het '+' teken wordt niet afgedrukt.
regel 90: de waarden van B en C worden afgedrukt. De ';' zorgt er voor dat de getallen direct achter elkaar worden afgedrukt; dat er toch ruimte is komt doordat na het getal B een spatie wordt afgedrukt en voor het getal C een ruimte voor het teken wordt opengehouden.

3.2 Variabelen, stringvariabelen en geheugenplaatsen

regel 100: Hier worden A, B en C afgedrukt. Het scherm van 40 tekens breed is verdeeld in 4 kolommen van elk 10 tekens en de ',' zorgt ervoor dat naar het begin van een volgende kolom wordt gesprongen.
regel 110: De waarde van D wordt afgedrukt. Let wel, dit is de uitkomst van de berekening in regel 30.
regel 120: A$ wordt afgedrukt.
regel 130: A$ en direct daarna C$ worden afgedrukt. Omdat het strings zijn wordt er geen ruimte voor het eventuele '+' teken opengehouden.
regel 140: A$ en C$ worden afgedrukt in opvolgende kolommen.
regel 150: A$ en C$ worden afgedrukt met een lege kolom tussenruimte. Immers, elke ',' betekent dat naar een volgende kolom gegaan wordt.
regel 160: D$ wordt afgedrukt. Let op hoe verraderlijk veel dit op een getal lijkt. U ziet het verschil door het ontbreken van een spatie voor het teken.
regel 170: F$ wordt afgedrukt.

```
 4
 5  6
 4          5          6
 12
PIET
PIETERMAN
PIET       ERMAN
PIET                  ERMAN
123
PRINT
```

Figuur 7/3.2-2: De uitvoer van het laatste voorbeeld

Laten we eens kijken wat er gebeurt als we een karakter (teken) willen toekennen aan een getalvariabele. Probeer ook een cijfer toe te kennen aan een stringvariabele. De volgende voorbeelden illustreren een paar van de voorkomende haken en ogen en de daarbij optredende foutmeldingen.

```
?TYPE MISMATCH  ERROR
READY.

A="123"

?TYPE MISMATCH  ERROR
READY.

A$="123"

READY.

A=123
```

Figuur 7/3.2-3: Voorbeelden van foutmeldingen.

Het volgende voorbeeld laat u zien dat de computer alleen maar rekening houdt met de eerste twee letters van de naam van een variabele.

```
10 A=4
20 AP=6
30 APPEL=8
40 APPELBOOM=10
50 PRINT A
60 PRINT AP
70 PRINT APPEL
80 PRINT APPELBOOM
90 END
READY.
```

Toelichting:
regel 10: aan A wordt de waarde 4 toegekend.
regel 20: aan AP wordt de waarde 6 toegekend.
regel 30: aan APPEL wordt de waarde 8 toegekend, maar de computer ziet de naam van de variabele als AP (eerste twee letters). Zowel bij afdrukken van AP als van APPEL zouden we nu 8 krijgen.
regel 40: aan APPELBOOM wordt de

3.2 Variabelen, stringvariabelen en geheugenplaatsen

waarde 10 toegekend, zie verder regel 30.
regels 50 t/m 80: de instructies om A, AP, APPEL en APPELBOOM af te drukken. We krijgen op het scherm 4, 10, 10 en 10.
regel 90: einde van het programma.

```
4
10
10
10
```

Figuur 7/3.2-4: De uitvoer van het vorige voorbeeld

7/3.3

Enkele mogelijkheden om het afdrukken op het scherm te beheersen

Het beeldscherm bestaat uit 25 regels, (de regels 0 t/m 24), van 40 tekens (genummerd 0 t/m 39). Erg handig is het om wat vellen papier in voorraad te hebben waarop een ruitjespatroon van 25 x 40 is getekend, u kunt dan de uitvoer die u op het scherm wilt hebben intekenen en verder uittellen wat er moet gebeuren. Enkele exemplaren worden in dit boek meegeleverd.
We gaan proberen om een programma te maken dat alleen onze naam afdrukt op regel nummer 10, dus de 11e regel op het scherm, te beginnen op positie 15, dus de 16e positie op de regel.

We beginnen met het intikken van NEW, gevolgd door RETURN om eventuele resten van een vorig BASIC programma te verwijderen.

Eerst moeten we het scherm schoon zien te krijgen als we het programma uitvoeren. We hebben gezien dat de controletoetsen in de 'quote mode' inverse tekens geven die naderhand als instructie worden opgevat. Het allereerst leggen we de naam van ons programma vast in een REM regel.

```
5 REM VOORBEELD 7
10 PRINT"♥"
20 PRINT"QQQQQQQQQQ]]]]]]]]]]]]]]]
   NAAM"
30 PRINT"←"
READY.
```

Toelichting:
regel 5: REM regel met de naam van het programma.
Na uitvoering van regel 10 staat de cursor op positie 0, 0 en is het scherm schoon, immers het inverse hartje krijgen we door SHIFT/CLS in de 'quote mode' te gebruiken. Bij uitvoering van het programma werkt het als instructie.
regel 20: Hier worden in een printregel de nodige reverse tekens geplaatst. De inverse 'Q' voor CURSOR NAAR BENEDEN en het inverse ']' voor CURSOR NAAR RECHTS. Uittellen en de goede controletoetsen gebruiken zorgen bijna voor het gevraagde resultaat. Alleen het woord READY stoort ons nog.
regel 30: Bij het bekijken van de tekens van de CBM 64 zagen we dat in de 'quote mode' de kleuren ook elk hun eigen inverse teken hadden. Hier lossen we ons probleem op door na het afdrukken van onze naam de tekenkleur te veranderen in de kleur van de ondergrond. Dit doen we door een printregel met CTRL 7 tussen de aanhalingstekens. De tekens worden nu in dezelfde kleur afgedrukt als de ondergrond. U kunt dit weer veranderen door CTRL of COMMODORE met de cijfertoetsen 1 tot en met 8 te gebruiken.

U kunt nu zelf verder oefenen met deze vorm van het controleren van de plaats der op het scherm af te drukken teksten.

3.3 Enkele mogelijkheden om het afdrukken op het scherm te beheersen

Zoals het spreekwoord al zegt: „Er zijn vele wegen die naar Rome leiden"
We kunnen u nog een oplossing geven om het gevraagde doel te bereiken. Het commando TAB(argument) werkt net als de tabulatorfunctie op een schrijfmachine, de printpositie gaat naar de plaats aangegeven door het argument. Het commando PRINT zonder meer zorgt er voor dat de cursor een regel naar beneden gaat.
Beide voorbeelden geven dus hetzelfde resultaat.

```
5 REM VOORBEELD 8
10 PRINT"□"
20 PRINT:PRINT:PRINT:PRINT:PRINT
   PRINT:PRINT:PRINT:PRINT:
   PRINT
30 PRINTTAB(15)"NAAM"
40 PRINT"□"
READY.
```

In het voorgaande gedeelte heeft u kennis gemaakt met de begrippen:

QUOTE MODE - toestand waarin de computer komt na het intikken van " .
INSERT MODE - toestand waarin de computer komt na het maken van ruimte d.m.v. INST/DEL met SHIFT.
ARGUMENT - noodzakelijk gegeven, variabele of constante bij functies die in de CBM 64 ingebouwd zijn.
ADRES - het getal dat een bepaalde geheugenplaats aanduidt.
VARIABELE - symboliseert een adres waar door u een waarde aan kan worden gegeven.
STRINGVARIABELE - als variabele, maar nu kunnen er alleen maar tekens in opgeslagen worden. De maximale lengte van een string kan 255 tekens zijn.
En het BASIC keyword:
TAB(argument) - brengt de cursor op een regel naar de door 'argument' bepaalde positie.

7/4

Variatie en flexibiliteit in uw programma's

Inhoud

Ongemerkt heeft u al zoveel van de taal BASIC opgestoken dat u eenvoudige programma's kunt schrijven. U heeft echter nog geen mogelijkheden om variaties in uw programma's aan te brengen, dit gaan we nu verhelpen.

7/4.1

Variatie in uw programma's

Een paar voorbeelden van de simpele programma's, zoals u die nu zou kunnen maken laten wij nu volgen.

```
5 REM VOORBEELD 9
10 PRINT"■"
20 PRINT"■■■■■■■■SOMMETJE"
30 A=5: B=12
40 PRINT"■■■■■■■5+12=";A+B
50 PRINT"■■■■■■■■GOED HE?"
60 END
READY.
```

Toelichting: alleen bijzondere toetsen binnen een string, dus in Quote mode, zijn gebruikt. Om het programma korter te maken hebben we meer instructies op een regel geplaatst.

```
SOMMETJE

5+12= 17

GOED HE?
```

Figuur 7/4.1-1: Zo ziet het sommetje er uit

```
5 REM VOORBEELD 10
10 PRINT"■"
20 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■
   PIET HEIN■"
30 PRINT"■■■■■■■■■■■■■■WAS EEN
   BEKENDE"
40 PRINT"■■■■■■■■■■■■■■ADMIRAAL"
50 PRINT"■■■■■■■■■■■■■■UIT ONZE■"
60 PRINT"■■■■■■■■■■■■■■GESCHIEDEN
   IS"
70 PRINT"■"
80 END
```

Toelichting: Hier zijn ook kleurinstructies en inverse in de strings meegegeven. Het storende woord READY is omzeild door de tekstkleur en ondergrondkleur gelijk te maken. U bent echter ook uw cursor weer kwijt. Door CTRL of COMMODORE samen met een van de cijfertoetsen 1 tot en met 8 in te tikken wordt uw cursor weer zichtbaar.

```
PIET HEIN

WAS EEN BEKENDE

ADMIRAAL

UIT ONZE

GESCHIEDENIS
```

Figuur 7/4.1-2: De uitvoer van het laatste voorbeeld

De uitvoer zoals hier is afgedrukt wijkt af van die op uw scherm. Kleuren vallen weg en ook de inverse teksten worden als normale letters afgedrukt. Dit is een gevolg van drukken in zwart/wit en de manier waarop het beeldscherm via de printer werd gereproduceerd.

```
5 REM VOORBEELD 11
10 PRINT"■"
20 A$="FLUIT"
30 B$="KETEL"
40 PF=4.95
50 C$="EMMER"
60 PE=8.70
```

4.1 Variatie in uw programma's

```
70 PRINTA$;B$;D$,PF
80 PRINTC$,D$,PE
90 PRINT"□□□□□□SAMEN KOSTEN ZE",
   PF+PE
100 END
```

Toelichting: Hier is gewerkt met ';' om strings direct achter elkaar af te drukken en met ',' om in kolommen af te drukken.

```
FLUITKETEL              4.95
EMMER                   8.7


SAMEN KOSTEN ZE         13.65
```

Figuur 7/4.1-3: Het boodschappenlijstje afgedrukt

Al deze programma's hebben een groot nadeel, als u ze in het jaar 2000 uitvoert gebeurt er precies hetzelfde als nu. Laten we eens kijken naar methodes om verschillende resultaten te krijgen met behulp van hetzelfde programma. U kunt dan bijvoorbeeld uw girorekening gaan bijhouden en andere nuttige dingen gaan doen.

Het moet toch mogelijk zijn dat er een soort dialoog tussen de computer en de gebruiker tot stand komt waarin de gebruiker het programma voorziet van de nodige gegevens.
Met het BASIC keyword 'INPUT' heeft u zo'n stuk gereedschap in handen. Als u in uw programma bijvoorbeeld de volgende regels heeft opgenomen:

150 INPUT A
250 INPUT A$
gebeurt er bij het uitvoeren van het programma het volgende.
Zodra de computer bij regel 150 is aangekomen stopt het programma, drukt een vraagteken op het scherm af en wacht met verder gaan tot u een gegeven heeft ingetikt gevolgd door RETURN.
Bij regel 250 gebeurt hetzelfde.

U ziet waarschijnlijk het probleem al, want in regel 150 moet er een getal ingevoerd worden en in regel 250 een string, maar tussen de vraagtekens is geen enkel verschil te ontdekken. Het is dan ook mogelijk om bij het INPUT commando tekst mee te geven als toelichting, zodat u weet wat het programma van u verwacht. De juiste syntax van het INPUT statement luidt dan ook:
INPUT["<toelichting>";]<lijst van in te voeren variabelen, gescheiden door komma's.>

Een paar voorbeelden:

10 INPUT"geef drie getallen";A,B,C
Let er op dat er tussen de aanhalingstekens na de toelichting en de ';' geen spatie mag staan. Als deze regel wordt uitgevoerd kunt u de drie gevraagde getallen tegelijk invoeren, mits gescheiden door komma's. U kunt ze echter ook een voor een invoeren. Zodra u het eerste getal heeft ingegeven drukt de computer 2 vraagtekens af ten teken dat u nog meer gegevens moet verstrekken.

20 INPUT"geef uw naam en adres";A$,B$
Hiervoor gelden dezelfde opmerkingen als bij regel 10, ook hier geen spatie tussen aanhalingstekens na de tekst en de ';' en

gegevens of in een keer of een voor een invoeren.

Bij het commando INPUT kunt u twee mededelingen van de computer krijgen. Geeft u gegevens die niet van het goede type zijn, als u bijvoorbeeld een string intikt als er een getal gevraagd wordt, dan geeft de computer de boodschap '?REDO FROM START' Geeft u meer gegevens dan de computer nodig heeft, dan meldt de computer dit door '?EXTRA IGNORED' op het scherm af te drukken.

Voorzichtigheid is geboden bij het invoeren van strings, want INPUT heeft problemen met diverse leestekens. U zult het al vermoeden, als in uw ingevoerde string een komma voorkomt dan ziet de computer dit als scheidingsteken tussen twee invoergegevens. Ook de dubbele punt geeft moeilijkheden. Als u de ingevoerde string tussen aanhalingstekens plaatst is dit probleem opgelost.

Bij de invoer van getallen is ',' ook het scheidingsteken en is verder alleen de decimale punt toegestaan.

7/4.2

Meer flexibiliteit in uw programma

Tot nu toe werden programma's uitgevoerd te beginnen bij regel 0 en vervolgens in oplopende volgorde van regelnummers. GOTO<regelnummer> of GO TO<regelnummer> brengen sprongen naar de aangegeven regelnummers te weeg. Het keyword is beslist niet geliefd bij programmeurs, want het maakt het programma niet overzichtelijker.

```
5 REM VOORBEELD 13
6 PRINT"␣"
10 PRINT"AAP"
15 GOTO 60
20 PRINT"EN NOG MEER CHAOS"
25 GOTO30
30 PRINT"DIT GEEFT SPAGHETTI"
35 GOTO70
40 PRINT"WIM"
45 GOTO20
50 PRINT"MIES"
55 GOTO40
60 PRINT"NOOT"
65 GOTO50
70 END
```

Als u in het bovenstaande voorbeeld de loop van het programma wilt volgen door het aanbrengen van verbindingslijnen, dan zult u zien dat u zelfs in dit korte programmaatje een behoorlijk onoverzichtelijke knoedel 'spaghetti' krijgt. Het is daarom verstandig om het GOTO statement zo min mogelijk toe te passen.

```
AAP
NOOT
MIES
WIM
EN NOG MEER CHAOS
DIT GEEFT SPAGHETTI
```

Figuur 7/4.2-1: Het resultaat door GOTO

7/4.3

Nog meer flexibiliteit in uw programma

Stelt u zich voor dat u een programma maakt dat uw girorekening bijhoudt, dan is het prettig dat u, indien uw saldo negatief zal worden als de afschrijvingen verwerkt zijn, van de computer de mededeling krijgt 'u moet zoveel gulden bijstorten'
Dit betekent niets meer of minder dat uw programma mogelijkheden moet hebben om 'zelf' beslissingen te nemen als zich bepaalde feiten voordoen.

Met de instructie IF ..... THEN .....(als..... dan.....) voorziet u Uw programma van een dosis 'eigen initiatief'.
De vormen waarin deze instructie gegeven kan worden zijn:
IF<uitdrukking>THEN<regelnummer>
IF<uitdrukking>GOTO<regelnummer>
IF<uitdrukking>THEN<instructie>
In de expressie kunt u de volgende wiskundige symbolen gebruiken:

voorbeeld	symbool	betekenis
A = 5	=	A is gelijk aan 5
A < 5	<	A is kleiner dan 5
A <= 5	<=	A is kleiner dan of gelijk aan 5
A > 5	>	A is groter dan 5
A >= 5	>=	A is groter dan of gelijk aan 5
A <> 5	<>	A is ongelijk aan 5

U kunt behalve getalvariabelen in de expressie ook stringvariabelen gebruiken. IF A$="PIET" THEN PRINT "DAG PIET" is volledig juist.
De instructie werkt als volgt: allereerst wordt de expressie na IF bekeken op 'juist' of 'onjuist'. Is het resultaat 'juist' dan wordt de rest van de instructie uitgevoerd, is het resultaat 'onjuist' dan wordt datgene wat na de expressie komt genegeerd en gaat het programma met de regel na de test verder.
U mag ook combinaties maken in de vorm: IF A>0 THEN IF A<10 THEN PRINT"A ligt tussen 0 en 10"

Nuttig gebruik van het voorgaande

Ongemerkt heeft u met het IF...THEN... statement een stuk programmeergereedschap in handen gekregen om programmagedeelten een door u te bepalen aantal malen te herhalen. Het is nuttig om de volgende in te tikken, te runnen en goed te bekijken.

```
5 REM VOORBEELD 14
10 PRINT"□":T=1
20 INPUT"GEEF EEN GETAL";G
30 PRINT" ";T;"MAAL";G;"=";T*G
40 T=T+1
50 IF T<=10 THEN 30
60 PRINT"U KUNT NU ALLE TAFELS
   MAKEN"
70 END
```

Dit voorbeeld volgen we van regel tot regel.
regel 5: geeft in een REM regel de naam van het programma.
regel 10: maakt het scherm schoon en zet een teller op 1.
regel 20: vraagt u om in te voeren welke tafel gemaakt moet worden.
regel 30: drukt een regel van de tafel af.
regel 40: de teller wordt met 1 opgehoogd.
regel 50: als de teller nog niet groter dan 10 is wordt teruggesprongen naar regel 30.

```
GEEF EEN GETAL? 8
  1 MAAL 8 = 8
  2 MAAL 8 = 16
  3 MAAL 8 = 24
  4 MAAL 8 = 32
  5 MAAL 8 = 40
  6 MAAL 8 = 48
  7 MAAL 8 = 56
  8 MAAL 8 = 64
  9 MAAL 8 = 72
  10 MAAL 8 = 80
U KUNT NU ALLE TAFELS MAKEN
```

Figuur 7/4.3-1: Zo kunt u tafels maken

```
5 REM VOOORBEELD 15
10 PRINT"□"
20 T=T+1
30 INPUT"GEEF UW NAAM";N$
40 INPUT"GEEF UW LEEFTIJD";L
45 REM KIJKEN OF DE LEEFTIJDEN MOGELIJK ZIJN
50 IF L<0 THEN GOTO 200
55 IF L>100 THEN GOTO 200
60 INPUT"MAN OF VROUW M/V";G$
65 REM KIJKEN OF HET GESLACHT JUIST IS INGEVOERD
70 IF G$<>"M"THEN IF G$<>"V"THEN GOTO200
75 REM ALS DE LEEFTIJDEN BUITEN 18-65 VALLEN ZIJN DE PERSONEN NIET
   INTERESSANT
80 IF L>65 THEN PRINT"U KRIJGT AOW":GOTO20
90 IF L<18 THEN PRINT"U BENT MINDERJARIG":GOTO20
100 IFG$="M"THENPRINT"U BENT EEN MAN":M=M+1
110 IF G$="V"THENPRINT"U BENT EEN VROUW":V=V+1
115 REM KIJKEN OF HET AANTAL DEELNEMERS DAT WE WILLEN HEBBEN
    BEREIKT IS
120 IF T<10THEN 20
130 PRINT"DE ENGUETE HAD";T"DEELNEMERS"
140 PRINT"WAARVAN";V;"MEERDERJARIGE VROUWEN EN";M;
```

4.3 Nog meer flexibiliteit in uw programma

```
   "MEERDERJARIGE MANNEN"
150 PRINT"DIE NOG GEEN 65 JAAR WAREN"
160 END
200 PRINT"ONWAARSCHIJNLIJK,OVERDOEN":GOTO30
```

Toelichting: De enige bijzonderheden zijn de meervoudige tests in regel 50 en een paar meervoudige programmaregels.

```
5 REM VOORBEELD 16
10 PRINT"{CLR}"
20 INPUT"GEEF UW NAAM ";N$
30 INPUT"GEEF UW ADRES";A$
40 INPUT"GEEF UW WOONPLAATS";W$
50 IF W$<>"AMSTERDAM" THEN 200
60 PRINT " U BENT AMSTERDAMMER"
90 PRINT"NU MAKEN WE UW VISITEKAARTJE"
120 PRINT"{CLR}"
130 PRINT"{DOWN}{RIGHT}";N$
140 PRINT"{DOWN}{RIGHT}";A$
150 PRINT"{DOWN}{RIGHT}";W$
154 REM WE HALEN HET WOORD READY WEG
155 PRINT"{UP}"
156 REM WE LATEN DE COMPUTER TELLEN OM TE VERTRAGEN. U LEERT NOG
    EEN BETERER
157 REM OPLOSSING
160 W=W+1:IF W<500 THEN GOTO 160
170 W=0
174 REM WE ZORGEN DAT DE CURSOR WEER TERUG KOMT
175 PRINT"{UP}"
180 GOTO10
200 PRINT"U BENT GEEN AMSTERDAMMER"
210 PRINT"IK  DRUK GEEN VISITEKAARTJE VOOR U"
215 REM OOK WEER VERTRAGEN
220 T=T+1: IF T<500 THEN GOTO 220
230 T=0
240 GOTO10
```

Toelichting: Hier wordt getest of iemand in AMSTERDAM woont of niet. Voor de Amsterdammers maken we een visitekaartje. We laten teksten langer op scherm staat door de computer te laten tellen en dit kost nu eenmaal tijd. In hoofdstuk 7.5 leert u een mooiere methode.

In het voorgaande gedeelte heeft u weer een aantal BASIC keywords leren kennen:

INPUT - Hierbij wordt het programma onderbroken en een vraagteken, al dan niet met toelichting op het scherm geplaatst. U kunt dan gegevens invoeren en het programma gaat daarna verder.
GOTO rn - zorgt voor een sprong naar het aangegeven regelnummer.
IF en THEN vormen samen een voorwaarde. Aan de hand van de juist- of onjuistheid van de expressie die in de IF.....THEN instructie voorkomt gaat het programma verder.

7/5

Het nut van lussen

Inhoud

7/5.1

Herhalen en sturen zonder moeite

Het zal geregeld voorkomen dat u bepaalde gedeelten van het programma dat u aan het schrijven bent ettelijke keren zult moeten uitvoeren. U heeft hiervoor al een methode leren kennen, namelijk het in het programma ophogen van een telvariabele en aan het einde van het te herhalen gedeelte met behulp van IF...THEN testen of het aantal herhalingen dat u wenste, uitgevoerd werd.
Er is nog een methode om programmagedeelten te herhalen, die erg flexibel is en bovendien diverse voordelen biedt boven de vorige manier.

Met behulp van de combinatie:
FOR<variabele>=<startwaarde->TO<eindwaarde >[STEP<stapgrootte>],
te herhalen programma gedeelte
NEXT[<dezelfde variabele>]
heeft u een stuk programmeergereedschap in handen met zeer veel mogelijkheden. U kunt er mee herhalen, beweging besturen, selecteren, vertragen, timen en nog veel meer.

U bent erg vrij in het bepalen van uw start en eindwaarde, want het maakt niet uit of de startwaarde groter is dan de eindwaarde, mits u uw stapgrootte dan maar negatief maakt (anders wordt het gedeelte tussen FOR en NEXT maar een maal uitgevoerd!). In vele gevallen kunt u de stapgrootte gewoon weglaten, de computer neemt dan aan dat de stapgrootte gelijk is aan een. U begrijpt uit het voorgaande zinnetje al dat uw stapgrootte ook een ander getal mag zijn, zelfs gebroken getallen zijn toegestaan.

Bij NEXT mag u de variabelenaam die u bij FOR vermeldde weglaten, dit maakt voor de computer niets uit.
Hoe werkt de constructie FOR ... NEXT ? Allereerst worden bij het tegenkomen van het woord FOR enige gegevens door de computer in zijn geheugen vastgelegd. Dit gebeurt in een bepaald stuk, de STACK, hierover vindt u in het gedeelte over machinetaal van dit boek veel meer informatie. Dit is een vrij delicaat stukje geheugen van beperkte omvang, waar u voorzichtig mee moet omspringen om niet de mededeling '?OUT OF MEMORY' te krijgen. De computer houdt vast: het regelnummer van FOR, de startwaarde, de eindwaarde en de stapgrootte van de betrokken variabele. De variabele wordt op de startwaarde gezet en het programma wordt voortgezet. Zodra de bijbehorende NEXT instructie wordt ontmoet wordt de variabele opgehoogd met de stapgrootte en wordt gecontroleerd of de waarde van de variabele de eindwaarde heeft overschreden. Is dit niet het geval dan keert de computer terug naar de bijbehorende

5.1 Herhalen en sturen zonder moeite

haalt dit net zolang tot de eindwaarde is overschreden. Hierna wordt het programma normaal verder uitgevoerd.

Het is zaak om de instructie volledig in het programma op te nemen; is er in uw programma een NEXT instructie zonder dat daar het bijbehorende FOR-gedeelte voor aanwezig is, dan volgt de mededeling '?NEXT WITHOUT FOR' en stopt uw programma.
De computer moet immers ophogen en eventueel terugspringen, maar hij heeft geen gegevens. Andersom volgt geen foutmelding, want als NEXT ontbreekt gaat het programma normaal verder en blijft op NEXT wachten. Uw STACK wordt dan wel zwaarder belast dan nodig is..... en hij is maar beperkt van omvang.

Probeer ook niet om uit een FOR...NEXT lus te springen met behulp van een IF...THEN test gevolgd door GOTO, want dan gaat het mis met de STACK. Dit soort lussen moet u met de Commodore versie van BASIC volledig doorlopen.
Het is ook niet mogelijk om van buitenaf in een lus te springen, want dan komt u 'NEXT WITHOUT FOR' tegen.
Probeert u maar eens:

```
30 NEXT A
40 PRINT"WE PROBEREN IN DE LUS
   TE SPRINGEN"
50 GOTO20

 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
 10 MAAL AFDRUKKEN
WE PROBEREN IN DE LUS TE SPRINGEN
 10 MAAL AFDRUKKEN

?NEXT WITHOUT FOR  ERROR IN 30
```

Figuur 7/5.1.2: Het resultaat

7/5.1.1 LUSSEN BINNEN LUSSEN

U mag diverse lussen binnen elkaar plaatsen, mits de ene lus maar volledig binnen de andere ligt.
De constructie:

```
5 REM VOORBEELD 17
10 FOR A=1 TO 10
20 PRINT" 10 MAAL AFDRUKKEN"
```

```
6 REM LUSSEN BINNEN LUSSEN
500 FOR Z=60 TO 75 STEP 5
510 PRINT"EERSTE LUS"
520 FOR T=1 TO 500:NEXT:REM
    TWEEDE LUS
530 PRINT Z,Z↑2
540 FOR Y=1 TO 5
550 PRINT"DERDE LUS"
560 PRINT Z,Y,Y*Z
570 NEXT Y
580 NEXT Z
```

is dus volkomen correct, maar als er in het voorgaande: 570 NEXT Z gevolgd door: 580 NEXT Y stond dan zou er een foutmelding volgen.

FOR...NEXT heeft zeer veel mogelijkheden; van enkele ervan volgt een korte bespreking.

7/5.2

Vertragingen in uw programma

Door het opnemen van de volgende regel: rn FOR T= 1 to N: NEXT, waarbij u de grootte van N zelf bepaalt, wordt uw programma vertraagd. De vertraging is afhankelijk van de grootte van N (Voortaan wordt met rn het regelnummer bedoeld.). Hier laat u de computer duidelijk een aantal keren totaal niets doen, maar het controleren of hij voldoende keren niets heeft gedaan kost tijd, vandaar de vertraging.

Het nu volgende programmaatje bevat deze vertragingslussen, later leert u een methode met minder schrijfwerk kennen.

```
5 REM VOORBEELD 18
10 PRINT"[illegible]"
20 PRINT"DIT"
30 FORT=1TO500:NEXT
40 PRINT" IS EEN"
50 FORT=1TO500:NEXT
60 PRINT"  STANDAARDWERK"
70 FORT=1TO500:NEXT
80 PRINT"   OVER DE CBM 64"
90 END
```

7/5.3

Het selecteren van bepaalde waarden

Als u bijvoorbeeld alle veelvouden van 23 wilt hebben binnen een bepaald bereik en daarmee verder wilt werken gebruikt u de volgende constructie:
FOR X= beginwaarde TO eindwaarde STEP 23. U werkt verder met de waarden die X krijgt en sluit de lus af met NEXT X. Achter NEXT mag u rustig X weglaten.

Som van de door 23 deelbare getallen tussen 0 en 2300

```
5 REM VOORBEELD 19
10 FOR T=0 TO 2300 STEP 23
20 A=A+T
30 NEXTT
40 PRINT"DE SOM IS";A
50 END
```

Toelichting:
Een eenvoudige oplossing om de som van de door 23 deelbare getallen tussen 0 en 2300 te bepalen. Regel 10 stelt begin- en eindwaarde en stapgrootte in, regel 30 bevat altijd het totaal van de getallen die aan de beurt zijn geweest.

De nu volgende voorbeelden tonen u dat de computer anders rekent dan wij. Bij uw Commodore hoeft het kwadraat van twee gehele getallen geen geheel getal te zijn, u kunt ook zien hoe u het probleem kunt omzeilen. Later bespreken we de functie INT.

```
5 REM VOORBEELD 20
10 PRINT"[CLR]"
20 PRINT"GETAL","KWADRAAT",
   "DERDE MACHT"
30 FOR X=0TO50 STEP 4
40 PRINTX,X↑2,X↑3
50 NEXTX
60 END
```

Bij het kwadraat van 24 en 28 komt de computer met een resultaat met 7 decimalen, samen met de vier posities voor de decimale punt (teken + 3 cijfers) betekent

```
GETAL       KWADRAAT   DERDE MACHT
 0           0          0
 4           16         64
 8           64         512
 12          144        1728
 16          256        4096
 20          400        8000
 24          576.000001             13824
 28          784.000001             21952
 32          1024       32768
 36          1296       46656
 40          1600       64000
 44          1936       85184.0002
 48          2304       110592
```

Figuur 7/5.3-1: Zo ziet het staatje er uit

```
GETAL       KWADRAAT   DERDE MACHT
 0           0          0
 4           16         64
 8           64         512
 12          144        1728
 16          256        4096
 20          400        8000
 24          576        13824
 28          784        21952
 32          1024       32768
 36          1296       46656
 40          1600       64000
 44          1936       85184
 48          2304       110592
```

Figuur 7/5.3-2: Zoals blijkt

5.3 Het selecteren van bepaalde waarden

dit dat we voorbij de eerste positie van de volgende kolom springen. De rest van de uitvoer verspringt dus ook.

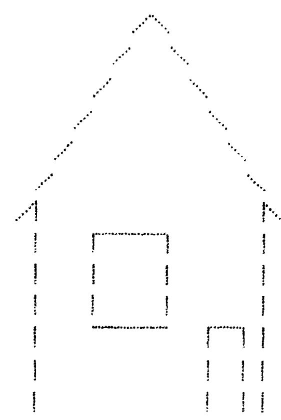
```
5 REM VOORBEELD 21
10 PRINT"[illegible]"
20 PRINT"GETAL","KWADRAAT",
   "DERDE MACHT"
30 FOR X=0TO50 STEP 4
40 PRINTX,INT(X↑2),INT(X↑3)
50 NEXTX
60 END
```

7/5.4

Het maken van figuren

Probeert u maar het volgende voorbeeld:

```
5 REM VOORBEELD 22
10 PRINT"♥"
20 PRINT"◘◘◘◘◘◘▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱╲"
30 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱▮▮▮╲"
40 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱▮▮▮▮▮╲"
50 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱▮▮▮▮▮▮▮╲"
60 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱▮▮▮▮▮▮▮▮▮╲"
70 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱▮▮▮▮▮▮▮▮▮▮▮╲"
80 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮╱▏ ▮▮▮▮▮▮▮▮▮▮▮▮ ▕╲"
90 FORA=1TO6
100 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮ ▮▮▮▮▮▮▮▮▮▮▮▮ ▏"
110 NEXTA
120 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮┌──────────────"
130 PRINT"□□□□□▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮ ▮▮▮ ▏"
140 PRINT"□□□▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮┌───┐"
150 FORA=1TO2
160 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮ ▮▮▮ ▏"
170 NEXTA
180 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮┌────▮▮┌─┐"
190 FORA=1TO2
200 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮ ▏"
210 NEXTA
```



Figuur 7/5.4-1: Het huis is klaar

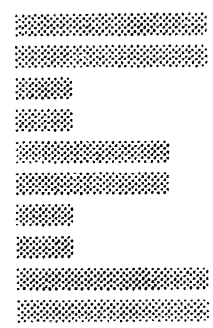
Toelichting:
regel 10 maakt het scherm schoon.
regels 20 t/m 80 maken het dak en de aansluiting met de muren.
regel 90 t/m 110 maken de muren met behulp van een FOR-NEXT lus.
regel 120 maakt de vloer.
regels 130 t/m 200 sturen de cursor naar boven en maken raam en deur.

Nog een voorbeeld van het maken van figuren met behulp van de FOR...NEXT instructie.

5.4 Het maken van figuren

Toelichting:
Hier wordt met behulp van grafische tekens, FOR-NEXT lussen en cursorbesturing een grote letter E gemaakt. Later zult u zien dat dit ook op een andere manier kan.

```
5 REM VOORBEELD 23
10 PRINT"▯▯▯▯▯▯▯▯▮▮▮▮▮▮▮▮▮░░░░░░"
20 FORA=1TO9
30 PRINT"▮▮▮▮▮▮▮▮▮░░░░░░"
40 NEXTA
50 PRINT"▯▯▯▯▯▯▯▯▯▯▯▮▮▮▮▮▮▮▮▮▮▮▮░░░░░░░░░░░░"
60 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮░░░░░░░░░░░░"
70 PRINT:PRINT
80 FORA=1TO2
90 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮░░░░░░░░░░"
100 NEXTA
110 PRINT:PRINT
120 FORA=1TO2
130 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮░░░░░░░░░░░░"
140 NEXTA
150 END
```



Figuur 7/5.4-2: Goed leesbaar

7/5.5

Het testen van de snelheid van uitvoering

Heel eenvoudig, u zet een FOR-instructie voor het te testen programmagedeelte en laat dit volgen door een NEXT. U kunt dan timen hoeveel tijd bijvoorbeeld 250 maal de uitvoering van de diverse mogelijkheden kost en zo zien welke het snelste werkt.

Hoe groot is uw oplossing?

Het allerbeste programma is natuurlijk hetgene dat het snelste werkt en bovendien de minste geheugenruimte in beslag neemt. U heeft net gezien hoe u van diverse mogelijkheden de beste qua snelheid kunt bepalen, maar hoe zit het met de lengte? Er is een functie ingebouwd in de Commodore 64 die u aangeeft hoeveel vrij geheugen u nog ter beschikking heeft.
PRINT FRE(< argument>) geeft u de nog beschikbare vrije ruimte voor BASIC. Als u bij het opvragen hiervan een negatief getal krijgt moet u niet in paniek raken, maar er gewoon 65536 bij optellen, dan heeft u het exacte aantal nog vrije bytes in het geheugen. Het geregeld gebruiken van deze functie vermindert het aantal 'OUT OF MEMORY' foutmeldingen aanmerkelijk! Het maakt niet uit wat U bij de functie FRE tussen de haakjes zet want met het argument zelf doet de computer niets, we spreken van een dummy argument. U zult er nog meer tegenkomen in de rest van dit verhaal.

U begrijpt zelf wel dat u dikwijls concessies zult moeten doen, het snelste programma is niet altijd het kortste en ook het omgekeerde geldt.

In dit hoofdstuk heeft u weer een paar BASIC commando's leren kennen:
FOR...NEXT - de lus die het u mogelijk maakt een bepaald programmagedeelte een door u te bepalen aantal malen uit te voeren. Met STEP bepaalt u de stapgrootte.
FRE(X), de functie die u het nog resterende beschikbare geheugen kan opgeven.
U heeft ook kennis gemaakt met het begrip dummy.

7/6

De strings aangepakt

Inhoud

7/6.1

Mogelijkheden en bewerkingen met strings

Bij het behandelen van de stringvariabelen leerde u reeds dat een string uit alle soorten tekens mag bestaan. Er is echter wel een beperking aan een string, namelijk de lengte. De maximaal toegestane lengte van een string is 255 tekens; wilt u hem langer maken dan krijgt u de foutmelding 'STRING TOO LONG ERROR'.

```
5 REM VOORBEELD 24
10 PRINT"■"
20 A$="EEN COMPUTER "
30 B$="IS EEN APPARAAT "
40 C$="ZONDER EIGEN INITIATIEF"
50 A=LEN(A$):B=LEN(B$):C=LEN(C$)
60 PRINTA$;B$;C$
70 PRINT
80 PRINT"DE TOTALE LENGTE VAN DEZE
   TEKST IS";A+B+C;"TEKENS"
90 END
```

U moet er dus achter komen hoe lang een string is. Hiervoor is een functie in de CBM 64 ingebouwd die u direct de lengte van een string geeft, namelijk LEN(A$). U doet dat meestal in de vorm:
rn L=LEN(A$). In de variabele L wordt dan een getal opgeslagen gelijk aan het aantal tekens dat A$ lang is. Met deze getalvariabele kunt u dan weer gaan rekenen. Deze functie wordt onder andere gebruikt bij het controleren of een ingevoerde string niet te lang is of bij het centreren van teksten.

```
5 REM VOORBEELD 25
10 PRINT"■"
20 A$="CENTREREN"
30 PRINT:PRINT:PRINT
40 A=LEN(A$)
50 PRINTTAB((40-A)/2);A$
60 END
READY.
```

7/6.2

Strings samenvoegen

Omdat een string een reeks tekens is, zult u wel begrijpen dat u met strings niet kunt rekenen. Het is bijvoorbeeld niet mogelijk om uit te rekenen hoeveel A$*B$ is, logisch want APPELEN*PEREN is ondenkbaar. Pas hiermee op als u een string heeft waarin cijfers als tekens zijn gebruikt. Als na INPUTA$ ingevoerd zou zijn 12345 en u laat A$ afdrukken dan staat er 12345. Dit lijkt verdacht veel op het getal 12345. Maar voor de computer is het eerste voorbeeld een reeks tekens bestaande uit een, twee, drie, vier en vijf die niets te maken hebben met de getallen 1,2,3,4 en 5.

```
5 REM VOORBEELD 26
10 PRINT"␣"
20 A$="SCHOEN"
30 B$="HERSTELLER"
40 C$="WERKPLAATS"
50 PRINTA$+B$;"S";C$
60 END
```

Het enige dat u met strings mag doen is ze samenvoegen. De Engelse naam hiervoor is 'concatenating', dit woord zult u bij strings beslist tegenkomen. Voor dit samenvoegen gebruikt u het '+' teken uit de rekenkundige bewerkingen.

```
SCHOENHERSTELLERSWERKPLAATS
```

Figuur 7/6.2-1: Zo ziet het resultaat er uit

7/6.3

Strings splitsen

Het is heel goed denkbaar dat u in een string wilt zoeken naar een bepaald teken of naar een bepaalde reeks tekens. Ook hiervoor kent de Commodore 64 een paar stringfuncties.

Stelt u zich voor dat u de string A$="WISSELWACHTER 328" heeft. Als u een stuk uit deze string zou willen afzonderen, dan kunt u de string op drie manieren aanpakken.

1) U kunt een stuk uit de string halen door aan de linkerkant te beginnen. Het enige dat u dan nog aan de computer moet meedelen is het aantal tekens dat u wilt hebben. B$=LEFT$(A$,4) zorgt er voor dat B$ gelijk wordt aan „WISS" namelijk de eerste vier letters vanaf de linkerkant.

2) U kunt een stuk midden uit de string halen. U zult dan aan de computer moeten vertellen op welke positie u wilt beginnen en hoeveel tekens u wilt hebben. C$=MID$(A$,4,5) geeft een C$ gelijk aan „SELWA", namelijk 5 letters, gerekend vanaf positie 4.

3) U kunt een stuk uit de string halen, uitgaande van het rechter uiteinde van de string. U moet dan wel weer aan de computer mededelen hoeveel tekens u wilt hebben.
D$=RIGHT$(A$,6) zorgt in het bovenstaande voorbeeld dat D$ gelijk wordt aan „ER 328". U ziet dat een spatie ook een teken is dat deel kan uitmaken van een string. „ „ Is dus ook een toegestane string, namelijk bestaande uit een spatie. Ook „" komt voor; dit is een lege string of nulstring.
Eigenlijk heeft u de functies LEFT$ en RIGHT$ niet nodig. U kunt deze simuleren door MID$. U geeft dan als eerste positie waar begonnen moet worden het cijfer 1. In het bovenstaande voorbeeld geeft LEFT$(A$,3) en MID$(A$,1,3) hetzelfde resultaat. Geeft u in MID$ niet aan hoeveel tekens u uit de string afgezonderd wilt hebben, dan neemt MID$ automatisch de rest van de string vanaf de aangegeven positie. MID$(A$,12) geeft

```
5 REM VOORBEELD 27
10 A$="WISSELWACHTER 328"
20 B$=LEFT$(A$,4)
30 C$=MID$(A$,4,5)
40 D$=RIGHT$(A$,6)
50 PRINTB$
60 PRINTC$
70 PRINTD$

WISS
SELWA
ER 328
```

Figuur 7/6.3-1: Het resultaat

6.3 Strings splitsen

dus hetzelfde resultaat als RIGHT$(A$,6). Algemeen geldt als vervanging voor RIGHT$ de formule MID$(A$,LEN(A$)-A+1) waarin A het aantal tekens voorstelt dat u wilt hebben.

```
5 REM VOORBEELD 28
10 A$="WISSELWACHTER 328"
20 B$=MID$(A$,1,3)
30 C$=MID$(A$,4,5)
40 D$=MID$(A$,12)
50 PRINTB$
60 PRINTC$
70 PRINTD$
```

7/6.4

Zoeken in een string

De stringfuncties die u hiervoor heeft leren kennen zijn erg nuttig bij het zoeken in strings, zoals de volgende voorbeelden u tonen.

```
5 REM VOORBEELD 29
10 PRINT"␣"
20 A$=" DIT IS ZO MAAR EEN STUK
   JE UIT DE LOSSE VUIST
   GESCHREVEN PROZA"
30 L=LEN(A$)
40 FOR P=1 TO L-3
50 B$=MID$(A$,P,3)
60 IF B$="LOS"THEN PRINT" DE
   REEKS LETTERS LOS KOMT IN A$
   VOOR"
70 NEXT
80 END
```

```
5 REM VOORBEELD 30
10 PRINT"␣"
20 A$=" DIT IS ZO MAAR EEN STUK
   JE UIT DE LOSSE VUIST
   GESCHREVEN PROZA"
30 PRINTA$
40 L=LEN(A$)
50 FOR P=1 TO L
60 B$=MID$(A$,P,1)
70 IF B$="I"THEN T=T+1
80 NEXT
90 PRINT"IN BOVENSTAANDE TEKST
   KOMT ";T;"MAAL DE LETTER 'I'
   VOOR"
100 END

DIT IS ZO MAAR EEN STUKJE UIT
DE LOSSE
VUIST GESCHREVEN PROZA
IN BOVENSTAANDE TEKST KOMT  4
MAAL DE LETTER 'I' VOOR
```

Figuur 7/6.4-1: Het resultaat

7/6.5

Van strings naar getallen en terug

U kunt ook een getal omzetten in een string, hiervoor gebruikt u de functie STR$ (uitgesproken als stringstring, het '$' teken betekent immers 'string'). De juiste vorm is: STR$(<getal>)

```
5 REM VOORBEELD 31
10 PRINT"␣"
20 A=123
30 B=10
40 A$=STR$(A)
50 B$=STR$(B)
60 PRINT"NU ZIJN DE GETALLEN OM
   GEZET IN STRINGS"
70 REM VOOR DE CIJFERS STAAT
   EEN SPATIE VOOR HET TEKEN
80 PRINTA$;B$
90 PRINT"MAAR OOK DE GETALLEN
   ZIJN"
100 PRINT"NOG AANWEZIG"
110 REM VOOR EN ACHTER DE GETAL
    LEN STAAT EEN SPATIE
120 PRINTA;B
130 PRINT"A=";A;"B=";B;
    "A*B=";A*B
140 END
```

Omgekeerd kunt u ook een string omzetten in een getal via VAL(<string>). Als het eerste teken van een string geen + teken, - teken of cijfer is dan is het resultaat van deze functie de waarde 0. De omzetting stopt als er een ander teken dan de cijfers of de decimale punt wordt gevonden, of wanneer het einde van de string is bereikt.

```
5 REM VOORBEELD 32
10 PRINT"␣"
20 A$="123":B$="10":C$="A10"
30 A=VAL(A$):B=VAL(B$):C=VAL(C$)
40 PRINTA,B,C
50 PRINTA*B
60 PRINT"NU HADDEN WE WEER GETAL
   LEN"
70 END
```

In dit gedeelte heeft u kennis gemaakt met de stringfuncties, te weten:
LEN($) - geeft de lengte van de tussen de haakjes aangegeven string.
LEFT$($,getal) - geeft het aangegeven aantal tekens uit de gegeven string, te beginnen vanaf de linkerkant.
MID$($,getal,getal) - geeft het aantal tekens aangegeven in het tweede getal, te beginnen op de positie aangegeven door het eerste getal uit de gegeven string.
RIGHT$($,getal) - geeft het door getal aangegeven tekens uit de betreffende string, vanaf het rechter uiteinde van deze string.
STR$(getal) - zet het gegeven getal om in een string. U kunt met de omgezette vorm dan niet meer rekenen!
CHR$(getal) - zet het gegeven getal om in het teken waarvan de ASCII code gelijk is aan dit getal.
ASC($) - zet het eerste teken van de betreffende string om in de bijbehorende ASCII code. Oppassen als u met een nulstring te maken heeft.

6.5 Van strings naar getallen en terug

VAL($) - zet een string om in een getal. Pas op, er zijn beperkingen. U krijgt een 0 als het eerste teken van de string geen + of - teken of cijfer is.

voorts leerde u het begrip CONCATENATING kennen, ofwel het samenvoegen van twee of meer strings.

7/6.6

Van string naar ascii en terug

Van een string kunt u het eerste teken omzetten in het bij dit teken behorende codegetal met behulp van de functie ASC. De juiste vorm is ASC(<string>). Als u te maken heeft met een string waarin geen enkel teken aanwezig is, geeft bovenstaande functie een foutmelding nl. '?ILLEGAL QUANTITY ERROR', waarna uw programma afbreekt. Om dit te voorkomen kunt u voor de zekerheid elke string waarop u de ASC-functie loslaat verlengen met CHR$(0). (zie verder) Zo kan T=ASC(B$+CHR$(0)) nooit een foutmelding geven ook al is B$ gelijk aan """", een nulstring of lege string.

```
5 REM VOORBEELD 33
10 PRINT"␣"
20 A$="KOKOSNOOT"
30 B=ASC(A$)
40 PRINTB
50 PRINT"CONTROLEER MAAR IN BIJ
   LAGE 4"
60 PRINT"DE ASCII CODE VOOR 'K'
   IS:75"
70 END
```

U kunt echter ook een getal (mits tussen 0 en 255) omzetten in het teken dat daar volgens de ASCII code mee overeenkomt. De juiste vorm is:
CHR$(<getal>). Als het getal niet binnen het toegestane bereik ligt volgt weer de foutmelding '?ILLEGAL QUANTITY ERROR'.

```
5 REM VOORBEELD 34
10 PRINT"␣"
20 REM ASCII CODE VOOR A=65
30 REM ASCII CODE VOOR P=80
40 PRINTCHR$(65);CHR$(65);CHR$
   (80)
50 PRINT"U KUNT DIT OOK GEBRUI
   KEN"
60 PRINT"OM TEKST TUSSEN"
70 PRINT"AANHALINGSTEKENS"
80 PRINT"AF TE DRUKKEN"
90 PRINTCHR$(34);CHR$(65);
   CHR$(65);CHR$(80);CHR$(34)
100 END
```

7/7

Onderbreken, functies en subroutines

Inhoud

7/7.1

Onderbreken van een programma

U kunt een programma onderbreken door het statement:
rn STOP in het programma op te nemen. Op dit punt wordt het programma nderbroken en heeft u de gelegenheid om de waarde(s) van variabelen en strings e onderzoeken door deze in de 'direct mode' te laten afdrukken. Wilt u hierna e programmauitvoering voortzetten, dan kan dit door CONT (van continue - oortzetten) gevolgd door RETURN te geven. Heeft u tijdens de onderbreking eranderingen in het programma aangebracht, dan is voortzetting niet mogelijk en krijgt u de foutmelding 'CAN'T CONTINUE'.
SGN(<getal>) geeft een waarde afhankelijk van het teken van het getal. Bij een positief getal is de waarde van deze functie 1; als het getal gelijk is aan 0 is de waarde van de SGN-functie ook 0 en bij negatieve getallen is de waarde -1.

```
5 REM VOORBEELD 37
10 PRINT"␣"
20 FOR A=1TO3
30 FORB=-1TO-3STEP-1
40 PRINTA/B;TAB(15)SGN(A/B)*INT
   (ABS(A/B)+0.5)
50 NEXTB
60 NEXTA
70 END
```

De functie ABS(<getal>) geeft de absolute waarde van het getal. Willen we nu een negatief getal correct afronden dan gebruiken we SGN(X)*INT(ABS(X)+0.5).
Volgt u maar: X=-10.41, dus SGN(X)=-1; ABS(X)=10.41; INT(10.41+0.5)=10, dus de formule SGN(X)*INT(ABS(X)+0.5) geeft -1*10=-10. De afronding was correct.

Ook in een gewenst aantal decimalen kunt u met behulp van INT(X) gaan afronden. Wilt u in twee decimalen afronden (belangrijk bij boekhouden) dan gebruikt u:
SGN(X)*INT(100*ABS(X)+0.5)/100.

7/7.2

De INTEGER functie en haar gebruik bij afronden

INT(X) geeft het grootste gehele getal dat gelijk is aan of kleiner dan X. U moet niet denken dat dit rekenkundig afronden op gehele getallen is, want INT(-10.3) geeft als uitkomst -11, een rekenmachine zou u -10 geven omdat deze wel correct afrondt. U kunt deze functie echter wel gebruiken om af te ronden, maar dan moet u wel een paar kunstgrepen toepassen.

```
5 REM VOORBEELD 35
10 A=110.75:B=9.99:C=-12.45
20 PRINTINT(A),INT(B),INT(C)
30 END


110          9           -13
```

Figuur 7/7.2-1: Het resultaat

INT(10.45) geeft als uitkomst 10, maar INT(10.51) geeft ook 10. Als u bij beide getallen 0.5 op telt, krijgt u INT(10.45+0.5), dit geeft 10, maar u krijgt INT(10.51+0.5) en dit geeft 11. De afronding is nu wel correct.

```
5 REM VOORBEELD 36
10 FOR A=1 TO 3
20 FOR B=1 TO 3
30 PRINTA/B;TAB(15)INT(A/B),INT
   (A/B+0.5)
40 NEXTB
50 NEXTA
60 END

1               1       1
.5              0       1
.333333333      0       0
2               2       2
1               1       1
.666666667      0       1
3               3       3
1.5             1       2
1               1       1
```

Figuur 7/7.2-2: Ziet u wel!

Afronden bij negatieve getallen gaat ook, maar alweer met kunstgrepen. Hierbij gebruiken we weer twee in de 64 ingebouwde functies namelijk SGN(X) en ABS(X).

7/7.3

Willekeurige getallen door de computer bepaald

De functie RND(<getal>) geeft een willekeurig getal tussen 0 en 1. Dat dit getal echter niet geheel willekeurig is zal u niet verbazen, omdat bij het aanzetten van de computer alles telkens in dezelfde beginsituatie wordt gebracht. Bij het opstarten van de computer wordt een reeks 'willekeurige' getallen gegenereerd, dus het uitganspunt is aldoor hetzelfde..... tenminste bij positieve getallen. Als het argument negatief is, krijgt u ook een willekeurig getal, maar telkens hetzelfde. Deze wetenschap is heel leuk om bijvoorbeeld een dobbelsteen aldoor dezelfde waarde te laten werpen, u kunt vrienden en kennissen verbaasd doen staan om uw gave van helderziendheid. RND(0) geeft een getal tussen 0 en 1 dat voor ons echt willekeurig is, want het bepalen van dit getal gebeurt op basis van de stand van de JIFFY CLOCK. Dit is een ingebouwde klok die 60 maal per seconde met 1 jiffy wordt opgehoogd en begint te lopen als u de computer aan zet. (Zie hoofdstuk 4.)

U kunt RND(1) gebruiken om getallen te laten genereren die binnen een bepaald bereik vallen.
100*RND(1) geeft een willekeurig getal tussen 0 en 100.
INT(100*RND(1)) geeft een willekeurig geheel getal van en met 0 tot en met 99.
INT(6*RND(1))+1 geeft een willekeurig geheel getal van 1 t/m 6 (de dobbelsteen).

```
5 REM VOORBELD 38
10 PRINT"␣"
20 FOR A=1 TO 10
30 O=INT(6*RND(1))+1
40 T=T+O
50 NEXTA
60 PRINT"IN 10 WORPEN GOOIDE U
   ";T;"OGEN"
70 END
```

Voor een willekeurig getal met als ondergrens X en als bovengrens Y gebruikt u de formule:
A=RND(1)*(Y-X)+X. De waarde X kan bereikt worden, de waarde Y wordt nooit gehaald.
Allerlei educatieve programma's gebruiken de RaNDom functie, ook in allerlei spelen wordt RND gebruikt. Kortom zodra iets verrassend moet zijn heeft u RND beslist nodig.

```
5 REM VOORBEELD 39
10 PRINT"␣"
20 REM STERREN WORDEN
   WILLEKEURIG AFGEDRUKT
30 P=INT(40*RND(1))
40 PRINTTAB(P);"*"
60 GOTO 30
```

7/7.4

Subroutines

Subroutines zijn uitgevonden om veel extra schrijfwerk te besparen. Het kan zijn dat u een bepaald gedeelte van een programma meerdere malen wilt laten uitvoeren. Als u voor elke keer dit programmastuk opnieuw moest schrijven zou dit uiterst onpraktisch zijn. Het is mogelijk om zo'n programmablok als het ware uit het programma te lichten - waarna we spreken van subroutine - en vanuit het programma naar deze subroutine toe te springen. Na het uitvoeren van de subroutine keert de computer terug naar het hoofdprogramma om dit verder uit te voeren.

U moet dus aan de computer vertellen waar de subroutine te vinden is en dit doet u met de opdracht:
rn GOSUB<regelnummer>. Bij deze instructie springt de computer naar het aangegeven regelnummer, zoals bij GOTO, maar onthoudt tevens het regelnummer waar de GOSUB opdracht stond.

U moet de computer ook mededelen wanneer de SUBROUTINE doorlopen is en teruggegaan moet worden naar het hoofdprogramma. U doet dit door als laatste instructie van de subroutine op te nemen:
rn RETURN
Zodra de computer deze instructie tegenkomt gaat de computer terug naar het hoofdprogramma en zet dit voort met de opdracht volgend op de GOSUB opdracht.

Net als bij de FOR..NEXT opdracht worden de gegevens die de computer moet onthouden bewaard op de STACK (zie hoofdstuk 9). Net zoals u met geneste FOR.. NEXT lussen kunt werken, kunt u met geneste subroutines werken, u springt dan vanuit de eerste subroutine naar de tweede, terug naar de eerste etcetera.

```
5 REM VOORBEELD 40
6 PRINT"□"
10 PRINT"GOEDENDAG"
20 GOSUB1000
30 INPUT"HOE HEET U";N$
40 GOSUB1000
45 GOSUB2000
50 PRINT"GOEDENDAG ";N$
55 PRINT:PRINT
60 PRINT"SUBROUTINE 1: ";A;"MAAL
   DOORLOPEN"
70 PRINT"SUBROUTINE 2: ";B;"MAAL
   DOORLOPEN"
80 PRINT"SUBROUTINE 3: ";C;"MAAL
   DOORLOPEN"
90 END
1000 PRINT"■WE ZIJN NU IN DE
     EERSTE SUBROUTINE":A=A+1
1010 FORT=1TO1500:NEXTT
1020 RETURN
2000 PRINT"■WE ZIJN NU IN DE
     TWEEDE SUBROUTINE":B=B+1
2005 GOSUB1000
2010 GOSUB2500
```

7.4 Subroutines

```
2020 RETURN
2500 PRINT"■WE ZIJN NU IN DE
     DERDE SUBROUTINE VANUIT DE
     TWEEDE":C=C+1
2510 GOSUB1000
2520 RETURN
```

Waar moet u de subroutines plaatsen, na het hoofdprogramma of er voor? Beiden is toegestaan, maar de programma uitvoering is sneller als de subroutines voor het hoofdprogramma staan. U dient dan voor het blok met subroutines een GOTO opdracht op te nemen zodat over het gehele blok subroutines heen gesprongen wordt en het hoofdprogramma gestart wordt.

Indien u de subroutines na het hoofdprogramma plaatst, dan moet u er zorg voor dragen dat het hoofdprogramma eindigt met END, want anders gaat de computer rustig door en komt een opdracht RETURN tegen zonder dat daar een bijbehorende GOSUB opdracht voor was. Resultaat: de foutmelding 'RETURN WITHOUT GOSUB'

```
5 REM VOORBEELD 41
6 PRINT"■"
10 PRINT"WE VERGETEND 'END' TE
   GEBRUIKEN"
20 GOSUB 1000
30 PRINT" U GOOIDE ";A;"OGEN"
1000 A=INT(6*RND(1)+1)
1010 RETURN
```

```
WE VERGETEND 'END' TE GEBRUIKEN
 U GOOIDE  2 OGEN

?RETURN WITHOUT GOSUB  ERROR IN 1010
READY.
```

Figuur 7/7.4-3: Het resultaat is 'einde'

Een gemakkelijk te volgen programma

```
5 REM VOORBEELD 42
10 PRINT"■"
20 T=T+1
30 IF T=6 THEN END
40 A=INT(20*RND(1))
50 B=INT(20*RND(1))
60 PRINT" HET EERSTE GETAL
   IS ";A
70 GOSUB 1000
80 PRINT" HET TWEEDE GETAL
   IS ";B
90 GOSUB 1000
100 PRINT" DE SOM IS ";A+B
110 GOSUB 2000
120 GOTO20
1000 FOR V=1 TO 1000:NEXT
1010 RETURN
2000 PRINT" DE LUS WERD ";T;"MAAL
     DOORLOPEN"
2010 RETURN
```

Toelichting: In dit korte programma worden 6 maal twee willekeurige getallen bepaald en telkens wordt de som van twee getallen uitgerekend en afgedrukt, door een in een subroutine opgenomen vertragingslus wordt alles gemakkelijker te volgen gemaakt.

7/7.5

Alweer een mogelijkheid om te vertakken

Als u zich voorstelt dat u een gecompliceerd programma aan het ontwikkelen bent, waarin diverse duidelijk van elkaar te onderscheiden gedeelten moeten zitten; hoe lost u dat dan op?
U kunt hierbij bijvoorbeeld denken aan een educatief programma waarin aparte stukken zitten voor het trainen van optellen, aftrekken, vermenigvuldigen en delen. Een ander voorbeeld is een bestandsprogramma waar u de keuze moet hebben tussen invoeren, wijzigen, bekijken, verwijderen en printen van records.

U heeft hier gelukkig een stuk gereedschap voor in handen. U kunt een kort hoofdprogramma schrijven, terwijl u de diverse onderdelen als subroutines schrijft.
Door gebruik te maken van de instructie
ON<variabele>(GOSUB/GO-
TO)<rn>[,<rn>].....
kunt u afhankelijk van de waarde van de variabele naar de door u gewenste regel springen. Als de variabele gelijk is aan 1 wordt naar het eerste regelnummer gesprongen, als de variabele gelijk is aan 2 naar de tweede mogelijkheid enzovoorts.

U moet wel de waarde van de variabele in de hand houden, als u zes mogelijkheden heeft, dan is het weinig zinnig om een groter getal dan 6 in te voeren; als de variabele negatief is slikt het programma dit ook niet en stopt met een foutmelding. U mag wel gebroken getallen invoeren, want de computer werkt verder met de INTEGER van het ingevoerde getal. Als u 0 invoert of een getal dat groter is dan het aantal mogelijkheden +1 (in verband met de INTEGER functie) dan gaat het programma verder met de volgende regel. In plaats van een variabele mag tussen ON en GOSUB (of GOTO) ook een bewerking staan, maar de uitkomst hiervan moet ook aan de bovengenoemde voorwaarden voldoen.

We noemen zo'n reeks keuzemogelijkheden een MENU. Een menu van een adresprogramma zou er als volgt kunnen uitzien:

ADRESLIJST
1. Invoeren
2. Wijzigen
3. Verwijderen
4. Sorteren
5. Zoeken
6. Afdrukken
7. Stoppen
MAAK NU UW KEUZE

Het is heel goed denkbaar dat mogelijkheid 4 u naar een SUBMENU brengt dat er als volgt zou kunnen uitzien:

7.5 Alweer een mogelijkheid om te vertakken

SORTEREN
1. Op naam
2. Op woonplaats
3. Op postcode
MAAK NU UW KEUZE

Deze mogelijkheid van selecteren van de juiste routine is een zeer sterk stuk programmeergereedschap waar u erg veel plezier van zult hebben.

```
1 REM 7.7.5.1
5 REM VOORBEELD 43
10 PRINT"␣":PRINT:PRINT
20 PRINT"WAT KIES JE?"
30 PRINT"1. OPTELLEN"
40 PRINT"2. AFTREKKEN"
50 PRINT"3. VERMENIGVULDIGEN"
60 PRINT"4. DELEN"
70 PRINT"5. STOPPEN"
90 INPUT"TYP JE KEUZE IN";Y
100 ONYGOTO500,1000,1500,2000,
    2500
110 GOTO10
500 REM OPTELLEN
510 A=INT(100*RND(1)):
    B=INT(100*RND(1))
530 PRINTA;"+";B;"=":GOSUB2200
550 IFA+B=CTHENGOSUB2100
560 IFA+B<>CTHENGOSUB2120
570 INPUT"DOORGAAN MET OPTELLEN?
    JA/NEE";A$:IFA$="JA"THEN510
590 GOTO10
1000 PRINT"MODULE AFTREKKEN":
     FORT=1TO1000:NEXT
1090 GOTO10
1500 PRINT"MODULE VERMENIGVUL-
     DIGEN":FORT=1TO1000:NEXT
1590 GOTO10
2000 REM DELEN
2010 A=INT(100*RND(1)):
     B=INT(100*RND(1))
2021 IFB=0THEN2010
2026 C=INT(A/B):IFC=1THEN2010
2028 A=B*C:PRINTA;"/";
     B"=":GOSUB2200
2050 IFA/B=CTHENGOSUB2100
2060 IFA/B<>CTHENGOSUB2120
2070 INPUT"DOORGAAN MET DELEN?
     JA/NEE";A$:IFA$="JA"THEN2010
2090 GOTO10
2100 PRINT"GOED":RETURN
2120 PRINT"FOUT":RETURN
2200 INPUT"GEEF JE ANTWOORD";
     C:RETURN
2500 END

READY.
```

Dit programma geeft u allereerst een keuzemenu en u heeft gelegenheid uw keuze te maken. Afhankelijk hiervan gaat het programma met een van de vier modules verder. De ontbrekende modules kunt u zelf toevoegen. Telkens na het oplossen van een sommetje krijgt u gelegenheid om aan te geven of u door wilt gaan met dit soort sommen of dat u naar het menu terug wilt gaan. Door met geneste subroutines te werken kan het programma korter worden gemaakt. U kunt er ook nog tellers inbouwen voor het aantal goede antwoorden, de percentages enzovoorts.

U heeft in dit hoofdstuk kennis gemaakt met de volgende BASIC keywords:
GOSUB<regelnummer> - laat uw programma naar het aangegeven regelnummer springen.
RETURN - stuurt het programma terug naar de regel volgend op die waarin

7.5 Alweer een mogelijkheid om te vertakken

GOSUB voorkwam.
ON in de vorm:
ON A GOSUB (OF GOTO) lijst van mogelijke regelnummers gescheiden door komma's. De waarde van A bepaalt het regelnummer waar naar toe gesprongen wordt.
STOP - om een programma te onderbreken.
CONT - om een programma dat onderbroken is, weer voort te zetten.

Verder de functies:
INTEGER(argument) - geeft het dichtstbijzijnde gehele getal dat kleiner is dan het argument.
SGN(argument) - geeft -1,0 of +1 al naar het argument negatief, nul of positief is.
ABS(argument) - geeft de absolute waarde van het argument, dus het argument zonder eventueel '-' teken.
RND(argument) - om willekeurige getallen te laten genereren.

Tenslotte heeft u het begrip MENU leren kennen en is er iets verteld over de JIFFY klok, een teller die 60 maal per seconde wordt opgehoogd.

7/8

Over indexen en vaste gegevens

Inhoud

7/8.1

Variabelen die verband met elkaar houden, lijsten en tabellen

Als u bijvoorbeeld een ledenlijst van een vereniging van 1000 leden wilt gaan samenstellen en u stopt de namen in stringvariabelen, dan wordt Uw fantasie betreffende het verzinnen van namen voor die variabelen zwaar op de proef gesteld. Er zou dus een andere mogelijkheid moeten zijn. Voor dit soort problemen is er de zogenaamde geïndiceerde variabele, oftewel een variabele met een volgnummer of index. Dit nummer is gebonden aan een maximum, namelijk 32767 en het mag niet negatief zijn. Het nummer mag ook vervangen worden door een variabele, u moet er dan wel zorg voor dragen dat de waarde hiervan voldoet aan de voorgaande eisen.
Een gebroken positief getal is wel toegestaan, de computer zet dit om in de bijbehorende INTEGER.

Waarschijnlijk interesseert het u wel waarom deze voorzieningen nodig zijn. Stelt u zich voor dat u voor uw gezin bioscoopkaartjes gaat halen. Er worden stoelen op een rij voor u gereserveerd en voor de bioscoopdirecteur is het totaal niet belangrijk hoe uw familie gaat zitten. Dat de zaak uitverkocht is als u later kaartjes voor de buren wilt bijbestellen zal u niet verbazen.

Iets dergelijks gebeurt bij de computer, voor een array (lijst of tabel van geïndiceerde variabelen) wordt geheugenruimte als een aaneengesloten stuk gereserveerd. Voor 11 elementen in een array heeft de computer ruimte, maar u moet als u meer dan 11 (0 t/m 10) elementen in uw array heeft van te voren via de DIM (van dimension- afmeting) instructie opgeven hoe groot het stuk geheugen moet zijn dat gereserveerd moet worden. Dit dimensioneren mag u maar één maal doen, logisch, omdat reserveringen niet zo maar kunnen worden uitgebreid.

```
5 REM VOORBEELD 44
10 DIM A(4)
20 REM U GAAT DE LIJST VULLEN
   MET EEN FOR NEXT LUS
30 FOR X=1 TO 10
40 INPUT A(X)
50 NEXTX
60 REM ZODRA DE INDEX GROTER
   WORDT DAN 4 VOLGT EEN FOUT
   MELDING
```

Toegestane vormen van geïndiceerde variabelen en het DIM statement zijn:
VARIABELE('VOLGNUMMER-VARIABELE(N)')
STRINGVARIABELE('VOLGNUM-

8.1 Variabelen die verband met elkaar houden, lijsten en tabellen

MERVARIABELE(N)')

Als u een meervoudige index gebruikt,dan moeten de indexen gescheiden worden door komma's.
DIM <VARIABELE>(<INDEXEN->)[,<VARIABELE>(<IN-DEXEN>)....].

U ziet dat u met meer indexen kunt werken (theoretisch maximaal 255, maar dit is bijna niet voor te stellen). Een tweevoudige index is gemakkelijk voor te stellen als een tabel (of matrix) waarbij de eerste index de kolom in de tabel en de tweede index de rij in de tabel aangeeft. Denk bijvoorbeeld aan vergelijken van prijzen van artikelen van diverse leveranciers.

Geïndiceerde variabelen besparen u massa's schrijfwerk bij het programmeren, kijkt u maar naar de volgende voorbeelden.

```
5 REM VOORBEELD 45
10 PRINT"␣ TYPE VIJFTIEN
   GETALLEN IN"
20 DIMA(15)
30 FORN=0TO15
40 PRINTN;:INPUTA(N)
50 NEXTN
60 PRINT"␣ DAAR KOMEN DE
   GETALLEN:"
70 GOSUB1000
80 FORK=0TO15
90 PRINT"A("K")=",A(K):GOSUB1000
100 NEXTK
110 GOSUB 2000
120 END
1000 FORT=1TO500:NEXTT:RETURN
2000 PRINT"BIJ EEN GROOTSTE
     INDEX VAN 15 HEEFT U 16
     GETALLEN!"
2010 PRINT"WANT INDEX 0 TELT
     OOK MEE"
2020 RETURN
```

```
5 REM VOORBEELD 46
10 PRINT"␣ TYPE VIJF WOORDEN IN"
20 FORN=0TO4
30 PRINTN;:INPUTX$
40 A$(N)=X$
50 NEXTN
60 PRINT"␣DAAR KOMEN DE WOORDEN"
70 FORK=0TO4
80 PRINT"A$("K")="A$(K):GOSUB1000
90 NEXTK
100 END
110 NEXTK
120 END
1000 FORT=1TO500:NEXTT:RETURN
```

Nog een paar tips:

DIMENSIONEER altijd, zelfs bij een index kleiner dan 10. De anders door de computer gereserveerde geheugenruimte voor 11 waardes komt dan, gedeeltelijk, vrij voor andere doeleinden.

Gebruik voor uw INDEXEN 'INTEGERVARIABELEN', want dit spaart u bij grote aantallen gegevens een gigantische hoeveelheid geheugenruimte. Een floating point index neemt veel meer geheugenplaats in beslag dan een integer index.

ARRAYS kunnen we vullen met behulp van INPUT, zoals u al zag en met de nog te behandelen statements READ, DATA en GET.

7/8.2

Nog meer gegevens in het programma

We hebben tot nu toe diverse methoden leren kennen om gegevens aan het programma mee te geven. De eerste was door het domweg toekennen van waarden aan variabelen via programmaregels als:
100 A=10:A$="PIET"
De tweede methode was via de instructie INPUT in de vorm:
100 INPUT"Geef Uw naam en leeftijd";N$,L

Het komt echter dikwijls voor dat we een grote hoeveelheid zelden of nooit veranderende gegevens in een programma hebben, zoals de namen van de leerlingen van een klas, het BTW-percentage, de maanden van het jaar enzovoorts.
Het is mogelijk om al deze gegevens, DATA, te concentreren in diverse zogenaamde 'DATA' regels, waarbij de afzonderlijke gegevens gescheiden worden door komma's. Via de instructie READ kunt u deze gegevens een voor een laten inlezen en toekennen aan de variabele achter READ. De verschillende DATA items hoeven niet tussen aanhalingstekens te worden geplaatst, tenzij ....

U zult het al begrijpen, de komma is het scheidingsteken tussen twee DATA items, zodra u dus een gegeven heeft dat een ',' bevat moet u dit gegeven wel tussen aanhalingstekens plaatsen. Ook krijgt u problemen met de dubbele punt, want dit is normaal de scheiding tussen twee instructies op een regel. Ook dit kunt u omzeilen door, indien uw gegeven een dubbele punt bevat, aanhalingstekens te gebruiken. Hetzelfde geldt voor spaties, letters aangeslagen met SHIFT, grafische tekens en controletekens. Uiteraard geven de aanhalingstekens zelf als onderdeel van de DATA ook een probleem, u kunt dit omzeilen door CHR$(34) te gebruiken, zoals in het voorbeeld uit 7.6.6.

De syntax van deze instructie:
READ<variabele>[,<variabele >]...
DATA<lijst van constanten>

De verwerking gaat volledig op volgorde, het eerste READ statement leest het eerste gegeven uit de eerste DATA regel in, enzovoorts. Voor twee dingen moet u oppassen, ten eerste moet het type gegeven dat aan de beurt is om ingelezen te worden overeenstemmen met de soort variabele die achter READ staat, want anders krijgt u een '?SYNTAX ERROR'; ten tweede moet u zorgen dat er voldoende gegevens in de DATA regels staan, want anders krijgt u de foutmelding '?OUT OF DATA'. U bent daar waarschijnlijk al mee geconfronteerd als u het woord READY als instructie heeft willen laten uitvoeren, het wordt dan opgevat als READY.

8.2 Nog meer gegevens in het programma

U bent geheel vrij in de plaats van uw DATA regels, meestal worden ze achterin het programma bij elkaar gezet, maar voor de computer maakt het niets uit of u ze voorin het programma concentreert of door het hele programma verspreidt. Bij elkaar houden is het makkelijkste als u een enkele keer gegevens moet gaan wijzigen.

U kunt de gegevens vaker dan een maal gebruiken door de instructie RESTORE in uw programma op te nemen, dan wordt weer begonnen met het inlezen van het allereerste gegeven. RESTORE is dus een 'alles of niets' instructie, u kunt niet opnieuw gaan inlezen; bij de vijfde DATA regel bijvoorbeeld, maar u gaat terug naar het eerste gegeven van de eerste DATA regel.

```
5 REM VOORBEELD 47
10 T%=5
11 GOSUB 1000
20 DIMA$(T%),D$(T%),P$(T%)
30 FOR A=1 TO 5
40 READ A$(A),D$(A),P$(A)
50 NEXT
60 PRINT"␣ INVENTARISLIJST"
70 PRINT"ARTIKEL",,"AANKOOP","PRIJS"
80 FOR A=1 TO 5
90 PRINTA$(A);TAB(20)D$(A),P$(A)
100 NEXT
110 DATA"KAST,HOUT","10-6-1980",1450.00,4 STOELEN,"10-6-1980","
    600.00"
120 DATA"SLAAPKAMER,STAAL",17-6-1980,1500.00
130 DATA"RADIO,DRAAGBAAR",17-6-1980," 650.00"
140 DATA"BURO,STAAL",17-6-1980," 965.00"
150 END
1000 PRINT"␣DIT IS EEN BEGIN VAN EEN INVENTAISLIJST":GOSUB2000
1040 PRINT"DE VOLGORDE DE DATAREGELS IS ARTIKEL,AANKOOPDATUM EN
     PRIJS":GOSUB2000
1030 PRINT"MAAR U MOET DAN WEL UW DIMENSIONERING AANPASSEN":
     GOSUB2000
1020 PRINT"DE GEGEVENS ZITTEN IN DATAREGELS DIE U KUNT UITBREIDEN"
     GOSUB2000
1010 PRINT"ER ZIJN 3 KOLOMMEN, ARTIKELEN, AANKOOPDATA EN PRIJZEN":
     GOSUB2000
1050 PRINT"ALS ER IN HET ARTIKEL EEN ',' VOORKOMT MOET HET GEHEEL
     TUSSEN "
1060 PRINT"AANHALINGSTEKENS":GOSUB2000
1070 PRINT" OM DE PRIJZEN ONDER ELKAAR TE KRIJGEN SPATIES GEBRUIKEN
    ,":GOSUB2000
1080 PRINT"MAAR DAN OOK AANHALINGSTEKENS":GOSUB2000
1090 RETURN
1100 RETURN
2000 FOR T=1TO5000:NEXT
2010 RETURN
```

8.2 Nog meer gegevens in het programma

In dit hoofdstuk heeft u kennis gemaakt met de begrippen:
ARRAY - een lijst of tabel van variabelen of stringvariabelen die dezelfde naam dragen, maar alleen een verschillend identificatienummer hebben.
INDEX - het identificatienummer van een geïndiceerde variabele of stringvariabele.

En met de BASIC instructies:
READ<variabelenaam> - die een gegeven uit een DATA regel leest.
DATA<lijst gegevens> - waarin de gegevens voor de READ statements worden opgeslagen.
DIM(<INDEX(en)> - waarmee u aan de computer moet meedelen wat de maximale indexen van Uw gedimensioneerde variabelen worden.

```
5 REM VOORBEELD 48
10 DIMA$(3,3):PRINT"␣":B$="  "
15 REM HET TWEEDIMENSIONALE ARRAY WORDT GEVULD MET BEHULP VAN TWEE
   GENESTE
16 REM LUSSEN,EERST DE EERSTE RIJ ETC.
17 REM DE GEGEVENS STAAN IN DATAREGELS
20 FORK=0TO3
30 FORR=0TO3
40 READA$(K,R)
50 NEXTR
60 NEXTK:GOSUB2000
65 REM OP DEZELFDE MANIER WORDT HET ARRAY AFGEDRUKT
66 REM NA HET AFDRUKKEN VAN 4 KOLOMMEN OP EEN RIJ VOLGT EEN
   PRINTOPDRACHT
67 REM EN WORDT DE VOLGENDE RIJ AFGEDRUKT
70 FORL=0TO3
80 FORS=0TO3
90 PRINTB$A$(L,S);:GOSUB2000
100 NEXTS
110 PRINT:PRINT
120 NEXTL
130 END
1000 DATA00,01,02,03,10,11,12,13,20,21,22,23,30,31,32,33
2000 FORT=1TO500:NEXTT:RETURN
```

8.2 Nog meer gegevens in het programma

```
5 REM VOORBEELD 49
10 REM VB 2-DIMENSIONELE ARRAY
20 DIMA$(5,3):REM 5 MENSEN MET ELK DRIE GEGEVENS
30 FOR I=1TO5
40 FOR J=1TO3
50 READ A$(I,J):REM INLEZEN GEGEVENS
60 NEXTJ
70 NEXTI
80 REM UITPRINTEN
90 FOR I=1TO5
95 PRINTI
100 PRINTI
110 FOR J=1TO3
120 PRINTA$(I,J)
130 NEXTJ
140 PRINT
145 REM GEBRUIK VOOR EEN NETTERE UITVOER BV DE TAB-FUNCTIE
150 NEXTI
160 REM GEBRUIK VOOR EEN NETTERE UITVOER BV DE TAB-FUNCTIE
170 DATA ZEGEL,POSTSTRAAT 12,GIRODORP
180 DATA VAN  BLAD,BOEKSTRAAT 2,KAFTSTAD
190 DATA SPIJKER,HAMERRING 12,GEREEDSCHAPSSTAD
200 DATA BASIC,STUDIELAAN 76,PIEKERDORP
210 REM IN 220 UW EIGEN GEGEVENS INVULLEN
220 DATA UW NAAM,UW ADRES,UW WOONPLAATS
```

7/9

Nog meer gegevensinvoer

Inhoud

7/9.1

Gegevens halen

Uw programma's hebben gegevens nodig om te kunnen werken. U maakte al kennis met het commando INPUT voor invoer van variabele gegevens en met de READ-DATA constructie voor het invoeren van weinig variërende gegevens. Er is nog een derde mogelijkheid, namelijk via het commando GET. Ook deze instructie heeft zijn voor- en nadelen, zoals u zult zien.

Voordat we hierop kunnen ingaan moeten we het eerst even hebben over de toetsenbordbuffer. Dit is een opslagplaats waar toetsaanslagen tijdelijk worden opgeborgen alvorens ze worden verwerkt. Zo'n 60 maal per seconde wordt het toetsenbord afgezocht naar een aangeslagen toets en deze wordt in de buffer opgeslagen alvorens verder te worden verwerkt. Maximaal kan de toetsenbordbuffer 10 aanslagen bevatten. De instructie GET haalt de toetsaanslagen een voor een uit de buffer (volgens het systeem first in first out - het teken dat het eerst werd opgeslagen, gaat er het eerste uit). Hierbij is het niet noodzakelijk om de RETURN-toets in te drukken.

Probeert u maar eens:

```
5 REM VOORBEELD 50
10 PRINT"DRUK SNEL EEN PAAR
   TOETSEN IN"
20 FOR T=1TO5000
30 NEXT
40 PRINT"DAAR KOMEN ZE"
50 FORK=1TO10
60 GETA$
70 PRINTA$
80 NEXTK
90 END
```

U kunt zowel GET gebruiken met getalvariabelen als met stringvariabelen. Omdat er nu eenmaal maar 10 cijfertoetsen zijn en veel meer andere, waaronder ook besturingstoetsen, is het gebruik van strings veel handiger. U moet dan echter wel selecteren.

9.1 Gegevens halen

Omdat het onmogelijk is om 60 tekens per seconde te tikken, moeten we iets doen om de zaak te vertragen. Daarom gebruiken we heel dikwijls regels van het volgende type:
100 GET A$:IF A$=""THEN 100.
Hier kijkt de computer in de toetsenbordbuffer, als A$ een nulstring is en er dus niets in de buffer staat, gaat de computer terug naar het begin van de programmaregel. Een lus waarin het programma blijft wachten totdat een toets wordt aangeslagen. U kunt GET dan ook gebruiken om een programma te onderbreken tot een toets wordt aangeslagen.

```
5 REM VOORBEELD 51
10 PRINT" DRUK EEN TOETS IN"
15 REM PROBEER DIT OOK MET DE
   BESTURINGSTOETSEN,
   DAAR REAGEERT GET OOK OP
16 REM ALTIJD GET BEVEILIGEN!
20 GET A$
30 IF A$=""THEN 20
40 PRINT"TOETS ";A$;" WAS
   INGEDRUKT"
50 PRINT" DRUK NOG EEN TOETS IN"
60 GET A$
70 IF A$=""THEN 60
80 PRINT"TOETS ";A$;" WAS
   INGEDRUKT"
90 END
```

U moet het geheel echter wel uitgebreid gaan beveiligen. Als u bijvoorbeeld een codewoord van 3 cijfers wilt gaan invoeren en U gebruikt in uw achteloosheid de deletetoets of de cursortoetsen, dan worden deze ook als aanslagen gezien en zit in de fout. U moet dus alle ongewenste toetsen uitsluiten. Dit kunt u vrij eenvoudig bereiken, want elke toets heeft immers zijn eigen ASCII code. Met IF.....THEN kunt u dan ongewenste toetsen uitsluiten. Hierbij is goed nadenken noodzakelijk, want een verkeerde toetsaanslag moet u wel kunnen corrigeren en u moet via RETURN ook kunnen aangeven dat u klaar bent met invoeren.

```
5 REM VOORBEELD 52
10 PRINT"IK WACHT ME SUF":PRINT:
   PRINT:PRINT
20 PRINT"DRUK OP V OM DOOR TE
   GAAN"
30 GET A$:IF A$="" THEN 30
40 IF A$="V" THEN 60
50 GOTO 10
60 PRINT"HE,HE, EINDELIJK WERD
   ER OP V GEDRUKT"
70 END
```

Via GET kunt u ook de keuzes bij menu's binnenhalen en dan het programma laten springen of vertakken met behulp van een IF/THEN constructie of met een ON A GOTO (of GOSUB) constructie. Als u met GETG$ werkt moet u bij ON G GOTO eerst G$ via VAL(G$) in een getal omzetten.

9.1 Gegevens halen

```
5 REM VOORBEELD 53
10 PRINT"DRUK OP 1,2 OF 3"
20 GETA$:IFA$=""THEN20
30 IFA$="1"THEN70
40 IFA$="2"THEN90
50 IFA$="3"THEN110
60 GOTO10
70 PRINT"EERSTE MOGELIJKHEID"
80 GOTO10
90 PRINT"TWEEDE MOGELIJKHEID"
100 GOTO10
110 PRINT"DERDE MOGELIJKHEID"
120 GOTO10
130 END
```

```
5 REM VOORBEELD 54
10 PRINT"■"
20 PRINT"1. OPTELLEN"
30 PRINT"2. DELEN"
40 PRINT"3. VERMENIGVULDIGEN"
50 PRINT"4. DELEN"
60 PRINT"5. STOPPEN"
70 PRINT:PRINT
80 PRINT"MAAK UW KEUZE"
90 GET A$:IFA$=""THEN90
100 A=VAL(A$)
110 IFA<1THEN90
120 IFA>=6THEN90
130 ONAGOSUB1000,2000,3000,
    4000,5000
140 FOR T=1TO2000:NEXT
150 GOTO10
1000 PRINT"GESPRONGEN NAAR
     SUBROUTINE OPTELLEN"
1010 RETURN
2000 PRINT"GESPRONGEN NAAR
     SUBROUTINE AFTREKKEN"
2010 RETURN
3000 PRINT"GESPRONGEN NAAR
     SUBROUTINE
     VERMENIGVULDIGEN"
3010 RETURN
4000 PRINT"GESPRONGEN NAAR
     SUBROUTINE DELEN"
4010 RETURN
5000 END
```

9.1 Gegevens halen

```
5 REM VOORBEELD 55
10 PRINT"1 VIA INPUT"
20 PRINT"2 VIA GET"
25 PRINT"3 STOPPEN"
30 PRINT"MAAK UW KEUZE"
40 GETA$:IFA$=""THEN40
50 A=VAL(A$)
60 ONAGOTO100,200,300
100 PRINT"PROBEER EEN STRING LANGER DAN 80 TEKENS IN TE VOEREN"
105 PRINT"VERGEET BIJ INPUT NIET OP RETURN TE DRUKKEN"
110 INPUT A$
120 PRINTA$
130 GOTO10
200 PRINT"NU KUNT U WEL MEER DAN 80 TEKENS INVOEREN"
210 GETA$:IFA$=""THEN210
220 B$=B$+A$
230 L=LEN(B$)
240 IFL=120THENPRINTB$:GOTO10
250 GOTO210
300 END
```

Een van de voordelen van GET boven INPUT is dat met INPUT maximaal 80 tekens ingevoerd kunnen worden, terwijl met GET teken voor teken wordt binnengehaald, een string kan dus opgebouwd worden tot de maximaal toegestane lengte van 255 tekens.

U heeft in dit gedeelte kennis gemaakt met de toetsenbordbuffer en met het BASIC commando:
GET - dat 1 teken uit de toetsenbordbuffer haalt.

7/10

De ingebouwde klok

Inhoud

7/10.1

Timing en registratie

U heeft eerder in dit boek al kennis gemaakt met de JIFFY CLOCK die 60 maal per seconde opgehoogd wordt. De stand van de JIFFY CLOCK wordt bijgehouden in de gereserveerde variabele TI en u kunt bijvoorbeeeld in de linker bovenhoek deze stand continu in beeld laten blijven.

```
5 REM VOORBEELD 56
10 PRINT"␣"
20 T=TI/60
30 PRINT"■"INT(T):GOTO20
```

U kunt ook kijken hoe lang uw programma draait door aan het begin en het einde van het programma de stand van TI op te nemen. Als u het verschil door 60 deelt heeft u de programmaduur in seconden.

```
5 REM VOORBEELD 57
10 PRINT"␣"
20 A=TI
30 B=(TI-A)/60
40 PRINT"■IK WERK NU"INT(B)
   "SECONDEN"
50 GOTO30
```

U kunt zo de snelheid van uitvoering van verschillende versies van uw programma's vergelijken.
Via de nu volgende SUBROUTINE kunt u ook uw programma een exact te bepalen aantal seconden vertragen.

```
5 REM VOORBEELD 58
500 S=30:GOSUB 900
600 PRINT"HET PROGRAMMA
    ONDERBRAK";S;"SECONDEN"
700 END
900 T=TI+60*S
910 IF T=TI THEN RETURN
920 GOTO 910
```

Naast TI kent uw computer ook nog TI$ en deze variabele geeft de tijd weer die de computer aanstaat in uren, minuten en seconden. Deze zijn niet door leestekens gescheiden. In tegenstelling tot TI is TI$ wel te beïnvloeden, deze kunt u namelijk gelijk zetten met uw horloge.

```
5 REM VOORBEELD 59
10 INPUT"␣HOE LAAT IS HET";
   TI$:PRINT"␣"
20 PRINT"■";TI$:GOTO20
```

10.1 Timing en registratie

U heeft nu kennis gemaakt met de gereserveerde variabelen:
TI - die de stand van de JIFFY CLOCK bevat en door u niet is te beinvloeden (1 seconde is 60 „jiffies").

TI$ - die de tijd in uren, minuten en seconden aangeeft die verstreken is na het inschakelen van de computer. Deze TI$ kunt U eventueel wel veranderen, bijvoorbeeld gelijk zetten met uw horloge.

7/11

Wiskunde, wiskundige functies en Commodore 64 BASIC

7/11.1

Goniometrische functies

Een paar van de wiskundige functies die in uw 64 ingebouwd zijn, hebben we al leren kennen, namelijk SGN, ABS en INT.
De Commodore 64 kan echter veel meer doen, er zijn een heel stel goniometrische functies ingebouwd. Voordat ik die ga behandelen moeten er eerst een paar begrippen duidelijk gemaakt worden.

Allereerst, wat is goniometrie? In de wiskunde wordt met goniometrie de leer der goniometrische of hoekverhoudingen bedoeld. U ziet het woord hoekverhoudingen, de vraag is hoe geven we een hoek aan?

Normaal geven we de grootte van een hoek aan in graden. Een volledige cirkel bestaat uit 360 graden. Zoals gewoonlijk is uw computer weer eigenwijs, hij kent alleen maar hoeken in radialen. Een volledige cirkel bestaat uit 2 * PI radialen. PI is het niet te berekenen getal dat het verband beschrijft tussen de omtrek en de straal van een cirkel. (omtrek = 2 * PI * straal). Het is inmiddels tot meer dan 1 miljoen decimalen berekend, maar het blijft een benadering. Houdt u het maar op 3.14 en realiseert u zich dat dit een benadering is. Overigens kent de 64 de waarde van PI, als u in een berekening PI gebruikt wordt die door deze waarde vervangen. Dit stukje over graden, radialen en PI geeft u in ieder geval het verband tussen deze 3 grootheden. Omdat 360 graden gelijk is aan 2 * PI radialen zit PI dus in graden in de 64!.
360 graden = 2 * PI radialen, ook als u in graden gegeven hoeken heeft kunt u de computer van een juiste invoer in radialen voorzien.

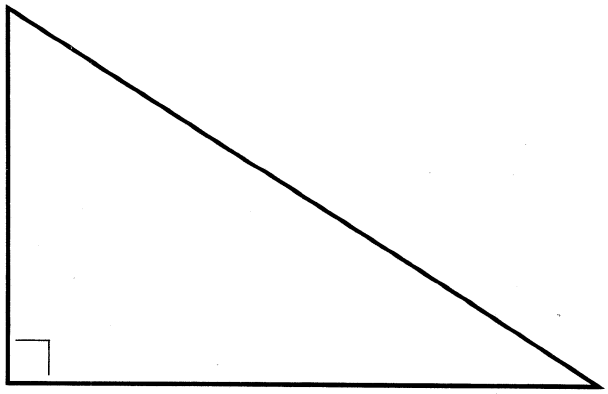
```
5 REM VOORBEELD 60
6 REM OMREKENING GRADEN-RADIALEN
10 PRINT"GRADEN","RADIALEN"
20 FOR G=0TO90STEP5
30 R=G*π/180
40 PRINTG,R
50 NEXTG
60 END
```

Definities uit de goniometrie:

We hebben een rechthoekige driehoek, als we vanuit een van de hoeken (nee, niet de rechte hoek nemen!) uitgaan dan geldt voor deze hoek het volgende:
sinus = overliggende rechthoekszijde/ schuine zijde. (de sinus ligt dus altijd tussen 0 en 1)

11.1 Goniometrische functies

cosinus = aanliggende rechthoekszijde/ schuine zijde. (ook de cosinus ligt tussen 0 en 1, maar als de sinus bijna 1 is, dan zal de cosinus bijna 0 zijn)
tangens = overliggende rechthoekszijde/ aanliggende rechthoekszijde.



Figuur 7/11.1-2: Een rechthoekige driehoek

Voor de bovenstaande drie grootheden hebben we de volgende functies:
COS(<getal>) geeft de cosinus van de door 'getal' aangegeven hoek - (aangegeven in radialen).
SIN(<getal>) geeft de sinus van de door 'getal' aangegeven hoek - (aangegeven in radialen).
TAN(<getal>) geeft de tangens van de door 'getal' aangegeven hoek - (aangegeven in radialen). De tangens kan varieren van heel klein tot reusachtig groot. Komt u in het gebied wat uw computer niet meer kan behappen dan krijgt u geen 'OVERFLOW ERROR' zoals u zou verwachten, maar een '?DIVISION BY ZERO ERROR'.

Als u van alle hoeken de bovengenoemde grootheden in een tabel verzamelt, dan zou u kunnen terugzoeken welke hoek bij een gegeven cosinus, sinus of tangens hoort.
De computer heeft alleen voor de tangens deze functie ingebouwd. ATN(<getal>) geeft u de hoek waarvan de tangens gelijk is aan 'getal'. Het resultaat is in radialen en zal altijd tussen -PI/2 en +PI/2 inliggen.
Voor de overeenkomstige handeling bij sinus en cosinus heeft de Commodore 64 geen ingebouwde functies.

```
5 REM VOORBEELD 61
10 INPUT"␣GEEF HOEK IN GEHELE
   GRADEN (NIET 90)";G
20 R=G*π/180
30 PRINT"AANTAL GRADEN","DE
   SINUS"
40 PRINTG,,SIN(R)
50 PRINT"AANTAL GRADEN","DE
   COSINUS"
60 PRINTG,,COS(R)
70 PRINT"AANTAL GRADEN","DE
   TANGENS"
80 PRINTG,,TAN(R)
90 PRINT"DRUK EEN TOETS OM DOOR
   TE GAAN"
100 GET A$:IFA$="" THEN 100
110 GOTO 10
```

7/11.2

Exponentiële en logaritmische functies

Ook hier moeten we het eens zijn over een paar begrippen. Wat is een exponent? De exponent is dat getal dat aangeeft tot welke macht een getal moet worden verheven. Bijvoorbeeld in 4 ↑ 3, voluit geschreven 4*4*4 en 2 ↑ 2, voluit 2*2 zijn 3 en 2 de exponenten.
Het begrip logaritme is iets ingewikkelder, nl. de exponent van de macht waartoe men een vast positief getal (het grondtal) moet verheffen om een willekeurig ander getal te krijgen. In ons decimale stelsel gebruiken we als grondtal meestal het getal 10. De logaritme van 100 is dan 2 (immers 10 ↑ 2 = 100) en de logaritme van 1000 is dan 3 (10 ↑ 3 = 1000).
Het zou eigenlijk abnormaal zijn als uw computer een zodanig grondtal hanteerde dat u de normale logaritmetafels kon gebruiken, (logaritmetafels zijn tabellen waarin de logaritmes van vele getallen, grondtal 10, met de bijbehorende getallen zijn verzameld). De 64 hanteert het natuurlijke getal 'e' als grondtal. Net als PI is e slechts te benaderen, houdt u het maar op 2.72.

De functie EXP(<getal>) geeft u e ^ 'getal'. Als de uitkomst te groot wordt dan geeft de computer een '?OVERFLOW ERROR'. De functie LOG(<getal>) geeft u de natuurlijke logaritme (met als grondtal dus e) van 'getal'. Als 'getal' 0 is of negatief dan bestaat LOG(getal) niet en krijgt u een '?ILLEGAL QUANTITY ERROR'.

```
5 REM VOORBEELD 62
10 INPUT"X-WAARDE";X
20 PRINT"E↑X=";EXP(X)
30 END
```

11.2 Exponentiële en logaritmische functies

```
5 REM VOORBEELD 63
10 INPUT"VOER X-WAARDE TUSSEN 0
   EN 88 IN ";X
20 IF X<=0 THEN 10
30 IF X>88 THEN 10
40 PRINT"X= ";X
50 PRINT"E↑X= ";EXP(X)
60 PRINT"LNX= ";LOG(X)
70 END
```

Er is nog een wiskundige functie in de Commodore 64 ingebouwd, namelijk: SQR(<getal>), deze functie geeft u de vierkantswortel van 'getal'. Ook hier wordt indien 'getal' negatief is uw inspanning gehonoreerd met een '?ILLEGAL QUANTITY ERROR'.

Mocht u denken dat u nu alle functies die de 64 voor u bezit heeft gehad, dan heeft u het mis. U heeft de beschikking over een heel arsenaal van functies, maar u dient ze wel zelf te definiëren. Als u een ingewikkelde wiskundige berekenig meerdere malen in het programma wilt laten uitvoeren, dan kunt u er beter in het begin van het programma een functie van maken. Dit spaart geheugenruimte en levert snelheid op, want de formule wordt maar een maal in het programma uitgerekend. De syntax van de door de gebruiker te definieren functies ziet er als volgt uit:
DEF FN<naam>(<variabele>)=<wiskundige uitdrukking>

'Naam' mag elke toegestane variabelenaam zijn, maximale lengte is echter 2 karakters. Het eerste teken mag geen cijfer zijn, maar alleen een letter. Een voorbeeld:

```
5 REM VOORBEELD 64
10 DEF FNA(X)=INT(RND(1)*X+1)
15 X=10
20 P=FNA(X)
30 PRINTP
```

Naderhand kunt u de gecreëerde functies aanroepen door de waarde voor X te geven en de functie aan te roepen, bv.

```
5 REM VOORBEELD 65
10 DEF FNA(X)=12*X-15/3
20 DEF FNB(X)=16*SQR(X)+X↑2
30 DEF FNC(X)=3*X↑3+12*X↑2-X↑2-7
40 FOR X=1 TO 5
50 P=FNA(X)
60 Q=FNB(X)
70 R=FNC(X)
80 PRINTX,P,Q;TAB(35);R
90 NEXT X
100 END
```

Dit geeft u een tabel als resultaat, maar u zou natuurlijk ook punten kunnen gaan plotten om een grafiek te maken. U kunt FN ook in de direct mode gebruiken, als tenminste het statement dat de functie definieert is uitgevoerd, anders volgt de foutmelding 'UNDEF'D FUNCTION'.

```
5 REM VOORBEELD 66
10 DEF FNA(X)=16*SQR(X)+X↑2
20 FOR X=1TO 3.25 STEP 0.25
30 P=FNA(X)
40 PRINTP;TAB(P);"*"
50 NEXT X
60 END
```

7/11.3

Nog meer gemak in uw computer ingebouwd

Als u op het scherm bezig bent en u weet niet op welke plaats de cursor staat, dan kunt u dit opvragen door de functie POS(<0>) te gebruiken. Het argument bij deze functie is een dummy-argument. U krijgt dan de cursorpositie op de regel waar deze staat. U verwacht een getal van 0-39, maar u kunt echter een getal van 0-79 krijgen. Dit komt door het verschil in lengte tussen schermregels en 'logische' regels. Een schermregel is 40 posities lang, een logische regel daarentegen 80. Daarom mag een programmaregel ook 80 tekens bevatten.

Bent u bezig met een programma en wilt u alle geheugenruimte in het RAM geheugen vrijmaken zonder dit programma aan te tasten dan kan dit door het statement CLR te gebruiken. Bij RUN en RUN gevolgd door een regelnummer (dezelfde werking als RUN, maar het programma start op het aangegeven regelnummer) is uitvoering van CLR automatisch ingebouwd, alle variabelen worden dan op 0 gesteld. U kunt echter redenen hebben om alleen CLR te gebruiken, denkbaar is dat u voor uitvoering van een bepaalde subroutine alle variabelen op 0 wilt zetten. Na terugkeer uit de subroutine kunt u het programma met STOP onderbreken, de waarden van de variabelen die in de subroutine zijn bewerkt opvragen en controleren op hun juistheid. Klopt alles, dan heeft u een grote kans dat de subroutine goed werkt. Dit is een handige foutzoekmethode.

U leerde in dit stuk de volgende wiskundige functies kennen:
COS(hoek) - die u de cosinus van de hoek geeft.
SIN(hoek) - die hetzelfde doet voor de sinus.
TAN(hoek) - die u de tangens van de hoek geeft.
ATN(getal) - die u de hoek die bij het getal hoort geeft.
LOG(getal) - die u de logaritme, op basis van grondtal 'e' van het getal geeft.
EXP(getal) - die u de exponent geeft waartoe u 'e' moet verheffen om het getal te krijgen.
SQR(getal) - die u de vierkantswortel van het getal geeft.
DEF FN<naam>(<variabele>)=<wiskundige uitdrukking> - die u de mogelijkheid geeft om zelf functies te definieren.
En verder de systeemfuncties:
POS(0) - die u de positie van de cursor op de logische regel mededeelt.
CLR - die alle geheugenruimte in het RAM geheugen vrij maakt. Het programma wordt hierbij niet aangetast.
En de begrippen graden, radialen, sinus, cosinus, tangens, logaritme, grondtal alsmede de, slechs te benaderen, getallen 'PI' en 'e'.

7/12

Talstelsels en logische operatoren; beïnvloeden van BITS

Inhoud

7/12.1

Iets over talstelsels

Hoewel talstelsels met het eigenlijke 'computeren' niets te maken hebben, moeten we er in verband met de werking van de computer wel iets van weten. Vooral het binaire, of tweetallige, stelsel is van belang.

De meeste talstelsels behoren tot de zogenaamde positiestelsels, dat wil zeggen dat er een direct verband is tussen de positie van een cijfer in het getal en de belangrijkheid ervan.

De invloed van de positie van een cijfer in een getal bekijken we aan de hand van het getal 352.
352 - 353 weinig invloed tengevolge van het veranderen van een cijfer (het aantal eenheden wijzigt).
352 - 362 meer invloed tengevolge van het veranderen van een cijfer (het aantal tientallen wijzigt).
352 - 452 meeste invloed tengevolge van de wijziging van een cijfer (het aantal honderdtallen wijzigt).
De positie van het cijfer is dus belangrijk, daarom spreken we van positiestelsel.

Over hoeveel stuks spreken we bij 4321 in ons tientallig stelstel?
Reken maar na 1 eenheid + 2 tientallen + 3 honderdtallen + 4 duizendtallen.

Als we naar het rijtje: 1 - 10 - 100 - 1000 kijken, dan zien we dat dit machten zijn van 10, het grondtal, namelijk de machten 0,1,2 en 3. (elk getal tot de macht 0 is volgens afspraak gelijk aan 1)
Van rechts naar links zijn in een getal de posities ook 0, 1, 2, 3 etc.
Voor andere talstelsels geldt deze regel ook.

Het getal 4321 in het vijftallig stelsel is dus
1*5↑0 + 2*5↑2 + 3*5↑3 + 4*5↑4
=
2926 stuks.
Het getal 11010110 in het binaire stelsel is dus 0*2↑0 + 1*2↑1 + 1*2↑2 + 0*2↑3 +
1*2↑4 + 0*2↑5 + 1*2↑6 + 1* 2↑7
= 214 in ons tientallige stelsel.

Het is dus noodzakelijk om te weten met welk grondtal we werken als we zaken met iemand gaan doen! Bestel maaar 195 vloertegels bij iemand die met het 20-tallig stelsel werkt, zonder dat u dit weet. Bij het uitpakken telt u tot uw schrik 585 tegels.

Wij werken met het tientallige stelsel, in de computerwereld komen voor: stelsels met 2 als grondtal (binair), 8 als grondtal (octaal) en 16 als grondtal (hexadecimaal).

7/12.1.1 Conversie

Een getal omzetten van het ene naar het andere talstelsel noemt men conversie. Het voert te ver om in te gaan op de trucs die bestaan in verband met conversie van hexadecimaal en octaal naar binair, maar de standaardprocedure is:
1) converteer het getal uit het ene stelsel naar het tientallige stelsel.
2) converteer het aldus gevonden tientallige getal naar het andere stelsel.

De eerste stap hebben we al gezien:
3527 (8-tallig)=3*8 ↑ 3+5*8 ↑ 2+2*8 ↑ 1+7*8-↑ 0 = 1879 (10-tallig).
Nu van het 10 tallig naar het ander stelstel (bijvoorbeeld het 5-tallige).
1) Deel het getal 1879 door het grondtal van het nieuwe stelsel, de rest is het meest rechtse cijfer van het gevraagde getal (1879/5=375, rest 4)
2) Deel het quotient uit 1) door het grondtal van het nieuwe stelsel, de rest is het volgende cijfer van rechts van het gevraagde getal (375/5= 75, rest 0)
3) Herhaal 2) tot het quotiënt niet meer deelbaar is.
(75/5=15, rest 0)
(15/5=3, rest 0)
4) Het laatste quotiënt is het meest linkse cijfer van het getal uit het nieuwe talstelsel.
In dit geval is het getal dus 30004 (5-tallig).

7/12.1.2 Algemeen talstelsels

Ons tientallig stelsel kent de cijfers 0 t/m 9
Het binaire stelsel kent de cijfers 0 t/m 1
Het octale stelsel kent de cijfers 0 t/m7
Het hexadecimale stelsel kent de cijfers 0 t/m 9, A,B,C,D,E,F

Als u terugdenkt aan het getal 256 (de mogelijkheden met 8 bits), dan ziet u misschien het verband al met 16 ↑ 2. Dit is namelijk ook 256, terwijl het grootste aantal dat u met twee tekens hexadecimaal kunt representeren FF is, ofwel 255.
Samen met de 0 zijn ook dit weer 256 mogelijkheden, voor te stellen met slechts 2 cijfers.

Ook het begrip CARRY willen we kort aanstippen. Als u in ons stelsel optelt en u zegt bijvoorbeeld: 8+5=3 en 1 onthouden dan is hier sprake van een overdracht of carry. Elk talstelsel kent dit verschijnsel bij optellingen. (zie hiervoor het hoofdstuk over machinetaal).

7/12.2

Logische bewerkingen

Behalve de normale rekenkundige bewerkingen kan uw computer ook nog logische bewerkingen verrichten, deze hebben in de volgorde van bewerkingen de laagste prioriteit.

```
5 REM VOORBEELD 68
10 PRINT"15 OR 194
   29 OR 32"
20 PRINT
30 PRINT"00001111      00011101"
40 PRINT"11000010      00100000
50 PRINT"--------      --------
60 PRINT"11001111      00111101
70 PRINT15 OR 194, 29 OR 32
80 END
```

We kennen daarvoor het keyword AND, de syntax luidt:
<uitdrukking>AND<uitdrukking>.
Het resultaat is waar als beide uitdrukkingen waar zijn, de numerieke waarde is dan -1

Ook kennen we de logische operator OR, deze geeft als resultaat 'waar' als een van beide uidrukkingen waar is of als beiden waar zijn. De syntax:
<uitdrukking>OR<uitdrukking>.
Het resultaat is -1 als een van de twee uitdrukkingen waar is.

Voorts kennen we nog de logische operator NOT die waar in onwaar omzet en omgekeerd.

7/12.3

Beïnvloeden van BITS

Als de AND en OR operatoren op twee getallen worden toegepast worden de corresponderende bits van de twee getallen, in het binaire stelsel geschreven, een voor een met elkaar vergeleken.
AND geeft als resultaat dat het bit dat werd vergeleken 1 wordt (wordt geset) als beide bits 1 waren. In alle andere gevallen wordt het betreffende bit 0 (gereset). Voorbeeld:

```
5 REM VOORBEELD 67
10 PRINT"␣15 AND 194   29 AND 16"
20 PRINT
30 PRINT"00001111      00011101"
40 PRINT"11000010      00010000
50 PRINT"--------      --------
60 PRINT"00000010      00010000
70 PRINT15 AND 194, 29 AND 16
80 END
```

In het tweede voorbeeld ziet u dat bit no. 4 op 1 gezet is in het resultaat, 16 = 2 ↑ 4. Als u zelf nog wat voorbeeldjes uitprobeert met machten van 2 na AND; dan ziet U dat het bitnummer aangegeven door de exponent 1 kan zijn in het resultaat, mits in het eerste getal dit bit ook 1 was. Alle andere bits worden 0. We kunnen AND dus gebruiken om bits op 0 te zetten.

Laten we dezelfde voorbeelden gebruiken bij OR, daar wordt het betreffende bit 1 als een van de twee bits 1 is.

```
5 REM VOORBEELD 68
10 PRINT"␣15 OR 194    29 OR 32"
20 PRINT
30 PRINT"00001111      00011101"
40 PRINT"11000010      00100000
50 PRINT"--------      --------
60 PRINT"11001111      00111101
70 PRINT15 OR 194, 29 OR 32
80 END
READY.
```

Hier werd in het tweede voorbeeld 2 ↑ 5 gebruikt, het resultaat was dat alle bits die 1 waren ook 1 gebleven zijn, maar dat ook bit no. 5 op 1 is gezet. Dus om bits op 1 te zetten gebruiken we OR.

In de hoofdstukken 10 en 11, graphics en geluid komen deze bewerkingen erg veel voor. Alle 8 sprites worden namelijk aan en uitgezet door de 8 bits van slechts een geheugenplaats. U kunt met OR bijvoorbeeld sprite no. 6 aanzetten door als tweede getal 2 ↑ 6 te nemen, ofwel 64. Uitzetten gaat weer dan met behulp van AND(255 - 2 ↑ 6). Veel van de graphics- en geluidsfuncties worden op een dergelijke wijze bestuurd.

12.3 Beïnvloeden van BITS

Probeert u zelf maar eens een constructie te bedenken om eerst alle bits op 0 te zetten en vervolgens bijvoorbeeld bits 2, 5 en 6 op 1 te zetten, daarna alle bits op 1 en vervolgens alle bits behalve bit 3 op 0. Succes!
(oplossing vindt u hieronder)

In dit deel maakte u kennis met de logische operatoren AND, NOT en OR en hun plaats in de volgorde van bewerkingen. Ook gingen we in op talstelsels, het omzetten van getallen van het ene in het andere talstelsen en het beïnvloeden van bepaalde bits in een byte.

```
GETAL   AND   0
0       OR    100
100     OR    255
255     AND   2↑3
```

Figuur 7/12.3-1: De beloofde oplossing

7/13

Randapparatuur en de communicatie daarmee

Inhoud

7/13.1

Algemeen

Tot nu toe zijn we er van uitgegaan dat u werkte met computer en cassetterecorder. Maar als u verder komt, zit u al snel met een diskdrive, een printer en eventueel zelfs nog andere rand-apparatuur.

Een van de merkwaardigheden van de organisatie van apparatuur bij de Commodore 64 is dat alles met elkaar doorverbonden is, elk signaal dat de computer uitstuurt, wordt door alle randapparaten -behalve de cassetterecorder- ontvangen. De cassetterecorder heeft een eigen aansluiting. Deze zorgt behalve voor het transport van data ook voor de besturing van de motor van de cassetterecorder. Alle andere randapparaten zijn op de seriële bus aangesloten door doorlussen. U zult dus moeten vertellen voor welk randapparaat de signalen estemd zijn. Net als bij een telefooncentrale heeft ook hier elke abonnee zijn eigen nummer. U moet er trouwens nog meer nummers bij geven, als eerste moet u een zogenaamd 'logical file number' ofwel in goed Nederlands 'logisch bestandsnummer' meegeven.

Dat wil zeggen dat u het bestand waarmee u gaat werken een eigen nummer geeft (vooral belangrijk bij de diskdrive, deze kan met meer bestanden tegelijk werken). Deze nummers mogen lopen van 0 tot 128, hogere nummers zijn bij de 64 intern bestemd voor andere doeleinden.

Een bestand is een verzameling bij elkaar horende records of blokken die als een eenheid wordt behandeld. Bij een factuur kan een regel een gegevenselement vormen, de volledige factuur een record en de verzameling facturen een bestand vormen. De wijze waarop de diverse records bereikbaar zijn bepaalt het type van het bestand. Bij een sequentieel bestand kunt u de records slechts op volgorde bereiken, bij andere vormen spreken we van relatieve bestanden en random bestanden. Ook hier komen we in de toekomst op terug.

Vervolgens moet u mededelen voor welk randapparaat de gegevens bestemd zijn, het rijtje is als volgt:
0 = toetsenbord
1 = cassetterecorder
2 = modem (RS232 interface)
3 = beeldscherm
4 of 5 zijn printers
6 = printer/plotter 1520
8,9 10 of 11 zijn nummers voor de diskdrive(s).

Tenslotte geeft u meestal een secundair adres mee, waarmee we eigenlijk aangeven via welk kanaal de communicatie gaat verlopen. Een paar van deze kanaalnummers zijn gereserveerd voor speciale doeleinden, bijvoorbeeld bij de diskdrive de nummers 0,1 en 15. De eerste twee cijfers

13.1 Algemeen

bepalen of gelezen of geschreven wordt: 0 voor lezen, 1 voor schrijven (door de commando's LOAD, SAVE en VERIFY worden deze nummers automatisch meegegeven.) Het nummer 15 mag U niet gebruiken voor normaal gegevenstransport, omdat dit het commandokanaal is voor de diskdrive. Via dit kanaal kunt u onder andere de foutmeldingen van de diskdrive uitlezen en uw diskcommando's geven; bijvoorbeeld file wissen, initialiseren, een nieuwe diskette formatteren enz. (zie verder hoofdstuk 6).

De communicatie met de randapparatuur is een zeer gecompliceerd gebied, waar niet alleen begrippen als lezen en schrijven aan te pas komen, maar ook vormen van bestanden, plaatsen van records en allerlei andere ingewikkelde zaken. Voor het moment zullen we ons beperken tot de cassetterecorder en heel voorzichtig de diskdrive aanroeren.
In de toekomst zullen we er veel meer aandacht aan besteden.

7/13.2

De cassetterecorder

De verschillende syntaxen bij cassette:
1) OPEN1,1,0,"naam" - open bestand „naam" op cassette om uit te lezen.
2) OPEN1,1,1,"naam" - open bestand „naam" op cassette om naar te schrijven.
3) CLOSE1, om het bestand te sluiten.

Een kanaal waarlangs gegevens worden getransporteerd vult een buffer. Als deze buffer vol is worden de gegevens weggeschreven. Het commando CLOSE geeft opdracht om ook een niet geheel gevulde buffer weg te schrijven. Alleen met toepassen van CLOSE bent u er zeker van dat alle gegevens zijn weggeschreven.

Doordat een cassetteband normaal maar in één richting loopt, kunt u alleen maar gegevens op volgorde inlezen of wegschrijven, dat wil zeggen dat u alleen met sequentiële bestanden kunt werken. U heeft echter geen probleem om, bij wijze van spreken, 10 maal een bestand met dezelfde naam op dezelfde cassette te zetten.

Bij de diskdrive is dit anders, op een diskette mag maar een bestand met een bepaalde naam voorkomen, anders krijgt u een foutmelding (via het knipperen van het rode lampje).

```
5 REM VOORBEELD 69
10 OPEN1,1,1,"LOONLIJST"
20 REM CASSETTEBESTAND
   'LOONLIJST' OM IN TE
    SCHRIJVEN
30 FORX=1TO10
40 READP$(X)
50 NEXTX
60 FORX=1TO10
70 PRINT#1,P$(X)
80 NEXTX
90 CLOSE1
100 DATA JANSEN,PIETERSE,
    VERDONK,WILLEMS,DE RUYTER
110 DATA KLAASSEN, THEUNISSEN,
    DIKSHOORN,ALBERTSEN, KOOT
```

```
5 REM VOORBEELD 70
10 PRINT"␣"
20 OPEN1,1,0,"LOONLIJST"
30 REM CASSETTEBESTAND
   'LOONLIJST' OM UIT TE LEZEN
40 FORX=1TO10
50 INPUT#1,P$(X)
60 NEXTX
70 FORX=1TO10
80 PRINTP$(X)
90 NEXTX
100 CLOSE1
```

7/13.3

De diskdrive

De verschillende syntaxen bij diskdrive:
LOAD"<programmanaam>",an[,sa]
SAVE"<programmanaam>",an[,sa]
VERIFY"<programmanaam>",dn[,sa]
voor BASIC programma's.
OPENlfn,an,sa,"dn:naam,t,w"

Hierin hebben de letters de volgende betekenis:
lfn=logical file number, zie boven.
an=apparaatnummer, het nummer van de diskdrive, zie boven.
sa=secundair adres, no's 2-14 zie boven
dn=het nummer van de drive, bij een enkele drive is dit 0, bij een dubbele drive of 1.
naam=naam van het bestand, maximaal 16 karakters
t=type van het bestand, er zijn drie types, namelijk „S" voor sequentieel,„U" voor user en „P" voor program.
w=wat wilt u doen? U vult hier in „R" voor lezen, „W" voor schrijven en „A" voor toevoegen (append).

Ook hier moet u bestand(en) weer sluiten als uw gegevensuitwisseling voltooid is.
De syntax is:
CLOSElfn
Dit moet u voor elk geopend bestand doen.

Hoe krijgt u nadat u bestanden geopend heeft de gegevens erin of eruit? Dit is vrij eenvoudig, u gebruikt dezelfde commando's PRINT, INPUT en GET, maar omdat u niet met het scherm werkt laat u ze volgen door het teken #, gevolgd door het bestandsnummer waarop de commando's betrekking hebben. PRINT mag u in dit geval niet afkorten tot '?'. Verder moeten we nog vermelden dat er bij INPUT#lfn,A$ maximaal een string van 80 karakters lengte binnengehaald kan worden; bij de opbouw van uw bestand moet u hier (via bijvoorbeeld LEN(A$)) rekening mee houden. Bij GET#,A$ wordt teken voor teken uit het bestand gehaald; via een formule B$=B$+A$ kunt u deze tekens opbouwen tot een string, de maximale lengte hiervan kan 255 tekens zijn.

Als u bij het werken met de diskdrive een foutmelding krijgt, dan kunt u met het volgende programma het foutkanaal uitlezen en in het handboek opzoeken wat er aan de hand was. Ik raad u aan dit in elk programma achteraan te zetten, met een regelnummer van bijvoorbeeld 60000, krijgt U dan een foutmelding, dan geeft u RUN 60000 en u bent klaar.
Dit uitlezen van het commandokanaal om de fout te achterhalen moet namelijk vanuit een programma gebeuren. In de direct mode is dit niet mogelijk.

13.3 De diskdrive

Kunstgrepen om dit via STATUS te behandelen vindt u in de aanvullingen op dit boek.

Het enige wat wij u nu nog over de diskdrive vertellen is de mogelijkheid om een tweede bestand met dezelfde naam op een diskette te krijgen zonder foutmelding, u moet dan voor de bestandsnaam tussen de aanhalingstekens meegeven het teken C gevolgd door het drivenummer: OPEN8,8,15, "C0:naam,s,w" overschrijft dan het oude bestand van dezelfde naam. Dit oude bestand is dan wel verloren gegaan. LET OP: Dit commando is zeer onbetrouwbaar. De kans bestaat dat de diskdrive alles door elkaar gaat gooien.

```
5 REM VOORBEELD 73
10 DIM M$(12)
20 OPEN8,8,8,"MAANDEN,S,R"
30 FOR X=1 TO 12
40 INPUT#8,M$(X)
50 PRINTM$(X)
60 NEXTX
70 CLOSE8
```

```
5 REM VOORBEELD 74
10 OPEN8,8,8,"MAANDEN,S,R"
20 GET#8,A$
30 PRINTA$;
40 IF A$=CHR$(13)THENPRINT:T=T+1
45 IF T=12 THEN CLOSE8 :END
50 GOTO20
```

7/13.4

De printer

Bij PRINT#lfn,A$ schrijft u alleen A$ naar het bestand weg, wilt u alle output naar het bestand schrijven, dan kunt u zich veel tikwerk besparen door het commando CMD#lfn te gebruiken, hierdoor wordt alle output naar dit bestand gestuurd.

Als u eenmaal klaar bent met de communicatie dan moet u eerst een lege printopdracht naar het betrokken bestand geven door PRINT#lfn en daarna door CLOSEfn het bestand afsluiten.

```
5 REM VOORBEELD 75
10 OPEN 3,4
20 PRINT"DEZE TEKST KOMT OP HET
   SCHERM"
30 PRINT#3,"DEZE TEKST GAAT
   NAAR DE PRINTER"
40 PRINT"ZO KUNT U WISSELEN"
50 PRINT#3,"WANT U WILT NIET
   ALLES OP PAPIER HEBBEN"
60 PRINT#3:CMD3
70 PRINT"NU GAAT ALLES NAAR
   DE PRINTER"
80 PRINT"EN DAT KAN MAKKELIJK
   ZIJN."
90 PRINT#3:CLOSE3
```

Bij de printer heeft u meestal geen secundair adres nodig, dit speelt alleen als U met interfaces gaat werken en paralelprinters gaat gebruiken. Normaal is de syntax:
OPENlfn,an
Dezelfde betekenis voor de letters als boven.

In dit gedeelte heeft u leren communiceren met diverse soorten randapparatuur, en kennis gemaakt met de volgende commmando's:
OPEN: om een bestand te openen.
PRINT#: om gegevens naar een bestand weg te schrijven.
INPUT#: om gegevens uit een bestand in een string in te lezen, maximaal 80 tekens per keer.
GET#: om gegevens uit een bestand in het computergeheugen in te lezen, dit gebeurt teken voor teken. Op deze wijze kunt u lange strings opbouwen.
CMD# om alle output naar een bepaald bestand te sturen.
CLOSE; om bestanden af te sluiten.
En met de begrippen bestand, record en sequentieel

7/14

Nog meer over het beeldscherm, wat beweging

Inhoud

7/14.1

Meer over schermindeling

In het voorgaande leerde u diverse methodes om teksten op de door u bepaalde plaats af te drukken; er is nog een slim kunstje voor, namelijk het volgende:
10 A$="25 maal cursor naar beneden"
20 B$="40 maal cursor naar rechts"
U kunt dan verder in uw programma sturen met PRINT LEFT$(A$,n1) en PRINT
LEFT$(B$,n2) om de tekst op de juiste plaats te krijgen.

```
5 REM VOORBEELD 76
10 A$="■■■■■■■■■■■■■■■■■■■■■■■■■
   ■"
   ■■■■■■■■■■■■■■■■■"
20 B$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
30 PRINT"■"
40 PRINTLEFT$(A$,12);LEFT$(B$,
   20);"ZO GAAT HET OOK"
```

Er zijn nog meer manieren om het afdrukken te controleren. U heeft geleerd dat het geheugen uit 65536 'postbussen' bestaat. Bepaalde van deze adressen controleren bepaalde handelingen. Zo wordt op adres 211 de horizontale positie van de cursor bijgehouden en op adres 214 de verticale positie. Via 'POKE adres, getal' kunt u zelf direct of vanuit uw programma waarden in het geheugen veranderen (getallen van 0 tm 255). Wat gebeurt er na POKE211,9 en POKE214,14? Helaas, er gebeurt niets. Er blijkt pas iets te gebeuren als u een machinetaalroutine laat werken die door Commodore in het ROM gedeelte is ingebouwd. Deze routine (er zijn er meer, zie het machinetaalgedeelte!) start u met het commando SYS gevolgd door het adres waar de routine begint, in dit geval 58732.

```
5 REM VOORBEELD 77
10 PRINT"■"
20 POKE211,15
30 POKE214,10
40 SYS 58732
50 PRINT"NAAM"
60 PRINT"■"
70 END
```

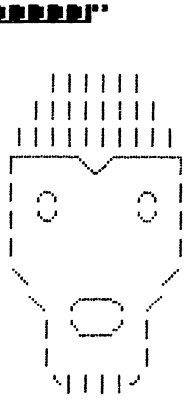
Overigens het commando POKE heeft een familielid, namelijk PEEK(geheugenplaatsnummer).
Door PRINT PEEK(adres) kunt u kijken welke waarde er in een bepaalde geheugenplaats staat.

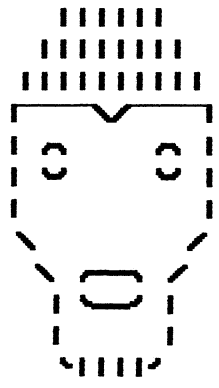
Tekeningen gemaakt met behulp van de grafische symbolen van de Commodore 64 in een programma opnemen is ook al niet zo moeilijk als het lijkt; het kunstje hiervoor:
1) maak de door u gewenste tekening op het scherm, u kunt dit het beste van tevoren leeg maken, maar u gebruikt BESLIST NIET de RETURN toets. Is de tekening naar uw zin; pas dan zet u op elke regel ervoor:
2) 'regelnummer PRINT"' en u drukt

14.1 Meer over schermindeling

vervolgens op RETURN. Let er wel op dat uw regelnummers hetzelfde aantal cijferposities hebben, want anders verloopt uw tekening in de breedte
3) u zet voor het stukje programma dat de tekening bevat een regel die de cursor op de juiste plaats zet om te beginnen, eventueel het scherm schoon maakt en het doel is bereikt.

```
5 REM VOORBEELD 78
10 PRINT"■"
15 FOR T=1 TO 2000:NEXT
20 PRINT"■■■■■■■■■■■■■■■■"
21 PRINT"
22 PRINT"
23 PRINT"
24 PRINT"
25 PRINT"
26 PRINT"
27 PRINT"
28 PRINT"
29 PRINT"
30 PRINT"
31 PRINT"
32 PRINT"
33 PRINT"
34 PRINT"■DIT IS EEN ZORGELIJK PROGRAMMEUR"
35 PRINT"■"
40 FOR T=1 TO 2000:NEXT
50 GOTO10
```





Figuur 7/14.1-1: De uitvoer

7/14.2

Beweging

Beweging op het scherm maken is niet zo moeilijk als het lijkt, met een zo simpel commando als TAB lukt het al, u mag namelijk tussen de haakjes zowel een waarde als een variabele zetten.
Wat dacht u van het volgende:

```
5 REM VOORBEELD 79
6 PRINT"■"
10 A=1
20 PRINT"■"
30 PRINTTAB(A)"*■■ ";
40 A=A+1
50 IFA=38THEN80
60 GOTO20
70 PRINT"■"
80 PRINTTAB(A)"* ";
90 A=A-1
100 IFA=1THEN20
110 GOTO70
```

Uiteraard is er wat beweging betreft nog veel en veel meer mogelijk met uw computer, in hoofdstuk 10 „GRAPHICS", wordt hieraan uitgebreid aandacht geschonken en leert u werken met SPRITES en dergelijke grappen meer. Het zou te ver gaan om dit allemaal in het hoofdstuk BASIC op te nemen, net zoals wij in dit gedeelte geen aandacht besteden aan het onderwerp geluid; dit komt in het hoofdstuk 11 „GELUID", uitgebreid aan de orde. Zowel van een bewegende SPRITE als van een geluidseffect volgt hier slechts een voorbeeld.

```
5 REM VOORBEELD 80
10 POKE2040,13:PRINT"■":POKE532
   69,PEEK(5369)OR(2↑0):
   POKE53249,100
20 FORI=0TO62:READA:POKE832+I,A:
   NEXT
30 POKE53277,PEEK(53277)OR(2↑0):
   POKE53271,PEEK(53271)
   OR(2↑0)
40 FORI=255TO0STEP-1:POKE53248,I
   :NEXT:GOTO40
60 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
   ,0,0,0,0,0,0,0,0,0,0,
   0,8,0,0,9,0,3
70 DATA255,240,3,24,240,3,24,240
   ,3,255,240,3,255,240,
   3,255,240,3,255,240
80 DATA0,96,192,0,96,192,0,0,0,0
   ,0,0,0,0,0
```

14.2 Beweging

```
5 REM VOORBEELD 81
10 PRINT"□"
20 PRINT"VOOR MEER KLANKEN:DRUK OP J"
30 PRINT"STOPPEN DOOR ANDERE TOETS.":FORT=1TO1500:NEXT
40 M=54272
50 FOR R=0TO24:POKEM+R,0:NEXT
60 POKE54296,15:FORT=1TO1000:NEXT
70 PRINT"□":S1=INT(255*RND(1))+1
80 POKE54273,S1
90 POKE54277,11:POKE54278,16
100 S2=INT(255*RND(1))+1
110 POKE54287,S2
120 FORN=1TO100:POKE54276,19
130 FOR T=1 TO 500:NEXT:POKE54276,40
140 GETA$:IFA$=""THEN140
150 IF A$="J" THEN 70
```

In dit gedeelte maakte u kennis met de BASIC commando's:
PEEK(adres) - om te kijken wat er op een bepaald adres voor waarde staat.
POKEadres,waarde - om het getal 'waarde' (0-255) te plaatsen in geheugenplaats 'adres'. Werkt alleen in het RAM geheugen.
SYSadres - om een in de 64 aanwezige machintaal routine die op geheugenplaats 'adres' begint te starten.
Verder maakte u heel vluchtig kennis met sprites, geluid en screen graphics.

7/15

De laatste BASIC commando's, het opsporen van fouten

Inhoud

7/15.1

De laatste BASIC commando's

De laatste nog ontbrekende, niet zo erg veel door beginnende programmeurs gebruikte commando's moeten we voor de volledigheid nog behandelen.

Allereerst nog een commando dat betrekking heeft op de schermindeling. Het is het commando SPC(A). Dit lijkt wat vorm betreft erg veel op het commando TAB(A), het verschil is echter dat TAB een absolute plaatsbepaling is en SPC een relatieve. De instructie SPC(10) zet 10 spaties tussen de laatst gebruikte printpositie en het volgende af te drukken teken.

U kunt hiermee een nog leuker bewegingseffect maken:

```
5 REM VOORBEELD 82
10 PRINT"␣"
20 FORT=1TO35
30 PRINT"*"SPC(T)"*"
40 NEXT
50 FOR T=35TO1STEP-1
60 PRINT"*"SPC(T)"*"
70 NEXT
80 GOTO20
```

WAIT is een commando dat een programma onderbreekt, totdat een bepaalde geheugenplaats een bepaalde inhoud heeft. Met andere woorden, het programma stopt totdat door een uitwendige invloed de bepaalde geheugenplaats een bepaalde waarde heeft gekregen. Bij weinig programmeerervaring kunt u het beter alleen gebruiken als remplacant voor GET om een programma te onderbreken tot een toets wordt ingedrukt.
WAIT 198,1 onderbreekt het programma tot er een toets wordt ingedrukt.

```
5 REM VOORBEELD 83
10 PRINT"␣EEN DEMONSTRATIE VAN
   WAIT;DRUK EEN TOETS IN
   OM VERDER TE GAAN"
20 WAIT198,1
30 PRINT"U HEEFT EEN TOETS
   INGEDRUKT"
40 FORT=1TO1000:NEXT
50 POKE198,0
60 WAIT198,1
70 PRINT"U HEEFT WEER EEN
   TOETS INGEDRUKT"
80 POKE198,0
90 GOTO20
```

USR is ook al geen commando voor beginners, dit start een machinetaalroutine waarvan het beginadres staat in de geheugenplaatsen 785 en 786. Hierbij worden variabelewaarden tussen machinetaalprogramma en BASIC gedeelte uitgewisseld. U moet echter het een en ander van machinetaal afweten om met dit commando te kunnen werken, in toekomstige aanvullingen zullen we er zeker aandacht aan besteden.

15.1 De laatste BASIC commando's

STATUS wordt wel veel gebruikt. We vertelden u al eerder dat STATUS een systeemvariabele is die u niet voor andere doeleinden mag gebruiken. STATUS geeft u informatie over het verloop van de laatste input/output operatie met een geopend bestand. De tabel in bijlage 3 geeft u enige gegevens over STATUS waarden als er iets is mis gegaan. U kunt bijvoorbeeld testen op de waarde van ST om te beslissen of U het programma wilt voortzetten of dat U wilt teruggaan omdat er iets niet goed is verlopen. We geven u een kort voorbeeld van het gebruik van status in een programma dat met de diskdrive werkt.

```
5 REM VOORBEELD 84
10 OPEN8,8,8,"TEST,S,W"
20 PRINTST:IFST>0THEN 70
30 CLOSE8
35 STOP
40 OPEN8,8,8,"TEST,S,W"
45 INPUT#8,A$
50 PRINTST:IFST>0 THEN60
60 OPEN15,8,15
70 INPUT#15,A$,B$,C$,D$
80 PRINTA$,B$,C$,D$
```

7/15.2

Fouten zoeken

Waarschijnlijk is er in de geschiedenis nog geen enkel groot programma geschreven dat direct al helemaal foutloos was. Hoe haalt u de fouten er uit? Hiervoor zijn diverse hulpmiddelen te bedenken:

1) De computer helpt u door een uitgebreid scala van foutmeldingen, die vrijwel allemaal in het voorafgaande aan de orde zijn gekomen.

2) De computer kan u echter niet helpen als u een denkfout heeft gemaakt en dus logische fouten heeft geïntroduceerd in uw programma. Een mogelijkheid om hiernaar te gaan speuren is het uitvoeren van een programmarun met een tevoren bekende uitkomst. Klopt het resultaat niet, dan is er duidelijk iets mis. U kunt dan proberen om op strategische plaatsen in het programma STOP statements in te voegen, daar de waarden van de diverse variabelen op te vragen en zo de fout proberen te localiseren.

3) Helpt dit ook niet, dan kunt u zelf computer gaan spelen. U maakt dan een zogenaamd geheugenschema en zet achter elk regelnummer in tabelvorm de waarde van alle numerieke en stringvariabelen. Door weer met STOP statements te werken kunt U alles controleren. Bij een lang programma met subroutines en for/next lussen is dit een erg tijdrovende klus, maar soms zal het noodzakelijk blijken hiertoe over te gaan.

4) Het inschakelen van een toolkit (dat wil zeggen een hulpprogramma) waarmee u extra commando's krijgt die niet in de Commodore 64 BASIC aanwezig zijn. Een van de commando's die u bij het opsporen van fouten kunt gebruiken is het commando TRACE waarbij u de regelnummers die op dat moment uitgevoerd worden op het scherm ziet afgedrukt. U kunt op die manier de loop van het programma controleren en zodoende kijken of alles volgens uw planning gaat. Meestal kennen deze toolkits ook de instructie DUMP, waarbij u na een STOP statement in een maal de waarde van alle tot op dat moment in het programma gebruikte variabelen te zien krijgt. U hoeft ze niet stuk voor stuk op te vragen, met de kans dat u er een of meer over het hoofd ziet.

Meer over het opsporen van fouten en het logisch opzetten van programma's vindt u in dit boek in hoofdstuk 12 „PROGRAMMEERKUNDE".

In dit laatste gedeelte maakte u kennis met de instructies: SPC(<argument>) - om een door u te bepalen aantal spaties tussen twee printresultaten te zetten.
WAIT- waarmee u onder andere een programma kunt onderbreken totdat een

15.2 Fouten zoeken

bepaalde situatie is opgetreden.
STATUS - een systeemvariabele die u informatie geeft over het verloop van de laatste INPUT/OUTPUT bewerking.

USR(<argument>) - een methode om een bepaalde machinetaalroutine te starten en daarbij waardes tussen machinetaalroutine en BASIC uit te wisselen.

7/16

Een eerste toepassing

Inhoud

7/16.1

Van keywords tot programma

Zoals u in hoofdstuk 12 kunt lezen, bestaat het belangrijkste werk bij het schrijven van een programma uit het definiëren en analyseren.

Een onmisbaar onderdeel bij inspanningen om een programma te schrijven is te zorgen dat er door de gebruiker van het programma geen dusdanige fouten worden gemaakt, dat de zaak ofwel verkeerde resultaten geeft, ofwel vastloopt met foutmeldingen. In deze en de komende aanvullingen zal een programma worden opgebouwd, dat u inzicht geeft in de kosten van uw auto, of eventueel van uw wagenpark.
Het zal in delen worden opgebouwd en u kunt de delen apart intikken en later aan elkaar koppelen met een 'merge' routine of u kunt telkens de eerste listing uitbreiden.

Als u even nadenkt over de definitie van een dergelijk programma ziet U dat er gelijk al twee dingen zijn die eenduidig moeten worden vastgelegd, namelijk om welke kosten gaat het en welke auto betreft het? De tweede vraag is snel beantwoord, een auto en zijn kentekennummer horen onafscheidelijk bij elkaar. De eerste vraag is aanmerkelijk moeilijker te beantwoorden. Er kan gedacht worden aan kosten per dag, kosten per kilometer, benzineverbruik in liters per kilometer, brandstofkosten per kilometer (dan telt het olieverbruik immers ook mee).

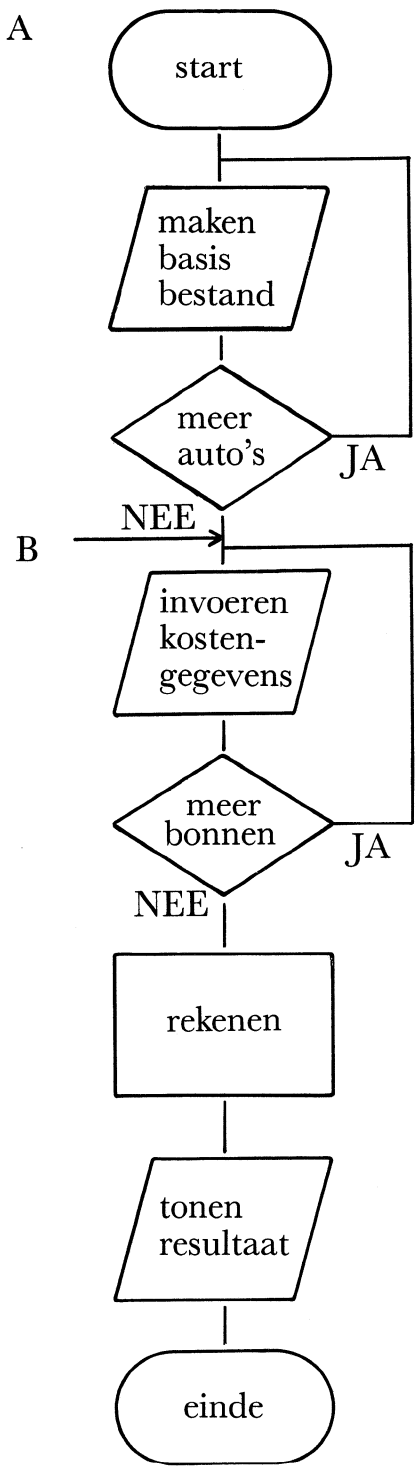
Ook zijn er extra's in te bouwen, zo kunt u denken aan het maken van een grafische voorstelling van de resultaten zodat u een eventueel omslagpunt kunt opsporen en zodoende kunt besluiten om de betreffende auto te gaan inruilen of verkopen. Beperking siert de meester en daarom nemen we niet alle mogelijkheden in het programma op. De resultaten worden in eerste instantie beperkt tot:

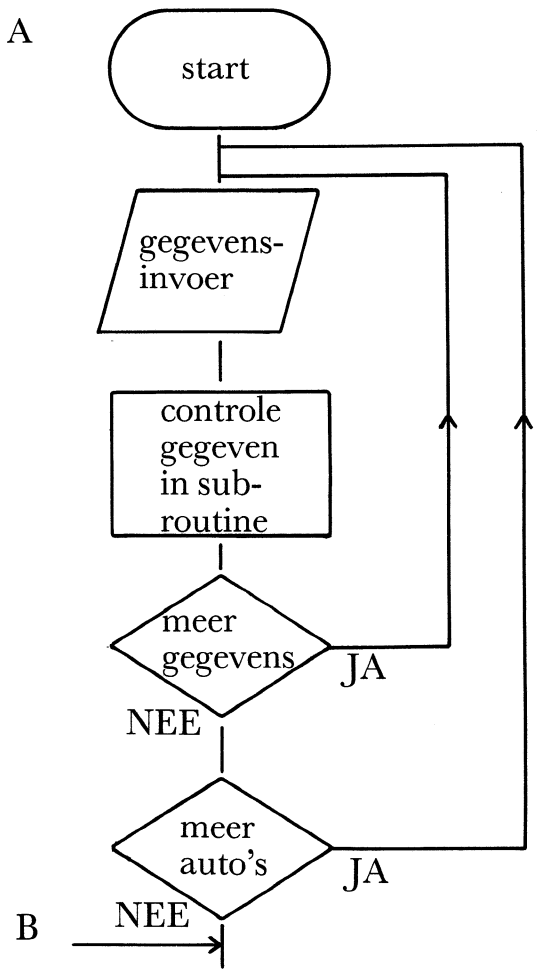
1) hoeveel kilometer rijdt de auto met een liter benzine.
2) wat zijn de brandstofkosten per kilometer.
3) wat zijn de totale kosten per kilometer.
4) wat zijn de totale kosten per dag.

U zult zien dat dit al gecompliceerd genoeg is. Immers, nu de taak van het programma is vastgelegd komt aan de orde welke gegevens noodzakelijk zijn. Ook kijken we of er al een grove structuur van het programma valt te maken. In hoofdstuk 12 is de term 'top-down' gevallen en dit is de meest praktische manier om een programma op te zetten.

Welke gegevens hebben we nodig en hoe moeten ze er uitzien. Hier zullen duidelijk keuzes gemaakt moeten worden en

16.1 Van keywords tot programma





eerste verfijning
eerste schema
tussen punten
A en B

Figuur 7/16-1: Een allereerste schema.

16.1 Van keywords tot programma

de gebruiker zal gedwongen moeten worden zich aan de gemaakte afspraken te houden.

Nodig zijn:
1) Het kentekennummer.
2) Aanschafprijs en restwaarde.
3) Aantal getankte liters benzine, kilometerstand bij tanken, datum bij tanken en totaalbedrag van de benzinebon.
4) Datum, bedrag en kilometerstand bij het aanvullen van het oliecarter. Het aantal gebruikte liters olie is in verband met de vraagstelling niet belangrijk.
5) Alle overige kosten met datum, omschrijving en bedrag.

Maar dit is niet alles, er is nog veel meer nodig. Kijkt u maar:
6) Datum van aanschaffen van de auto.
7) Als u er een lening voor heeft gesloten dan zijn de rentekosten simpel in te boeken, heeft u de auto contant betaald, dan moet u het renteverlies opnemen in de kosten van de auto.
8) Kilometerstand bij aankoop, anders heeft u geen basis om de computer te laten rekenen.

Nu u hierboven een eisenpakket heeft opgesteld voor het programma kunt u gaan kijken of er al een splitsing te maken valt in diverse modules. Het is verstandig een programma in kleine modules op te splitsen die na het schrijven, indien mogelijk, afzonderlijk getest moeten worden.

Houdt eveneens rekening met de hardware, die beschikbaar is. Deze heeft ook een belangrijke invloed op het programma. Hier rijst ook de vraag: 'heeft u nog uitbreidingsplannen?'.

Als u geen printer heeft maar wel overweegt om er een aan te schaffen is het zinnig om nu al rekening te houden met het uitprinten van rapporten.

Een paar dingen zijn duidelijk.
a) Bij aanschaf van de auto moet een gegevensbestand worden gemaakt.
b) Dit gegevensbestand moet worden bewaard om later te kunnen worden uitgebreid met kostenposten tijdens het gebruiken van de auto. Opslaan van gegevens kunt u op twee manieren doen, namelijk op cassette of op diskette. In dit programma worden beide mogelijkheden opgenomen, niet iedereen heeft de beschikking over een diskdrive.
c) Als u de resultaten wilt zien dan moet u de opgeslagen gegevens aanvullen met een paar noodzakelijke toevoegingen.

Om tot kostprijzen te kunnen komen heeft u de datum, waarop u rekent, nodig en de restwaarde van de auto op die datum. Er van uitgaande dat een deel van de lezers (nog) geen printer heeft worden de resultaten naar keuze op het scherm of op papier gezet.

In het voorgaande gedeelte over BASIC is uitgebreid gesproken over subroutines daar deze zeer veel werk kunnen besparen. U kunt nu al gaan kijken welke onderdelen van het programma geschikt zijn om in een subroutine te plaatsen. Ook hieruit volgt weer een rijtje punten.
1) Niemand is volmaakt. Elke invoer moet gecontroleerd worden op juistheid. Is de invoer juist, dan moet dit bevestigd worden. Dit kan via het toetsenbord als antwoord op een vraag gebeuren. Om te voorkomen dat gebruikers het programma opza-

delen met gegevens die niet onjuist, maar wel onlogisch zijn wordt alle invoer als extra zekerheid "gezeefd"; de onderlinge logica moet worden gecheckt.

2) Diverse gegevens die in alle delen van het programma voorkomen – te weten opzetten bestand, aanvullen bestand en resultaten tonen – vragen erom in subroutines te worden geplaatst. Voor subroutines komen in aanmerking datumcontrole, controle op invoer van bedragen, controle op kilometerstanden enzovoorts.

Nu moet ook beslist worden waar de subroutines komen te staan, voor of na het hoofdprogramma. Hoewel het de overzichtelijkheid niet ten goede komt kiezen we – het werkt nu eenmaal sneller op die wijze – ervoor om de subroutines voor het hoofdprogramma te plaatsen.

Ook belangrijk zijn de namen van de gebruikte variabelen, hiervan vanaf het begin van het programma een lijst bijhouden spaart veel puzzelen en zoeken. Behalve het verslag van het denken over het programma (dit leest u op dit moment) is het erg nuttig om in een eerste versie van het programma veel documentatie in de vorm van REM statements op te nemen zodat u later weet wat u heeft gedaan. Ook is het nuttig om in de eerste versie van uw programma maar één instructie per regel op te nemen. Blijkt het programma na uitgebreid testen feilloos te lopen, dan kunnen er zoveel mogelijk instructies op één programmaregel (zoals u weet twee regels op het scherm) worden geplaatst om geheugen te sparen en snelheid te winnen.

Het menu.

Omdat er in figuur 7/16-1 een duidelijke scheiding te zien is tussen de diverse verwerkings-fasen wordt het programma menugestuurd. Het menu kan geschreven worden zodra er regelnummers zijn vastgelegd.

```
10 goto 2000
100 for t = 1 to 1000
110 next t
120 return
200 g$=""
210 get g$: if g$ = "" then 210
220 g = val(g$)
230 if g <=3 or g >=0 then if g>=1 then return
240 rem regel 250 wordt na testen gewijzigd.
250 if g$="j"org$="J"org$="n"org$="N"org$="C"org$="c"thenprint"ok":
    gosub 100
260 return
2000 rem autokosten
2010 rem jan auteur
2020 rem ***1985***
2030 rem weka basic tot machinetaal
2040 rem aanvulling 1
3000 rem menu
3010 rem strings om uitvoer te positioneren
3020 ho$="[REVERSE]                                                    "
3030 ve$="[REVERSE]                                                    "
```

16.1 Van keywords tot programma

```
3040 print"■"
3050 print left$(ho$,5) ; left$(ve$,5) ; "■autokosten"
3060 print
3070 print left$(ho$,5) ; "■1■ ■maken van basisbestand(en)"
3080 print left$(ho$,5) ; "■2■ ■kosten invoeren"
3090 print left$(ho$,5) ; "■3■ ■resultaten tonen"
3100 print left$(ho$,5) ; "■4■ ■stoppen"
3110 print
3120 print left$(ho$,5) ; "■Maak nu uw keuze"
3130 gosub 200
3140 on g gosub 4000,5000,7000,10000
3150 goto 3000
4000 rem maken basisbestanden
4010 rem de nu volgende regels kunnen in definitieve versie
     verwijderd worden
4020 print "■" ; left$(ho$,5) ; left$(ve$,10) ;
     "■maken basisbestanden"
4030 gosub 100
4040 return
5000 rem invoeren van kosten
5010 rem de nu volgende regels kunnen na testen verwijderd worden
5020 print "■" ; left$(ho$,5) ; left$(ve$,10) ; "■kosten invoeren"
5030 gosub 100
5040 return
7000 rem resultaten tonen
7010 rem de nu volgende regels kunnen na testen verwijderd worden
7020 print "■" ; left$(ho$,5) ; left$(ve$,10) ; "■resultaten tonen"
7030 gosub 100
7040 return
10000 rem einde
10010 end
```

Toelichting:
Regel 10 springt over de ruimte voor subroutines naar het hoofdprogramma. Dit begint op regel 2000. Om ruimte te hebben voor mededelingen, dimensioneren van geïndiceerde variabelen enzovoorts start de eerste module met regel 4000. Titel: maken van basisbestand(en).

De volgende module volgt op regel 5000, titel: invoeren van kosten. Omdat deze module ook het wegschrijven en inlezen van gegevens omvat rekent u op een omvangrijk gedeelte. Het rekenen en tonen van resultaten begint op regel 7000, einde van het programma op regel 10000. Op deze manier zijn er nog mogelijkheden om boven regel 10000 nuttige dingen op te nemen zoals een 'merge' (samenvoegen), een 'delete' (verwijderen), een 'renumber' (hernummeren) en een 'foutlees' routine voor de diskdrive.

Na het testen van dit menu kunnen overbodige regels verwijderd worden en blijft over:

16.1 Van keywords tot programma

```
10 goto 2000
100 fort=1to1000:next:return
200 getg$:ifg$=""then200
210 g=val(g$)
220 ifg<=3org>=0thenifg>=1thenreturn
230 if g$="j"org$="J"org$="n"org$="N"org$="C"org$="c"thenreturn
250 if g$="j"org$="J"org$="n"org$="N"org$="C"org$="c"thenprint"ok":
    gosub 100
3000 ho$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
3010 ve$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
3020 print"■":printleft$(ho$,5);left$(ve$,5);"■autokosten":print
3030 print left$(ho$,5) ; "■1■ ■maken van basisbestand(en)"
3040 print left$(ho$,5) ; "■2■ ■kosten invoeren."
3050 print left$(ho$,5) ; "■3■ ■resultaten tonen."
3060 print left$(ho$,5) ; "■4■ ■stoppen.":print
3070 print left$(ho$,5) ; "■maak nu uw keuze"
3080 gosub200:on g gosub 4000, 5000, 7000, 10000:goto3000
4000 rem basisbestanden
4010 return
5000 rem invoeren van kosten
5010 return
7000 rem resultaten tonen
7010 return
10000 rem einde
10010 end
```

Er zijn zoals u weet twee mogelijkheden om gegevens in te voeren, namelijk met INPUT en met GET. Allebei hebben ze voor- en nadelen. Met INPUT kunt u een hele string tegelijk invoeren, maar het gaat mis als het type van de invoer niet overeenstemt met de soort variabele. Met GET kunt u maar één teken tegelijk invoeren, bovendien reageert de computer op controletoetsen. Hiertegen moet u beveiligen en gegevens van meer tekens lengte moet u teken voor teken opbouwen. In deze aflevering wordt INPUT gebruikt voor de invoer. Later leert u ook met GET werken.

De eerste module, het maken van de basisbestand(en).

Zoals bekend zijn een aantal gegevens noodzakelijk. Ten eerste *het kenteken*. Dit slaan we op in de variabele KENT$. Let u bij de naamgeving van de variabelen erop dat er geen keywords in de naam zitten? Dit geeft gegarandeerd foutmeldingen. Kentekens hebben de vorm 'LL-CC-LL'. Behalve een controle via het scherm maken we ook een kentekencontrole die alle invoer die niet de voorgeschreven vorm heeft weigert. Kentekens kunnen ook gelijk als bestandsnaam gebruikt worden. Ze zijn dan nodig in de module basis, invoer en uitvoer. Invoer en controle schrijft u dus het beste als subroutine.

16.1 Van keywords tot programma

```
300 rem subroutine kenteken
310 print"■"
320 print left$(ho$,5) ; left$(ve$,5) ; "voer het kenteken in"
330 print left$(ho$,5) ; " in de vorm "
340 print
350 print left$(ho$,5) ; " 11-cc-11 "
360 print
370 input"voer kenteken in";kent$
380 rem onjuiste invoer weigeren
390 if len(kent$) <> 8 then gosub 600:goto 300
410 rem eerste en laatste twee karakters moeten letters zijn
420 rem derde en zesde karakter moet een streepje zijn
430 rem middelste twee karakters moeten cijfers zijn
440 n=1
450 h$ = mid$(kent$,n,1)
460 ifn=1and h$="0" or n=1 and val(h$)<>0thengosub600:goto300
470 ifn=2and h$="0" or n=2 and val(h$)<>0thengosub600:goto30
480 if n=3and asc(h$)<>45 then gosub600:goto300
490 if n=4and asc(h$)<48 or n=4 andasc(h$)>57 then gosub 600:goto300
500 if n=5and asc(h$)<48 or n=5 andasc(h$)>57 then gosub 600:goto300
510 if n=6and asc(h$)<>45 then gosub600:goto300
520 ifn=7and h$="0" or n=7 and val(h$)<>0thengosub600:goto300
530 ifn=8and h$="0" or n=8 and val(h$)<>0thengosub600:goto300
540 n=n+1
550 if n<=len(kent$) then 450
560 return
```

Toelichting:
De eerste en laatste twee karakters moeten letters zijn, de middelste twee moeten cijfers zijn. De scheidingen moeten streepjes zijn in verband met later printen van rapporten. De eenvoudigste controle is via de ASCII-tabel. Ook de functie VAL(A$) kunt u toepassen, want letters geven een waarde 0. Hoofdletters, zoals normaal voor kentekens, hebben codes van 65 tot en met 57. Het streepje heeft nummer 45. Als een kenteken een lengte heeft van meer of minder dan 8 karakters dan is het zeker niet goed.

Hier dient zich een volgende subroutine aan. Als een invoer niet juist is, door het programma gecontroleerd op logische criteria, moet hiervan een mededeling op het scherm komen zodat de gebruiker weet wat er aan de hand is. Omdat u vanuit de ene subroutine de andere kunt aanroepen ontstaat er al een samenhangend stuk programma.
Ook al is een gegeven voor de computer acceptabel dan hoeft het nog niet juist te zijn. Alle invoer wordt daarom nogmaals op het scherm afgedrukt en dan moet bevestigd worden of deze invoer al dan niet correct is. Net als een keuze maken uit het menu gebeurt dit via het toetsenbord. Met behulp van GET kunt u een karakter uit de toetsenbordbuffer halen. GET accepteert echter ook controletoet-

16.1 Van keywords tot programma

sen! Ook hier is dus beveiliging tegen ongelukken nodig. De enige karakters die u nodig heeft zijn 'J' of 'j' voor ja, 'N' of 'n' voor nee en de cijfers 1, 2, 3 en 4.

```
100 fort=1to1000:next:return
200 getg$:ifg$=""then200
210 g=val(g$)
220 ifg<=3org>=0thenifg>=1thenreturn
230 if g$="j"org$="J"org$="n"org$="N"org$="C"org$="c"thenreturn
250 goto 200


600 rem subroutine die mededeelt dat invoer niet juist was
610 print"■"
620 print left$(ho$,5) ; left$ (ve$,5) ; "■deze invoer was
    onmogelijk."
630 print left$(ho$,5) ; "■ opnieuw invoeren "
640 gosub 100
650 return
```

Toelichting: De computer wacht gedwongen op invoer. De invoer wordt opgeslagen in G$ en getest op kwaliteit. Bij onjuiste invoer volgt weer een mededeling.

Merk van de auto. Dit is een gegeven dat alleen maar voor het eventuele rapport nodig is. Hier volstaat een controle via het beeldscherm. Het merk wordt opgeslagen in MERK$.

Kilometerstand bij aanschaf. Dit moet duidelijk een getal zijn. Als enige beperkingen gelden hier een logisch gegeven – een kilometerstand is nooit negatief – en een door de programmeur te bepalen beperking. U kunt stellen dat u nooit een auto zal kopen die meer dan 50000 kilometer heeft gereden. Invoer als string is het prettigste. De kilometerstand bij aanschaf wordt opgeslagen in de variabele KILA$.

Is de auto *contant betaald of gefinancierd?* Hiervoor is een letter voldoende, namelijk 'C' of 'c' voor contant en 'F' of 'f' voor gefinancierd. Ook deze letters moeten dus in de subroutine voor toetsenbordinvoer worden geaccepteerd!
Opslaan in CONT$ kan niet, dit is immers een keyword, CASH$ is bruikbaar.

```
700 print "■"
710 print left$(ho$,5) ; left$(ve$,10) ; "voer kilometerstand bij
    aanschaf in"
720 print "als string dus tussen aanhalingstekens"
730 input"kilometerstand?";kila$
740 rem er mogen alleen cijfers in staan. decimale punt mag ook.
```

16.1 Van keywords tot programma

```
750 n=1
760 h$=mid$(kila$,n,1)
770 if asc(h$)<46orasc(h$)>46andasc(h$)<48 or asc(h$)>57
    then gosub600:goto710
790 n=n+1
800 if n<len(kila$) then 760
810 if val(kila$)>50000 then 760
820 return
900 rem routine om aanschaffingsprijs in te voeren en te behandelen
905 d1$=".00":d2$="0":d3$="00"
910 print"■"
920 print left$(ho$,5) ; left$(ve$,5) ; "voer aanschaffingsprijs
    in "
930 inputaans
935 if aans=<0 or aans> 15000then gosub600:goto910
940 aans$ = str$(aans)
950 rem kijken waar decimale punt zit
960 n = 1
970 h0$ = mid$(aans$,n,1)
980 if asc(h0$) = 46 then 1030
990 n = n + 1
1000 if n <= len(aans$) then 970
1015 aans$=aans$+d1$:goto1060
1030 if n = len(aans$) - 1 then aans$=aans$+d2$:goto1060
1040 if n = len(aans$) - 2 then aans$=aans$+d3$:goto1060
1050 ifn<len(aans$)-2thenaans=int(100*aans+0.5)/100:
     aans$=str$(aans):goto1060
1060 rem aans$ aanvullen tot 9 cijfers met shift spaties
1070 d4$="          "
1080 aans$=left$(d4$,(len(d4$)-len(aans$)))+aans$
1090 return
1100 rem subroutine contant betaald
1110 print"■"
1120 print left$ (ho$,5) ; left$(ve$,5) ; "voer bij contant
     betaling een c in "
1140 gosub 200
1150 if g$="c" then cash$=g$
1160 return
1200 rem subroutine datuminvoer
1210 print "■"
1220 print left$(ho$,5) ; left$(ve$,5) ; " voer datum in "
1230 print left$(ho$,5) ; " in de vorm ";chr$(34);"dd-mm-jjjj";
     chr$(34)
1240 print left$(ho$,5) ; " dus als string tussen
     aanhalingstekens "
1250 inputadat$
1260 if len(adat$)<>10 then gosub 600:goto 1200
1270 rem er mogen alleen cijfers in staan of streepjes
1280 n=1
1290 h$=mid$(adat$,n,1)
1300 if asc(h$)=45 then 1310
```

16.1 Van keywords tot programma

```
1305 if asc(h$)<48 or asc(h$)>57 then gosub600:goto1200
1310 n=n+1
1320 if n<= len(adat$) then 1290
1330 sc = asc(mid$(adat$,6,1))
1340 if sc <> 45 then gosub 600:goto1200
1350 j = val(right$(adat$,4))
1360 if j < 1980 or j > 1990 then gosub 600: goto 1200
1370 d=val(left$(adat$,2))
1380 if d < 1 or d > 30 then gosub 600:goto1200
1390 sc = asc(mid$(adat$,3,1))
1400 if sc <> 45 then gosub 600:goto1200
1410 m = val(mid$(adat$,4,2))
1420 if m<1orm>12thengosub600:goto1200
1430 return
```

De aankoopdatum. Data kunnen op veel manieren worden ingevoerd. Goed bruikbaar is 'DD-MM-JJJJ' (D=dag, M=maand, J=jaar). Aankoopdatum wordt opgeslagen in ADAT$. Ook hier datumcontrole in een subroutine. Er zullen vaker data moeten worden ingevoerd. Wel zijn er logische beperkingen, het aantal dagen 1 tot en met 31, het aantal maanden 1 tot en met 12 en het jaar. Dit programma maakt u in 1985/1986. Als u geen jaartal groter dan 2000 accepteert kunt u 15 jaar plezier hebben van het programma. Onwaarschijnlijk is dat u een auto koopt ouder dan vijf jaar. Jaartallen kleiner dan 1980 kunt u dus rustig weigeren. Nu moet u ook beslissen of u rekening houdt met schrikkeljaren en met het exacte aantal dagen van een jaar, dit met het oog op de kosten per dag en een eventueel renteverlies. Het eenvoudigste is het jaar op 360 dagen te stellen en de maanden op maximaal 30 dagen. Een eventuele datum met als eerste cijfers 31 wordt geweigerd.

Aanschafprijs. Behalve dat deze niet negatief kan zijn kunt u zelf in uw portemonnaie kijken wat uw maximum te besteden bedrag zal zijn. Hier wordt dit gesteld op 12500 gulden. Omdat geregeld bedragen moeten worden ingevoerd beslist u nu hoe deze er uit moeten zien en hoe ze wel en niet worden geaccepteerd. Bedragen moeten twee cijfers achter de decimale punt hebben. Is er geen '.' te vinden bij de laatste karakters, dan moet achter de prijs '.00' geplaatst worden (of '00' of '0', al naar gelang de positie van de decimale punt). U moet wel heel lang met uw auto rijden om een totaal bedrag van kosten te krijgen dat groter is dan 9 cijfers. Elk bedrag, dat minder posities inneemt, moet aan de voorzijde met het noodzakelijke aantal spaties worden verlengd. Invoeren als getal AANS en opslaan in AANS$.

Als al deze gegevens zijn ingevoerd worden ze nogmaals afgedrukt op het scherm voor een visuele controle. Na akkoord moeten de gegevens worden opgeslagen. Hier moet een keuze gemaakt worden tussen diskdrive en cassetterecorder en moet een bestandsnaam worden gekozen. Hiervoor zou u het kenteken gebruiken. Denkt u erom dat u bij de diskdrive foutmeldingen kunt krijgen, onder andere als er al een bestand met

16.1 Van keywords tot programma

dezelfde naam op de schijf staat. Als u bij regel 50000 de foutleesroutine van de diskdrive (Deel 7 hoofdstuk 15.1 blz. 2) plaatst ziet u via GOTO 50000 direct wat er aan de hand is.

Om een nette uitvoer te krijgen is het praktisch om na het schoonmaken van het scherm gebruik te maken van twee strings om de cursor te positioneren, namelijk HO$ voor de horizontale positie en VE$ voor de verticale positie. Via LEFT$ kunt u dan gebruiken wat u nodig acht.

Rest nog het wegschrijven van de gegevens en het kijken of er nog meer auto's zijn gekocht. Het laatste is uiteraard een scherm-mededeling en een J/N invoer, voor het eerste maakt u weer het gemakkelijkst gebruik van een subroutines. De gegevens die u moet wegschrijven zijn namelijk hetzelfde bij cassetterecorder en diskdrive.

```
1500 rem keuze tussen diskdrive of cassetterecorder
1510 print"{clr}"
1520 print left$(ho$,5) ; left$(ve$,5) ; "{rvs on}1.{rvs off} {rvs on}diskdrive"
1530 print left$(ho$,5) ; "{rvs on}2.{rvs off} {rvs on}cassetterecorder"
1540 print
1550 print left$(ho$,5) ; "{rvs on}maak uw keuze"
1560 gosub 200
1570 on g gosub 1600
1580 on g gosub 1700
1590 gosub 200
1600 open2,8,2,"0:"+kent$+",s,w"
1610 return
1700 open2,1,1,kent$
1710 return
```

```
2000 rem
3000 ho$="{38 crsr right}"
3010 ve$="{24 crsr down}"
3020 print"{clr}":printleft$(ho$,5);left$(ve$,5);"{rvs on}autokosten":print
3030 print left$(ho$,5) ; "{rvs on}1{rvs off} {rvs on}maken van basisbestand(en)"
3040 print left$(ho$,5) ; "{rvs on}2{rvs off} {rvs on}kosten invoeren."
3050 print left$(ho$,5) ; "{rvs on}3{rvs off} {rvs on}resultaten tonen."
3060 print left$(ho$,5) ; "{rvs on}4{rvs off} {rvs on}stoppen.":print
3070 print left$(ho$,5) ; "{rvs on}maak nu uw keuze"
3080 gosub200:on g gosub 4000, 5000, 7000, 10000:goto3000
4000 gosub300
4010 rem eindelijk een invoer zonder problemen
4020 print "{clr}"
4030 print left$(ho$,5) ; left$(ve$,10) ; " voer nu het merk van
     de auto in"
4040 input merk$
4050 gosub 700
4060 gosub 900
```

16.1 Van keywords tot programma

```
4070 gosub1100
4080 gosub1200
4090 rem visuele controle op invoer
4100 print "■"
4110 print left$(v$,5) ;kent$
4120 print merk$
4130 print kila$
4140 print aans$
4150 print cash$
4160 print adat$
4170 print
4180 print "■is deze invoer accoord j/n"
4190 gosub 200
4200 if g$="n" then 4000
4300 gosub1500
4310 print#2,kent$
4320 print#2,merk$
4330 print#2,kila$
4340 print#2,cash$
4350 print#2,adat$
4360 close2
5000 rem invoeren van kosten
5010 return
7000 rem resultaten tonen
7010 return
10000 rem einde
10010 end
50000 open15,8,15
50010 input#15,a$,b$,c$,d$
50020 printa$,b$,c$,d$
```

Toelichting: Om zoveel mogelijk geheugen te sparen en snelheid te winnen zijn alle REM statements verwijderd en zijn de regels zo vol mogelijk gemaakt.

Lijst met de tot nog toe gebruikte variabele-namen

HO$	voor horizontale positie van uitvoer.
VE$	voor verticale positie van uitvoer
G$	variabele om keuze in menu's op te slaan.
G	waarde van G$.
T	teller in vertragingslus.
KENT$	kenteken, tevens bestandsnaam.
MERK$	merk van de auto.
KILA$	kilometerstand bij aanschaf.
KMREK$	kilometerstand op de 'uitvoer' dag.
KM	gereden aantal kilometers.
AANS$	aanschaffingsprijs als string.
AANS	aanschaffingsprijs.

16.1 Van keywords tot programma

REST$	restwaarde op 'uitvoer' dag.
ADAT	datum van aanschaf.
RDAT$	datum van berekenen, 'uitvoer' dag.
A$	in gebruik bij opsporen van diskfouten.
B$	idem
C$	idem
D$	idem
CASH$	geeft aan of de auto contant betaald is.
N	teller in lusconstructies.
S	teller bij het verwerken van bonnen.
BDAT$(N)	datum van bon nummer N.
SOT$(N)	kostensoort van bon N.
LITER$(N)	aantal liters benzine van bon N.
BEDR$(N)	bedrag van bon N.
DR	aantal dagen van 'uitvoer' datum.
D	aantal dagen van aanschafdatum.
DG	aantal dagen dat we de auto gebruiken.
RT	renteverlies.
RV	rentevoet.
IB	geinvesteerd bedrag.
LT	totaal aantal liters benzine.
TK	totale kosten van de auto.
H$	hulpstring
HO$	hulpstring
D1$	om getallen te verlengen met decimale punt en nullen.
D2$	om getallen te verlengen met een 0.
D3$	om getallen te verlengen met twee nullen.
D4$	om getallen op 9 posities lengte te brengen.
SC	value van scheidingskarakter.
J	aantal jaren.
M	aantal maanden.

In de komende aanvulling wordt het programma verder uitgebreid met het invoeren van de kosten en het toevoegen van deze kosten aan het basisbestand.

Vergeet u niet om het ontwikkelde gedeelte geregeld te saven? Het zou jammer zijn als u een stroomstoring kreeg op het moment dat uw programma bijna klaar is. Als u met cassetterecorder werkt, kunt u gewoon over de oude versie van het programma heen schrijven. Met de diskdrive is het iets gecompliceerder. Hiervoor zijn drie mogelijkheden.

1) Gebruik de 'REPLACE' optie met '@', zie de handleiding van uw drive. Dit commando is echter onbetrouwbaar. U kunt daarom beter een van de andere methodes volgen.
2) Geef het oude programma een andere naam via RENAME en SAVE daarna uw nieuwe versie. Er kan dan niets gebeuren als de stroom uitvalt.

16.1 Van keywords tot programma

3) SCRATCH eerst het programma op schijf en SAVE daarna uw huidige versie. Als uw computer op dit moment in staking gaat heeft u geen programma meer. Methode twee is de beste.

U kunt ook alle problemen oplossen door tijdens uw werk te saven onder de namen 0 t/m 9, uw huidige versie als 'VI' en daarna alle bestandsnamen van 1 karakter lengte te scratchen met het commando:
OPEN15,8,15,"SO:?":CLOSE15

7/17

Tips, truuks en utilities

Inhoud

7/17.1

POKE-opdrachten

Het BASIC POKE-commando wordt gebruikt om geheugenplaatsen een nieuwe inhoud te geven. Nu zijn er in de 64 bepaalde geheugenplaatsen die belangrijke gegevens bijhouden. Door deze gegevens te veranderen kan de 64 verteld worden bepaalde dingen juist wel of niet te doen. Het is bijvoorbeeld mogelijk om alle toetsen repeterend te maken. De lijst hieronder bevat een aantal POKE-instructies met hun werking.

POKE 788,49	Schakel RUN/STOP uit
POKE 788,52	Schakel RUN/STOP in
POKE 808,234	Schakel RUN/STOP-RESTORE uit
POKE 808,237	Schakel RUN/STOP-RESTORE in
POKE 816,32	Verandert SHIFT-RUN/STOP in RUN(!)
POKE 816,165	SHIFT-RUN/STOP weer normaal
POKE 775,200	Schakel LIST uit
POKE 775,167	Schakel LIST in
POKE 649,0	Schakel toetsen uit
POKE 649,10	Schakel toetsen in
POKE 19,64	INPUT zonder vraagteken
POKE 19,0	INPUT normaal
POKE 24,37	LIST-en zonder regelnummers
POKE 650,255	Alle toetsen repeterend
POKE 650,127	Geen enkele toets repeterend
POKE 650,0	Cursor/spatiebalk repeterend (normaal)
POKE 818,32	Schakel SAVE uit
POKE 818,237	Schakel SAVE in
POKE 816,32	Schakel LOAD uit
POKE 816,165	Schakel LOAD in
POKE 646,X	Geef cursor kleur X
POKE 56325,X	Verander snelheid van de cursor
POKE 56325,58	Cursorsnelheid normaal
POKE 56334,0	Zet interrupt uit
POKE 56334,1	Zet interrupt aan
POKE 657,128	Schakel SHIFT-Commodore toets uit
POKE 657,0	Schakel SHIFT-Commodore toets in
POKE 204,0	Zet cursor aan (bij GET)
POKE 204,1	Zet cursor uit
POKE 198,0	Leeg toetsenbordbuffer
POKE 212,0	Zet quote-mode uit
POKE 212,1	Zet quote-mode aan
POKE 199,01	Zet RVS on
POKE 199,0	Zet RVS off
POKE 1,15	Schakel cassettemotor aan
POKE 1,47	Schakel cassettemotor uit
POKE 1,54	Schakel BASIC uit
POKE 1,53	Schakel BASIC en O.S. uit
POKE 1,51	Schakel tekenROM in
POKE 1,55	Schakel BASIC en O.S. in
POKE 53265,11	Schakel scherm uit

17.1 POKE-opdrachten

	(computer 10% sneller)
POKE 53265,27	Schakel scherm in
POKE 56,X	Verlaag MEMTOP naar X*256
POKE 56,160	Zet MEMTOP normaal

Er zijn ook 'dubbele' POKE-opdrachten en POKE-opdrachten gevolgd door een SYS-commando. Dat houdt in dat na zo'n opdracht de 64 nog niet weet dat er iets veranderd is. De SYS-opdracht maakt een en ander dan kenbaar. Hieronder volgt een lijstje met 'gecompliceerde' POKE-opdrachten:

POKE 2051,255:POKE 2052,255
Deze POKEs geven de eerste regel van het programma het nummer 65535. Deze is normaal niet meer te verwijderen en kan dus voor copyright-doeleinden gebruikt worden.

POKE 56324,28:POKE 56325,0
Deze POKEs vertragen de 64 in zijn werking zodat bijvoorbeeld listings langzamer over het scherm lopen.

POKE 2050,8:SYS 42291 <Return>
POKE 46,PEEK(35)–(PEEK(781)>253):POKE 45,PEEK(781)+2 AND 255:CLR <Return>
Deze commando's maken een NEW of een reset ongedaan. Het BASIC-programma komt weer terug en is weer te gebruiken. Maak geen typefouten, anders is het programma voorgoed verloren.

POKE 781,X:POKE 782,Y:SYS 65520 of POKE 214,X:POKE 211,Y:SYS 58732
Deze twee series commando's doen het zelfde, ze brengen de cursor namelijk naar de positie X,Y op het scherm.

POKE 781,R:SYS 59903
Deze opdrachten wissen regel R op het scherm.

Om twee programma's aan elkaar te plakken (MERGE) kan na het laden van het eerste programma de volgende constructie gebruikt worden:
A=PEEK(45)+256*PEEK(46)–2:
POKE 43,A AND 255:POKE 44,A/256
Nu kan programma 2 geladen worden, waarna alleen nog
POKE 43,1:POKE 44,8
gegeven hoeft te worden. Let er wel op dat de regelnummers van programma 2 hoger moeten zijn dan die van programma 1.

7/17.2

Data-creator

Veel lezers hebben ons gevraagd hoe het mogelijk is om machinetaal in BASIC-programma's te verwerken. Een ervan is om de gewenste machinetaal gewoon te laden met het commando
IF A=0 THEN A=1:LOAD"ML",8,1
(of ,1,1)
De machinetaal moet dan wel met behulp van een monitor o.i.d. op de disk of tape gezet zijn.

Soms echter is het niet mogelijk om een machinetaalfile van tape of disk te laden en moet de machinetaal dus bijvoorbeeld in data-regels staan. Ook voor sprite-definities worden data-regels veelvuldig gebruikt. Wanneer de data echter al in het geheugen staat moet deze op het scherm gezet worden (PRINT PEEK (I);) en op papier overgenomen worden. Daarna kan het moeizame intypen van data-regels beginnen.
De data-creator maakt een eind aan deze problemen. Na het maken van de data met een monitor of sprite-editor wordt de data-creator geladen en geRUNd. Start- en eindadres ingeven, alsmede het regelnummer waar de DATA's moeten beginnen. Na een tijdje is het programma klaar en hoeven alleen de regels van de data-creator zelf verwijderd te worden. Dit zijn er slechts achttien, dus dat is niet al te veel werk. Het programma is zeer kort, maar des te functioneler. Een voorbeeld: alle BASIC-laders bij onze machinetaallistings zijn met de data-creator gemaakt. Zelfs de checksum wordt gegenereerd!

```
10 INPUT"START ADRES";SA
15 INPUT"EIND ADRES";EA:IFEA<=SATHEN10
20 INPUT"STARTREGEL:  1000■■■■■■■";FL:IFFL<200THEN20
30 LN=FL
50 I=SA
55 IFI>EATHEN150
60 PRINT"■"MID$(STR$(LN),2)"DATA ";
70 FORJ=0TO(((EA-I)<15)*(I-EA))-14*((EA-I)>14)
80 PRINTMID$(STR$(PEEK(I+J)),2)",";:S=S+PEEK(I+J)
90 NEXT
100 PRINT"■ "
110 PRINT"S="S":LN="LN+10"■:I="I+15"■:EA="EA"■:SA="SA"■:GOTO55"
130 POKE631,19:POKE632,13:POKE633,13:POKE634,147:POKE198,4
140 END
150 PRINT"■"MID$(STR$(LN),2)"FOR I="MID$(STR$(SA),2)" TO"EA"■";
160 PRINT":READ A:S=S+A:POKE I,A:NEXT"
170 PRINTMID$(STR$(LN+10),2)"IF S<>"MID$(STR$(S),2)" THEN PRINT"CHR$(34);
180 PRINT"FOUT IN DATA"CHR$(34)":END":GOTO130

READY.
```

17.2 Data-creator

De werking

De data-creator maakt gebruik van de keyboard-buffer. Het scherm wordt schoon gemaakt, er wordt een regelnummer gePRINT, gevolgd door het woord DATA. Dan verschijnen er vijftien waardes gescheiden door komma's, deze waardes worden rechtstreeks uit het geheugen gelezen. Het probleem is nu om deze 'DATA-regel' aan het programma toe te voegen. Daartoe wordt de input-buffer met twee toetsen gePOKEt. Een HOME en een RETURN. Wanneer nu het programma beëindigd wordt (d.m.v. END) zal de computer denken dat er ondertussen nieuwe toetsen zijn ingedrukt en deze ook uitvoeren; de cursor verdwijnt naar boven en er volgt een RETURN. Gevolg: de regel wordt aan het programma toegevoegd.

Tot zover is het allemaal nog vrij eenvoudig. Het probleem doet zich echter voor dat door het toevoegen van een regel alle variabelen zijn gewist en dat het programma ook beëindigd is. Het zou alleen leuk zijn als er meer dan 1 regel toegevoegd kon worden. Dat kan dan ook, daartoe wordt op de tweede regel de volgende informatie gePRINT:
S= 123 :LN= 1010:I= 49167:EA= 50000:
SA= 49152:GOTO 55
(de waardes kunnen anders zijn).
Door nu een tweede RETURN in de inputbuffer te POKEn zullen de variabelen hun oude waardes terugkrijgen en zal het programma op regel 55 verder gaan.
Wanneer I (de lopende variabele) groter dan EA is geworden tenslotte worden er twee andere regels gePRINT, namelijk de regels FOR I= . . .: NEXT en IF S <> . . .:END.

De commando's in regel 130 vullen de inputbuffer inderdaad met een HOME (19), twee RETURNS (13) en een CLR (147). De POKE 198,4-opdracht zet de inhoud van de inputbuffer op vier (dit is noodzakelijk!).
Regel 70 ziet er wellicht een beetje ingewikkeld uit, dit is gedaan om het aantal resterende geheugenplaatsen te berekenen. Normaal gaan er 15 DATA's op een regel, wanneer er nog slechts 10 geheugenplaatsen over zijn hoeft er dus slechts tot (I-EA) doorgeteld te worden. De eindwaarde van de NEXT-loop is namelijk I-EA als EA-I<15 en 14 als EA-I>14. De vergelijkingen (EA-I)<15 en (EA-I)>14 leveren namelijk 0 of -1 op (is de ene 0 dan is de andere -1 en omgekeerd).

7/17.3

Menu-programma

In deze paragraaf zullen we het menu-programma van de BASIC-diskette eens nader onder de loep nemen. Op die manier is dit programma voor eigen gebruik in te zetten en naar believen uit te breiden. Het is niet onmogelijk om het programma tot een gebruikersvriendelijk disk operating system uit te breiden, zoals we later ook zullen zien. Allereerst echter zal de essentie van het programma uitgelegd worden zodat we weten waar we over praten.

Het zal uiteraard opgevallen zijn dat, wanneer de ↑-toets ingedrukt wordt, het menu direct verschijnt. Dit verraadt een permanenete aanwezigheid in het geheugen. Dit kan niet anders, want normale BASIC-programma's 'draaien' gewoon, echter het menu-programma is ook in BASIC geschreven. Zodra de gebruiker het wenst kan het normale BASIC-programma verlaten worden en kan, middels het ↑-commando, het menu opgestart worden.

Het tweede punt dat opvalt is dat de eerste keer dat het menu gedraaid wordt er een inleidende tekst verschijnt. Wordt later echter het menu opgeroepen dan is die hele tekst verdwenen. Kennelijk wordt of niet het hele menu-programma bewaard of wordt een gedeelte telkens overgeslagen. Zoals later zal blijken is het eerste waar, het menu-programma zoals dat van disk gelezen wordt wordt slechts gedeeltelijk 'opgeborgen', ten eerste scheelt dit geheugenruimte en ten tweede is de inleidende tekst leuk voor 1 keer, daarna gaat ze echter vervelen.

Een volgende vraag die rijst is: waar wordt het menu-programma opgeslagen? Mensen die wat dieper in de 64 thuis zijn kunnen hier zonder problemen een antwoord op geven: onder het ROM. Zoals wellicht bekend is bezit de 64 evenzovele kilobytes aan RAM als zijn naam doet vermoeden. Slechts 38 zijn er voor BASIC beschikbaar. De overige 26 zijn er wel maar zijn voornamelijk gereserveerd voor ROM-gedeeltes en BASIC-uitbreidingen. Nu is het echter mogelijk om dat RAM (want het is er wel, ook al zie je het niet), toch te gebruiken. Het kan op de normale manier worden volgePOKEt, het is alleen niet meer te PEEKen, tenminste niet zonder meer. In machinetaal kan het gelukkig wel, vandaar ook dat het menu-programma iets aan machinetaal bevat. Drie gedeeltes om precies te zijn: één om het programma op te bergen, één om het programma terug te lezen en één om de disk-directory in nette kolommen op het scherm te tonen. Dit laatste kan ook in BASIC, maar dat gaat dan akelig traag. Vandaar. Waar die machinetaal is opgeborgen is niet zo belangrijk, wellicht verdiepen we ons daar later nog eens in.

17.3 Menu-programma

Behalve bovengenoemde problemen en vragen is er nóg een punt waar we rekening mee moeten houden: Commodore 64-BASIC kent geen chaining. Wat dat inhoudt? Soms is het nodig en/of nuttig om vanuit het ene programma een volgend programma te laden en te RUNnen. In feite gebeurt dat hier ook: het menu-programma laadt het door ons gekozen programma. Ook dat gebeurt door middel van een truuk, omdat dit normaal niet mogelijk is. De volgende constructie
10 LOAD "PROGRAMMA2",8
werkt namelijk niet omdat bij het laden vanuit een ander programma de pointers naar het eind van het programma niet opnieuw ingesteld worden. Wanneer het tweede programma korter is dan het eerste is dat niet erg, echter wanneer het tweede programma langer is dan wordt dit dus over de originele eindpointers heen geschreven. Op die plaats staan helaas de variabelen, dus wanneer in het tweede programma een nieuwe variabele gedefinieerd wordt (bijvoorbeeld in FOR I=1 TO 100:NEXT) komt deze midden in het tweede programma te staan met als gevolg dat het tweede programma verminkt wordt.

Een betere oplossing is dus om niet vanuit het menu het tweede programma te laden, maar om dit ná het menu-programma te doen. Dit kan door de computer te laten denken dat bijvoorbeeld de volgende commando's gegeven worden na dat het menu-programma is afgelopen:
LOAD "PROGRAMMA",8
RUN
Dit is mogelijk door deze twee commando's op het scherm te PRINTen en er dan met twee RETURNs overheen te gaan. Hoe dat in zijn werk gaat zien we dadelijk.

Nu de diverse (eigen)aardigheden van het programma duidelijk zijn kunnen we de programmalisting aan een nadere beschouwing onderwerpen.

```
10 poke56,159:clr
15 fori=832toi+41:pokei,.:next:fori=itoi+20:readz:pokei,z:next:mv=2000
20 data255,255,255,170,8,65,170,56,193,130,9,65,130,56,221,130,8,85
30 data255,255,255
40 poke53280,6:poke53281,6:v=53248:pokev,32:pokev+1,65:poke2040,
   13:pokev+23,1
50 pokev+29,1:pokev+39,0
60 print"■■■┌──────┐":a$="│      │":forj=1to6:printa$:next
70 pokev+21,1:print"└──────┘"
80 print"■■■"tab(18)"Weka-Uitgeverij"
90 printtab(18)"Donker Curtiusstraat 7";
100 printtab(58)"1051 JL Amsterdam"
110 printtab(18)"020-867131■■":print"──────────────────────────────────────■"
120 print"Geachte abonnee,■"
130 print"Een belofte waargemaakt! Enige aanvul-"
140 print"lingen geleden hebben we onze abonnees"
150 print"een WEKA-diskette beloofd met daarop"
160 print"de originele 'Van Basic tot Machinetaal'";
```

17.3 Menu-programma

```
170 print"software. Welnu, dit is 'm dan. Volge-"
180 print"stouwd met zowat alle programma's uit de";
190 print"hoofdstukken 6, 7, 9, 10 en 11. WEKA"
200 print"zou echter WEKA niet zijn als er geen"
210 print"'extraatjes' zouden zijn, vandaar dat"
220 print"deze disk nog enige kerstcadeautjes be-"
230 print"vat, waaronder o.a. versie 0.1 van onze"
240 print"BASIC-uitbreiding."
250 fori=1tomv:next:poke198,0
260 printtab(30)"<TOETS>▮"
270 fori=1to250:next:poke646,1-peek(646):ifpeek(198)=.then260
280 forx=10to24:poke781,x:sys59903:next:poke198,0
290 print"▮▮▮▮▮▮▮▮▮▮▮▮▮Alle programma's kunnen vanuit dit menu"
300 print"geladen worden, het is echter ook moge-"
310 print"lijk ze los van de disk te laden. De"
320 print"programma's ontlenen hun naam aan de"
330 print"paragraaf in het boek waar ze vandaan"
340 print"stammen, eventueel gevolgd door een     ";
350 print"extra getal indien er meer dan een pro-"
360 print"gramma in die paragraaf staat. We hopen ";
370 print"zo dat u veel plezier aan deze disk zult";
380 print"beleven en dat u er ook nog het een en"
390 print"ander van opsteekt.▮"
400 print"Prettige kerstdagen en een gelukkig"
410 print"nieuw jaar!"
420 printtab(15)"WEKA Uitgeverij B.V.";
425 fori=1tomv-1000:next:fori=896toi+126:pokei,.:next:poke198,0
430 fori=896+46toi+17step3:reada:pokei,a:next
435 fori=1004toi+17step3:reada:pokei,a:next:fori=49152to49318:reada:
    pokei,a
436 next:fori=40797to40959:reada:pokei,a:next:sys40797:sys40930
440 printtab(36)"<T>▮"
450 fori=1to250:next:poke646,1-peek(646):ifpeek(198)=.then440
460 data4,6,255,255,6,4    ,32,96,255,255,96,32
490 d1=49152:v=53248:poke53280,6:poke53281,6
500 pokev+21,0:print"▮▮▮▮┌──────────────────────────────────────┐";
510 a$="|                                      |"
520 fori=1to21:printa$;:next
530 print"└───┘                                  └───┘"
540 print"▮▮▮▮▮▮▮▮┌──────────────────────────────────┐▮▮▮▮";
550 print"▮▮▮▮|                                   |"
560 print"▮▮▮▮▮|                                  |▮▮▮▮";
570 print"▮▮▮▮└──────────────────────────────────┘";
580 print"▮▮"tab(40);:sysd1:ym=peek(214)
600 print"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮Maak een keuze m.b.v. de cursor-"
610 print"▮▮▮▮ toetsen.  Druk op return om te"
620 ifpeek(2)thenprint"▮▮▮▮▮▮laden en op F1 voor het vervolg":goto640
630 print"▮▮▮▮▮▮▮▮laden en op F7 om te stoppen";
640 poke198,0:pokev+23,0:poke2040,14:poke2041,15:pokev+29,3
```

17.3 Menu-programma

```
645 pokev,8:pokev+1,44:pokev+21,1:pokev+39,1:pokev+40,1:k=0:y=0
650 geta$:ifa$=""then650
660 ifa$=""andpeek(2)thendl=49309:goto500
670 ifa$=""thensys49301:print"";:poke53272,21:poke53280,14:
    pokev+21,0:end
680 ifa$=""ora$=""thenk=1-k
690 ifa$=""theny=y-1:ify<0theny=ym-1:ifpeek(1066+y*40+k*20)=32theny=y-1
700 ifa$=""theny=y+1:ify>ym-2thenifpeek(1066+y*40+k*20)=32theny=0
710 ifpeek(1066+y*40+k*20)=32theny=yo:k=ko
720 ifa$<>chr$(13)then950
750 sys49301
780 n$="":fori=1066+y*40+k*20toi+15:a=peek(i):ifa>31anda<64then800
790 ifa<32thena=a+64
800 n$=n$+chr$(a):next
810 fori=16to1step-1:ifmid$(n$,i,1)<>" "thenne=i:i=0
820 next
830 n$=left$(n$,ne):pokev+21,0
840 print"┌──────────────────────────────────┐"
850 fori=1to3:print"│                                  │":next
860 print"└──────────────────────────────────┘"
870 printtab(12-len(n$)/2)n$" wordt geladen..."
880 poke631,19:poke632,13:poke633,13:poke634,76:poke635,73:poke636,83
890 poke637,84:poke198,7
930 print"load"chr$(34)n$chr$(34)",8"
940 print"print"chr$(34)""chr$(34)";:poke53272,21:poke53280,14":end
950 pokev,8+k*160:ko=k
960 pokev+1,44+y*8:yo=y
1000 goto650
1010 data169,36,133,251,169,251,133,187,169,0,133,188,169,1,133
1020 data183,169,8,133,186,169,96,133,185,32,213,243,165,186,32
1030 data180,255,165,185,32,150,255,169,36,141,193,2,169,0,133
1040 data144,169,29,32,210,255,32,210,255,160,18,132,251,32,165
1050 data255,133,252,164,144,208,82,32,165,255,164,144,208,75,164
1060 data251,136,208,233,32,165,255,166,144,208,63,170,201,34,208
1070 data244,169,0,141,192,2,238,192,2,32,165,255,166,144,208
1080 data43,201,34,240,5,32,210,255,208,237,56,169,21,237,192
1090 data2,170,169,29,32,210,255,202,208,250,206,193,2,208,10
1100 data169,36,141,193,2,169,1,133,2,96,160,2,208,163,32
1110 data66,246,169,0,133,2,96,169,29,32,210,255,32,210,255,208,234
3000 data169,234,133,20,169,1,133,21,165,43,166,44,32,23,166
3010 data166,95,165,96,134,251,133,252,162,0,169,168,134,253,133
3020 data254,165,45,141,254,191,165,46,141,255,191,56,233,7,170
3030 data160,0,177,251,145,253,136,208,249,230,252,230,254,202,208
3040 data242,96,169,1,133,253,169,8,133,254,169,0,133,251,169
```

17.35 Menu-programma

```
3050 data168,133,252,169,54,133,1,173,255,191,133,46,32,134,159
3060 data173,254,191,133,45,169,55,133,1,32,51,165,32,96,166
3070 data173,33,208,141,134,2,160,3,185,222,159,153,119,2,136
3080 data16,247,169,4,133,198,76,116,164,82,213,13,58,169,237
3090 data141,8,3,169,159,141,9,3,96,32,115,0,201,174,240
3100 data6,32,121,0,76,231,167,32,115,0,76,155,159
ready.
```

In regel 10 wordt een stukje geheugen vrijgemaakt voor de routine die ons ↑-commando kan detecteren en ook kan uitvoeren. Hier komt dus ook de routine te staan die het programma onder het ROM vandaan haalt en op de juiste plaats in het geheugen terugzet.
Regel 15, 20 en 30 definiëren een sprite die het WEKA-logo voorstelt. Iets wat alleen belangrijk is in deze versie van het menuprogramma, eigen versies hebben dit logo uiteraard niet nodig.
De regels 40 t/m 240 zijn in feite ook niet belangrijk, het enige wat hier gebeurt is dat het eerste gedeelte van de tekst getoond wordt. Dit gedeelte is dus naar believen aan te passen.
De regels 250, 260 en 270 bevatten een wachtlusje en zorgen er voor dat het woord ‹TOETS› knippert zolang er geen toets ingedrukt wordt. Het knipperen wordt veroorzaakt door de tekst "‹TOETS›" de ene keer in wit en de andere keer in zwart af te drukken. Het wisselen van de kleur gebeurt door de constructie:
POKE 646,1-PEEK(646)
Is de inhoud van locatie 646 gelijk aan 0 (zwart) dan wordt deze dus 1 en omgekeerd. Geheugenplaats 646 bevat de huidige PRINT-kleur, 198 (regel 270) bevat het aantal ingedrukte toetsen waarbij geldt dat er een 0 in staat wanneer er geen toets is ingedrukt.
Regel 280 bevat een leuke truuk die we ook in de vorige aanvulling zijn tegengekomen: SYS 59903 kan gebruikt worden om een regel te wissen. In regel 280 worden dus de regels 10 t/m 24 gewist, wat inhoudt dat het 'briefhoofd' blijft staan.
De regels 290 t/m 420 zijn wederom niet van belang, die tekst kennen we zo langzamerhand wel.
Regels 425, 430 en 435 zorgen er voor dat de twee pijltjes (ook dit zijn sprites) gedefinieerd worden. De data voor deze pijltjes staat in regel 460. Verder zorgt regel 435 voor het POKEn van de directory-machinetaal. Regel 440 doet hetzelfde, maar dan voor de machinetaal die van belang is voor het ↑-commando en het verplaatsen van het programma.
Het SYS 40797-commando schakelt het pijltje in, het SYS 40930-commando verplaatst een gedeelte van het oorspronkelijke menu-programma naar onder het ROM.
De regels 440 t/m 460 tenslotte wachten voor de tweede maal op een toets-druk.

Het verhaal wordt hier even onderbroken omdat we nu op het punt zijn aangekomen waar het 'tweede' of te wel ingekorte menu-progamma begint. Wanneer name-

17.3 Menu-programma

lijk op een gegeven moment het menu opgeroepen wordt en dit onderbroken wordt (via de RUN/STOP-toets) dan zal een listing verschijnen van de regels 490 en verder. Dit is een gevolg van het feit dat de machinetaalroutine de regels 490 en verder opslaat, alles wat daar voor staat vervalt dus. Dit is een gevolg van de DATAs 234 en 1 in regel 3000, want 234 + 1 * 256 = 490. Door dus die getallen te veranderen is het 'startadres van opbergen' aan te passen. Een 232 en een 3 respectievelijk leveren als resultaat 232+3*256=1000, dus dan wordt de boel vanaf regel 1000 opgeborgen. Op die manier is de routine naar eigen wens aan te passen.

Het menuprogramma

Alles vanaf 490 en verder omvat dus het eigenlijke menu-programma. De regels 490 t/m 570 tekenen het scherm en zijn dus minder belangrijk. Regel 580 start de directory op door middel van het SYS DL-commando. DL is op 49152 geïnitialiseerd, dit is wel belangrijk. Wanneer namelijk een directory nog niet actief is, oftewel bij het eerste scherm, ligt het startadres op 49152. Na het uitvoeren van de routine kan het zijn dat nog niet alle entries de revue gepasserd zijn, omdat er teveel files op de disk staan. Zulks is het geval bij bijvoorbeeld de WEKA-diskette. Om nu een reeds gestarte directory is vervolgen moet niet 49152 aangeroepen worden, maar 49309. In geheugenplaats 2 kan bekeken worden of de directory is afgelopen. Staat hier een 0 dan is de directory ten einde.

Het programma maakt hier ook gebruik van, in regel 620 wordt gekeken of geheugenplaats 2 een 0 bevat. Is dit niet het geval dan wordt er aangegeven dat er meer is, staat er wel een nul dan wordt regel 630 uitgevoerd.

De regels 640 t/m 650 zetten de beide pijltjes op het scherm en wachten op een toetsdruk. Er zijn nu de volgende mogelijkheden:

F1 (regel 660)

Als er een vervolg-directory is dan wordt DL op 49309 gezet en kan het voorgaande opnieuw beginnen via 500.

F7 (regel 670)

Het programma stopt door het directorykanaal te sluiten. Dit gebeurt met het SYS 49301-commando. Doe dit in een eigen programma ook altijd!

CRSR LEFT of *CRSR RIGHT* (regel 680)

De kolom waar de pijltjes staan verandert. De kolomwaarde staat in K en is gelijk aan 0 of 1.

CRSR UP (regel690)

De waarde van Y (de pijltjesregel) wordt met één verlaagd (een regel hoger). Wordt Y dan kleiner dan 0 dan wordt naar beneden gesprongen (wrap-around). De waarde waar naartoe gesprongen moet worden staat in YM, dat is de waarde van de laatste regel die de directory-routine heeft aangedaan (zie regel 580). Blijken de pijltjes in een korte kolom te staan (één entry minder dan de andere kolom) dan moet dus nog een regel omhoog gegaan worden, anders staan de pijltjes bij een 'lege' naam. Aan geheugenplaats 1066+Y*40+K*20 is te zien of de naam leeg is, want hier staat dan een spatie. Het blijkt dus niet mogelijk te zijn om namen te hebben die met een spatie beginnen! In de praktijk komt dit echter ook nooit voor.

CRSR DOWN (regel 700)

Hier geldt zo'n beetje het zelfde als voor CRSR UP, komen de pijltjes te laag uit dan wordt naar boven gesprongen, tenzij daar ook een lege naam staat (in de praktijk niet mogelijk). Anders wordt automatisch regel 710 aangedaan welke tot ge-

17.3 Menu-programma

volg heeft dat, mochten de pijltjes toch bij een lege naam staan, de oude kolom- en regelwaardes weer ingevuld worden. Deze oude waardes worden in respectievelijk K0 en Y0 bijgehouden.

RETURN (regel 720)

Wordt op RETURN gedrukt dan moet de aangewezen file geladen worden en worden de regels 750 t/m 940 uitgevoerd. In het geval van een andere toets (geen van de voorgaande) worden de pijltjes 'geupdate' en worden de nieuwe oude waardes vastgelegd in de regels 950 en 960. Regel 1000 tenslotte maakt de hoofdlus compleet.

Het laden

De regels 750 t/m 940 zijn verantwoordelijk voor het laden van het gekozen programma. Daartoe wordt allereerst het directory-kanaal gesloten (regel 750).Regel 780 bepaalt de filenaam door deze uit het schermgeheugen te lezen. De POKE-codes moeten daar tot ASCII-codes omgezet worden, dit gebeurt in 780 en 790. Regel 810 vervolgens telt de overtollige spaties, waarop in regel 830 de filenaam geheel bekend is.

Het is nu mogelijk om op het scherm de volgende BASIC-commando's te PRINTen:

LOAD"NAAM",8

PRINT"(CLR)(COMM-7)(CTRL-I)"

De input-buffer wordt dan met een HOME, twee RETURNs en de tekst LIST gePOKEt waarop het programma eindigt. De computer echter denkt dat er nog een aantal toetsen is ingedrukt en voert die ook uit. Het gevolg is dat het gewenste programma geladen wordt. Indien een en ander niet duidelijk is dan kan het beste regel 930 zo aangepast worden dat de tekstkleur niet blauw gemaakt wordt. Sla het tweede teken in de PRINT-opdracht (CTRL-7) dus over een kijk nauwkeurig wat er gebeurt. Het is een truuk die veelvuldig toegepast wordt!

Een vriendelijk operating system

Als variant op het WEKA menuprogramma volgt nu een versie waar de extra's uitgehaald zijn. Het stelt de gebruiker in staat om op de bekende manier programma's te laden. Daarna volgen een aantal programma's die in samenwerking met BOS (Beter Operating Systeem) gebruikt kunnen worden, onder andere een formatteringsprogramma en een renameprogramma. Op die manier kunnen mensen die niet bekend zijn met het reilen en zeilen van het 1541/1571-DOS toch min of meer gebruik maken van de 64 met bijbehorende diskdrive. Op die manier kennen wij hier bij WEKA al diverse niet-Commodore 64 gebruikers die toch meteen met de 64 en de diskdrive overweg konden!

```
10 POKE56,159:CLR
20 PRINT"■":POKE53280,6:POKE53281,6:PRINT"■■■■■■■■■■■■■■"TAB(10);
30 PRINT"■■■♥ INITIALISATIE"
40 FORI=896TOI+62:POKEI,.:NEXT
50 FORI=896+46TOI+17STEP3:READA:POKEI,A:NEXT
60 FORI=49152TO49318:READA:POKEI,A:NEXT
70 FORI=40797TO40959:READA:POKEI,A:NEXT:SYS40797:SYS40930
80 DATA4,6,255,255,6,4
200 REM HIER BEGINT DE BODY
210 POKE56334,0:POKE788,49:POKE789,234:POKE56334,1
490 DL=49152:V=53248:POKE53280,6:POKE53281,6
```

17.3 Menu-programma

```
500 POKEV+21,0:PRINT"┌────────────────────────────────────────────────┐";
510 A$="|                                      |"
520 FORI=1TO21:PRINTA$;:NEXT
530 PRINT"└──┤                                   ├──┘"
540 PRINT"┌──────────────────────────────────────┐";
550 PRINT"|                                      |"
560 PRINT"|                                      |";
570 PRINT"└──────────────────────────────────────┘";
571 IFPEEK(49153)=36THEN575
572 PRINT"":PRINTTAB(15)"\OMENTJE..."
573 FORI=49152TO49318:READA:POKEI,A:NEXT
575 IFPEEK(40800)=20THEN580
577 IFPEEK(646)=0THENFORI=49152TO49318:READA:NEXT
578 IFPEEK(646)=0THENPRINT"":PRINTTAB(15)"\OMENTJE..."
579 FORI=40797TO40959:READA:POKEI,A:NEXT
580 PRINT""TAB(40);:SYSDL:YM=PEEK(214)
600 PRINT"\AAK EEN KEUZE M.B.V. DE CURSOR-"
610 PRINT" TOETSEN.  -RUK OP RETURN OM TE"
620 IFPEEK(2)THENPRINT"LADEN EN OP -1 VOOR HET VERVOLG":GOTO640
630 PRINT"LADEN EN OP -7 OM TE STOPPEN";
640 POKE198,0:POKEV+23,0:POKE2040,14:POKE2041,15:POKEV+29,3
645 POKEV,8:POKEV+1,44:POKEV+21,1:POKEV+39,1:POKEV+40,1:K=0:Y=0
650 GETA$:IFA$=""THEN650
660 IFA$=""ANDPEEK(2)THENDL=49389:GOTO580
670 IFA$=""THENSYS49301:PRINT"";:POKE53272,21:POKE53280,
    14:POKEV+21,0:END
680 IFA$=""ORA$=""THENK=1-K
690 IFA$=""THENY=Y-1:IFY<0THENY=YM-1:IFPEEK(1066+Y*40+K*20)=32THENY=Y-1
700 IFA$=""THENY=Y+1:IFY>YM-2THENIFPEEK(1066+Y*40+K*20)=32THENY=0
710 IFPEEK(1066+Y*40+K*20)=32THENY=YO:K=KO
720 IFA$<>CHR$(13)THEN950
750 SYS49301
780 N$="":FORI=1066+Y*40+K*20TOI+15:A=PEEK(I):IFA>31ANDA<64THEN800
790 IFA<32THENA=A+64
800 N$=N$+CHR$(A):NEXT
810 FORI=16TO1STEP-1:IFMID$(N$,I,1)<>" "THENNE=I:I=0
820 NEXT
830 N$=LEFT$(N$,NE):POKEV+21,0
840 PRINT"┌──────────────────────────────────┐"
850 FORI=1TO3:PRINT"|                                  |":NEXT
860 PRINT"└──────────────────────────────────┘"
870 PRINTTAB(12-LEN(N$)/2)N$" WORDT GELADEN..."
880 POKE631,19:POKE632,13:POKE633,13:POKE634,82:POKE635,85:POKE636,78
890 POKE198,6
930 PRINT"LOAD"CHR$(34)N$CHR$(34)",8"
940 PRINT"PRINT"CHR$(34)""CHR$(34)";:POKE53272,21:POKE53280,14":END
950 POKEV,8+K*160:KO=K
960 POKEV+1,44+Y*8:YO=Y
1000 GOTO650
1010 DATA169,36,133,251,169,251,133,187,169,0,133,188,169,1,133
1020 DATA183,169,8,133,186,169,96,133,185,32,213,243,165,186,32
```

17.3 Menu-programma

```
1030 DATA180,255,165,185,32,150,255,169,36,141,193,2,169,0,133
1040 DATA144,169,29,32,210,255,32,210,255,160,18,132,251,32,165
1050 DATA255,133,252,164,144,20PRINT#,82,32,165,255,164,144,208,75,164
1060 DATA251,136,208,233,32,165,255,166,144,208,63,17STOP,201,34,208
1070 DATA244,169,0,141,192,2,238,192,2,32,165,255,166,144,208
1080 DATA43,201,34,240,5,32,210,255,208,237,56,169,21,237,192
1090 DATA2,170,169,29,32,210,255,202,208,250,206,193,2,208,10
1100 DATA169,36,141,193,2,169,1,133,2,96,160,2,208,163,32
1110 DATA66,246,169,0,133,2,96,169,29,32,210,255,32,210,255,208,234
3000 DATA169,200,133,20,169,0,133,21,165,43,166,44,32,23,166
3010 DATA166,95,165,96,134,251,133,252,162,0,169,168,134,253,133
3020 DATA254,165,45,141,254,191,165,46,141,255,191,56,233,7,170
3030 DATA160,0,177,251,145,253,136,208,249,230,252,230,254,202,208
3040 DATA242,96,169,1,133,253,169,8,133,254,169,0,133,251,169
3050 DATA168,133,252,169,54,133,1,173,255,191,133,46,32,134,159
3060 DATA173,254,191,133,45,169,55,133,1,32,51,165,32,96,166
3070 DATA173,33,208,141,134,2,160,3,185,222,159,153,119,2,136
3080 DATA16,247,169,4,133,198,76,116,164,82,213,13,58,169,237
3090 DATA141,8,3,169,159,141,9,3,96,32,115,0,201,174,240
3100 DATA6,32,121,0,76,231,167,32,115,0,76,155,159
READY.
```

```
10 print"[CLR]"
20 print"Format formatteert een schijf, eventuele"
30 print"aanwezige data gaat hierbij verloren.[DOWN]"
40 print"De diskettenaam mag maximaal 16 tekens[DOWN]"
50 print"bevatten maar hoeft niet uniek te zijn,[DOWN]"
60 print"de ID moet per schijf echter wel uniek[DOWN]"
70 print"zijn en uit twee willekeurige tekens[DOWN]"
80 print"bestaan, bijvoorbeeld a0. Gebruik bij[DOWN]"
90 print"voorkeur geen SHIFT-tekens in naam of[DOWN]"
100 print"ID.[DOWN]"
110 n$="":input"Naam";n$:iflen(n$)>16thenpoke781,19:sys59903:
    print"[UP]":goto110
120 id$="":input"ID";id$:iflen(id$)<>2thenpoke781,20:sys59903:
    print"[UP]":goto120
130 print"[DOWN]Formatteren van "n$","id$"[DOWN]"
140 print"Dit neemt ca. 90 seconden in beslag."
150 open1,8,15,"n:"+n$+","+id$
160 input#1,a,b$,c,d
170 print"[DOWN]Diskstatus:"a;b$;c;d:close1
180 print"[DOWN]Nog een (j/n)?"
190 poke198,0
200 geta$:ifa$=""then200
210 ifa$="n"then:†
220 ifa$="j"then10
230 goto200
ready.
```

17.3 Menu-programma

```
10 print"■■■■■■"
20 print"BOS is een eenvoudig besturingssysteem■"
30 print"voor de Commodore 64. Programma's kun-■"
40 print"nen geladen worden door ze met de■"
50 print"cursor-toetsen aan te wijzen en op■"
60 print"RETURN te drukken. Na het gebruiken van■"
70 print"een programma kan BOS weer opgeroepen■"
80 print"worden door op het pijltje (↑) te druk-■"
90 print"ken, gevolg door RETURN."
100 printtab(27)"■■<TOETS>":poke198,0
110 geta$:ifa$=""then110
120 ↑
ready.
```

```
10 print"■■■■■■"
20 print"Delete verwijdert files van de schijf.■"
30 print"Geef de naam van de te verwijderen■"
40 n$="":input"file";f$:iflen(f$)>16thenpoke781,5:sys59903:
   print"■■":goto40
50 print"■"f$" wordt verwijderd."
60 open1,8,15,"s:"+f$
70 input#1,a,b$,c,d
80 print"■Diskstatus:"a;b$;c;d:close1
90 ifa=1thenprint"■Aantal verwijderde files is"c
100 print"■Nog een (j/n)?"
110 poke198,0
120 geta$:ifa$=""then120
130 ifa$="n"then:↑
140 ifa$="j"then10
150 goto120
ready.
```

```
10 print"■■■■■■"
20 print"Rename geeft een file een andere naam.■■"
30 o$="":print"■■"
40 input"Oude naam";o$:iflen(o$)>16oro$=""thenpoke781,3:sys59903:
   goto30
50 print"■"
60 n$="":print"■■"
70 input"Nieuwe naam";n$:iflen(n$)>16orn$=""thenpoke781,5:
   sys59903:goto60
80 open1,8,15,"r:"+n$+"="+o$:input#1,a,b$,c,d
90 print"■Diskstatus:"a;b$c;d
100 fori=1to1500:next:↑
ready.
```

17.3 Menu-programma

```
10 print"■■■■■"
20 print"Validate herorganiseert een schijf.■"
30 print"Plaats de te 'validaten' schijf in de■"
40 print"de drive en druk op een toets.■":poke198,0
50 geta$:ifa$=""then50
60 open1,8,15,"v"
70 print"Even geduld A.U.B.■"
80 input#1,a,b$,c,d
90 print"Diskstatus:"a;b$c;d:close1
100 fori=1to1500:next:↑
ready.
```

7/17.4

Voorwoord-programma

In dit deel dat tips, truuks en utilities behandelt is ook plaats voor een oude bekende: het programma uit het voorwoord. Dit programma (blz. 1/1-2) is een onderdeel van het basiswerk en is het enige programma in dit boekwerk dat niet uitvoerig behandeld is. Dat heeft twee redenen gehad: ten eerste komen er nogal wat niet alledaagse truuks in kijken en ten tweede is het een voorbeeldprogramma, niet zozeer een programma waar iets van geleerd hoeft te worden. Echter welbeschouwd mag enige uitleg omtrent dit programma niet ontbreken omdat er toch wel wat elementaire zaken in zitten, vooral op het grafische gebied. Echter omdat er meer dan alleen grafische aspecten aan het programma zitten zal het in deze paragraaf besproken worden.

Om het geheugen even op te frissen is hier de listing nog een maal:

```
4 print"■"
5 poke 56,160:dima(7000):clr:poke56,32:clr:dimb(7),w(7),b1(7),b2(7)
6 w(7)=128+64:w(6)=32+16:w(5)=8+4:w(4)=2+1:fort=0to3:w(t)=w(t+4):next
10 input"wat is jouw naam";a$:si=54272:pokesi+6,240:pokesi+4,17:
   pokesi+24,15
15 f=si+1
20 if len(a$)>20 then print "minder dan 20 letters graag":goto 10
30 sp=int((20-len(a$))/2)
40 ifsp>0thena$=a$+" ":a$=" "+a$:sp=sp-1:goto 40
50 print"■":poke53272,24:poke53265,59
60 fori=1to len(a$)
70 m$=mid$(a$,i,1):ifm$=" "then 190
80 s=8192+320*11+(i-1)*16
90 l=asc(m$):ifl>=64thenl=l-64
95 c=53248+1*8
100 poke 56334,0:pokel,5l:forj=0to7:b(j)=peek(c+j):next:pokel,55:poke56334,1
110 forj=0to7:b1(j)=0:b2(j)=0
120 forb=7to4step-1:bw=sgn(b(j)and(2↑b)):b1(j)=(b1(j)or(bw*w(b))):next
130 forb=3to0step-1:bw=sgn(b(j)and(2↑b)):b2(j)=(b2(j)or(bw*w(b))):next
140 pokes+j*2-312*(j>3),b1(j):pokes+8+j*2-312*(j>3),b2(j):pokef,rnd(1)*256
150 pokes+1+j*2-312*(j>3),b1(j):pokes+1+8+j*2-312*(j>3),b2(j)
160 next
190 next
200 fort=1024+11*40tot+39:poket+40,int(rnd(1)*15+1)*16
205 poket,int(rnd(1)*15+1)*16:pokef,rnd(1)*256:next
210 geta$:ifa$=""then 200
300 print"■":poke53272,21:poke53265,27:pokesi+4,0:pokesi+24,0
ready.
```

17.4 Voorwoord-programma

Een nogal ondoorzichtige listing, maar dat is ook niet te vermijden: zonder BASIC-uitbreiding is de 64 nu eenmaal een hulpeloze machine als het gaat om geluid en graphics. Echter om wat verlichting in de duisternis te scheppen volgt hier een uitleg van regel tot regel:
In regel 5 wordt het HIRES-scherm op een slimme manier gewist door er een array van 7000 variabelen overheen te DIMensioneren. De op die manier verbruikte geheugenruimte wordt meteen weer teruggenomen door het tweede CLR-commando in die regel. Daarna volgt een declaratie van 4 arrays.
Regel 6 bevat de initialisatie voor de W-array. Deze array bevat de POKE-waardes om dubbele puntjes op het scherm te kunnen zetten. Het programma zet tenslotte dubbelgrote letters op het scherm.
Regel 10 vraagt om een naam en initialiseert de SID-chip, regel 15 kent aan F het frequentieregister toe (SID+1).
De regels 20 t/m 40 kijken of de naam te lang is, zo niet dan worden aan allebei de kanten van de string evenveel spaties toegevoegd.
Regel 50 maakt het scherm schoon en schakelt de hires-mode in.
De lus 60 t/m 190 doet het eigenlijke werk. Is de te tekenen letter een spatie dan wordt alles in ieder geval overgeslagen, dit is in regel 70 te zien. Regel 80 rekent de geheugenlokatie uit waar het dubbelgrote teken gePOKET moet worden. Let er op dat we op een hires-scherm bezig zijn en dat elke letter dus gePOKET moet worden! Daartoe zal verderop ook het teken-ROM uitgelezen moeten worden om de letter-definities te bepalen.
Regel 90 vervangt een ASCII-code door zijn scherm-POKE-code, dit is nodig omdat de scherm-POKE-code overeen komt met de plaats van het teken in het teken-ROM. Regel 95 rekent de precieze plaats uit van de tekendefinitie in het character-ROM (elk teken bestaat uit 8 bytes, vandaar L*8). Om het teken-ROM uit te kunnen lezen moet dit eerst ingeschakeld worden. Ook de interrupts dienen uitgeschakeld te worden. Dit alles gebeurt in regel 100, de definitie wordt tijdelijk in de B-array opgeslagen.
Het is dan mogelijk om de B1- en B2-array uit te rekenen, deze bevatten respectievelijk de linker- en rechterhelften van de dubbelbrede letters. De letters worden twee keer zo hoog door elk gevonden byte twee maal onder elkaar te POKEn.
Regel 140 zet de gevonden dubbelbrede POKE-codes daadwerkelijk neer en laat daarbij een toontje horen door F te POKEn met een RND-waarde. Regel 150 doet hetzelfde, alleen dan 1 geheugenplaats verschoven. Op die manier worden de letters twee maal zo hoog.
Wanneer de hele string neergezet is kunnen de kleurtjes opdraven, dit gebeurt in de regels 200 en 205. Er worden gewoon random-waardes in het kleurengeheugen gePOKEt. Tevens wordt er op een toetsdruk gewacht, zodat het programma ook netjes kan eindigen.

7/18

Een tekstverwerker in BASIC

INHOUD

7/18.1

Inleiding

Onderzoeken bewezen, dat de inzet van tekstverwerkingsprogramma's niet alleen voor commerciële doeleinden gebruikt wordt, maar dat ook heel veel home computers hiervoor gebruikt worden, bijv. voor het schrijven van brieven of lijsten voor hobby's en de huishouding. Vooral deze thema's zullen wij bespreken: een eenvoudig te programmeren tekstverwerkingsprogramma met uitgebreide, echter niet te veel, mogelijkheden, die een eenvoudige bediening bewerkstelt.

Omdat de C 64 vaak enkel 40 tekens per regel verwerken kan is hij als hardware ook niet het beste toestel om omvangrijke teksten te verwerken.

Omdat het tekstverwerkingsprogramma, die wij nu in de eerste plaats uitgebreid in BASIC programmeren, zich in het geheugen bevindt, is ook de geheugenruimte niet voor grotere teksten geschikt.

In de volgende aanvullingen zullen wij de tekstverwerker in gedeelten in machinetaalprogramma veranderen, waarbij wij, indien mogelijk, nieuwe BASIC bevelen gebruiken, die ook door andere programma's gebruikt kunnen worden. Dit bespaart aan de ene kant waardevolle geheugenruimten en versnelt aan de andere kant de programmauitvoering.

Ook de basis-BASICversie laat zich nog meer versnellen en in het geheugen samenpersen als een compiler gebruikt wordt. Bij het tekstprogramma is het gebruik van een compiler vanwege de complexiteit, zeker aan te raden.

Op grond van de programmadocumentatie hebben wij de opmerkingen in het oorspronkelijke programma niet geschrapt. Veel REMs kunnen echter probleemloos weggelaten worden, dit scheelt aanzienlijk in de geheugenruimte!

De basisversie wordt eerst in BASIC gehouden, om alle lezers en niet enkel machinetaalprogrammeerders, een aanvulling op en aanpassing van de tekstverwerker op de speciale wensen van iedere particulier mogelijk te maken.

Nu nog een paar algemene mededelingen voor ons tekstverwerkingsprogramma: De teksten worden weliswaar gedeeltelijk intern in de computer verwerkt en opgeslagen, er is echter een beeldschermgeoriënteerde verwerking voor de gebruiker mogelijk. Het gedeeltelijk opslaan gebeurt in een veld. Iedereen die al veel met tekenregels gewerkt heeft, moet het begrip Garbage Collection kennen. Het betekent niet anders dan een grote schoonmaak in het geheugen, waarbij de niet meer benodigde tekenregels verwijderd worden.

18.1 Inleiding

Daar bij elke toewijzing aan een tekenregel-variabele nieuwe geheugenplaats nodig is, worden deze toewijzingen tot een minimum beperkt. Mocht ondanks dat uw C 64 niet werken, dan is hij echt niet stuk, maar bevindt zich in het bedrijfssysteem-Routine Garbage Collection, die - helaas - ongeveer 3 minuten kan duren. Bereikt worden deze enkele stringtoewijzingen door een tweede veld, die enkel regelnummers bevat. Het verwisselen van regels wordt ook niet in het tekenregelveld uitgevoerd, maar enkel de aanwijzing in het wijzerveld worden verwisseld.

7/18.2

Basisversie 'TEXT 1.0'

De basisversie van de tekstverwerker zullen wij in delen behandelen, d.w.z. wij beginnen eerst met een bedieningshandleiding., daarna de programmabeschrijving met aansluitend een overzicht van variabelen en ter afsluiting de Listing.
Zoals vermeld is de basisversie geheel in BASIC geprogrammeerd, waarbij de teksten weliswaar regel voor regel opgeslagen worden, een beeldschermgeoriënteerde bewerking toch mogelijk is.

Het tekstverwerkingsprogramma kan tot aan 150 regels en tot en met 254 tekens verwerken. Wilt u meer regels verwerken, dan is de variabele 'max' overeenkomstig te veranderen, aangenomen dat er voldoende geheugenruimte voorradig is. De breedte van tot aan 254 tekens is minder geschikt voor tekstuitgave dan voor tabellen, die beter op een overeenkomstige printer uitgegeven kunnen worden. De teksten kunnen ook regel voor regel opgeslagen of geladen worden, wat zelfs een blokverwerking mogelijk maakt. Verdere programmafunkties blijken uit het volgende commando overzicht. De juiste werking vindt u in de daaropvolgende bedieningshandleiding.

KOMMANDO'S z!=regels exact aangeven

A..	vrije geheugenruimte, aantal regels	
B..	regels wissen	
C..	beeldscherm wissen	z!
D..	regels afdrukken	
E..	regels wijzigen	
F..	woord zoeken, voortzetten:	W
H..	tekst laden	
I..	inhoudsopgave disk	
L..	regels lezen/wijzigen	
M..	woord verwisselen	z!
Q..	programma beëindigen	
R..	tekst wegschrijven	
S..	regels sorteren	z!
V..	regels verplaatsen	z!
Z..	regels op beeld printen	

18.2 Basisversie 'TEXT 1.0'

F1 ..... terugbladeren
F3 ..... vooruitbladeren
CRSR onder .... Cursor naar beneden
Nummer met leesteken ':'....regel inlezen
Toets drukken!

FUNKTIES VOOR HET WIJZIGEN

Toetsen	Werking
CRSR rechts	Cursor naar rechts
CRSR links	Cursor naar links
CRSR omhoog	Cursor naar boven
CRSR omlaag	Cursor naar beneden
HOME	kommando L,E,Z beëindigen
←	begin alinea
CTRL	einde alinea
F1	terug bladeren
F3	vooruit bladeren
INST	spatie invoegen
DEL	teken verwijderen
F5	scheiden: links boven
F7	scheiden: rechts onder
CTRL-D	regel kopiëren
CTRL-N	nieuwe regel
CTRL-L	inhoud wissen
CTRL-Y	regel wissen
CTRL-A	codegetal inlezen

Het zal zeker erg handig zijn dit blad naast de computer te leggen, zodat u niet steeds de menu's op het beeldscherm hoeft op te roepen. Met een vraagteken in de controlmodus van het tekstverwerkingsprogramma laten zich deze beide overzichten ten alle tijden weer op het beeldscherm halen.

Bedieningshandleiding
De bedieningshandleiding kunnen wij voor een beter overzicht in drie delen indelen. Eerst zullen wij ons met het laden van verschillende programmaversie en hun index bezighouden en aansluitend in een overzicht alle commando's incl. voorbeelden exact beschrijven. Tot slot zullen wij op het aanpassen van printerdata ingaan.

Laden van de tekstverwerker
Als u het BASICprogramma van de tekstverwerker vanuit het boek uittypt, dan kunt u aan dit programma een naam naar uw eigen keuze geven en het programma in aansluiting daarop met

LOAD "naam",8 (RETURN-toets)

18.2 Basisversie'TEXT 1.0'

inladen en in aansluiting daarop met

RN (RETURN-toets)

starten.

Het programma schakelt zelfstandig in de voor de tekstverwerking benodigde groot/ kleinschrift -modus om.

VOORBEREIDENDE BEWERKINGEN
Nadat u het programma geladen en gestart heeft verschijnt het volgende menu afbeelding:

```
---------------------------------------
              kleurkeuze
---------------------------------------

     F1: kader
     F3: achtergrond
     F5: tekst

     RETURN: klaar
---------------------------------------
     tekstkleur
---------------------------------------
```

Met dit kleine menu kunt u de kleuren voor uw tekstverwerker bepalen. Opdat u niet steeds de gewenste kleuren vooruit hoeft in te stellen, kunt u in het BASIC-programma in de variabelen f1, f2 en f3 kleurnummers van te voren ingeven, die u met het getoonde menu nu nog steeds veranderen kunt. Door op een van de funktietoetsen F1, F3 resp F5 te drukken wordt de kleurnummer van dat moment één eenheid verhoogd en na het drukken op de toets dadelijk toegewezen, waarmee u de gekozen kombinatie ook tegelijk met betrekking tot haar leesbaarheid kunt beoordelen.

Na het kleurnummer 15 wordt weer naar kleurnummer 0 (zwart) teruggegaan.

Daar een opnieuw tonen van het menu na iedere druk op de toets een flikkeren veroorzaakt laat de tekstkleur zich aan het woord 'Tekstkleur' aflezen. Door het bedienen van de RETURN-toets gaat hij verder.

Vervolgens wordt om het printertype gevraagd.

Waarop u een voorhanden zijnd printertype (zonder de toevoeging.PRN) moet aangeven. Drukt u enkel op de RETURN-toets, dan wordt de data PRINTER.PRN geladen. Op het instellen van een printertype zullen wij tot afsluiting van de bedieningshandleiding nog ingaan.

Nadat de printer aanpassing geladen wordt, verschijnt de vermelding:

toelaatbare regelnummers: 1 tot 150

als men als commando een vraagteken ingeeft (zonder RETURN) verschijnt op het beeldscherm een korte bedieningshandleiding.
willekeurige toets indrukken.

Als u met meer dan 150 regels met de tekstverwerker wilt verwerken, kunt u de variabele 'max' in het BASIC-programma overeenkomstig veranderen. Let op dat er genoeg geheugenruimte voorhanden is. De vermelding van de toelaatbare regelnummers wordt dan aangepast.

Als u nu een willekeurige toets indrukt,

18.2 Basisversie 'TEXT 1.0'

bevindt u zich nu reeds midden in de tekstverwerker. Op het beeldscherm ziet u:
*
en de cursor flikkert achter de ster. Hier verwacht de computer één van de vooraf voorgestelde commando's. Als voornaamste commando wordt eerst het '?' gebruikt, waarmee u een menu met alle commando's getoond krijgt. Als u daarna een toets indrukt, worden ook de toetsfuncties tot het editeren getoond en het programma gaat weer in de commando-ingave-modus over.

COMMANDO'S
Er staan principieel de reeds toegewezen commando's ter beschikking die slechts met de in het menu aangegeven letters, zonder RETURN-toets opgeroepen worden. Voordat wij tot een nauwkeurige beschrijving van alle commando's komen, iets over het thema regelopgaven, die bij meerdere commando's nodig zijn.

REGELOPGAVEN
Enkele commando's hebben nauwkeurige regelopgaven nodig, zoals bijv. het wissen van regels, het wijzigen van regels, zowel als het sorteren en verplaatsen van regels. Andere commando's komen ook zonder of met enkel onvolledige regelopgaven uit. Als regelopgaven kunnen afzonderlijke regels, regelgroep en regelgroep met tussenstappen ingegeven worden. Zijn i, j en k hele regels binnen het bereik van de geldige regelnummers, dan zijn de volgende ingaven mogelijk:

i : regel i moet verwerkt worden.
i-j : de regels i tot j (aansluitend) moeten verwerkt worden
i-j/k : verwerken van de regels i tot j (weer aansluitend) met tussenstap k, wat de Basicaanwijzing FOR...NEXT met stapgrootteopgave STEP overeenkomt. Wordt k niet ingegeven, dan neemt het programma automatisch de waarde 1 aan.

Buiten concrete regelnummers zijn nog 3 speciale ingaven toelaatbaar:

. : staat voor de actuele (laatstgebruikte) regel
! : staat voor gebruikte regelnummer met de hoogste waarde

Beide speciale tekens kunnen binnen de regelopgave gebruikt worden.
Hier volgen enkele voorbeelden:

Nadat u het commando 'L' ingegeven heeft, hebben de volgende regelopgaven de desbetreffende betekenis:

5 : lees regel 5
5-10 : lees regel 5 tot 10
1-13/3 : lees de regels 1, 4, 7 en 13

Heeft u de laatste toepassing van de regelopgave bij het 'L' commando gebruikt, dan betekenen zowel de ingaven van '.' als ook '!' enkel de regel 13. Als u tussendoor L en regelnummer 5 ingeeft, behoudt u dus de volgende toevoeging:

. : Lees regel 5
! : Lees regel 13
k : Stoppen
Er zijn echter ook combinaties mogelijk:

1-. : Alle regels tot aan de huidige regel
1-! : Gehele tekst
.-! : Alle regels vanaf de huidige tot aan de laatste regel.

18.2 Basisversie 'TEXT 1.0'

In principe zijn alle regelingaven met de RETURN-toets af te sluiten.
Het volgende commando overzicht zullen wij niet alfabetisch behandelen, echter in die volgorde, die u ook tegen komt bij het gebruik van de tekstprogramma's

COMMANDO: A Vrije geheugenruimte, aantal regels
Regel vermelding:-
Beschrijving: Door ingave van het commando 'A' verkrijgt u enige informatie, die voor het werken met een tekst zeker van belang zijn. Dit is de overgebleven vrije geheugenruimte, aangegeven in bytes, het aantal tekstregels, het hoogste regelnummer van de teksten, het hoogst toelaatbare nummer van een tekstregel en het actuele regelnummer. Het hoogste regelnummer en het actuele regelnummer kunnen ook d.m.v. '.' en '!' bij regelopgaven bereikt worden. Het beeldscherm na het starten van het programma ziet er als volgt uit:

```
*
Vrije geheugenruimte..... xxxxxBytes
aantal tekstregels.....0
hoogste nummer.....0
hoogst toelaatbare nummer...150
```

Voorbeeld: Als u met 'L' en '1-13/3' vijf regels registreert, dan ziet de opgave er als volgt uit:

```
*
Vrije geheugenruimte..... xxxxxBytes
aantal tekstregels.....5
hoogste nummer.....13
hoogst toelaatbare nummer.. 150
actuele regel.....13
```

COMMANDO: L regels lezen/wijzigen
Regelvermelding: ja
Beschrijving: Het commando 'L' dient voor het registreren van de tekst. Er kunnen regels opeenvolgend geregistreerd worden, echter ook regels met tussenstappen (zie regelopgaven) zijn mogelijk. Bovendien wordt met een cursorbesturing rekening gehouden (zie edit commando's).

Voor het overschrijven van een regel is het niet nodig deze regel uit te wissen, zoals ook bij het BASICprogrammeren, kan men de regel gewoon overschrijven. Lege regels moeten niet met open tekens ingegeven worden.

Voorbeeld: Door ingave van 'L' en '1-13/3' zullen wij enkele regels registreren, waarbij wij als tekst telkens het begrip 'regel' en de regelnummers van dat moment toepassen. Dit mag dan in de eerste plaats een zwakke tekst zijn, toch bewijst deze tekst bij de volgende voorbeelden waardevolle diensten, zodat zij overzichtelijk worden.

Na het intikken van het commando 'L' verschijnt in de bovenste regel de tekst

Regels lezen

waarna wij ingeven

1-13/3

in de rechter bovenhoek van het beeldscherm vinden wij vervolgens de vermelding

veld 1

een verwijzing naar het veld, waarin wij

18.2 Basisversie 'TEXT 1.0'

ons op het moment bevinden. Deze aankondiging wordt vooral bij het rangschikken van het beeldscherm bij regels langer dan 35 tekens van betekenis, daar alle mogelijke 254 tekens van een regel ook in een regel op het beeldscherm bewerkt worden en daartoe de tekst naar rechts en links verschoven wordt.

Onder de begrenzingsstreep in de derde beeldschermregel vinden wij de tekst

1:

waarachter de cursor flikkert. Wij geven nu

Regel 1

in en drukken op de RETURN-toets waarop in de vierde beeldschermregel

4:

verschijnt: Wij geven

Regel 4

in, en volgens dit schema beschrijven wij ook de regels 7, 10 en 13. Na ingave van regel 13 verschijnt het volgende op het beeldscherm:

```
*
-------------------------------------------
  1: Regel 1
  4: Regel 4
  7: Regel 7
 10: Regel 10
 13: Regel 13
```

Nu geven wij weer het commando 'L' en na de vermelding 'regels lezen' de ingave '1-!'.

Nu ziet het beeldscherm er als volgt uit:

```
                                        Veld
-------------------------------------------
1: Regel 1
```

waarbij de cursor op de 'R' van regel flikkert.

Wij willen onze regel niet veranderen en drukken dus weer direkt op de RETURN-toets. Op de volgende regel verschijnt nu

2:

en wij geven in

Regel 2

Hetzelfde doen wij met regel 3 en als wij op regel 4 komen verschijnt de al van te voren ingegeven tekst, die wij door de RETURN-toets overnemen. De reeds ingegeven regels kunnen op deze manier ook gewijzigd worden, zonder het commando 'E' op te roepen.

Hebben wij in een regel een fout getypt, dan kunnen wij ook met de cursor naar boven gaan en daar de verandering direct doorvoeren. Door op de RETURN-toets te drukken gaan wij dan weer een regel verder. Door op regel 13 de RETURN-toets weer in te drukken zijn wij weer in de commando-modus gekomen. Van een afbeelding van een menu is er nu geen sprake.

Door ingeven van '1-15' bij het comman do 'L' kunnen wij een blok van reeds ingegeven regels afbeelden resp. veranderen. Daarbij wordt eerst een 1 afgebeeld en na iedere druk op de RETURN-toets een

18.2 Basisversie 'TEXT 1.0'

een nieuwe regel.

COMMANDO: Z Regels op beeldscherm afbeelden

Regelvermelding: ja
Beschrijving: Door het ingeven van het commando 'Z' en een overeenkomstige regelopgave worden de gewenste regels op het beeldscherm afgebeeld. Worden er meer dan 22 regels afgebeeld dan worden eerst enkel de eerste 22 regels getoond en de cursor flikkert in de linker onderste beeldschermhoek. Door op de RETURN-toets te drukken gaat het afbeelden verder. Na de afbeelding van het laatst ingegeven regelnummer bevindt het tekstverwerkingsprogramma zich weer in de commando-modus.

Voorbeeld: Door het ingeven van 'Z' en '1-!' worden alle tot nu toe ingevoerde tekstregels afgebeeld. Wie het doorbladeren wil gadeslaan kan met 'L' en '14-30' meer regels rangschikken en met 'A' afbeelden.

COMMANDO: B Regels wissen

regelnummer: ja (exact)
Beschrijving: Nadat op toets 'B' gedrukt is, verwacht de computer weer een regelopgave.

Voorbeeld: Willen wij in ons voorbeeld de regels 6 tot 9 laten wissen, dan geven wij na 'B' (hier wordt ook de eerste beeldschermkant van de tekst meteen getoond) '6-9' in. Om de uitwerking van ons commando te bekijken, hoeven wij niet het commando 'Z' op te roepen, omdat het programma de tekst zonder de gewiste regel toont.

Om de gehele tekst te wissen kunt u '1-!' ingeven.

Door de toetsencombinatie 'L' '5' en 2x 'RETURN' maken wij de regel 5 tot actuele regel. Na ingave van het commando 'B' en regelopgave '.' wordt nu regel 5 gewist, wat het programma ook meteen laat zien.

Door ingave van 'B' en '!' wordt regel 13 gewist.

Onze korte voorbeeldtekst ziet er nu als volgt uit:

```
*
  1: regel 1
  2: regel 2
  3: regel 3
  4: regel 4
 10: regel 10
 11: regel 11
 12: regel 12
```

COMMANDO: C Beeldscherm wissen

Regelvermelding: -
Beschrijving: De eenvoudige ingave van commando 'C' wist het beeldscherm, op de ster en de streep na, in de tweede beeldschermregel.

COMMANDO: E Regels wijzigen

Regelvermelding: ja
Beschrijving: Het edit commando dient voor het corrigeren en veranderen van de regels op dezelfde voorafgaande wijze, zoals bij commando 'L'.

COMMANDO: D Regels printen

18.2 Basisversie 'TEXT 1.0'

Regelvermelding: ja
Beschrijving: Na het invoeren van het commando 'D' vraagt de computer eerst

Regels afdrukken?

Hier geeft u het gewenste regelbereik in de gebruikelijke notatie aan. Vervolgens verschijnt de vraag.

Afdruk met regelnummer (j/n)

die u overeenkomstig met 'j' beantwoordt, als de regelnummers mee afgedrukt moeten worden en 'n' als de regels in de afdruk niet mogen verschijnen. De eerste mogelijkheid is zeker voor correctie interessant, of als vaak hetzelfde schrijven voor het verwerken gebruikt wordt, zodat men niet eerst op het beeldscherm het overeenkomstige regelnummer moet opzoeken. Voor brief of lijstopgave hoeft het regelnummer natuurlijk niet te verschijnen.

Vervolgens vraagt de computer

linker kantlijn?

waarmee u softwarematig de afstand van de tekst vanaf de linkerkant kunt aangeven. Indien u de printer al met behulp van besturingscodes een instelling voor de linkerkant ingegeven heeft, kunt u hier een nul ingeven.

Tenslotte vraagt de computer

‹RETURN›.. printen ‹Q›..stoppen
Waarmee u eventueel verkeerd ingegeven printerparameters nog ongedaan kan maken.

Na het printen komt niet automatisch een bladzijde aanduiding, zodat u ook meedere teksten direkt achter elkaar kunt printen. Wilt u toch na de tekst een bladzijde aanduiding, dan moet u als laatste tekstregel een '?' ingeven, waardoor deze bladzijde aanduiding geplaatst wordt: Let op: De tekst achter een '!' wordt niet meer uitgeprint.

De printerbesturing zullen wij verder bespreken.

Voorbeeld: Eerst voegen wij door het ingeven van 'L', '13', 'RETURN', '!' en 'RETURN' nog de regel 13 met een '!' teken bij ons tekstvoorbeeld aan.

Eerst zullen wij de gehele tekst met bladzijde aanduiding 10 tekens van de linkerkant zonder regelnummers printen.
Daarvoor wordt achterelkaar ingegeven:

D
1-!
N
10
RETURN

Als de linkerkant van te voren is ingesteld en wij met regelnummer, echter zonder afgesloten bladzijde printen willen, wordt nu

D
1-12
J
0
RETURN

ingegeven.

U dient nu zelf andere mogelijkheden uit te proberen.

18.2 Basisversie 'TEXT 1.0'

Op bijzondere printerbesturingen komen wij aan het einde van de bedieningshandleiding terug.

COMMANDO: R Regels in file redden

Regelvermelding: ja
Beschrijving: Met het Commando 'R' kunnen tekstblokken naar voorkeur op disk opgeslagen worden. Nadat u de gebruikelijke regelopgave doorgegeven heeft, vraagt de computer u:

File naam?

waarna u een geldige filenaam volgens de gebruikelijke regels moet ingeven. Ook hier wordt u, zoals bij commando 'D' gevraagd:

Uitgave met nummer (j/n)

die u overeenkomstig moet beantwoorden. Daarna verschijnt de vermelding:

volgende regels zijn toegewezen:

gevolgd door de opgave van de regelnummers in de vorm:

1: 2: 3: 4; ...

en na beëindiging van het wegschrijven op de disk:

einde opslaan tekst.

Voorbeeld: met de ingaven

r
1-4
Testb1b4
J

slaat u het eerste gedeelte van onze tekst op in de file 'Test1b4' met regelnummers en met

R
10-12
Test10L12
N

het tweede deel in de file Test10L12 zonder regelnummers.

COMMANDO: I Inhoudsopgave diskette

Regelvermelding:-
Beschrijving: Het commando 'I' toont de inhoudsopgave van de disk, waarbij enkel met sequentiële file's (de tekstfile's zijn allemaal sequentieel) rekening gehouden wordt. Bovendien wordt nog het aantal vrije blokken getoond, wat zeker voor de gebruiker een klein hulpmiddel is, of een tekst nog op de disk past ja of nee.

Voorbeeld: Door intikken van 'I' in de commando-modus kunt u zich daarvan overtuigen, dat de bij het voorgaande commando opgeslagen voorbeeldfile's werkelijk op disk bestaan.

COMMANDO: H File's laden

Regelvermelding: ja (wordt later voor het nummeren van de disk gevraagd)
Beschrijving: Nadat u het commando 'H' ingegeven heeft, wordt u gevraagd:

File naam?

waarna u de naam van een voorhanden zijnd bestand moet ingeven.

18.2 Basisversie 'TEXT 1.0'

Is de ingegeven filenaam als tekstfile niet voorhanden, dan wordt de foutmelding 'file not found' gegeven.

Bevindt zich in het geheugen van de computer al een tekst, dan wordt de vermelding

in het geheugen is reeds tekst aanwezig
hoogste regel Nr.:'

gevolgd door het daadwerkelijke hoogst uitgegeven regelnummer. Hierdoor kunnen opgeslagen tekstblokken met nieuwe tekstaanvullingen uitgebreid worden, zodat u bijv. ook grote stukken uit fragmenten samen kunt voegen. Vervolgens krijgt u de melding

welke regels moeten opnieuw geladen worden (nummeren op disk)?

waarna u de gewenste regelnummers op de gebruikelijke schrijfwijze ingeeft. Daarbij worden de afzonderlijke tekstregels niet beslist in hun oorspronkelijke regelnummers geladen, maar u kunt ook nieuwe regelnummers en ook een tussenstap aangeven. Daarvoor geeft de computer eerst de volgende melding.

In het geval dat de geladen regels nieuwe nummers moeten krijgen, gelieve eerste nummer en eventuele tussenstap aangeven:

waarna de gewenste ingave uitgevoerd kan worden. Net als bij regelnummeren moet een tussenstap achter het eerste regelnummer met een schuine streep aangegeven worden. Moeten de regels vanaf nummer 10 met een tussenstap van 2 gereserveerd worden dan wordt ingegeven '10/2'.

Dan vraagt de computer nog

geladen regels tonen (J=ja)?

die u overeenkomstig beantwoordt. Geeft u 'j' dan meldt de computer zich met

geladen regels:

De computer toont alle tekstregels met van te voren gekozen regelnummers en ook de zojuist van disk gelezen regelnummers, zowel als de nieuwe regelnummers, bijv.

Nr. disk: 4 Nr. geheugen: 10

Normaal geldt het puntteken (.) en '!' voor actuele en hoogste regelnummer niet voor het inlezen van disk, echter enkel in werkgeheugen.

Voorbeeld: Wij nemen aan, dat wij een leeg geheugen hebben, waarin wij uit de file 'Test1L4' de regels 1 tot 4 willen inlezen. Daarvoor zijn de volgende ingaven nodig:

H	(commando
Test1-4	(filenaam)
1-4	(te lezen regels)
1	(vanaf regel 1 in tekst)
J	(geladen regels tonen)

Vervolgens moeten de regels 10-12 in de regels 10, 14 en 18 ingelezen worden:

H	(commando)
Test10-12	(filenaam)

Hier verschijnt de melding:

18.2 Basisversie 'TEXT 1.0'

in het geheugen is al tekst aanwezig
laatste regel Nr.: 4

10-12 (te lezen regels)
10/4 (vanaf regel 10/tussenstap 4)
J (geladen regels tonen)

Als u de regels 10 tot 12 vanaf regel 1 met tussenstap 1 zou laden, behoudt u de volgende tekst:

1: regel 10
2: regel 11
3: regel 12
4: regel 4

omdat bij het laden van teksten bestaande regels overschreven worden, wat in vele gevallen ook zinvol kan zijn.

COMMANDO: F Woord zoeken, voorzetten: W

Regelvermelding: ja
Beschrijving: Na drukken op 'F' verschijnt de vermelding

Regels doorzoeken/

die u met een gebruikelijke regelnummeropgave beantwoordt. Daarna wordt u naar de te zoeken letterreeks

Letterreeks zoeken?

gevraagd. Heeft u dit aangegeven, dan verschijnt de eerste regel waarin deze letterreeks voorkomt, waarbij de gezochte letters met '=' onderstreept zijn. De computer bevindt zich daarop weer in de commando-modus en u kunt door de toets 'W' de volgende regel met de gezochte letterreeks laten tonen.

Voorbeeld: : Als u in onze teksttekst (regel 1-4 en 10-12) de letterreeks 're' in de eerste 4 regels wilt zoeken, wordt dus

F
1-4
re

ingegeven en u behoudt als beeld

```
*
1: regel 1
   ==
```

als u nu 'W' indrukt verschijnt:

```
*
2: regel 2
   ==
enz.
```

COMMANDO: M letterreeks verwisselen

Regelvermelding: ja (exact)
Beschrijving: Ook na ingave van het commando 'M' wordt er naar een regelopgave gevraagd:

Regels wijzigen?

Met het 'M' commando kunt u letterreeksen door anderen vervangen (zie onderstaand voorbeeld). Heeft u een juiste regelopgave bewerkstelligd, dan wordt u naar de vervangende letterreeks

oude letterreeks?

gevraagd. Nadat u deze ingegeven heeft, vervolgt de afvraag

nieuwe letterreeks?

18.2 Basisversie 'TEXT 1.0'

die ook overeenkomstig ingegeven wordt. Daarna toont de computer alle veranderde regels en keert terug in de commandomodus.

Voorbeeld: Wij willen het begrip 'regel' in de regels 1-4 door de woordvolgorde 'Dit is regel' vervangen. Daarvoor zijn de volgende ingaven nodig:

```
M
1-4
Regel
Dit is Regel
```

Het resultaat is dan:

```
*
1:    Dit is Regel 1
2:    Dit is Regel 2
3:    Dit is Regel 3
4:    Dit is Regel 4
```

COMMANDO: V Regels verplaatsen

Regelvermelding: ja (exact)
Beschrijving: Door het commando 'V' kunt u regelblokken verplaatsen. Na oproep van het commando wordt eerst het eerste deel van de tekst getoond en u wordt gevraagd

regels verplaatsen?

Hier wordt het gewenste regelblok ingegeven. Dan vraagt de computer u

Regelnr. van de eerste verplaatste regel?

die u overeenkomstig beantwoordt. Let op dat u niet per ongeluk aanwezige regels overschrijft, wat een van de volgende voorbeelden toont.

Voorbeeld: In het eerste voorbeeld willen wij de regels 1-4 naar regels 21-24 verschuiven. Volgende ingaven zijn daarvoor nodig:

```
V
1-4
21
```

en u krijgt het resultaat:

```
*
10: Regel 10
11: Regel 11
12: Regel 12
21: Dit is Regel 1
22: Dit is Regel 2
23: Dit is Regel 3
24: Dit is Regel 4
```

Het tweede voorbeeld toont, dat u bij het verplaatsen van regels voorzichtig moet zijn, daar aanwezige regels overschreven worden:

```
V
1-2
10
```

met het resultaat

```
*
 3:  Dit is Regel 3
 4:  Dit is Regel 4
10:  Dit is Regel 1
11:  Dit is Regel 2
12:  Regel 12
```

COMMANDO: S Regels sorteren

Regelvermelding: ja (exact)
Beschrijving: Ook bij het commando 'S' wordt eerst naar een regelblok gevraagd

18.2 Basisversie 'TEXT 1.0'

met

Welke regels moeten gesorteerd worden?

Daarna wordt nog het eerste en het laatste veld gevraagd met de overeenkomstige beeldschermmeldingen. Dit maakt het mogelijk, dat u niet na een gehele tekstregelnaam enkel maar een deel daarvan kan sorteren. Het heeft zeker geen zin brieftekst regel voor regel alfabetisch te sorteren.
Daar het tekstverwerkingsprogramma echter ook voor lijsten en tabellen gebruikt kan worden is de inzet van een sorteercommando daardoor zinvol, zoals volgende voorbeelden aantonen.

Voorbeeld: Gegeven is volgende kleine tekstfile:

```
*
1: Schneider        Hans Lorenz      München
2: Interest         Verlag           Kissing
3: Friese           Manfred          Hildesheim
4: Eberl            Werner           München
```

Door ingave van:

```
S
1-!
1
15
```

krijgt u het volgende resultaat

```
*
1: Eberl            Werner           Munchen
2: Friese           Manfred          Hildesheim
3: Interest         Verlag           Kissing
4: Schneider        Hans Lorenz      Munchen
```

Als u de laatste beide opgaven door '16' en '30' vervangt komt het volgende aansluitend op het beeldscherm:

```
*
4: Schneider        Hans Lorenz      Munchen
2: Friese           Manfred          Hildesheim
3: Interest         Verlag           Kissing
1: Eberl            Werner           Munchen
```

COMMANDO: Q Programma afbreken

Regelvermelding: -

Beschrijving: Met het commando 'Q' kunt u het tekstverwerkingsprogramma verlaten. Om het per vergissing verlaten

18.2 Basisversie 'TEXT 1.0'

en daarmee het dataverlies te verhinderen, verschijnt aansluitend nog de melding

<Q>.. programma stoppen
<RETURN>.. verder

Daar het programma enkel via het END-bevel verlaten wordt, kunt u ook na de beeldschermmelding 'ready' weer in het programma terug, +.w met het CONT-bevel. Ook de data blijft daarbij behouden, wat u door ingave van het Z-commando kunt uitproberen.

COMMANDO'S

De meeste edit-commando's behoeven niet afzonderlijk besproken te worden zoals bij de cursor-commando's voor rechts, links, omhoog en omlaag, en de toets HOME voor terugkeer in de commando-modus. De toets (pijl naar links) en de toetsencombinatie CTRL (pijl naar links) creëeren de bewerkelijke cursorbewegingen naar regelbegin en regeleinde. Ook het snelle voor- en achteruit bladeren in de tekst wordt gemakkelijk, waarbij de funktietoetsen F1 en F3 ter beschikking staan. Ook de funktie van de toetsen INST en DEL is bekend, waarbij toch een iets ander resultaat volgt. Bij DEL wordt het teken onder de cursor gewist en alle tekens rechts van de cursor worden een stap naar links verschoven. Tegenover de normale Basic-Edit modus, waarbij het teken links van de cursor gewist wordt en de cursor dan ook een stap naar links gaat, is dit dus fundamenteel veranderd. Hetzelfde geldt voor de INST-toets.

De funktie van de toetsen F5 en F7 kunt u zelf wellicht uitproberen, als u de cursor in het midden van een tekstregel plaatst en deze beide toetsen gebruikt. Er blijven nog de edit-commando's in combinatie met de CTRL-toets. Bespreken wij eerst CTRL-L (regelinhoud wissen) en CTRL-Y (regel afvoeren). Bij CTRL-L wordt nu de tekst in een regel gewist. De regel blijft als lege regel-met toegewezen regelnummer-. Wilt u toch een tekstregel afvoeren, zonder dat een lege regel achterblijft, moet u de toetsencombinatie CTRL-Y gebruiken, die uw wens uitvoert. Daarbij worden alle daaronderliggende tekstregels één regelnummer naar boven geschoven, zoals volgend voorbeeld toont

```
*
1: Regel 1
2: Regel 2
3: Regel 3
4: Regel 4
```

Als u nu de cursor op regel 2 beweegt en het commando CTRL-Y (de verbindingsstreep geeft aan dat de beide toetsen gelijktijdig ingedrukt moeten worden) dan ziet het beeldscherm er als volgt uit:

```
*
1: Regel 1
2: Regel 2
3: Regel 3
```

De tegenhanger van CTRL-Y is CTRL-N, het invoegen van een nieuwe regel. In de regel waar zich de cursor bevindt, wordt een lege regel gecreëerd en alle regels vanaf de cursor neerwaarts worden een regelnummer verhoogd. Als u bij het laatste voorbeeld met de cursor weer in regel 2 gaat, en de CTRL-N ingeeft, verandert het beeldscherm als volgt:

18.2 Basisversie 'TEXT 1.0'

```
*
1:  Regel 1
2:
3:  Regel 3
4:  Regel 4
```

Deze regel kunt u zelf met een tekst opvullen, echter ook de inhoud van een andere regel in een nieuwe regel overkopiëren. Daar kunt u het commando CTRL-D voor gebruiken. Na ingave van dit commando vraagt de computer u

kopiëren regel?

en u geeft een regelnummer aan. De tekst uit het aangegeven regelnummer wordt nu in de regel gekopieerd waar de cursor staat. Zou zich in de tekstregel, waar zich de cursor bevindt, een tekst bevinden, wordt deze overgeschreven. De cursor staat altijd nog in regel 2 en u geeft het commando CTRL-D, waarna u de regelafroep met '4' beantwoordt. Ook hier weer de beeldschermafbeelding na uitvoering:

```
*
1:  Regel 1
2:  Regel 4
3:  Regel 3
4:  Regel 4
```

PRINTERBESTURING

Ons tekstprogramma TEXT1.0 biedt natuurlijk ook de mogelijkheid van printerbesturing. Voor zover uw printer het toelaat kunt u teksten onderstrepen, in vet- of breedshrift, tekstdelen hoog of laag plaatsen, of ook de tekst in kleinschrift, schoonschrift of proportionaalschrift uitgeven. Bovendien is het nog mogelijk vast te stellen of het om een printer met enkelblad of kettingformulier gaat en ook de regels per drukzijde en de lege regels aan de kop en voet kunt u vastleggen.
De schriftgegevens kunt u op de gewenste plaats in de tekst verwerken en wel door een apostroph (') gevolgd door een letter, zoals getoond in onderstaand tabel:

U/u	:	Onderstrepen
F/f	:	Vetschrift
B/b	:	Breedschrift
H/h	:	Hoog instelling
T/t	:	Laag instelling
N	:	Normaal schrift
S/s	:	Kleinschrift
L/l	:	Schoonschrift (letter kwaliteit)
P/p	:	Proportionaalschrift

Door de hoofdletter wordt telkens het lettertype ingeschakeld en door de kleine letter uitgeschakeld. Bovendien kan door 'W' het printen gestopt worden (wachten).

De aanpassing van de afzonderlijke lettertypen op verschillende printertypen gebeurt in een printerfile, die wij u later aan de hand van een voorbeeld zullen tonen. Voor iedere printer kan zo'n file vervaardigd worden, zodat teksten bij de afdruk op andere printers niet meer veranderd hoeven te worden.

Als u meerdere printeropties in het programma wilt inbouwen, bent u op de regels 3000 tot 3960 (inlezen van printerfile) en 19000 tot 19740 (opvragen van de printergegevens bij het printen) aangewezen. De printerfile is dan overeenkomstig aan te passen.

De printerfile kan een willekeurige naam

18.2 Basisversie 'TEXT 1.0'

tot aan twaalf tekens bevatten, gevolgd door '.prn', zodat men in de inhoudslijst gelijk de betekenis van de file aflezen kan. Een programma, dat een printerfile bevat, dient de overeenkomstige naam, gevolgd door '-prn' te bezitten.

Nu een voorbeeld van een printerfile:

```
  100 rem programma epson-prn
  110 rem bevat de file 'epson.prn'
  120 rem met de printeraanpassingen
  130
  140 open15,8,15,"i0"
  150 open1,8,2,"@0:epson.prn,u,w"
  160
  170 data 3,27,45,1 :rem onderstrepen aan
  180 data 3,27,45,0 :rem onderstrepen uit
  190 data 2,27,69   :rem vet schrift aan
  200 data 2,27,70   :rem vet schrift uit
  210 data 2,27,14   :rem breed schrift aan
  220 data 1,20      :rem breed schrift uit
  230 data 3,27,83,0 :rem verhogen aan
  240 data 2,27,84   :rem verhogen uit
  250 data 3,27,83,1 :rem verlagen aan
  260 data 2,27,84   :rem verlagen uit
  270 data 2,27,80   :rem normaal schtift
  280 data 1,15      :rem smal schrift aan
  290 data 1,18      :rem smal schrift uit
  300 data 2,27,77   :rem schoon schrift aan
  310 data 2,27,80   :rem schoon schrift uit
  320 data 0,0       :rem proportioneel schrift aan
  330 data 0,0       :rem proportioneel schrift uit
  350 data 1         :rem 0=enkel blad
  360 data 68        :rem regels per pagina
  370 data 2         :rem blanko regels per pagina (kop en
staart)
  390 data-1
  410 readi:ifi<0then430
  420 print#1,i:goto410
  430 close1:close15
```

Het programma met de naam EPSON-PRN stelt een printerfile voor van een EPSON-RX80-Printer. Opgevraagd wordt de file EPSON.PRN. In regel 140 wordt het Floppyloopwerk geïnitialiseerd en in regel 150 de file voor de uitgave van de printer gegevens geopend. Dan volgen de gegevens voor de overeenkomstige printer parameters die telkens uit de opmerkingen duidelijk zijn. De gegevens zijn in die volgorde opgeslagen, zoals zij bij TEXT1.0 in de regels 3000 tot 3960 in

18.2 Basisversie 'TEXT 1.0'

gelezen worden. In iedere regel bevindt zich eerst het aantal parameters (1,2 of 3) gevolgd door de gespecificeerde printercode.

Tenslotte worden nog de kenmerken voor kettingformulier/enkelblad, het aantal regels per bladzijde en het aantal van kop- resp. voetregels vastgelegd. Als markering voor de afsluiting van printer gegevens dient de ingave '-1'. Vanaf regel 410 worden de gegevens ingelezen en in de file ingevoerd.

PROGRAMMABESCHRIJVING
TEXT 1.0 is een tekstverwerkingsprogramma, dat geheel in BASIC geschreven is. Dit heeft het voordeel dat het makkelijk te begrijpen en te veranderen is, maar helaas ook het nadeel, dat het bijzonder traag is. Daarvoor is TEXT 1.0 echter ook een beeldscherm georiënteerd programma, d.w.z. de tekst op het beeldscherm kan altijd aangesproken en veranderd worden.

De commentaarregels worden door TEXT 1.0 niet aangesproken. Daardoor is het programma ook zonder deze regels geheel geschikt te doorlopen.

Wie de mogelijkheid heeft, moet TEXT 1.0 bovendien met behulp van een Compiler compileren (d.w.z. in machinetaal omzetten). Het programma wordt daardoor korter en wezenlijk sneller. Een goed voorbeeld is de Austro-Compiler, Petspeed wijst het programma in de gedrukte vorm af. Andere Compilers werden niet getest.

Op grond van de lengte van het programma, kan met TEXT 1.0 enkel gewerkt worden, als tenminste alle commentaarregels en het inspringen van regels van de FOR-NEXT loops verwijderd zijn. U moet daarvoor het programma geheel met een Compiler bewerken, of deze zaken weglaten.

TEXT 1.0 slaat de tekstregels in het veld A$() op. In de voorafgaande versie is de tekst tot 150 regels begrensd. Dit kan door veranderen van de variabels MAX veranderd worden. Daarbij moet u echter opletten, dat de C 64 niet genoeg vrij geheugenruimte voor wezenlijk meer regels tot zijn beschikking heeft, dat deze, naast de zekere velddefinitie voor de letterreeksen, die door de gebruiker ingegeven worden, geheugenruimte gebruikt.

De ingegeven regels hebben allen een eigen regelnummer. Om veelvuldig reorganiseren van de stringvariabels te vermijden, worden de nummers van de regels niet met de index van het veld A$() samengevoegd, maar in veld IZ() vastgelegd. In dit veld wordt de index van de stringvariabels vastgelegd. Geeft men nu de regel met het cijfer 10 als eerste regel in, dan wordt de inhoud van de regel A$(1) opgeslagen, en IZ(10) behoudt de waarde één. Om ook vanuit de inhoud van de tekstregel op het regelnummer te kunnen terugvallen, is er nog een veld IS() nodig. IS(I) slaat de regelnummers van de tekstregel A$(I) op.

In de regels 1000-1340 worden naast deze velden ook de bevelen van TEXT 1.0 bewaard, waarin de letters, degenen waarmee de bevelen opgeroepen worden, in het veld K$() vastgelegd zijn.

Het vastleggen van de kleuren gebeurt in

18.2 Basisversie 'TEXT 1.0'

de regels 1500 tot 1870. De toegepaste kleuren zijn in de variabelen F1 tot F3 vastgelegd. Wie het programma dikwijls gebruikt moet de door hem gewenste kleuren hier inzetten.

In de volgende DATA-regels staat de opbouw van het beeldscherm (bij de C 64 25*39) en de lijst van de toetsen, die bij het editen toepassing vinden. Dit zijn de cursor-toetsen, de toets met de pijl naar links, de Home-toets, de funktietoetsen, en de verschillende toetsen in samenhang met de Control-toets. Deze inhoud wordt in het veld IC() ingelezen en in het veld CTEXT$() de reservering van de toetsen als hulptekst.

Als data worden vanaf regel 2500 de initialisatie van de disk en de printer in overeenstemming gerealiseerd. Omdat zij voortdurend nodig zijn, blijven zij tot het einde van het programma open. Nu worden de printer aanpassingsgegevens van de disk gelezen. Deze worden in de strings U1$, U2$, F1$, F2$, etc. ingelezen. Bij het printen kunnen dan de strings uitgegeven worden. Zo laat zich elke printer aan het programma aanpassen.

Vanaf regel 4000 wordt een korte informatie voor de gebruiker getoond en het programma gaat bij regel 4500 in het hoofdletterschrift over. Hier wordt eerst de commandoregel getoond.
Let op, dat het programma enkel de positie's 1 tot 39 kan gebruiken. Een uitgave in positie 40 wordt door de regel voor het bedrijfssysteem van de C 64 direkt als dubbele regel gekenmerkt. Ingaven in de commandoregel worden zo fout geïnterpreteerd.

De hoofdprogramma-loop accepteert een commando van de gebruiker. Dit commando zoekt zij in het veld K$(). Bestaat het commando, dan wordt het bijbehorende onderprogramma van de ON...GOSUB-constructie vanaf regel 4970 aangesproken. Anders wordt de ingave genegeerd.

Het eenvoudigste van dit onderprogramma staat bij regel 5500. Het wist het gehele beeldscherm.

Een andere informatie voor de gebruiker kan vanaf regel 6000 opgeroepen worden. Hier worden de vrije geheugenruimte, de laatst benutte filenaam, het aantal tekst regels, de hoogste regelnummers, de hoogst toegestane nummers en de actuele regel uitgegeven.

Om het programma te verlaten, moet het programma onderdeel vanaf regel 6500 opgeroepen worden. Dit gebeurt d.m.v. het Q-commando. Na nog een 'Q' worden de file's gesloten en het programma beëindigd. De RETURN-toets voert daarentegen weer naar de commandoregel. Deze beveiliging dient voor het foutieve ingeven van het Q-commando.

Als verdere informatie kan men opvragen welke file's er zich op de disk bevinden. Dit toont het deelprogramma vanaf regel 7000. Dit wordt bereikt als de disk drive gesignaleerd wordt, dat de inhoudslijst in het geheugen geladen mag worden. In plaats de inhoud te laden, wordt hij echter ingelezen en getoond. Daar TEXT 1.0 alle teksten als sequentiële file's opslaat, worden ook enkel de sequentiële file's getoond. Dit kan men heel eenvoudig berei

18.2 Basisversie 'TEXT 1.0'

ken, als men als file naam niet zo als gewoonlijk '$' voor de inhoudslijst aangeeft, echter de naam '$*=S'. Overeenkomstig zal '$*=p'. de programmafile's leveren.

Om de regels te kunnen wijzigen, wordt het deelprogramma vanaf regel 10000 opgeroepen. Dit deelprogramma kan òf direct opgeroepen worden, òf ook via het commando 'cursor-down', dat vanaf regel 7500 afgehandeld wordt. Ook de opdracht tot inlezen van tekstregels, die in de regels 8000-8490 gerealiseerd wordt, bedient zich van het deelprogramma tot het wijzigen.

Voordat men regels op het beeldscherm verwerken kan, moeten de regels eerst een keer op het beeldscherm getoond worden. Voor deze opgave staat het commando 'regels tonen' ter beschikking. In de regels 8000-8850 worden de regels op het beeldscherm gebracht. Dit geldt natuurlijk enkel voor het zichtbare gedeelte van de regels. Een andere mogelijkheid de tekst te laten tonen kan met het commando 'Page-up' resp. 'Page-down', dit in de regels 9000-9300 resp. 9500-9750 afgewerkt worden. De genoemde deelprogramma's brengen in een For-loop eerst de regelnummers, en gevolgd door het zichtbare gedeelte van de regel op het beeldscherm. Bovendien worden in het veld ZZ() de regelnummers van de momenteel op het beeldscherm getoonde regels opgeslagen, zodat het programma steeds weet, welke regel op welke positie op het beeldscherm staat.

Bij het wijzigen van de regels worden de ingegeven tekens in de regels 10540-10700 verwerkt. Is het ingegeven teken geen normaal teken, dan kan het om een commando gaan. Dit gebeurt bij het teken met nummer 95, zowel als bij tekens waarbij het laagste bit een code kleiner dan 32 hebben. In het andere geval kan het ingegeven teken in de actuele regel, (zo ook in A$(IZ) opgeslagen worden. Naast deze hoofdopgave wordt nog de positie in de bovenste rechter beeldschermhoek uitgegeven. Bovendien moet gecontroleerd worden of de regel nog op het beeldscherm getoond kan worden. Eventueel moet de actuele regel een stap naar links verschoven worden. Dit gebeurt in de regels vanaf 10630.

Vanaf regel 10710 worden de tijdens de editfase ingegeven opdrachten afgehandeld. Tot aan regel 11500 zijn eerst de opdrachten behandeld, die enkel veranderingen in de regels bewerkstellen, ook de opdrachten 'cursor rechts/links, 'naar regelbegin/regeleinde', 'Insert', 'Delete', 'regel kopiëren', 'regelinhoud wissen' en 'ingave ASCII-waarde'. De opdracht 'regel kopiëren' en 'ingave ASCII-waarde', worden daarbij in verdere deelprogramma's vanaf regel 12500 resp. vanaf regel 13000 gerealiseerd.

Daarna volgen de opdrachten die een regelwissel bewerkstelligen. De opdracht 'Page-up' en Page down' roepen de reeds doorgenomen deelprogramma's op. Ook de overige bevelen worden, voor zover mogelijk, naar voorliggende programma's terug gevoerd, daar voor het wissen en invoegen van de regels overeenkomstige deelprogramma's ook voor het systeem nodig zijn.

Voor het doorlopen van tekst staan vanaf regel 17000 twee deelprogramma's ter beschikking, die de beide mogelijke door

18.2 Basisversie 'TEXT 1.0'

loopwijzen verwezenlijken. Naast hun eigen opgave, het doorlopen, moeten de deelprogramma's de nieuw geproduceerde regels weer tonen. Dit gebeurt voor beide routines samen vanaf regel 17950. Het doorlopen zelf gebeurt door opsplitsen van de actuele regel. Doorlopende wijze wordt de opvolgende regel resp. de voorafgaande regel op het rechter resp. linker actuele gedeelte van de regel opgevuld. De actuele regel behoudt het overgebleven gedeelte.

Gewist worden de regels in de routine van af regel 15000. In een FOR-loop worden de te wissen regels met de opvolgende regels overschreven. De in de velden IS() en IT() opgeslagen waarden moeten daarbij natuurlijk overeenkomstig gecorrigeerd worden.

Als nog een opdracht van de commandoregel kunnen regels verschoven worden. Deze opdracht wordt door het deelprogramma vanaf regel 15550 gerealiseerd. Ook voor deze operatie moeten alle bovengenoemde velden gereorganiseerd worden. Voordat deze reorganisatie plaatsvindt, wordt echter getest of de operatie helemaal doorvoerbaar is. Anders zou de opdracht tot een regelverlies kunnen leiden.

Als bijzondere funktie staat de gebruiker van TEXT 1.0 het sorteren van regels ter beschikking. Deze funktie wordt in de regels 20000-20980 uitgevoerd. De voor het sorteren toonaangevende deelstring van de regels wordt in het veld S$() gekopiëerd. Dan worden de regels gesorteerd, waarin de velden IS() en IZ() door sorteren van het veld S$() gereorganiseerd worden.

Het zoeken van de letterreeks in de tekst handelt de routine vanaf regel 21000 af. De te zoeken letterreeks wordt met behulp van het deelprogramma van regel 25200 in ieder relevante tekstregel gezocht. Het deelprogramma vanaf regel 25200 simuleert de op andere computers ter beschikkingstaande INSTR-funtie. Zij levert de positie van een letterreeks binnen een string. Deze funktie is ook nog op andere plaatsen van het programma nodig. Wordt de letterreeks gevonden, dan kan men de desbetreffende regel tonen. De gevonden reeks wordt daarbij met '=' onderstreept.

Het verplaatsen van letterreeksen funktioneert in principe precies zo. Nu wordt de letterreeks niet enkel getoond, maar de gevonden reeks gewist, en de nieuwe, door de gebruiker ingegeven reeks in de tekstregel ingevoegd. Dit werk wordt door het deelprogramma in de regels 22000-22470 overgenomen.

In de volgende regels 22500-22910 wordt nog de uitgave van de tweede hulptekst gerealiseerd, die de gebruiker uitleg over de besproken toetsen en mogelijke commando's geeft.

Het deelprogramma vanaf regel 23000 leest het regelbereik in, dat praktisch voor ieder commando nodig is. Het herkent de ingave van het regelbereik, met een eventuele tussenstap, zowel als de speciale ingaven ., ':', en '!'. Het zoeken van de afzonderlijke waarden in de ingave-string gebeurt weer met behulp van de gesimuleerde INSTR-funktie. De routine levert

18.2 Basisversie 'TEXT 1.0'

in I1 de startregel, in 12 de laatste regel en in 13 de tussenstap. De ingave gebeurt d.m.v. een normale INPUT-opdracht. Omdat een ingave met dubbelpunt mogelijk is, moet de ingave met behulp van het teken ' als string gekenmerkt worden. Dit neemt TEXT 1.0 automatisch over.

Vanaf regel 24000 volgen nog algemene hulproutines om de cursor te plaatsen, de ingave van een tekstenlijst van disk, de gesimuleerde INSTR-funktie en andere routines.

Drie andere opdrachten staan nog ter beschikking, om de tekst op te slaan en af te drukken. De routine vanaf regel 13500 slaat de regel op in een aan te geven file. Is de file al op de disk aanwezig kan men deze overschrijven, of een nieuwe filenaam aangeven. Bovendien kan men besluiten of de tekst met of zonder regelnummers in de file geschreven moet worden. In een FOR-loop worden dan de regels in de file geschreven en de regelnummers ter controle op het beeldscherm getoond.

Vanaf regel 14000 volgt de routine voor het inlezen van opgeslagen tekstblokken. De routine leest eerst de filenaam in en opent de file. Dan worden vragen voor de bereiken gesteld, zoals zij in de bedieningshandleiding beschreven zijn. Nu kunnen de regels ingelezen worden. Daar de regels het speciale teken zoals bijv. de komma bevatten kunnen, moet de regel middels GET ingelezen worden. Dit schakelt een van de algemene hulproutines uit. Een ingelezen regel krijgt eventueel nog een nieuw regelnummer en wordt dan in de voorhanden zijnde tekst ingevoegd. Op verzoek wordt zij bovendien op het beeldscherm ter controle uitgegeven.
De belangrijkste routine van het programma is de routine vanaf regel 18500. Daarin wordt de opgeslagen tekst op de printer uitgegeven. Daarbij wordt elke regel onderzocht, of deze het teken '/' bevat. Dit gebeurt weer met behulp van de gesimuleerde INSTR-funktie. Bevat de regel deze tekens niet, dan kan zij gewoon naar de printer gezonden worden (eventueel met regelnummer). Anders wordt het volgende teken als printeropdracht geïnterpreteerd en de aan het teken toebehorende string naar de printer gezonden. Daar deze strings uit de printerfile gelezen worden, is het mogelijk een tekst op verschillende printers uit te geven, zonder het stuurteken in de tekst te moeten veranderen.

Bij de enkelblad verwerking wordt aan het eind van een blad gewacht, totdat de gebruiker de verder verwerking beveelt. Het commando 'W' veroorzaakt de routine eveneens op een opdracht van de gebruiker te wachten.

18.2 Basisversie 'TEXT 1.0'

VARIABLE	BETEKENIS
A	Vrije geheugenruimte
A$	Algemeen teken
A1$, A2$	Hulpstring voor het wijzigen van tekst
AR$	Hulpstring voor het editeren van regels
B$	Hulpstring algemeen
B1$	Printstring: Breedschrift aan
B2$	Printstring: Breedschrift uit
CH$	ingelezen teken bij het editeren
CL$	Konstante voor het wissen van beeldscherm
DATEI$	Aktuele filenaam
DS	Diskstatus
F	Foutnummer disk
F1-F3	Kleuren tekst, kader en achtergrond
F$	Fouttekst disk
F1S	Printerstring: vetschrift aan
F2$	printerstring: vetschrift uit
FD$	Foutmeldingstekst van de floppy
A1$	Printerstring: verhogen aan
A2$	Printerstring: verhogen uit
I	Hulpindex
I1$	Regelnummer bij ingave in de kommandoregel
I1	Bereik van
I2	Bereik tot
I3	Tussenstap
I4-I9	Hulpvariable voor bereikingave
IC	ingelezen teken uit de printerfile
ID	ingegeven codegetal
II	aktuele regel
IJ	Loopindex voor FOR-loop
IK	Hulpindex
IL	Aantal tekstregels
IM	Hoogste regelnummer
IN	Hulpindex
IPO	Positie van een in de string gezochte tekens
IS	Hulpindex
IV	Loopindex voor FOR-loop
IW	Loopindex voor wachtlus
IX	Hulpindex
IY	Aantal regels op het beeldscherm
J	Loopindex voor FOR-loop
J1-J3	Hulpvariables algemeen
JC	Hulpvariables voor scheiden

18.2 Basisversie 'TEXT 1.0'

```
JJ          Loopindex voor FOR-Loop
JZ          Hulpindex algemeen
K           Regelindex
K1          Tussenopslag voor regelindex
KALT        Oude opdrachtnummer
KB          Vlag voor bereikopgave
KD          Hulpvariable
KE          Kenmerk voor kettingformulier
KI,KJ,KK    Hulpvariables
KL          Vlag voor ' geladen regels tonen'
KM$         ingegeven kommando
KNEU        aktuele opdrachtnummer
KN$         ingelezen waarde bij 'met regelnummer'
KOPF$       Kopregel "--------------------"
KT$         Konstante: 'Geen tekst voorhanden'
KZ          Vlag voor editeren
L           Hulpvariable voor positie in de string
L$          Hulpstring
L1$         Printerstring: schoonschrift aan
L2$         Printerstring: schoonschrift uit
L0-L9       Loopindex voor FOR-loop
La,LE       Hulpvariables voor stringlengte
LH          Positie bij het editeren
LJ          Stringlengte bij het scheiden
LL          Loopindex voor FOR-loop
LQ          Loopindex voor FOR-loop
LS          Hulpvariable
LS$         Wisstring
LV          Beeldschermregel
LX,LY       Loopindex voor FOR-loop
LZ          Aantal lege regels per pagina
LZ$         Wisstring
M$          tussenopslag voor MID$-funktie
MAX         Maximale aantal tekstregels
MMAX        Maximale af te drukken regels per pagina
MZ          Hulpvariable 4
N           Aantal tekens van de printerstring
N1$         Printerstring: normaalschrift aan
N2          Hulpvariable
NA          Hulpvariable voor stringpositie
NB,NC,NI    Hulpvariables
ND          Konstante: 4
NJ          Tussenopslag
NK          Aantal opdrachten (20)
NL          Aantal lege tekens linkerkantlijn bij het printen
NL$         NL open teken
NP          Regels per pagina bij het printen
NR          Nummer van de door de disk geladen regels
NS          Kolommen vanhet beeldscherm
NU          Konstante: 5
NW          Wachttijd van de wachtlus
NZ          Regelaantal van het beeldscherm
P1$         Printerstring: proportionaalschrift aan
```

18.2 Basisversie 'TEXT 1.0'

```
P2$         Printerstring: proportionaalschrift uit
Q$          Tekens (van disk gelezen)
QA          Hulpvariable voor algemene deelprogramma's
QI          Loopindex voor algemene deelprogramma's
QL          Stringlengte bij INSTR-funktie
QQ          Loopindex voor INSTR-funktie
QZ,QS       Cursorregel, Cursorkolom
QS$         String voor INSTR-Funktie
QZ$         Hulpstring
R$          Tussenopslag bij het scheiden
S1$         Printerstring: kleinschrift aan
S2$         Printerstring: kleinschrift uit
T$          Scheidingsstring: "-" of " "
T1$         Printerstring: laagstellen aan
T2$         Printerstring: laagstellen uit
TEXT$       Toon tekst bij het inlezen van disk
U1$         Printerstring: onderstrepen aan
U2$         Printerstring: onderstrepen uit
US$         String voor het onderstrepen bij het zoeken
W$          Ingavevariable bij de printerpauze
W1          Loopindex voor FOR-loop
Z$          te zoeken tekenreeks
Z1$         oude tekenreeks bij het vervangen
Z2$         nieuwe tekenreeks bij het vernangen
ZAHL        uitgave aantal bij het printen
ZAHL$       ZAHL als string
ZL          tekenpositie bij het editeren
ZL$         Hulpstring voor de bereiksberekening
```

```
VELDVARIABLE        BETEKENIS

A$()                Tekstregels
CTEXT$()            Hulptekst
IC()                Tabel van de editeeropdrachten
IS%()               Regelnummeren
IZ%()               Regels aaneenschakelen
K$()                Tabel van de tekstopdrachten
S$()                Hulpveld van de te sorteren regels
ZZ()                Tabel van de regels op het beeldscherm
```

18.2 Basisversie 'TEXT 1.0'

```
1000 rem ********************************************************************
1010 rem ***                                                              ***
1020 rem ***                          tekstverwerker                      ***
1030 rem ***                                                              ***
1040 rem ********************************************************************
1050 :
1060 rem ------------------------------------------------------------------------
1070 rem ---          variablen vastleggen & opdrachtlijst uitgeven          ---
1080 rem ------------------------------------------------------------------------
1090 :
1100 dim ic(20),ctext$(20),k$(20)
1110 :
1120 nk=20
1130 :
1140 data l,z,,,e,c,q,r,h,i,b,v,,d,s,a,f,m,?,w
1150 :
1160 for i=1 to 20
1170 : read k$(i)
1180 next
1190 :
1200 max=20
1210 :
1220 dim a$(max),iz%(max),is%(max),s$(max)
1230 dim zz(25)
1240 :
1250 iz%(0)=1
1260 nw=250
1270 :
1280 f$=">>> Ingave fout <<<"
1290 kt$=">>> Geen Tekst Aanwezig <<<"
1300 nu=5
1310 nd=4
1320 ii=1
1330 il=0
1340 im=0
1500 :
1510 rem ------------------------------------------------------------------------
1520 rem ---                          goleurkeuze                           ---
1530 rem ------------------------------------------------------------------------
1540 :
1550 cl$=chr$(147)+chr$(14)
1560 printcl$;
1570 print"┌──────────────────────────────────────┐"
1580 print"                kleurkeuze "
1590 print"└──────────────────────────────────────┘"
1600 print
1610 print"  F1 : Kader"
1620 print"  F3 : Achtergrond"
1630 print"  F5 : Tekst"
1640 print
1650 print" RETURN : klaar"
1660 print
1670 print"┌──────────────────────────────────────┐"
1680 print
1690 print
1700 print
1710 print"└──────────────────────────────────────┘"
1720 :
1730 f1=11
1740 f2=0
1750 f3=7
1760 :
1770 get a$
```

18.2 Basisversie 'TEXT 1.0'

```
1780 if a$=chr$(13) then 1890
1790 if a$=chr$(133) then f1=f1+1:if f1=16 then f1=0
1800 if a$=chr$(134) then f2=f2+1:if f2=16 then f2=0
1810 if a$=chr$(135) then f3=f3+1:if f3=16 then f3=0
1820 poke53280,f1
1830 poke53281,f2
1840 poke 646,f3
1850 print "■■■■■■■■■■■■■■■■  Tekstkleur"
1860 :
1870 goto 1770
1880 :
1890 data 25,39
1900 readnz,ns
1910 :
1920 data 29,157,145,17,19,95,6,133,134,148,20,135,136,4,14,12,25,1,0,0
1930 :
1940 fori=1to20
1950 : readic(i)
1960 next
1970 :
1980 ctext$(01)="CHRS rechts.....Cursor naar rechts"
1990 ctext$(02)="CHRS links......Cursor naar links"
2000 ctext$(03)="CHRS omhoog.....Cursor naar boven"
2010 ctext$(04)="CUSOR omlaag....Cursor naar beneden"
2020 ctext$(05)="HOME............Kommando L,E,Z beeindigen"
2030 ctext$(06)="←...............Naar regel begin"
2040 ctext$(07)="CTRL←...........Naar regel einde"
2050 ctext$(08)="F1..............Terug bladeren"
2060 ctext$(09)="F3..............Vooruit bladeren"
2070 ctext$(10)="INST............Lege plaats invoegen"
2080 ctext$(11)="DEL.............Teken wissen"
2090 ctext$(12)="F5..............Scheiden: Links boven"
2100 ctext$(13)="F7..............Scheiden: rechts onder"
2110 ctext$(14)="CTRL-D..........Regels kopieren"
2120 ctext$(15)="CTRL-N..........Nieuwe regel"
2130 ctext$(16)="CTRL-L..........Regelinhoud wissen"
2140 ctext$(17)="CTRL-Y..........Regel wissen"
2150 ctext$(18)="CTRL-A..........Codegetal inlezen"
2160 :
2170 kopf$="-----------------------------------------"
2180 lz$="                                         "
2430 rem ------------------------------------------------------------------------------
2500 :
2510 rem ------------------------------------------------------------------------------
2520 rem ---                          file's openen                                ---
2530 rem ------------------------------------------------------------------------------
2540 :
2550 open15,8,15 : rem diskdrive
2560 open4,4     : rem printer
3000 :
3010 rem ------------------------------------------------------------------------------
3020 rem ---                       printerfile inlezen                             ---
3030 rem ------------------------------------------------------------------------------
3040 :
3050 k$(3)=chr$(ic(8))
3060 k$(4)=chr$(ic(9))
3070 k$(13)=chr$(ic(4))
3080 :
3090 n2=nz-2
3100 ls=ns-nu
3110 ls$=""
3120 fori=1tols-1
```

18.2 Basisversie 'TEXT 1.0'

```
3130 : ls$=ls$+" "
3140 next
3150 :
3160 poke198,1:poke631,34

3170 input"Welk type printer";file$
3180 if file$=""then file$="epson"
3190 open1,8,2,file$+".prn,u,r"
3200 input#15,f,f$
3210 iffthenprintf,f$:close1:foriw=1tonw:nextiw:goto3160
3220 :
3230 u1$="":input#1,n
3240 fori=1ton
3250 : input#1,ic:u1$=u1$+chr$(ic)
3260 next
3270 u2$="":input#1,n
3280 fori=1ton
3290 : input#1,ic:u2$=u2$+chr$(ic)
3300 next
3310 f1$="":input#1,n
3320 fori=1ton
3330 : input#1,ic:f1$=f1$+chr$(ic)
3340 next
3350 f2$="":input#1,n
3360 fori=1ton
3370 : input#1,ic:f2$=f2$+chr$(ic)
3380 next
3390 b1$="":input#1,n
3400 fori=1ton
3410 : input#1,ic:b1$=b1$+chr$(ic)
3420 next
3430 b2$="":input#1,n
3440 fori=1ton
3450 : input#1,ic:b2$=b2$+chr$(ic)
3460 next
3470 h1$="":input#1,n
3480 fori=1ton
3490 : input#1,ic:h1$=h1$+chr$(ic)
3500 next
3510 h2$="":input#1,n
3520 fori=1ton
3530 : input#1,ic:h2$=h2$+chr$(ic)
3540 next
3550 t1$="":input#1,n
3560 fori=1ton
3570 : input#1,ic:t1$=t1$+chr$(ic)
3580 next
3590 t2$="":input#1,n
3600 fori=1ton
3610 : input#1,ic:t2$=t2$+chr$(ic)
3620 next
3630 n1$="":input#1,n
3640 fori=1ton
3650 : input#1,ic:n1$=n1$+chr$(ic)
3660 next
3670 s1$="":input#1,n
3680 fori=1ton
3690 : input#1,ic:s1$=s1$+chr$(ic)
3700 next
```

18.2 Basisversie 'TEXT 1.0'

```
3710 s2$="":input#1,n
3720 fori=1ton
3730 : input#1,ic:s2$=s2$+chr$(ic)
3740 next
3750 l1$="":input#1,n
3760 fori=1ton
3770 : input#1,ic:l1$=l1$+chr$(ic)
3780 next
3790 l2$="":input#1,n
3800 fori=1ton
3810 : input#1,ic:l2$=l2$+chr$(ic)
3820 next
3830 p1$="":input#1,n
3840 fori=1ton
3850 : input#1,ic:p1$=p1$+chr$(ic)
3860 next
3870 p2$="":input#1,n
3880 fori=1ton
3890 : input#1,ic:p2$=p2$+chr$(ic)
3900 next
3910 :
3920 input#1,ke : rem kenmerk,kettingformulier
3930 input#1,np : rem aantal regels per pagina
3940 input#1,lz : rem aantal blanco regels per pagina (kop + staart)
3950 :
3960 close1
4000 :
4010 rem ------------------------------------------------------------------------
4020 rem ---                        gebruikers informatie                      ---
4030 rem ------------------------------------------------------------------------
4040 :
4050 print
4060 print "Toelaatbare regelnummers: 1 tot";max
4070 print
4080 print"Wanneer men als Kommando een Vraagteken ingeeft (zonder Return) , ";
4090 print"verschijnt op het beeldscherm een korte Bedienings"
4100 print"handleiding":print
4110 print"Toets indrukken.";
4120 :
4130 gosub24120
4140 a$=qz$
4150 print cl$;
4160 iy=2
4170 file$=""
4180 print
4190 i1$=""
4200 kz=0
4500 :
4510 rem ------------------------------------------------------------------------
4520 rem ---              commandoregel/hoofdlus                               ---
4530 rem ------------------------------------------------------------------------
4540 :
4550 qz=1:qs=1:gosub24060
4560 print lz$
4570 print kopf$;
4580 qz=1:qs=1:gosub24060
4590 print "* ";
4600 gosub24120
4610 km$=qz$
4620 ifkm$=k$(3)orkm$=k$(13) then4760
4630 print km$;
```

18.2 Basisversie 'TEXT 1.0'

```
4640 kk=asc(km$)
4650 if kk<>13 and (kk<48 or kk>58) then4740
4660 ifkk<>13andkk<>58theni1$=i1$+km$:goto4600
4670 i1=val(i1$)
4680 ifi1<1ori1>maxthen4190
4690 i2=i1
4700 i3=1
4710 knie=1
4720 kz=1
4730 goto4830
4740 ifkk>127thenkk=kkand127:km$=chr$(kk)
4750 :
4760 forknie=1tonk
4770 : ifk$(knie)=km$then4810
4780 next knie
4790 goto4190
4800 :
4810 ifknie=3orknie=4or(knie>=6andknie<=10)orknie=16orknie=17then4970
4820 ifknie=19orknie=20then4970
4830 ifkoud<8orkoud=11orkoud=13orkoud=14then4970
4840 printcl$;
4850 qz=2:qs=1:gosub24060
4860 printkopf$
4870 ifiy=2then4930
4880 forij=3toiy
4890 : i=zz(ij)
4900 : iz=iz%(i)
4910 : printright$(" "+str$(i)+": ",5);left$(a$(iz),ls)
4920 next ij
4930 qz=1:qs=1:gosub24060
4940 print"* ";
4950 iflen(k$(knie))=1thenprintk$(knie);
4960 :
4970 onkniegosub7610,8550,9050,9550,10050,5550,6550,13550
4980 ifknie>8thenonknie-8gosub14050,7050,15050,15550,7550,18550,20050,6050
4990 ifknie>16thenonknie-16gosub21050,22050,22550,20930
5000 koud=knie
5010 goto 4190
5500 :
5510 rem ------------------------------------------------------------------------------
5520 rem ---                          wis opdracht                                  ---
5530 rem ------------------------------------------------------------------------------
5540 :
5550 printcl$;
5560 iy=2
5570 return
6000 :
6010 rem ------------------------------------------------------------------------------
6020 rem ---                    uitkomst geven  (a-opdracht)                        ---
6030 rem ------------------------------------------------------------------------------
6040 :
6050 a=fre(0)
6060 printcl$;
6070 qz=4:qs=1:gosub24060
6080 :
6090 if datei$<>""then print "Naam van de file.....:";datei$
6100 print"Vrije Geheugenruimte.....:";a;"Bytes."
6110 print"Aantal Tekstregels.....:";il
6120 print"Hoogste Nummer.....:";im
```

18.2 Basisversie 'TEXT 1.0'

```
6130 print"Hoogsttoelaatbare Nummer.:";max
6140 if il>0 then print"Aktuele Regel.............:";ii
6150 return
6500 :
6510 rem ------------------------------------------------------------------------------
6520 rem ---                              stop opdracht                               ---
6530 rem ------------------------------------------------------------------------------
6540 :
6550 qz=1:qs=1:gosub24060
6560 print"<Q>..Programma einde  <Return>..verder";
6570 gosub24120
6580 a$=qz$
6590 printa$;
6600 ifa$<>"Q"anda$<>"q"thenreturn
6610 :
6620 printcl$;
6630 close15
6640 close4
6650 end
7000 :
7010 rem ------------------------------------------------------------------------------
7020 rem ---                          diskette inhoud uitgeven                        ---
7030 rem ------------------------------------------------------------------------------
7040 :
7050 printcl$;
7060 qz=4:qs=1:gosub24060
7070 open2,8,0,"$*=s"
7080 get#2,a$,a$
7090 get#2,a$,a$,a$,b$
7100 ifst<>0then7160
7110 printasc(a$+chr$(0))+256*asc(b$+chr$(0));
7120 get#2,a$
7130 ifa$<>""thenprinta$;:goto7120
7140 print
7150 ifst=0then7090
7160 close2
7170 close1
7180 return
7500 :
7510 rem ------------------------------------------------------------------------------
7520 rem ---                          editeren na 'crsr down'                         ---
7530 rem ------------------------------------------------------------------------------
7540 :
7550 i2=max
7560 i3=1
7570 if iy>2 then i1=zz(3):goto8180
7580 i1=1
7590 if iz%(ii)>0 theni1=ii
7600 goto 8180
7610 if kz=1 then8180
8000 :
8010 rem ------------------------------------------------------------------------------
8020 rem ---                              regels  inlezen                             ---
8030 rem ------------------------------------------------------------------------------
8040 :
8050 printcl$;
8060 iy=2
8070 print"* L";
8080 print"    Regels lezen";
8090 gosub 23050
8100 print kopf$
8110 if i1=-1 then return
8120 if i3=0 then i3=1
```

18.2 Basisversie 'TEXT 1.0'

```
8130 if i1=0 then i1=im+i3:if i2=0 then i2=max
8140 if i2=0and kb=1 then i2=max
8150 if i2=0 then i2=i1
8160 ifi2>maxthenprint">>>Regelnummer tegroot.<<<":foriw=1tonw:next:return
8170 :
8180 qz=1:qs=1:gosub24060
8190 print lz$;
8200 forlx=i1toi2stepi3
8210 : iz=iz%(lx)
8220 : if iz>0 then8310
8230 : il=il+1
8240 : iz=il
8250 : iz%(lx)=iz
8260 : is%(il)=lx
8270 : qz$=" "
8280 : qa=ns-nu
8290 : gosub24220
8300 : a$(iz)=qz$
8310 : ii=lx
8320 : if ii>im then im=ii
8330 : j1=i1
8340 : j2=i2
8350 : j3=i3
8360 : i1=lx
8370 : i2=lx
8380 : i3=1
8390 : kj=0
8400 : gosub 10290: rem --> kommando e
8410 : lx=ii
8420 : i1=j1
8430 : i2=j2
8440 : i3=j3
8450 : lx=ii
8460 : if kj>im thenlx=i2
8470 : if ic=ic(5)thenlx=i2
8480 nextlx
8490 return
8500 :
8510 rem ------------------------------------------------------------------------
8520 rem ---                          regels afbeelden                        ---
8530 rem ------------------------------------------------------------------------
8540 :
8550 if il=0 then print kt$;:foriw=1tonw:nextiw:return
8560 print"    Afbeelden Regels";
8570 gosub 23050
8580 if i1<0 or i1>im then return
8590 gosub 23840
8600 printcl$;
8610 iy=2
8620 mz=1
8630 forly=i1toi2stepi3
8640 : iz=iz%(ly)
8650 : if iz=0 then8840
8660 : mz=mz+1
8670 : if mz<=n2 then8730
8680 : mz=1
8690 : gosub24120
8700 : ch$=qz$
8710 : ic=asc(ch$)
8720 : if ic=81 or ic=113 or ic=ic(5) thenly=i2:goto 8840
8730 : iy=i
```

18.2 Basisversie 'TEXT 1.0'

```
          y+1
8740 : qz=iy:qs=1:gosub24060
8750 : printright$(" "+str$(ly)+": ",5);
8760 : print left$(a$(iz),ls)
8770 : zz(iy)=ly
8780 : if iy<nz then 8830
8790 : for ij=3 to nz-1
8800 :   zz(ij)=zz(ij+1)
8810 : next ij
8820 : iy=nz-1
8830 : ii=ly
8840 nextly
8850 return
9000 :
9010 rem ------------------------------------------------------------------------------
9020 rem ---                    pagina omhoog / terugbladeren                      ---
9030 rem ------------------------------------------------------------------------------
9040 :
9050 printcl$;
9060 if il=0 then print kt$;:foriw=1tonw:nextiw:return
9070 qz=2:qs=1:gosub24060
9080 iy=2
9090 print kopf$;
9100 mz=1
9110 if koud=4 then ii=i1
9120 i2=ii
9130 forw1=ii to 1 step -1
9140 : iz=iz%(w1)
9150 : if iz=0 then 9190
9160 : mz=mz+1
9170 : i1=w1
9180 : if mz=n2 thenw1=1
9190 nextw1
9200 :
9210 forw1=i1 to ii
9220 : iz=iz%(w1)
9230 : if iz=0 then 9280
9240 : iy=iy+1
9250 : qz=iy:qs=1:gosub24060
9260 : printright$(" "+str$(w1)+": ",5);left$(a$(iz),ls)
9270 : zz(iy)=w1
9280 nextw1
9290 ii=i1
9300 return
9500 :
9510 rem ------------------------------------------------------------------------------
9520 rem ---                    pagina omlaag / vooruitbladeren                    ---
9530 rem ------------------------------------------------------------------------------
9540 :
9550 printcl$;
9560 if il=0 then print kt$;:foriw=1to nw:nextiw:return
9570 qz=2:qs=1:gosub24060
9580 iy=2
9590 print kopf$;
9600 if koud=3 then ii=i2
9610 i1=ii
9620 forw3=ii to im
9630 : iz=iz%(w3)
9640 : if iz=0 then 9730
9650 : iy=iy+1
9660 : qz=iy
9670 : qs=1
9680 : gosub24060
9690 : printright$(" "+str$(w3)+": ",5);left$(a$(iz),ls)
9700 : zz(iy)=w3
```

18.2 Basisversie 'TEXT 1.0'

```
9710 : ii=w3
9720 : if iy>n2 thenw3=im
9730 nextw3
9740 i2=ii
9750 return
10000 :
10010 rem ------------------------------------------------------------------------------
10020 rem ---                          regels editeren                              ---
10030 rem ------------------------------------------------------------------------------
10040 :
10050 if i1=0 then print kt$;:foriw=1tonw:nextiw:return
10060 print"    Editeren  Regels";
10070 gosub 23050
10080 if i1<0 or i1>im then return
10090 if i2>im then i2=im
10100 if kb=1 and i2=0 theni2=im
10110 if i1=0 and i2=0 then i2=im
10120 if i1=0 and iy>2 and i2>=zz(3)then i1=zz(3)
10130 if i1=0 then i1=1:if iz%(ii)>0 then i1=ii
10140 if i2=0 then i2=i1
10150 if i3=0 then i3=1
10160 qz=1:qs=1:gosub24060
10170 print lz$;
10180 jj=0
10190 :
10200 forij=3toiy
10210 : ifzz(ij)=i1thenjj=ij:ij=iy
10220 nextij
10230 :
10240 ifjj<>0thenij=jj:goto10290
10250 printcl$;
10260 iy=2
10270 print
10280 print kopf$
10290 forlq=i1 to i2 step i3
10300 : ic=-1
10310 : iz=iz%(lq)
10320 : if iz=0 then13160
10330 : if ic<>ic(3)then zl=0:rem aansl.linkse tekens, die niet zichtbaar zijn
10340 : le=len(a$(iz))
10350 : nb=ls-le+1
10360 : if nb<1then nb=1
10370 : if nb>1 then nb=nb-1
10380 : qz$=" "
10390 : qa=nb
10400 : gosub24220
10410 : a$(iz)=a$(iz)+qz$
10420 : le=le+nb
10430 : if ic=ic(3)or ic=ic(4) or ic=ic(13)then10540
10440 : if ic=ic(12) then 10530
10450 : if iy=2 then lv=3:goto 10500
10460 : for ij=3 to iy
10470 :   if zz(ij)=lqthen lv=ij:goto 10530
10480 : nextij
10490 : lv=iy+1
10500 : qz=lv:qs=1:gosub24060
10510 : printright$(" "+str$(lq)+": ",5);
10520 : print left$(a$(iz),ls);
10530 : lh=nu+1
10540 : qz=1:qs=ns-11:gosub24060
10550 : print"  Spatie:";right$("  "+str$(lh-nu+zl),3);
10560 : qz=lv:qs=lh:gosub24060
10570 : gosub24120
10580 : ch$=qz$
10590 : ic=asc(ch$)
```

18.2 Basisversie 'TEXT 1.0'

```
10600 : if(icand127)<32oric=ic(6)then10710
10610 : printch$;
10620 : a$(iz)=left$(a$(iz),lh-nu+zl-1)+ch$+mid$(a$(iz),lh-nu+zl+1)
10630 : if le<254 or lh<ns then lh=lh+1
10640 : iflh<=nsthen10540
10650 : lh=ns
10660 : zl=zl+1
10670 : qz=lv:qs=nu+1:gosub24060
10680 : printmid$(a$(iz),zl+1,ls);
10690 : ifle<zl+lsthena$(iz)=a$(iz)+" ":le=le+1:print" ";
10700 : goto10540
10710 : ific<>ic(1)then10750
10720 : lh=lh+1
10730 : goto24410
10740 : goto10540
10750 : ific<>ic(2)then10790
10760 : lh=lh-1
10770 : goto24500
10780 : goto10540
10790 : ific=ic(6)thenlh=nu+1:goto10540
10800 : ific=ic(7)thenlh=ns:goto10540
10810 : ific=ic(16)thenlh=nu+1:a$(iz)=ls$:qz=lv:qs=lh:gosub24060:printls$;:zl=0:
goto10540
10820 : ific<>ic(10)then10910
10830 : ifle=254then10540
10840 : ar$=" "+right$(a$(iz),le-lh-zl+nu+1)
10850 : printleft$(ar$,ns-lh+1);
10860 : nb=0
10870 : ifright$(a$(iz),1)=" "thennb=1
10880 : a$(iz)=left$(a$(iz),lh+zl-nu-1)+left$(ar$,len(ar$)-nb)
10890 : le=le+1-nb
10900 : goto10540
10910 : ific<>ic(11)then11000
10920 : b$=""
10930 : l=-1
10940 : ifle<=ls+zlthenb$=" ":l=0
10950 : ar$=right$(a$(iz),le-lh-zl+nu)+b$
10960 : printleft$(ar$,ns-lh+1);
10970 : a$(iz)=left$(a$(iz),lh+zl-nu-1)+ar$
10980 : le=le+1
10990 : goto10540
11000 : ific<>8then11120
11010 : b$=""
11020 : l=-1
11030 : ifle<=ls+zlthenb$=" ":l=0
11040 : if lh=nu+1then10540
11050 : ar$=right$(a$(iz),le-lh-zl+nu+1)+b$
11060 : lh=lh-1
11070 : qz=lv:qs=lh:gosub24060
11080 : printleft$(ar$,ns-lh+1);
11090 : a$(iz)=left$(a$(iz),lh+zl-nu-1)+ar$
11100 : le=le+1
11110 : goto 10540
11120 : ific=ic(14)then12560
11130 : ific=ic(18)then13050
11500 :
11510 rem hier worden de kodes verwerkt,die een regel wissel bewerken
11520 rem dat zijn return,ic(3),ic(4),ic(5),ic(8),ic(9),ic(12),ic(13),
11530 rem ic(15) und ic(17):
11540 :
11550 : ific=ic(3)andlv=3then10540
11560 : ific=ic(4)andlv>=iythen10540
11570 : ifzl<0oric=ic(12)oric=ic(13)then11610
11580 : qz=lv:qs=nu+1:gosub24060
11590 : printleft$(a$(iz),ls);
11600 : zl=0
```

18.2 Basisversie 'TEXT 1.0'

```
11610 : ii=lq
11620 : nb=0
11630 : forj=le tolstep-1
11640 :   ifmid$(a$(iz),j,1)<>" "thenj=1:goto11660
11650 :   nb=nb+1
11660 : nextj
11670 : ifnb>0thena$(iz)=left$(a$(iz),le-nb)
11680 : ific=ic(4)thenlq=zz(lv+1):lv=lv+1:goto10310
11690 : goto24590
11700 : iflv>iytheniy=lv
11710 : zz(lv)=ii
11720 : ific=13then13160
11730 : ific=ic(5)thenlq=i2:goto13160
11740 : ific=333then24690
11750 : ific=ic(3)thenlq=zz(lv-1):lv-1:goto10310
11769 : ific<>ic(12)andic<>ic(13)then11850
11770 : le=len(a$(iz))
11780 : k1=i1
11790 : k=lq
11800 : i1=lq
11810 : gosub17050
11820 : i1=k1
11830 : lq=k
11840 : goto10340
11850 : ific=ic(15)thenkk=ii:ki=zz(3):i1=ii:i2=im:i3=1:i4=i1+1:gosub24770
11860 : ific=ic(15)goto24860
11870 : ific=ic(17)andii<imthenkk=ii:ki=zz(3):i1=ii+1:i2=im:gosub25000:goto10300
11880 : ific<>ic(17)then12040
11890 : ki=zz(3)
11900 : i1=im
11910 : i2=im
11920 : i3=1
11930 : kj=ki
11940 : ifiy>3thenkj=zz(lv-1)
11950 : gosub15200
11960 : ii=kir
11970 : i2=ki
11980 : gosub9550
11990 : lq=kj
12000 : i1=kj
12010 : i2=im
12020 : i3=1
12030 : goto10300
12040 : ific=ic(8)theni1=ii:k=ii:gosub9050:lq=k:i1=k:i2=im:i3=1:goto10300
12050 : ific=ic(9)theni2=ii:k=ii:gosub9550:lq=k:i1=k:i2=im:i3=1:goto10300
12060 : lq=ii
12070 : goto10300
12500 rem einde regel wissel-verwerking.
12510 :
12520 rem ------------------------------------------------------------------------
12530 rem ---                          regels kopieren                          ---
12540 rem ------------------------------------------------------------------------
12550 :
12560 : qz=1:qs=1:gosub24060
12570 : printlz$;
12580 : qz=1:qs=1:gosub24060
12590 : input"    Kopieren Regel";kk
12600 : ifkk<1orkk>maxthen12560
12610 : jz=iz%(kk)
12620 : ifjz>0then12700
12630 : qz=1:qs=1:gosub24060
12640 : printlz$;
12650 : qz=1:qs=1:gosub24060
12660 : gosub25100
12670 : qz=1:qs=1:gosub24060
12680 : printlz$
```

18.2 Basisversie 'TEXT 1.0'

```
12690 : goto10540
12700 : zl=0
12710 : a$(iz)=a$(jz)
12720 : le=len(a$(iz))
12730 : nb=ls-le+1
12740 : ifnb<1thennb=1
12750 : qz$=" ":qa=nb:gosub24220
12760 : a$(iz)=a$(iz)+qz$
12770 : le=le+nb
12780 : goto25130
13000 :
13010 rem ------------------------------------------------------------------------
13020 rem ---                          kodegetal inlezen                        ---
13030 rem ------------------------------------------------------------------------
13040 :
13050 : qz=1:qs=1:gosub24060
13060 : printlz$;
13070 : qz=1:qs=1:gosub24060
13080 : input"    kodegetal";id
13090 : ifid<0orid>255then13050
13100 : ch$=chr$(id)
13110 : qz=lv:qs=lh:gosub24060
13120 : printch$;
13140 : printlz$;
13150 : goto10620
13160 nextlq
13170 ific=ic(15)andknie=5then10290
13180 return
13500 :
13510 rem ------------------------------------------------------------------------
13520 rem ---                           regels opslaan                          ---
13530 rem ------------------------------------------------------------------------
13540 :
13550 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
13560 printcl$;
13570 print"* R";
13580 print"    regels opslaan";
13590 gosub23050
13600 ifil<0oril>imthenreturn
13610 gosub23840
13620 print
13630 :
13640 input"File Naam ";file$
13650 le=len(file$)
13660 ifle=0thenreturn
13670 open1,8,2,file$+",s,w"
13680 input#15,f,fd$
13690 iff=63thenprint"file overschrijven?";
13695 :gosub24120:ifqz$<>"j"andqz$<>"j"thenclose1:return
13700 iff=63thenclose1:print#15,"s0:"+file$:goto13670
13710 iff<>0thenprintf,fd$:foriw=1tonw:next:close1:return
13720 print"Uitgave met nummers(j/n) ? ";
13730 gosub24120
13740 kn$=qz$
13750 printkn$
13760 ifkn$="j"thenkn$="J"
13770 ifkn$="n"orasc(kn$)=13thenkn$="N"
13780,ifkn$<>"J"andkn$<>"N"then13720
13790 :
13800 print
13810 print"Volgende Regels zijn opgeslagen:"
13820 print
13830 forl1=i1toi2stepi3
13840 : iz=iz%(l1)
13850 : ifiz=0then13890
13860 : ifkn$="J"thenprint#1,right$(" "+str$(l1)+": ",5);
```

18.2 Basisversie 'TEXT 1.0'

```
13870 : print#1,a$(iz)
13880 printright$(" "+str$(l1)+": ",5);
13890 nextl1
13900 close1
13910 print:print
13920 print"Einde Opslag procedure."
13930 return
14000 :
14010 rem ------------------------------------------------------------------------------
14020 rem ---                          regels van disk lezen                         ---
14030 rem ------------------------------------------------------------------------------
14040 :
14050 printcl$;
14060 print"* H";
14070 input"   File Naam ";file$
14080 le=len(file$)
14090 ifle=0thenreturn
14100 open1,8,2,file$+",s,r"
14110 input#15,f,fd$
14120 iff<>0thenprintf,fd$:foriw=1tonw:next:close1:return
14130 jb=0
14140 text$="In Werkgeheugen reeds tekst aanwezich"+chr$(13)+"HoogsteRegelnr.:"
14150 ifil>0thenprint:printcl$;text$;im
14160 print
14170 print"Welke Regels moeten geladen worden"
14180 print"(Nummeren op Diskette)";
14190 gosub23050
14200 j1=i1
14210 ifj1<0thenclose1:return
14220 j2=i2
14230 ifkb=1andj2=0thenj2=32767
14240 ifj1=0thenj1=1:ifj2=0thenj2=32767
14250 ifj2=0thenj2=j1
14260 print
14270 print"Indien de geladen Regels nieuw genummerd moeten worden,";
14280 print" eerste Nummer en eventueel tussenstap aangeven:"
14290 gosub23050
14300 ifi1<0thenclose1:return
14310 ifi3=0theni3=1
14320 lq=i1-i3
14330 print
14340 print"Geladen Regels afbeelden(J=ja)?";
14350 gosub24120
14360 a$=qz$
14370 printa$;
14380 k1=0
14390 ifa$="J"ora$="j"thenk1=1
14400 :
14410 print:print"geladen Regels:"
14420 print
14430 ipo=nu
14440 kd=1
14450 k=0
14460 nr=0
14470 ds=0
14480 ifds<>0then14880
14490 gosub24320
14500 ifkd=0then14550
14510 ifmid$(a$,nd,1)<>":"thenkd=0:goto14550
14520 nr=val(left$(a$,nd-1))
14530 ifnr=0thenkd=0:goto14550
14540 goto14560
14550 nr=nr+1
14560 ifnr<j1then14490
14570 ifnr>j2then14880
14580 lq=lq+i3
```

18.2 Basisversie 'TEXT 1.0'

```
14590 ifil=0thenlq=nr
14600 iflq>maxthen14880
14610 iz=iz%(lq)
14620 ifiz>0then14670
14630 il=il+1
14640 iz=il
14650 iz%(lq)=iz
14660 is%(il)=lq
14670 ifkd=1thena$(iz)=right$(a$,len(a$)-ipo)
14680 ifkd<>1thena$(iz)=a$
14690 ifright$(a$(iz),1)=" "thena$(iz)=left$(a$(iz),len(a$(iz))-1:goto14690
14700 ii=lq
14710 ifii>imthenim=ii
14720 qz=14
14730 qs=1
14740 gosub24060
14750 print"Nr. Diskette:"nr""r. eheugen";lq;
14760 ifkl=0then14480
14770 qz=16+k
14780 qs=1
14790 gosub24060
14800 printlz$;
14810 qz=16+k
14820 qs=1
14830 gosub24060
14840 printright$(" "+str$(lq)+": ",5);left$(a$(iz),ls);
14850 k=k+1
14860 ifk>nz-17thenk=0
14870 goto14480
14880 close1
14890 iy=2
14900 return
15000 :
15010 rem ------------------------------------------------------------------------------
15020 rem ---                          regels wissen                                 ---
15030 rem ------------------------------------------------------------------------------
15040 :
15050 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
15060 print"    Regels afvoeren";
15070 gosub23050
15080 ifi1=-1then return
15090 ifi1>0andkb=0then15150
15100 qz=1:qs=1:gosub24060
15110 printlz$;
15120 qz=1:qs=1:gosub24060
15130 print">>>Exakte regelaanduiding vereist.<<<";
15140 foriw=1tonw:nextiw:return
15150 ifi2=0theni2=i1
15160 ifi3=0theni3=1
15170 ifi2>imtheni2=im
15180 qz=1:qs=1:gosub24060
15190 printlz$;
15200 forll=i1toi2stepi3
15210 : iz=iz%(ll)
15220 : ifiz=0then15360
15230 : ifiz=ilthen15280
15240 : a$(iz)=a$(il)
15250 is=is%(il)
15260 iz%(is)=iz
15270 : is%(iz)=is
15280 : a$(il)=""
15290 : is%(il)=0
15300 : il=il-1
15310 : iz%(ll)=0
15320 : ifll<imthen15360
15330 : forij=im-1to0step-1
```

18.2 Basisversie 'TEXT 1.0'

```
15340 :    ifiz%(ij)>0thenim=ij:ij=0
15350 : nextij
15360 nextll
15370 f=fre(0)
15380 ifil=0theniy=2
15390 ifiy<>2thenii=zz(3):i2=ii:gosub9550
15400 ifiy=2thenprintcl$;
15410 return
15500 :
15510 rem ------------------------------------------------------------------------------
15520 rem ---                          regels verplaatsen                            ---
15530 rem ------------------------------------------------------------------------------
15540 :
15550 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
15560 print"    Regels verplaatsen";
15570 gosub23050
15580 ifi1=-1thenreturn
15590 ifi1>0andkb=0then15650
15600 qz=1:qs=1;gosub24060
15610 printlz$;
15620 qz=1:qs=1:gosub24060
15630 print">>>Exakte Regelaanduiding vereist.<<<";
15640 foriw=1tonw:nextiw:return
15650 ifi2=0theni2=i1
15660 ifi3=0theni3=1
15670 ifi2>imtheni2=im
15680 i5=abs(i2-i1)
15690 delta=abs(i5/i3)
15700 ifdelta=int(delta+.0001)andi2>=i1then15740
15710 qz=1:qs=ns-25:gosub24060
15720 printf$;
15730 foriw=1tonw:nextiw:return
15740 zw$=""
15750 ifi2<>i1thenzw$="eerste "
15760 qz=1:qs=1:gosub24060
15770 printlz$;
15780 qz=1:qs=1:gosub24060
15790 print"eind nr.";zw$;"verpl. regel";
15800 inputi4
15810 ifi4>=1andi4<=maxthen15890
15820 qz=1:qs=1:gosub24060
15830 printlz$;
15840 qz=1:qs=1:gosub24060
15850 printf$;
15860 foriw=1tonw:nextiw
15870 return
15880 :
15890 ifi4=i1thenreturn
15900 qz=1:qs=1:gosub24060
15910 printlz$;
15920 ic=-1
15930 ni=i4-i1
15940 ifi2+ni<=maxthen16000
15950 qz=1:qs=1:gosub24060
15960 print">>>Niet uitvoerbaar.<<<";
15970 foriw=1tonw:next
15980 return
15990 :
16000 ifni>0theni5=i1:i1=i2:i2=i5:i3=-i3
16010 ii=0
16020 fori0=i1toi2stepi3
16030 : iz=iz%(i0)
16040 : in=i0+ni
16050 : ix=iz%(in)
16060 : ifix=0then16160
16070 : ifix=ilthen16130
```

18.2 Basisversie 'TEXT 1.0'

```
16080 : a$(ix)=a$(il)
16090 : is=is%(il)
16100 : iz%(is)=ix
16110 : is%(ix)=is
16120 : iz=iz%(10)
16130 : a$(il)=""
16140 : is%(il)=0
16150 : il=il-1
16160 : iz%(in)=iz
16170 : iz%(10)=0
16180 : is%(iz)=in
16190 : ifin>imandiz>0thenim=in
16200 : ifi3<0andi1=0andiz>0thenii=in
16210 : ifi3>0andiz>0thenii=in
16220 nextl0
16230 ifni>0ori2<>imthen16270
16240 forqa=im-1to0step-1
16250 : ifiz%(qa)>0thenim=qa:qa=0
16260 nextqa
16270 ific=ic(15)thenreturn
16280 printcl$;
16290 iy=2
16300 return
17000 :
17010 rem ------------------------------------------------------------------------------
17020 rem ---                              scheiden                                  ---
17030 rem ------------------------------------------------------------------------------
17040 :
17050 ifle<1thenreturn
17060 is=lh+zl-nu
17070 zl=0
17080 ific=ic(13)then17550
17100 :
17110 rem ------------------------------------------------------------------------------
17120 rem ---                          scheiden naar boven                           ---
17130 rem ------------------------------------------------------------------------------
17140 :
17150 ifis>le thenis=le
17160 j1=0
17170 forl9=i1-1to1step-1
17180 : jz=iz%(19)
17190 : ifjz=0then17450
17200 : lj=len(a$(jz))
17210 : m$=mid$(a$(iz),is,1)
17220 : t$="-"
17230 : ifm$=" "orm$="-"oris=le thent$=""
17240 : ifis=le then17260
17250 : ifmid$(a$(iz),is+1,1)=" "thent$="":is=is+1
17260 : b$=" "
17270 : r$=right$(a$(jz),1)
17280 : iflj=0orr$=" "orr$="-"thenb$=""
17290 : ifr$="-"thena$(jz)=left$(a$(jz),lj-1)
17300 : a$=left$(a$(iz),is)
17310 : ifright$(a$,1)=" "thena$=left$(a$,len(a$)-1):goto17310
17320 : l=0
17330 : ifleft$(a$,1)=" "thenl=l+1:a$=right$(a$,len(a$)-1):goto17330
17340 : iflen(a$)=0then17440
17350 : qz$=" "
17360 :'qa=1
17370 : gosub24220
17380 : l$=qz$
17390 : a$=a$(jz)+b$+a$+t$
17400 : a$(jz)=a$
17410 : ifle>isthena$(iz)=l$+right$(a$(iz),le-is)
17420 : ifle<=isthena$(iz)=""
17430 : j1=19
```

18.2 Basisversie 'TEXT 1.0'

```
17440 : 19=1
17450 next19
17460 ifj1=0thenreturn
17470 goto17950
17500 :
17510 rem ------------------------------------------------------------------------------
17520 rem ---                        scheiden naar beneden                           ---
17530 rem ------------------------------------------------------------------------------
17540 :
17550 ifis>le thenreturn
17560 j1=0
17570 ifi1=imthenim=im+1:il=il+1:jz=il:iz%(im)=jz:is%(il)=im:a$(jz)=""
17580 for17=i1+1toim
17590 : jz=iz%(17)
17600 : ifjz=0then17930
17610 : lj=len(a$(jz))
17620 : m$=mid$(a$(iz),is,1)
17630 : t$="-"
17640 : ifm$=" "orm$="-"oris=1thent$=""
17650 : ifm$="-"thenis=is+1
17660 : ifis=1then17700
17670 : m$=mid$(a$(iz),is-1,1)
17680 : ifm$=" "orm$="-"thent$=""
17690 : ifm$=" "thenis=is-1
17700 : b$=" "
17710 : r$=right$(a$(iz),1)
17720 : ifle-is<0then17920
17730 : l=0
17740 : jc=0
17750 : iflj>1thenjc=asc(mid$(a$(jz),l+1,1)):ifjc=32thenl=l+1:goto17740
17760 : ifr$="-"and((jcand127)>=65and(jcand127)<=93)thenb$="":goto17780
17770 : ifr$="-"thenb$="":a$(iz)=left$(a$(iz),le-1):le=le-1
17780 : a$=right$(a$(iz),le-is+1)
17790 : ifleft$(a$,1)=" "thena$=right$(a$,len(a$)-1):goto17790
17800 : iflen(a$)=0then17920
17810 : l=0
17820 : iflen(a$(jz))=0then17840
17830 : ifmid$(a$(jz),l+1,1)=" "thenl=l+1:goto17830
17840 : ifl=0thena$=a$+b$+a$(jz):goto17860
17850 : a$=left$(a$(jz),l)+a$+b$+right$(a$(jz),len(a$(jz))-1)
17860 : a$(jz)=a$
17870 : ifis>1thena$(iz)=left$(a$(iz),is-1)+t$
17880 : ifis<=1thena$(iz)=""
17890 : le=len(a$(iz))
17900 : ifright$(a$(iz),1)=" "thena$(iz)=left$(a$(iz),le-1)
17910 : j1=17
17920 : l7=im
17930 next17
17940 ifj1=0thenreturn
17950 kprin=0
17960 qz=lv:qs=nu+1:gosub24060
17970 printls$;
17980 qz=lv:qs=nu+1:gosub24060
17990 printleft$(a$(iz),ls);
18000 forij=3toiy
18010 : ifzz(ij)<>j1then18080
18020 : qz=ij:qs=nu+1:gosub24060
18030 : printls$;
18040 : qz=ij:qs=nu+1:gosub24060
18050 : printleft$(a$(jz),ls);
18060 : kprin=1
18070 : ij=iy
18080 nextij
18090 ific<>ic(13)orkprin<>0oriy>=nz-1thenreturn
18100 iy=iy+1
18110 qz=iy:qs=1:gosub24060
```

18.2 Basisversie 'TEXT 1.0'

```
18120 printright$(" "+str$(j1)+": ",5);left$(a$(jz),ls);
18130 zz(iy)=j1
18500 :
18510 rem ------------------------------------------------------------------------
18520 rem ---                          tekst printen                           ---
18530 rem ------------------------------------------------------------------------
18540 :
18550 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
18560 print"    Regels afdrukken";
18570 gosub23050
18580 print
18590 ifi1<0ori1>imthenreturn
18600 gosub23840
18610 qz=1:qs=1:gosub24060
18620 printlz$;
18630 qz=1:qs=1:gosub24060
18640 print"Afdruk met Regelnummers (m/z) ?";
18650 gosub24120
18660 kn$=qz$
18670 printkn$
18680 ifkn$="m"thenkn$="M"
18690 ifkn$="z"orasc(kn$)=13thenkn$="Z"
18700 ifkn$<>"M"andkn$<>"Z"then18610
18710 nl=6
18720 qz=1:qs=1:gosub24060
18730 printlz$;
18740 qz=1:qs=1:gosub24060
18750 input"spatie 's linkerkantlijn";nl$
18760 ifnl$<>""thennl=val(nl$)
18770 qz$=" "
18780 qa=nl
18790 gosub24220
18800 nl$=qz$
18810 qz=1:qs=1:gosub24060
18820 printlz$;
18830 qz=1:qs=1:gosub24060
18840 print"<Return>..Printen     <Q>..Stoppen";
18850 gosub24120
18860 w$=qz$
18870 printw$;
18880 ifw$="Q"orw$="q"then return
18890 qz=1:qs=1:gosub24060
18900 printlz$;
18910 qz=1:qs=1:gosub24060
18920 mz=0
18930 mmax=np-lz
18940 kl=1
18950 for18=i1toi2stepi3
18960 : iz=iz%(18)
18970 : ifiz=0then19730
18980 : ifa$(iz)<>"!"andmz<mmaxthen19130
18990 : forj=1tonp-mz
19000 :   print#4
19010 : nextj
19020 : ifke<>0then19110
19030 : qz=1:qs=1:gosub24060
19040 : print"<Return>..nieuw Blad     <Q>..Stoppen";
19050 : gosub24120
19060 : w$=qz$
19070 : ifw$="Q"orw$="q"thenreturn
19080 : qz=1:qs=1:gosub24060
19090 : printlz$;
19100 : qz=1:qs=1:gosub24060
19110 : mz=0
19120 : ifa$(iz)="!"then19730
19130 : mz=mz+1
```

18.2 Basisversie 'TEXT 1.0'

```
19140 : ifkl=1thenprint#4,nl$;
19150 : kl=1
19160 : ifkn$="M"thenprint#4,right$(" "+str$(18)+": ",5);
19170 : ipo=-1
19180 : na=ipo+2
19190 : qa=na
19200 : qs$=a$(iz)
19210 : qz$="'"
19220 : gosub25220
19230 : ipo=qa
19240 : ifipo=0then19690
19250 : print#4,mid$(a$(iz),na,ipo-na);
19260 : teken$=mid$(a$(iz),ipo+1,1)
19270 : ifteken$="+"thenkl=0:goto19720
19280 : ifteken$="E"thenprint#4,chr$(27);:goto19180
19290 : ifteken$="U"thenprint#4,u1$;:goto19180
19300 : ifteken$="u"thenprint#4,u2$;:goto19180
19310 : ifteken$="F"thenprint#4,f1$;:goto19180
19320 : ifteken$="f"thenprint#4,f2$;:goto19180
19330 : ifteken$="B"thenprint#4,b1$;:goto19180
19340 : ifteken$="b"thenprint#4,b2$;:goto19180
19350 : ifteken$="H"thenprint#4,h1$;:goto19180
19360 : ifteken$="h"thenprint#4,h2$;:goto19180
19370 : ifteken$="T"thenprint#4,t1$;:goto19180
19380 : ifteken$="t"thenprint#4,t2$;:goto19180
19390 : ifteken$="N"thenprint#4,n1$;:goto19180
19400 : ifteken$="S"thenprint#4,s1$;:goto19180
19410 : ifteken$="s"thenprint#4,s2$;:goto19180
19420 : ifteken$="L"thenprint#4,l1$;:goto19180
19430 : ifteken$="l"thenprint#4,l2$;:goto19180
19440 : ifteken$="P"thenprint#4,p1$;:goto19180
19450 : ifteken$="p"thenprint#4,p2$;:goto19180
19460 : ifteken$<>"W"then19550
19470 : qz=1:qs=1:gosub24060
19480 : printlz$;
19490 : qz=1:qs=1:gosub24060
19500 : print"Pause. RETURN-Toets drukken.";
19510 : inputw$
19520 : qz=1:qs=1:gosub24060
19530 : printlz$;
19540 : goto19180
19550 : getal$=""
19560 : n=0
19570 : le=len(a$(iz))
19580 : ifipo+n+1>le then19640
19590 : ic=asc(mid$(a$(iz),ipo+n+1,1))
19600 : ific<48oric>57then19640
19610 : getal$=getal$+chr$(ic)
19620 : n=n+1
19630 : ifn<3then19580
19640 : getal=val(getal$)
19650 : ifgetal>255thenn=0
19660 : ifn>0thenprint#4,chr$(getal);
19670 : na=ipo+1+n
19680 : ifna<le then19190
19690 : le=len(a$(iz))-na+1
19700 : ifle<=0thenprint#4:goto19720
19710 : print#4,right$(a$(iz),le)
19720 : ii=18
19730 next18
19740 return
20000 :
20010 rem ------------------------------------------------------------------------
20020 rem ---                          regels sorteren                          ---
20030 rem ------------------------------------------------------------------------
20040 :
```

18.2 Basisversie 'TEXT 1.0'

```
20050 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
20060 ifil=1thenreturn
20070 print"    Regels sorteren";
20080 gosub23050
20090 ifi1=-1thenreturn
20100 ifi1>0andkb=0then20170
20110 qz=1:qs=1:gosub24060
20120 printlz$
20130 qz=1:qs=1:gosub24060
20140 print">>>Exakte regelaanduiding vereist.<<<";
20150 foriw=1tonw:nextiw
20160 return
20170 ifi2=0thenreturn
20180 ifi2>imthen20110
20190 ifi3=0theni3=1
20200 ifi1>i2then20110
20210 ifi1=i2thenreturn
20220 ifiz%(i1)=0theni1=i1+1:goto20210
20230 ifiz%(i2)=0theni2=i2-1:goto20210
20240 i5=abs(i2-i1)
20250 delta=abs(i5/i3)
20260 ifdelta<>int(delta+.0001)then20110
20270 j1=0
20280 j2=0
20290 qz=1:qs=1:gosub24060
20300 printlz$;
20310 qz=1:qs=1:gosub24060
20320 input"eerste positie ";j1
20330 qz=1:qs=1:gosub24060
20340 printlz$;
20350 qz=1:qs=1:gosub24060
20360 input"laatste Positie ";j2
20370 ifj1>=0andj2>=0andj1<=255andj2<=255then20440
20380 qz=1:qs=1:gosub24060
20390 printlz$;
20400 qz=1:qs=1:gosub24060
20410 printf$
20420 foriw=1tonw:nextiw
20430 goto20290
20440 ifj1=0thenj1=1
20450 ifj2=0thenj2=j1+7:ifj2>255thenj2=255
20460 ifj2>=j1then20530
20470 qz=1:qs=1:gosub24060
20480 printlz$;
20490 qz=1:qs=1:gosub24060
20500 printf$;
20510 foriw=1tonw:nextiw
20520 goto20290
20530 nc=j2-j1+1
20540 a$=" "
20550 forl6=i1toi2stepi3
20560 : iz=iz%(l6)
20580 : nj=nc
20590 : le=len(a$(iz))
20600 : ifle<j2thennj=le-j1+1
20610 : ifle<j1then20630
20620 : s$(iz)=mid$(a$(iz),j1,nj)
20630 nextl6
20640 forl4=i1+i3toi2stepi3
20650 : iz=iz%(l4)
20660 : ifiz=0then20840
20670 : k=l4-i3
20680 : ik=iz%(k)
20690 : ifik=0thenk=k-i3:goto20680
20700 : ifs$(iz)>=s$(ik)then20790
20710 : j=k+i3
```

18.2 Basisversie 'TEXT 1.0'

```
20720 : ifiz%(j)=0thenj=j+i3:goto20720
20730 : iz%(j)=ik
20740 : is%(ik)=j
20750 : k=k-i3
20760 : ik=iz%(k)
20770 : ifik=0then20750
20780 : ifk>=i1then20700
20790 : j=k+i3
20800 : ifiz%(j)=0thenj=j+i3:goto20800
20810 : ij=iz%(j)
20820 : iz%(j)=iz
20830 : is%(iz)=j
20840 next14
20850 forl5=i1toi2stepi3
20860 : iz=iz%(15)
20870 : ifiz=0then20890
20880 : s$(iz)=""
20890 next15
20900 printcl$;
20910 iy=2
20920 return
20930 i1=ii+1
20940 i2=i8
20950 i3=i9
20960 z$=zalt$
20970 ifil=0orz$=""thenreturn
20980 goto21220
21000 :
21010 rem ------------------------------------------------------------------------
21020 rem ---                          tekenreeks zoeken                        ---
21030 rem ------------------------------------------------------------------------
21040 :
21050 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
21060 print"    Regels doorzoeken";
21070 gosub23050
21080 ifi1<0ori1>imthenreturn
21090 gosub23840
21100 i8=i2
21110 i9=i3
21120 qz=1:qs=1:gosub24060
21130 printlz$;
21140 qz=1:qs=1:gosub24060
21150 input"Teken volgorde zoeken";z$
21160 ifz$=""thenz$=zalt$:ifz$=""thenreturn
21170 le=len(z$)
21180 qz$="="
21190 qa=le
21200 gosub24220
21210 us$=qz$
21220 forl3=i1toi2stepi3
21230 : iz=iz%(13)
21240 : ifiz=0then21540
21250 : qa=1
21260 : qs$=a$(iz)
21270 : qz$=z$
21280 : gosub25220
21290 : ipo=qa
21300 : ifipo=0then21540
21310 : la=len(a$(iz))
21320 : qz$=" "
21330 : qa=la
21340 : gosub24220
21350 : l$=qz$
21360 : l$=left$(l$,ipo-1)+us$+mid$(l$,ipo+1+le)
21370 : qa=ipo+1
21380 : qs$a$(iz)
```

18.2 Basisversie 'TEXT 1.0'

```
21390 : qz$=z$
21400 : gosub25220
21410 : ipo=qa
21420 : ifipo>0then21360
21430 : printcl$;
21440 : print:u:print
21450 : printright$(" "+str$(13)+": ",5);left$(a$(iz),ls)
21460 : qz$=" "
21470 : qa=nu
21480 : gosub24220
21490 : printqz$;left$(l$,ls)
21500 : ifla>lsthenprintright$(a$(iz),la-ls):printright$(l$,la-ls)
21510 : print
21520 : ii=13
21530 : 13=i2
21540 next13
21550 zalt$=z$
21560 iy=2
21570 return
22000 :
22010 rem ------------------------------------------------------------------------
22020 rem ---                        tekenreeks vervangen                        ---
22030 rem ------------------------------------------------------------------------
22040 :
22050 ifil=0thenprintkt$;:foriw=1tonw:nextiw:return
22060 print"    Regels modificeren";
22070 gosub23050
22080 ifi1=-1thenreturn
22090 ifi1>imthenreturn
22100 ifi1>0andkb=0then22170
22110 qz=1:qs=1:gosub24060
22120 printlz$;
22130 qz=1:qs=1:gosub24060
22140 print">>>Exakte Regelaanduiding vereist.<<<";
22150 foriw=1tonw:nextiw
22160 return
22170 ifi2=0theni2=i1
22180 ifi3=0theni3=1
22190 ifi2>imtheni2=im
22200 printcl$;
22210 inpyt" oude Tekenreeks";z1$
22220 l1=len(z1$)
22230 ifl1=0thenprint:return
22240 input" nieuwe Tekenreeks";z2$
22250 print
22260 forl2=i1toi2stepi3
22270 : iz=iz%(12)
22280 : ifiz=0then22460
22290 : qa=1
22300 : qs$=a$(iz)
22310 : qz$=z1$
22320 : gosub25220
22330 : ipo=qa
22340 : ifipo=0then22460
22350 : a1$=a$(iz)
22360 : la=len(a1$)
22370 : a2$=""
22380 : a2$=a2$+left$(a1$,ipo-1)+z2$
22390 : la=la-ipo+1-l1
22400 : a1$=right$(a1$,la)
22410 : ifla>0thenqa=1:qs$=a1$:qz$=z1$:gosub25220:ipo=qa:ifipo>0then22380
22420 : a$(iz)=a2$+a1$
22430 : printright$(" "+str$(12)+": ",5);
22440 : printa$(iz)
22450 : ii=12
22460 next12
```

18.2 Basisversie 'TEXT 1.0'

```
22470 return
22500 :
22510 rem ------------------------------------------------------------------------
22520 rem ---                          hulp tekst                              ---
22530 rem ------------------------------------------------------------------------
22540 :
22550 printcl$;
22560 print "KOMMANDO'S    z!=Regels exakt aangeven "
22570 print "====================================="
22580 print"A...Rest Geheugenruimte,Regelaantal"
22590 print"B...Regels wissen                    z!"
22600 print"C...Beeldscherm wissen"
22610 print"D...Regels printen"
22620 print"E...Regels editeren"
22630 print"F...Tekenreeks zoeken,doorgaan:W"
22640 print"H...File laden"
22650 print"I...Inhouds opgave Diskette"
22660 print"L...Regels lezen/editeren"
22670 print"M...Tekenreeks verwisselen          z!"
22680 print"Q...Programma beeindigen"
22690 print"R...Regels in File opslaan"
22700 print"S...Regels sorteren                 z!"
22710 print"V...Regels verplaatsen              z!"
22720 print"Z...Beeldschermafbeelding van de regels"
22730 print
22740 printctext$(8)
22750 printctext$(9)
22760 printctext$(4)
22770 print"nummer met ':'.. regel inlezen"
22780 print
22790 print"Toets indrukken!";
22800 gosub24120
22810 a$=qz$
22820 printcl$;
22830 qz=3:qs=1:gosub24060
22840 print"TOETSEN OM TE EDITEREN"
22850 print"====================="
22860 print"Toetsen         Werking"
22870 printkopf$
22880 forqa=1to18
22890 : printctext$(qa)
22900 nextqa
22910 return
23000 :
23010 rem ------------------------------------------------------------------------
23020 rem ---                      regel bereik inlezen                        ---
23030 rem ------------------------------------------------------------------------
23040 :
23050 poke198,1:poke631,34:inputz$
23060 i1=0
23070 i2=0
23080 i3=0
23090 kb=0
23100 iflen(z$)=0thenreturn
23110 ifleft$(z$,1)=" "thenz$=right$(z$,len(z$)-1):goto23110
23120 zl$=left$(z$,1)
23130 ifzl$="k"orzl$="K"theni1=-1:return
23140 ifzl$="."theni1=ii:goto23380
23150 ifzl$<>":"then23250
23160 fori1=ii+1toim
23170 : ifiz%(i1)>0then23380
23180 nexti1
23190 i1=-1
23200 qz=1:qs=ns-25:gosub24060
23210 printf$;
23220 foriw=1tonw:nextiw
```

18.2 Basisversie 'TEXT 1.0'

```
23230 return
23240 :
23250 ifz1$="!"theni1=im:goto23380
23260 ifz1$="-"then23380
23270 ifz1$="/"then23730
23280 i1=val(z$)
23290 ifi1<>0then23380
23300 qz=1
23310 qs=ns-25
23320 gosub24060
23330 printf$;
23340 i1=-1
23350 foriw=1tonw:nextiw
23360 return
23370 :
23380 qa=1
23390 qs$=z$
23400 qz$="-"
23410 gosub25220
23420 ipo=qa
23430 ifipo=0then23730
23440 z$=right$(z$,len(z$)-ipo)
23450 ifz$=""thenkb=1:return
23460 ifleft$(z$,1)=" "thenz$=right$(z$,len(z$)-1):goto23460
23470 ifz$=""thenkb=1:return
23480 z1$=left$(z$,1)
23490 ifz1$="."theni2=ii:goto23730
23500 ifz1$<>":"then23620
23510 fori2=ii+1toim
23520 : ifiz%(i2)>0then23730
23530 nexti2
23540 i1=-1
23550 qz=1
23560 qs=ns-25
23570 gosub24060
23580 printf$;
23590 foriw=1tonw:nextiw
23600 return
23610 :
23620 ifz1$="!"theni2=im:goto23730
23630 ifz1$="/"thenkb=1:goto23730
23640 i2=val(z$)
23650 ifi2<>0then23730
23660 qz=1:qs=ns-25:gosub24060
23670 qs=ns-25
23680 printf$;
23690 i1=-1
23700 foriw=1tonw:nextiw
23710 return
23720 :
23730 qa=1
23740 qs$=z$
23750 qz$="/"
23760 gosub25220
23770 ipo=qa
23780 ifipo=0thenreturn
23790 z$=right$(z$,len(z$)-ipo)
23800 i3=val(z$)
23810 ifi3=0thenqz=1:qs=ns-25:gosub24060:printf$;:i1=-1:foriw=1tonw:nextiw
23820 return
23830 :
23840 ifi2>imtheni2=im
23850 ifkb=1andi2=0theni2=im
23860 ifi1=0theni1=1:ifi2=0theni2=im
23870 ifi2=0theni2=i1
23880 ifi3=0theni3=1
```

18.2 Basisversie 'TEXT 1.0'

```
23890 return
24000 :
24010 rem ------------------------------------------------------------------------------
24020 rem ---                        algemene hulproutines                             ---
24030 rem ------------------------------------------------------------------------------
24040 :
24050 rem --- cursor plaatsen ---
24060 poke214,qz-1
24070 poke211,qs-1
24080 sys58732
24090 return
24100 :
24110 rem --- get qz$ ---
24120 getqz$
24130 ifqz$=""thenpoke204,0:goto24120
24140 poke205,2
24150 wait207,1,1
24160 poke204,1
24170 return
24200 :
24210 rem --- string met lengte qa met teken qz$ produceren ---
24220 ifqa=0thenqz$=""
24230 ifqa<2thenreturn
24240 qq=asc(qz$)
24250 forqi=2toqa
24260 qz$=qz$+chr$(qq)
24270 nextqi
24280 return
24300 :
24310 rem --- a$ van disk lezen ---
24320 a$=""
24330 get#1,q$
24340 ifq$=chr$(13)thends=st:return
24350 a$=a$+q$
24360 goto24330
24400 :
24410 iflh<=nsthen10740
24420 lh=ns
24430 ifle<=zl+lsthen24480
24440 zl=zl+1
24450 qz=lv:qs=nu+1:gosub24060
24460 printmid$(a$(iz),zl+1,ls);
24470 goto10740
24480 iflv<iythenlh=nu+1:ic=ic(4):goto11570
24490 goto10740
24500 iflh>=nu+1then10780
24510 lh=nu+1
24520 ifzl>0then24550
24530 iflv>3thenlh=ns:ic=333:goto11570
24540 goto10780
24550 zl=zl-1
24560 qz=lv:qs=lh:gosub24060
24570 printmid$(a$(iz),zl+1,ls);
24580 goto10780
24590 iflv<=nz-1then11700
24600 print
24610 qz=1:qs=1:gosub24060
24620 printlz$
24630 printkopf$;
24640 lv=nz-1
24650 forjj=3tonz-1
24660 : zz(jj)=zz(jj+1)
24670 nextjj
24680 goto11700
24690 ic=ic(3)
24700 lq=zz(lv-1)
```

18.2 Basisversie 'TEXT 1.0'

```
24710 le=len(a$(iz%(lq)))
24720 ifle<=lsthen11750
24730 zl=le-ls
24740 qz=lv-1:qs=nu+1:gosub24060
24750 printright$(a$(iz%(lq)),ls);
24760 goto11750
24770 gosub15930
24780 ii=kk
24790 il=il+1
24800 iz=il
24810 iz%(ii)=iz
24820 is%(il)=ii
24830 a$(iz)=ls$
24840 ifiy<nz-1theniy=iy+1:zz(iy)=zz(iy-1)+1
24850 return
24860 foriv=lvtoiy
24870 : qz=iv:qs=nu+1:gosub24060
24880 : printls$;
24890 : qz=iv:qs=1:gosub24060
24900 : lq=zz(iv)
24910 : printright$(" "+str$(lq)+": ",5);
24920 : printleft$(a$(iz%(lq)),ls)
24930 nextiv
24940 lq=im+1
24950 i1=kk
24960 i2=im
24970 i3=1
24980 ii=ii-1
24990 goto13160
25000 i3=1
25010 i4=ii
25020 gosub15810
25030 i1=ki
25040 i2=ki
25050 gosub9550
25060 lq=kk
25070 i1=kk
25080 i2=im
25090 return
25100 print">>>Regel";kk;"bestaat niet.<<<";
25110 foriw=1tonw:nextiw
25120 return
25130 qz=lv:qs=nu+1:gosub24060
25140 printls$;
25150 qz=lv:qs=nu+1:gosub24060
25160 printleft$(a$(iz),ls);
25170 qz=1:qs=1:gosub24060
25180 printlz$
25190 goto10530
25200 :
25210 rem --- instr-funktie ---
25220 ql=len(qz$)
25230 forqq=qatolen(qs$)-ql+1
25240 : ifmid$(qs$,qq,ql)=qz$thenqa=qq:qq=998
25250 next
25260 ifqq=999thenreturn
25270 qa=0
25280 return
29570 : ifiz=0then20630

ready.
```

7/19

Een database in BASIC

Een van de makkelijkste dingen van een computer is dat deze gegevens vast kan houden. Zodoende kunnen we bijvoorbeeld alle adressen van onze vrienden, kennissen en familie aan de computer toevertrouwen, waarna we alles makkelijk terug kunnen vinden.
Dit doen we met behulp van een database programma. Een database betekent letterlijk een gegevensbak. De gegevens kunnen bestaan uit allerlei dingen zoals adresgegevens of de titels van de platenverzameling. Ook een adresboekje is een database. Een paar jaar geleden bestonden deze gegevensverzamelingen nog voor het grootste gedeelte op papier in zogenaamde kaartenbakken. Hierbij werden de gegevens in een bepaalde volgorde op een kaart geschreven waarbij voor elk hoofdgegeven een nieuwe kaart genomen werd. Bijvoorbeeld bij een verenigingsadministratie werd per lid een kaart gemaakt met al zijn gegevens en die werd dan met de hand op alfabet in de bak gezet. Deze methode was vreselijk bewerkelijk en ook niet zo snel want telkens moesten vele tientallen kaarten doorlopen worden als er iets gezocht werd. Hierdoor begonnen grote bedrijven al in een vroeg stadium de computer te gebruiken om deze gegevens vast te houden en makkelijker toegankelijk te maken.

Het volgende voorbeeld illustreert het verschil tussen het met de hand opzoeken en het met de computer opzoeken.
De overheid wil weten hoeveel ambtenaren 56 jaar of ouder zijn en dit moet in een verslag gezet worden met aantallen per leeftijd. Indien dit niet via de computer kan gebeuren komen deze cijfers nooit beschikbaar. Voor 100.000 personen (kaarten) met een gemiddelde van 2 minuten doorleestijd kost het met de hand ruim 3300 uur of ca. 1 1/2 jaar door 1 persoon. Hierbij van uitgaande dat de gegevens binnen 1 ruimte aanwezig zijn. Met de computer gebeurt dit binnen 1 dag en levert dus een actueel beeld van de situatie.

Hiervoor worden per bedrijf meestal zelf de programma's geschreven maar met het kleiner worden van de computers worden ook de toepassingen groter.
Steeds meer bedrijven kunnen een computer aanschaffen en er ontstaan steeds meer database programma's. Ook voor de microcomputers worden deze programma's geschreven maar meestal gebeurt dit door grote bedrijven waarbij de vele mogelijkheden gebruikt worden die de computer biedt.
Een nadeel van deze programma's is wel dat je altijd zelf nog een paar zaken aan moet geven maar het voordeel is dat hetzelfde programma voor diverse soorten verzamelingen gebruikt kan worden.

Het Programma

Het nu volgende programma is een BASIC programma dat geschreven is voor het bewaren van adresgegevens. Voor de iets geoefende programmeur is het programma makkelijk aan te passen voor andere zaken.
Dit programma zullen we nu stap voor stap op gaan bouwen zodat het voor iedereen duidelijk is.

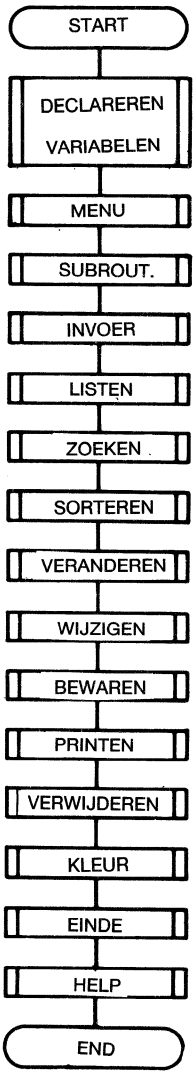
Voor een adressenprogramma hebben we minstens naam, adres, postcode, woonplaats en telefoon nodig. Tevens is het meestal makkelijk om nog minstens één rubriek extra erbij te hebben voor gegevens die men zelf wil bepalen zoals bijvoorbeeld geboortedatum.
Tevens zullen we, om het programmeren makkelijker te maken, afspreken dat er een paar specifieke variabelen gebruikt zullen worden om de lussen in het programma te controleren.
Dit is namelijk beter omdat dan zonder problemen uit een lus gesprongen kan worden. Dit komt omdat de computer op een speciale plaats bijhoudt welke lussen gedaan moeten worden en welke variabele daar bij hoort. Als er nu een nieuwe lus gestart wordt controleert de computer eerst of er nog een onafgemaakte lus is van die variabele. Indien dat zo is wordt deze verwijderd en alle daarna komende lussen ook en wordt de nieuwe lus gestart.
Nu we bepaald hebben welke rubrieken we nodig hebben, moeten we gaan bepalen wat het programma moet gaan doen.

Natuurlijk moeten we de adressen in kunnen voeren, opbergen en ophalen want anders hebben we helemaal niets aan het programma.
Maar buiten deze drie dingen moet het programma ook nog wat meer kunnen doen. We moeten de adressen kunnen onderhouden, dus veranderen, verwijderen en afdrukken op scherm en papier.
Het veranderen is nodig als iemand verhuisd is, zodat we zijn nieuwe adres in kunnen brengen zonder alle andere gegevens opnieuw in te moeten brengen.
Het verwijderen moet gedaan worden als we iemand niet meer in het bestand willen hebben.
Zonder het onderdeel afdrukken kunnen we met het ingevoerde bestand niets doen omdat we dan anders nog alles zelf moeten opschrijven.
Een van de makkelijkste zaken bij een programma met zoveel mogelijkheden is om alles via een menu te laten sturen. Een menu is een stuk tekst wat op het scherm komt in een overzichtelijke vorm met daarin de mogelijkheden van het programma. Hiervandaan kan men kiezen wat men wil doen door het intoetsen van de opdracht.

De Struktuur

Het totale programma gaat nu 12 onderdelen bevatten buiten het gedeelte van subroutines en dergelijke.
Voordat we verder gaan volgt eerst de hoofdstruktuur van het programma. Dit is, omwille van de duidelijkheid in schemavorm.

Het declareren van de variabelen is niet voor alle variabelen nodig maar, als alle variabelen in één keer opgegeven worden en in alfabetische volgorde staan loopt het programma sneller.
Tevens weten we dan ook meteen welke variabelen we in gebruik hebben als we nog wat willen veranderen.
De verschillende onderdelen van het programma zullen ieder op een veelvoud van 100 beginnen zodat we ook hier precies



weten waar we welk onderdeel kunnen vinden. We beginnen met het invoergedeelte op 1000 en eindigen op 12000 bij de helpfunctie.
Voordat we met het programma beginnen eerst nog een paar opmerkingen. Ten eerste dient de printer, als u die heeft, altijd aan te staan daar het programma niet controleert of deze aan staat en het dan fout kan gaan.
Ten tweede is door het gecombineerd gebruik van dezelfde routine voor wegschrijven naar band of floppy geen controle mogelijk op het goed wegschrijven naar schijf. Indien de foutmelding van de floppydrive knippert is er wat aan de hand en is het bestand niet weggeschreven. Probeer dan het bestand eerst op een andere schijf weg te zetten en kijk dan wat er aan de hand is. Ten derde is het menu dusdanig opgezet dat elke eerst karakter van de keuzes uniek is. Om dit voor elkaar te krijgen zijn soms engelse woorden gebruikt.
De declaratie van de variabelen, het menu en de subroutine's bevinden zich vanaf regel 0 t/m 990. Hiervan worden nu nog niet alle regels gebruikt maar de mogelijkheid is aanwezig om het programma later uit te breiden. Tevens werkt het programma sneller als de subroutine's aan het begin van het programma staan.
Dit omdat dan niet het hele programma afgezocht hoeft te worden. Ook weten we altijd welke subroutines we reeds hebben.
Tijdens het bespreken van het programma, zal op bepaalde plaatsen een schematische opstelling gemaakt worden van wat er gebeuren moet in dat programmaonderdeel. Dit is dan wel voor diegene die deze schema's kunnen lezen. De uitleg wat welk symbool betekent wordt te groot om hier te behandelen.

De Variabelen
Zoals gezegd beint het programma met het declareren van de variabelen inclusief de tabellen voor de adressen. Het aantal op te nemen adressen is gesteld op 250, zie variabele A1, maar kan voor de CBM 128 wel verdubbeld worden naar de 500. Het enige wat dan gedaan moet worden is variabele A1=250 te veranderen in A1=500.
Het declareren van de variabelen loopt tot en met regel 50 waarna aan het einde een sprong gemaakt wordt naar het menu.
Regels 60 en 70 worden in beslag genomen door de meest gebruikte subroutine. Deze routine wordt telkens aangeroepen als er een enkel teken gevraagd wordt.

Menu Opbouw
Regels 80 tot en met 200 zorgen ervoor dat het menu op het scherm gezet wordt. Regel 210 vraagt aansluitend om de keuze, maar deze regel zal ook vaak aan het eind van een onderdeel aangeroepen worden. Tevens wordt er een RESTORE op de data regels uitgevoerd.
Vervolgens wordt in regel 220 een leesopdracht uitgevoerd en gecontroleerd of de laatste data gelezen is. Indien dit zo is wordt er een boodschap op het scherm gezet en daarna weer het volledige menu.
Tijdens het ingeven van de opdracht mag elke opdracht op elke gewenste plaats afgebroken worden. Het makkelijkste is om maar 1 karakter in te toetsen daar deze toch allen uniek zijn.
Regel 230 zoekt de goede opdracht op in samenwerking met regel 220 en als de goede gevonden is wordt regel 240 aangeroepen die de juiste module aanroept.
Regels 250 en 260 bevatten de dataregels met de opdrachten.

```
0 rem bestandsprogramma door p.rijnaard
10 print"■■■■■■■■even geduld a.u.b."
20 a1=250:dimn$(a1),a$(a1),p$(a1),w$(a1),t$(a1),k$(a1)
30 a=0:ag=0:be=0:d=0:f=0:f1=0:n=0:ra=0:s=0:t=0:te=5:tm=0:va=0:x=0:y=0
40 a$=chr$(13):c$="":d$="":f$="":g$="":i$="":k$="":n$="":o$="":p$=""
50 s$="":sp$="                                        ":t$="":w$="":z$="":print"■":goto80
60 getc$:ifc$=""then60
70 print" ";c$:return
80 print"kies opdracht:":print
90 print"invoer      (nieuw of van cas/floppy)"
100 print"list        (op het scherm)
110 print"zoek        (in het bestand)"
120 print"sorteer     (op alle rubrieken)"
130 print"verander    (per adres)"
140 print"wijzigen    (per item)"
150 print"bewaar      (op cassette/floppy)"
160 print"printen     (afdruk op papier)"
170 print"remove      (verwijderen uit bestand)"
180 print"kleur       (instelen scherm kleuren)"
190 print"einde       (stoppen van het programma)"
200 print"help        (toont dit menu)"
210 print:f1=0:input"opdracht ";o$:print:restore
220 reads$,t:ifs$="**"thenprint"opdracht onbekend":goto80
230 ifleft$(s$,1)<>left$(o$,1)then220
```

```
240 ontgoto1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000
250 data"invoer",1,"list",2,"zoek",3,"sorteer",4,"verander",5,"wijzigen",6
260 data"bewaar",7,"printen",8,"remove",9,"kleur",10,"einde",11,"help",12,"**",0
270 rem start van de subroutine's
280 print"welke adressen wilt u hebben?":print:va=0:tm=0
290 print"alles(a),een groep(g),of 'een'(e) adres":print
300 gosub60:ifc$="a"thenva=1:tm=n
310 ifc$="e"theninput"adresnr ";va:tm=va:ifva>norva<1thengosub360:goto310
320 ifc$="g"theninput"start.,eind.";va,tm:ifva>tmorva<1ortm>nthengosub350:goto32
0
330 ifva=0thenprint"uw keuze kan niet;kies nogmaals!":goto270
340 return
350 print:print"laatste adres < eerste adresnr."
360 print:print"het bestand bevat maar ";n; "adressen":return
370 print:printtab(10)"< ";y;">":printtab(10)"======="
380 printn$(y);tab(28);t$(y)
390 printa$(y)
400 printp$(y);"  ";w$(y)
410 printk$(y):return
420 rem verwissel routine
430 n$=n$(x)
440 a$=a$(x)
450 p$=p$(x)
460 w$=w$(x)
470 t$=t$(x)
480 k$=k$(x)
490 n$(x)=n$(y)
500 a$(x)=a$(y)
510 p$(x)=p$(y)
520 w$(x)=w$(y)
530 t$(x)=t$(y)
540 k$(x)=k$(y)
550 n$(y)=n$
560 a$(y)=a$
570 p$(y)=p$
580 w$(y)=w$
590 t$(y)=t$
600 k$(y)=k$
610 return

ready.
```

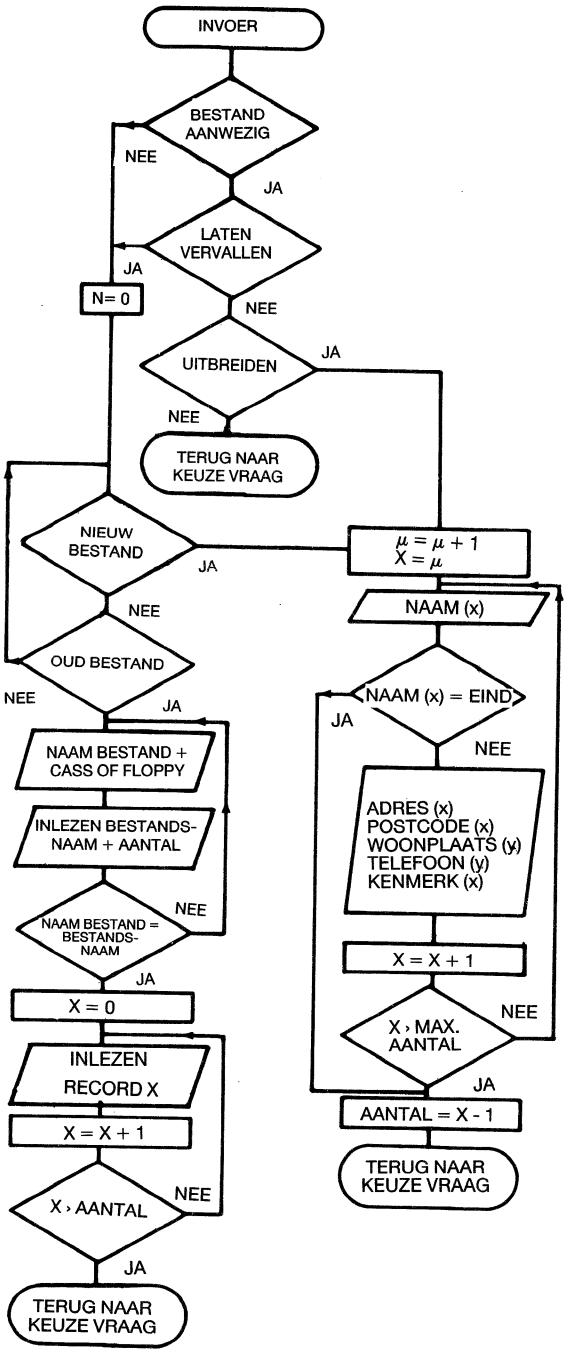
Subroutine's Algemeen

Vanaf regel 270 tot en met 640 zijn een paar subroutine's. Regels 280 tot en met 340 bepalen de selektie van de adressen die afgedrukt moeten worden. Regels 350 en 360 zijn een gecombineerde subroutine. De ene keer wordt alleen regel 360 aangeroepen en de andere keer worden de regels 350 en 360 aangeroepen.
Regels 370 tot en met 410 zorgen voor het op het scherm zetten van een adres. Vanaf regel 430 tot en met regel 610 bevindt zich de verwisselroutine.
Deze routine wordt aangeroepen vanuit de sorteermodule telkens als er twee adressen omgewisseld moeten worden. Om het overzichtelijk te houden is er per regel maar 1 opdracht gezet. Het is natuurlijk ook mogelijk deze routine in twee of drie regels te maken door alle opdrachten achter elkaar te zetten en deze te scheiden door de ':'. Verder is er hier nog ruimte voor verdere subroutine's maar op dit moment is er niets meer nodig.

Invoer

Vanaf regel 1000 krijgen we nu een van de belangrijkste modules. Deze module is tevens de grootste module en controleert zowel de invoer vanaf het toetsenbord als ook de invoer van cassette of floppy.

Eerst wordt er gecontroleerd of er een bestand aanwezig is. Dit gebeurt in regel 1000 op het eind van die regel. Als er geen bestand is wordt er doorgegaan met regel 1080 en anders verschijnt er een melding dat er een bestand is en wordt er gevraagd of dat moet vervallen (regels 1010 en 1020).
Als het bestand moet vervallen, antwoord is C$=J, dan wordt N op nul gezet en wordt er gesprongen naar regel 1080 (zie regel 1030).



Hierna wordt in regel 1040 gekeken of er anders een 'N' ingetoetst is. Indien dit niet het geval is, wordt er teruggesprongen naar het begin van de module. Hierna wordt er gevraagd of het bestand dat aanwezig is soms uitgebreid moet worden, regel 1050. Als het antwoord 'N' is kunnen we weer kiezen uit de keuzes van het hoofdmenu.
Regel 1060 springt naar 210.
Regel 1070 zorgt er alleen voor dat N opgehoogd wordt en springt dan naar regel 1110 omdat we op de volgende regels eerst moeten afvragen of we een nieuw bestand aanmaken of een oud bestand van cassette of floppy halen (regel 1080).
Als het antwoord '0' is vervolgt het programma bij regel 1250. Regel 1100 kontroleert daarna, of er soms een 'N' als antwoord gegeven is en als dit niet zo is wordt de vraag opnieuw gesteld.
Regels 1110 tot en met 1150 zorgen er voor dat een passende boodschap op het scherm verschijnt als er een nieuw bestand ingebracht wordt of als er een bestand uitgebreid wordt.
Regel 1160 vraagt de naam van een persoon en controleert of er 'eind' opgegeven is. Indien dit zo is wordt de invoer beëindigd en vervolgt het programma op 1220. Anders worden de overige gegevens gevraagd en wordt daarna met behulp van de lus weer terug naar de invoer van een naam gesprongen (Regels 1170 tot en met 1210).
Regel 1220 verlaagt alleen en kent de waarde naderhand toe aan N. Vervolgens geeft regel 1230 aan hoeveel adressen er nu in het bestand zitten en gaat dan terug naar de keuzevraag op regel 210.

Nu volgt het gedeelte om een bestand van schijf of cassette te halen. (Regels 1250 tot en met 1380).

```
1000 print:print"■*** ■invoer■ ***":ifn=0then1080
1010 print"het geheugen bevat reeds een bestand"
1020 print"moet dat vervallen? (j/n)";:gosub60
1030 ifc$="j"thenn=0:goto1080
1040 ifc$<>"n"then1000
1050 print"moet het bestand uitgebreid worden(j/n)";
1060 gosub60:ifc$="n"then210
1070 n=n+1:goto1110
1080 n=1:print"nieuw bestand maken of een":print"oud bestand ophalen? (n/o)"
1090 gosub60:ifc$="o"then1250
1100 ifc$<>"n"then1080
1110 print"■*** invoeren ***":print:be=1
1120 print"beeindig de invoer door eind in te typen":print
1130 print"indien niets bekend dan '-' typen":print
1140 forx=ntoa1:print:print"** ";x;" **";tab(25);"** ";a1-x;" **"
1150 print"==========";tab(25);"==========="
1160 input"naam        :";n$(x):ifn$(x)="eind"then1220
1170 input"adres       :";a$(x)
1180 input"postcode    :";p$(x)
1190 input"woonplaats  :";w$(x)
1200 input"telefoon    :";t$(x)
1210 input"kenmerk     :";k$(x):nextx
```

```
1220 n=x-1
1230 print:print"in het bestand zitten nu ";n;" adressen":goto210
1240 rem start ophalen
1250 print"wat is de naam van het op te halen"
1260 input"bestand";f$:f=3:d=8:s=5:be=0:print"van schijf of cassette (s/c)";
1270 gosub60:ifc$="c"thend=1:s=0:d$=f$:goto1310
1280 ifc$="s"then1300
1290 print"fout antwoord typ opnieuw ";:goto1270
1300 d$="":d$="0:"+f$+",seq,read"
1310 openf,d,s,d$:input#f,n$:input#f,g$:n=val(n$)
1320 iff$<>g$thenclosef:goto1250
1330 print:print"bestand **  ";g$;" ** gevonden":print
1340 print"het aantal adressen is :";n:print:print
1350 forx=1ton
1360 input#f,n$(x):input#f,a$(x):input#f,p$(x)
1370 input#f,w$(x):input#f,t$(x):input#f,k$(x)
1380 print"■";tab(15);"> ";x:nextx:closef:goto210

ready.
```

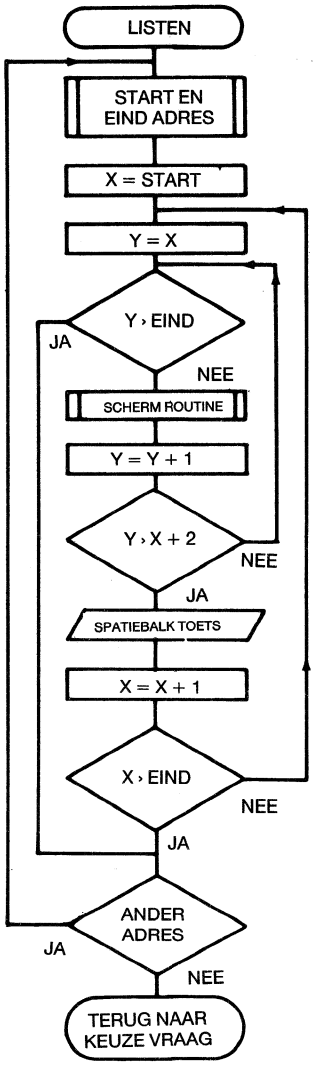
Regel 1250 begint met het vragen wat de bestandsnaam is, waarna in regel 1260 een stel variabelen een waarde krijgen toegekend. Deze waarden zijn voor gebruik van schijven, en de vraag verschijnt of er van schijf of cassette gelezen moet worden.
Regel 1270 haalt het antwoord op en controleert of dit gelijk is aan 'C'. Als dit zo is krijgen de variabelen de waarden voor het gebruik van cassette en wordt er gesprongen naar regel 1310.
Als de controle in regel 1280 niet aangeeft dat er een S ingetypt is, vervolgt regel 1290 met een foutmelding en wordt een nieuw antwoord gevraagd via regel 1270.
Als bij regel 1280 de controle wel S aangeeft wordt er gesprongen naar regel 1300 waar de laatste variabele zijn waarde krijgt toegekend.
Regel 1310 zorgt voor het openen van het bestand en haalt de eerste twee gegevens op.
In regel 1320 wordt gecontroleerd of de naam die in het eerste gegeven staat, gelijk is aan de opgegeven naam.
Als dit niet zo is, wordt het bestand gesloten en wordt teruggegaan naar regel 1250.
Regels 1330 tot en met 1340 geven op het scherm aan dat het bestand gevonden is en hoeveel adressen er zijn.
Regels 1350 tot en met 1380 lezen met behulp van een lus het bestand in en geven tevens op het scherm aan welk record gelezen is.

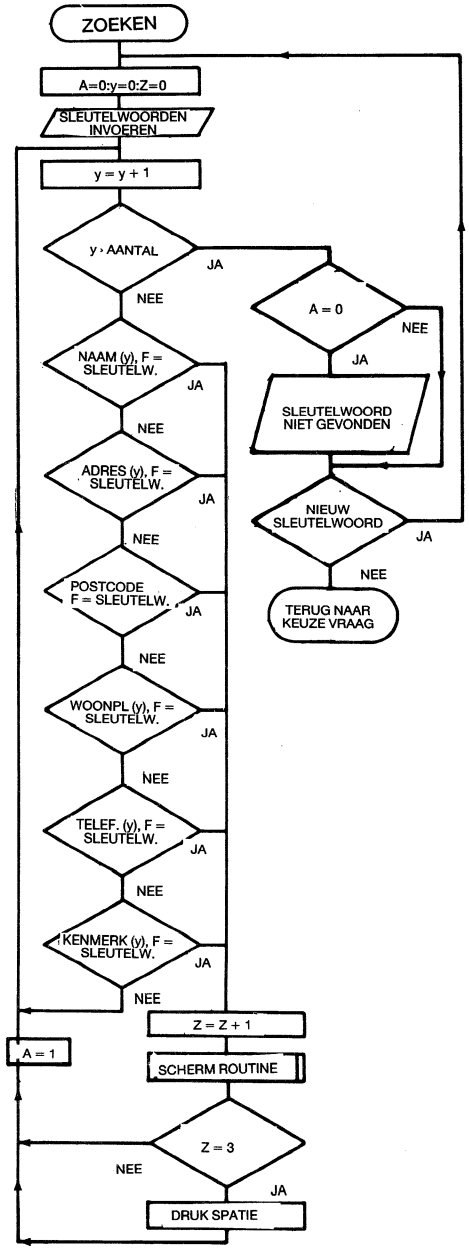
Als het gehele bestand ingelezen is, wordt het bestand gesloten en wordt er gesprongen naar de keuzeregel (regel 210). Dit is tevens het einde van de invoer module.
De volgende module die we gaan behandelen is de module die de adressen op het scherm zet. Deze module is erg klein omdat hier volop gebruik gemaakt wordt van de reeds eerder besproken subroutine's.

Adressen op het scherm
Als eerste wordt op regel 2000 het scherm schoongemaakt en wordt bovenin het woord 'inhoud' gezet.
Vervolgens wordt er naar de routine op regel 280 gesprongen. Deze zorgt voor de selectie van welk(e) adres(sen) er getoond moet(en) worden.
In regel 2010 begint een dubbele lus met de x en y waarden. De y-waarde in de tweede lus wordt gebruikt om maximaal 3 adressen op het scherm te zetten.
Tevens wordt nagegaan of y niet boven de waarde van TM uitkomt. Regel 2020 maakt een sprong naar de subroutine op regel 370 en als de y-waarde zijn eind bereikt komt er op het scherm de melding om op de spatiebalk te drukken.
Dit wordt gedaan omdat anders alle adressen voorbij flitsen zonder dat er wat gezien kan worden.
Regel 2030 sluit de x-lus af. In regel 2040 wordt dan nog gevraagd of er nog een ander adres gezien moet worden.
Als het antwoord in regel 2050 gekontroleerd is en 'J' was, dan gaan we terug naar het begin van de module.
Anders gaan we in regel 2060 naar de keuze regel op 210.

```
NAAM                    ADRES                   POSTCODE  WOONPLAATS
    TELEFOON                KENMERK
GAK HOOFDKANTOOR        POSTBUS 8300            1005 CA   AMSTERDAM
    020-879111              -
S.C.N. HOT-NEWS         POSTBUS 6999            1009 AR   AMSTERDAM
    020-934699              COMPUTER CLUB
WEKA UITGEVERIJ         D. CURTIUSSTR. 7        1051 JL   AMSTERDAM
    020-867131              -
```

```
NAAM                    TELEFOON
GAK HOOFDKANTOOR        020-879111
S.C.N. HOT-NEWS         020-934699
WEKA UITGEVERIJ         020-867131
```


LISTEN
START EN EIND ADRES
X = START
Y = X
Y › EIND
JA
NEE
SCHERM ROUTINE
Y = Y + 1
Y › X + 2
NEE
JA
SPATIEBALK TOETS
X = X + 1
X › EIND
NEE
JA
ANDER ADRES
JA
NEE
TERUG NAAR KEUZE VRAAG


ZOEKEN
A=0:y=0:Z=0
SLEUTELWOORDEN INVOEREN
y = y + 1
y › AANTAL
JA
NEE
A = 0
NEE
JA
SLEUTELWOORD NIET GEVONDEN
NIEUW SLEUTELWOORD
JA
NEE
TERUG NAAR KEUZE VRAAG
NAAM (y), F = SLEUTELW.
JA
NEE
ADRES (y), F = SLEUTELW.
JA
NEE
POSTCODE F = SLEUTELW.
JA
NEE
WOONPL (y), F = SLEUTELW.
JA
NEE
TELEF. (y), F = SLEUTELW.
JA
NEE
KENMERK (y), F = SLEUTELW.
JA
NEE
Z = Z + 1
A = 1
SCHERM ROUTINE
Z = 3
NEE
JA
DRUK SPATIE

```
2000 print"■*** ■inhoud■ ***":print:gosub280
2010 forx=vatotmstep3:print"■":fory=xtox+2:ify>tmthen2030
2020 gosub370:nexty:print:print"■druk op de spatiebalk■":gosub60
2030 nextx
2040 print:print"■ander adres?■ (j/n)";:gosub60
2050 ifc$="j"then2000
2060 goto210

ready.
```

```
3000 print"■*** ■zoek■ ***":print:a=0:z=0
3010 input"sleutelwoord";z$:f=len(z$)
3020 fory=1ton
3030 ifleft$(n$(y),f)=z$then3150
3040 ifleft$(a$(y),f)=z$then3150
3050 ifleft$(p$(y),f)=z$then3150
3060 ifleft$(w$(y),f)=z$then3150
3070 ifleft$(t$(y),f)=z$then3150
3080 ifleft$(k$(y),f)=z$then3150
3090 nexty
3100 ifa=0thenprint:print"sleutelwoord niet gevonden"
3110 print:print"nieuw sleutelwoord? (j/n)";:gosub60
3120 ifc$="j"then3000
3130 ifc$="n"then210
3140 print"fout antwoord":goto3110
3150 z=z+1:gosub370:ifz=3thenz=0:print:print"■druk op de spatiebalk":gosub60:pri
nt"■"
3160 a=1:goto3090

ready.
```

De derde module van het programma begint op regel 3000 en eindigt op regel 3160.

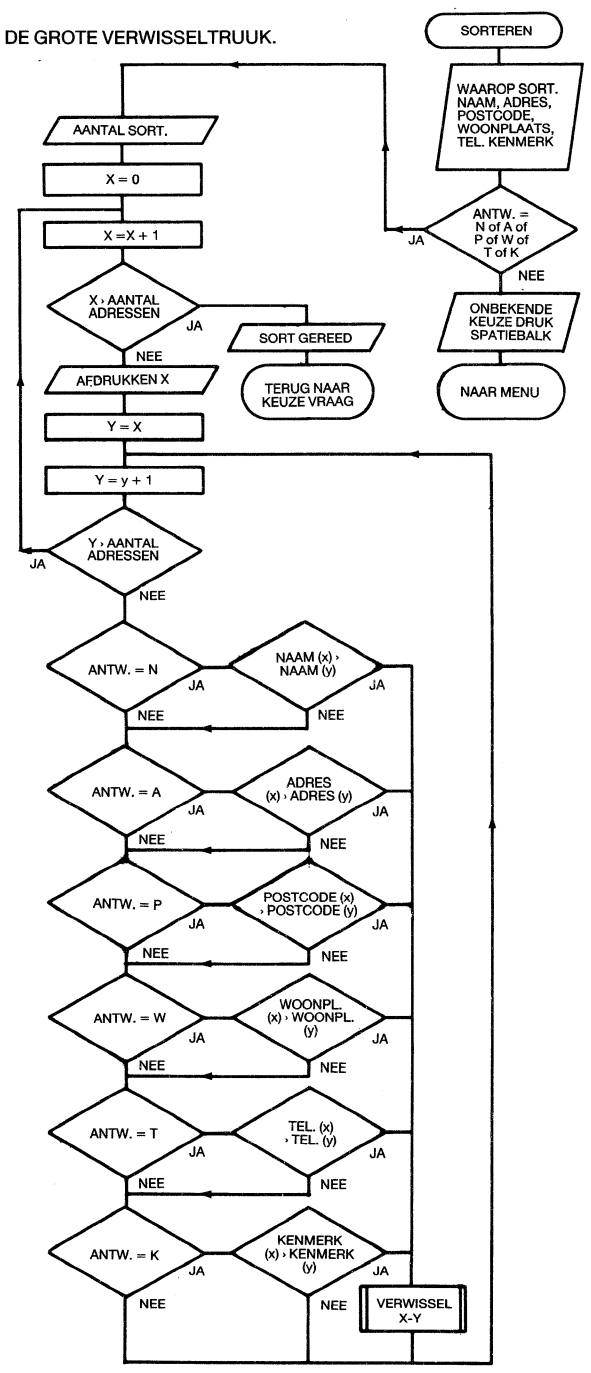
Zoals bij elke module, begint ook deze module op regel 3000 met het schoonmaken van het scherm en het afdrukken boven aan het scherm van het woord 'zoek'. Tevens worden de variabelen A en Z op nul gezet.
Regel 3010 vraagt naar het sleutelwoord. dit is het woord wat gezocht wordt in het bestand.
Het is wel zo dat het sleutelwoord niet ge-

zocht kan worden in het midden van een veld maar alleen vanaf het begin.
Er kan wel op een deel van het adresveld gezocht worden. In variabele F moet de lengte van het sleutelwoord (F$) gezet worden, zodat in de vergelijkingen de lengte van dat sleutelwoord vergeleken kan worden met een zelfde lengte van de andere velden.
In regel 3020 wordt een lus gestart zodat het hele bestand doorgelopen kan worden.
Vanaf regel 3030 tot en met 3080 zijn de vergelijkingen opgenomen om te kijken of het sleutelwoord in dat betreffende adres zit. Indien dit zo is wordt er gesprongen naar regel 3150.
Regel 3090 geeft het eind van de lus.
Als de lus afgelopen is en de variabele A nog nul is wordt de boodschap 'sleutelwoord niet gevonden' afgedrukt. (regel 3100). Op 3110 wordt er gevraagd of er een nieuwe sleutelwoord ingevuld moet worden en als het antwoord J is, regel 3120, wordt er na het begin van deze module gesprongen.
Indien het antwoord 'N' is, regel 3130, wordt er naar de keuze vraag van het menu gesprongen.
Als er in regel 3140 aangekomen wordt is er geen 'J' of 'N' ingevoerd, zodat een fout boodschap verschijnt en de vraag opnieuw gesteld wordt.
Regel 3150 verhoogt eerst teller Z, gaat dan naar de schermafdrukroutine en controleert daarna de waarde in Z.
Als Z de waarde 3 heeft betekent dit dat er reeds 3 adressen op het scherm staan zodat de waarde van Z nu weer nul moet worden en er op de spatiebalk gedrukt moet worden om verder te kunnen gaan.
Regel 3160 geeft aan variabele A de waarde 1 en gaat daarna naar regel 3090. Dit is ook meteen de laatste regel van deze module.

Sorteren
De volgende module is de sorteermodule. Deze module is volledig in BASIC opgebouwd en zodoende niet een van de snelste maar wel de duidelijkste.
Deze module loopt vanaf regel 4000 tot en met 4190.
Als eerste wordt natuurlijk het scherm schoongemaakt en er wordt bovenin het woord 'sorteren' afgedrukt.
De variabele BE die in het begin van regel 4000 de waarde 1 krijgt wordt gebruikt als het programma beëindigd gaat worden.
In de regels 4010 tot en met 4030 wordt gevraagd waarop gesorteerd moet worden en er wordt aangegeven welke letters daarbij horen. Als de keuze gemaakt is wordt gecontroleerd of de goede letter ingedrukt is, zo ja wordt er verder gegaan in regel 4060. Deze controle vindt plaats in regel 4040. Als niet de goede letter ingetoest is komt er een passende boodschap en wordt, nadat de spatiebalk ingedrukt is, naar het keuzemenu gesprongen (regel 4050).
Regels 4060 en 4070 geven aan hoeveel adressen er gesorteerd moeten worden.
Regel 4080 start de eerste lus met de variabele X en omvat het gehele bestand.
Regel 4090 brengt eerst de cursor een regel omhoog en print dan de waarde van X.
In regel 4100 wordt de tweede lus gestart met de variabele Y.
Vanaf regel 4110 tot en met 4160 wordt de vergelijking gedaan of het adres in de tabel positie X groter is dan het adres in de tabelpositie van Y.
Als dit zo is wordt er naar de subroutine op regel 420 gesprongen. Dit is de verwissel routine die zorgt dat het adres op X verwisseld wordt met het adres op Y.
Afhankelijk van de ingetoetste letter wordt er vergeleken met het juiste veld.
Regel 4170 sluit alle twee de lussen af

waarna, als de laatste X geweest is, in regel 4180 de boodschap komt dat het sorteren gereed is en er een toets ingedrukt moet worden.
Regel 4190 maakt weer de sprong naar de keuzeregel.

Veranderen
De volgende module is de eerste van twee modules om wijzigingen in het adressenbestand aan te brengen.
Zoals bij alle modules wordt ook in deze module op de eerste regel, 5000, het scherm schoongemaakt en de boodschap 'veranderen' bovenaan het scherm gezet.
In regel 5010 wordt eerst gecontroleerd of er een bestand aanwezig is.
Dit gebeurd door te testen of de variabele N de waarde nul heeft. Dit betekent dan dat er nog geen bestand is omdat N het aantal adressen aangeeft.
Als er geen bestand aanwezig is wordt er een boodschap op het scherm gezet en naar de keuzeregel gesprongen.
Deze module kan alle gegevens van een adres wijzigen. De andere module wijzigt hetzelfde gegeven in meerdere adressen.
In regel 5020 wordt, net als bij de sorteermodule, aan de variabele BE de waarde 1 toegekend. Tevens wordt, door aan C$ de waarde 'E' toe te kennen, de subroutine aangeroepen om een adresnummer op te vragen.
In de regels 5050 tot en met 5100 wordt extra informatie op het scherm gezet.
Vanaf regel 5050 tot en met 5100 wordt het adres op het scherm gezet met de omschrijving van dat veld.
Door nu in regel 5110 de cursor 6 regels omhoog te brengen en dan op positie 11 van die regel te zetten kan het adres gewijzigd worden.
Als er niets gewijzigd hoeft te worden kan rustig op return gedrukt worden zodat de

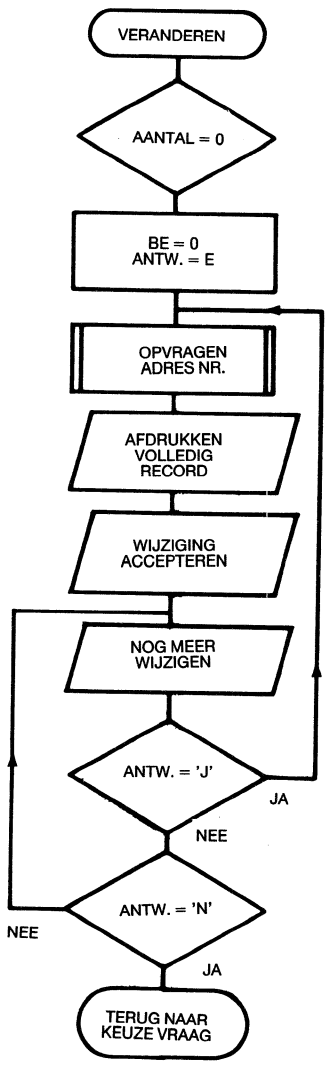


```
4000 be=1:print"■*** ■sorteren■ ***":print
4010 print"waarop moet gesorteerd worden?"
4020 print"naam(n), adres(a), postcode(p),"
4030 print"woonplaats(w), telefoon(t), kenmerk(k)":print"maak uw keuze";:gosub60
4040 ifc$="n"orc$="a"orc$="p"orc$="w"orc$="t"orc$="k"then4060
4050 print"onbekende keuze ■druk op de spatiebalk■":gosub60:goto80
4060 print:print"thans worden er < ";n;" > adressen":print"gesorteerd"
4070 print:print
4080 forx=1ton
4090 print"■";tab(15);"> ";x
4100 fory=xton
4110 ifc$="n"thenifn$(x)>n$(y)thengosub420
4120 ifc$="a"thenifa$(x)>a$(y)thengosub420
4130 ifc$="p"thenifp$(x)>p$(y)thengosub420
4140 ifc$="w"thenifw$(x)>w$(y)thengosub420
4150 ifc$="t"thenift$(x)>t$(y)thengosub420
4160 ifc$="k"thenifk$(x)>k$(y)thengosub420
4170 nexty:nextx
4180 print:print:print"sorteren gereed. druk een toets";:gosub60
4190 goto210

ready.
```

gegevens op het scherm overgenomen worden.
Deze invoer/wijziging loopt tot en met regel 5160.
In regel 5170 wordt gevraagd of er nog een adres gewijzigd moet worden (Voorbeeld wijzigen scherm).
Als het antwoord 'N' is, regel 5180, wordt er naar de keuzeregel van het menu gesprongen. Als het antwoord niet 'N' is maar 'J', regel 5190, wordt deze module opnieuw aangeroepen.
Als er geen 'N' en geen 'J' als antwoord gegeven is wordt in regel 5200 een foutboodschap gegeven en teruggesprongen naar regel 5170.

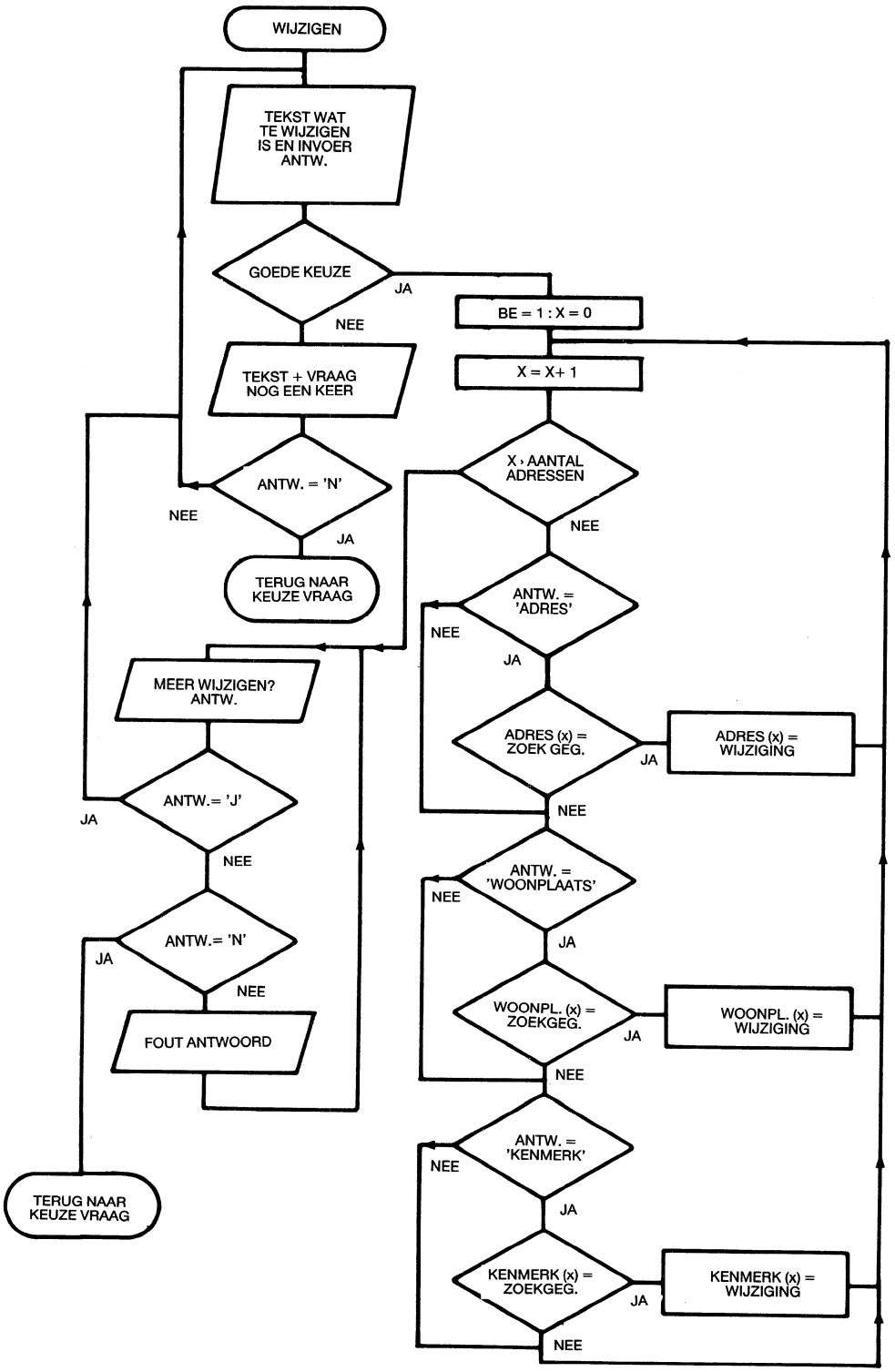
Wijzigen

Dan volgt nu de tweede module om wijzigingen in het adressenbestand aan te brengen.
Er verschijnt nu boven in het scherm het woord 'wijzigen', regel 6000, waarna vanaf regel 6010 tot en met 6030 wat korte informatie gegeven wordt over deze module.
Met behulp van deze module kan men door het hele bestand eenzelfde gegeven wijzigen. Bijvoorbeeld in het item woonplaats staat 'AMSTERDAM' en men wil dit veranderen in 'A'DAM'. Nu is het normaal zo dat elk adres handmatig aangepast moet worden, echter deze module loopt zelf het gehele bestand door.
Tijdens dit doorlopen controleert het programma of er in het opgegeven item het te wijzigen gegeven staat.
Als dit zo is dan plaatst de computer zelf de nieuwe gegevens hierin.
Regel 6040 vraagt welk item er gewijzigd moet worden (I$). Dit is dus het adres, woonplaats of kenmerk.



Hierna wordt in regel 6050 gevraagd wat er gewijzigd moet worden, in het bovenstaande voorbeeld dus AMSTERDAM. Hierna komt in regel 6060 de vraag wat er voor in de plaats komt, dit wordt dan in het voorbeeld A'DAM.
Regel 6070 controleert of hetgeen ingetypt bij item overeenkomt met de mogelijkheden. Als dit zo is dan wordt er naar regel 6120 gesprongen. Als het opgegeven item fout is verschijnt in regel 6080 een boodschap waarna in regel 6090 gevraagd wordt of er nog een keer geprobeerd moet worden. Als het antwoord 'N is, regel 6100, wordt er naar de keuzeregel gesprongen. Anders wordt via regel 6110 weer naar het begin van deze module gesprongen.
Regel 6120 begint weer met de variabele BE op 1 te stellen waarna een lus gestart wordt met behulp van variabele X.
Tevens wordt de waarde van X op het scherm gezet. Hierna wordt gecontroleerd of I$ (item naam) *niet* gelijk is aan 'ADRES' (regel 6130). Als dit zo is wordt er naar regel 6150 gesprongen. Anders wordt er gecontroleerd of het adres gelijk is aan Z$ en als dit zo is wordt aan A$(x) de waarde van W$ toegekend waarna naar regel 6190 gesprongen wordt.
Regels 6150 en 6160 zijn gelijk aan 6130 en 6140, maar dan voor item woonplaats.
Regels 6170 en 6180 zijn ook gelijk van strekking als regels 6150 en 6160.
Regel 6190 eindigt de lus waarna in regel 6200 gevraagd wordt of er nog wat gewijzigd moet worden. Bij 'J' als antwoord, regel 6210, wordt deze module weer opnieuw aangeroepen. Als het antwoord 'N' is wordt er weer naar de keuzeregel gesprongen. Als er geen 'J' of 'N' geantwoord is verschijnt er een boodschap dat de keuze fout was en kan er opnieuw ingevoerd worden.

```
5000 print"■*** ■veranderen■ ***":print
5010 ifn=0thenprint"geen bestand aanwezig!!":print:goto210
5020 be=1:c$="e":gosub310
5030 print:print"wijzig desgewenst elke regel,"
5040 print:print"geen wijziging op een regel? druk op";print"return/enter":print
5050 print"naam      :  ";n$(va)
5060 print"adres     :  ";a$(va)
5070 print"postcode  :  ";p$(va)
5080 print"woonplaats:  ";w$(va)
5090 print"telefoon  :  ";t$(va)
5100 print"kenmerk   :  ";k$(va)
5110 print"■■■■■■■■";tab(11);:inputn$(va)
5120 printtab(11);:inputa$(va)
5130 printtab(11);:inputp$(va)
5140 printtab(11);:inputw$(va)
5150 printtab(11);:inputt$(va)
5160 printtab(11);:inputk$(va)
5170 print:print"nog een adres veranderen? (j/n)";:gosub60
5180 ifc$="n"then210
5190 ifc$="j"then5000
5200 print"foute keuze":goto5170

ready.


6000 print:print"■*** ■wijzigen■ ***":print
6010 print"hiermee kan u eenzelfde item in het"
6020 print"totale bestand wijzigen. de te wijzigen"
6030 print"items zijn: adres, woonplaats en kenmerk"
6040 input"welk item wijzigen";i$
6050 input"wat moet er gewijzigd worden";z$
6060 input"wat moet het worden";w$:print:print
6070 ifi$="adres"ori$="woonplaats"ori$="kenmerk"then6120
6080 print"gekozen item niet mogelijk"
6090 print"nog een keer?(j/n)";:gosub60
6100 ifc$="n"then210
6110 goto6000
6120 be=1:forx=1ton:print"■";tab(25)"■ "x"■"
6130 ifi$<>"adres"then6150
6140 ifa$(x)=z$thena$(x)=w$:goto6190
6150 ifi$<>"woonplaats"then6170
6160 ifw$(x)=z$thenw$(x)=w$:goto6190
6170 ifi$<>"kenmerk"then6190
6180 ifk$(x)=z$thenk$(x)=w$
6190 nextx
6200 print"nog meer wijzigen? (j/n)";:gosub60
6210 ifc$="j"then6000
6220 ifc$="n"then210
6230 print"foute keuze":goto6200

ready.
```


WIJZIGEN
TEKST WAT TE WIJZIGEN IS EN INVOER ANTW.
GOEDE KEUZE
JA
NEE
TEKST + VRAAG NOG EEN KEER
ANTW. = 'N'
NEE
JA
TERUG NAAR KEUZE VRAAG
MEER WIJZIGEN? ANTW.
ANTW.= 'J'
JA
NEE
ANTW.= 'N'
JA
NEE
FOUT ANTWOORD
TERUG NAAR KEUZE VRAAG
BE = 1 : X = 0
X = X+ 1
X › AANTAL ADRESSEN
NEE
ANTW. = 'ADRES'
NEE
JA
ADRES (x) = ZOEK GEG.
JA
ADRES (x) = WIJZIGING
NEE
ANTW. = 'WOONPLAATS'
NEE
JA
WOONPL. (x) = ZOEKGEG.
JA
WOONPL. (x) = WIJZIGING
NEE
ANTW. = 'KENMERK'
NEE
JA
KENMERK (x) = ZOEKGEG.
JA
KENMERK (x) = WIJZIGING
NEE

Bewaren

De volgende module is, na de invoermodule, de belangrijkste module. Deze module is de bewaarmodule zodat we de adressen op schijf of cassette kunnen zetten.

Regel 7000 zet, net als bij elke eerste regel van een module, bovenin het scherm het woord 'bewaren'.

In regel 7010 wordt met behulp van variabele D bepaald of de vraagstelling in de volgende twee regels gesteld wordt.

De variabele D bevat namelijk het nummer van de cassette recorder, de floppydrive of de waarde nul.

De waarde nul komt alleen voor als er nog geen invoer van schijf of cassette gebeurd is of als er nog geen bewaar gedaan is.

Als D de waarde nul heeft wordt er naar regel 7040 gesprongen en anders wordt er doorgegaan met regel 7020.

In regel 7020 wordt gevraagd of het bestand met dezelfde naam weggeschreven moet worden. Dit is dus de naam die het bestand had bij het ophalen of bij de laatste keer bewaren.

Indien in regel 7030 het antwoord 'J' is wordt er direct doorgegaan met regel 7100 en anders met regel 7040.

Regel 7040 vraagt wat de naam van het bestand is en in regel 7050 wordt het antwoord opgehaald en krijgt D de waarde acht.

In regel 7060 wordt vervolgens gevraagd of er naar schijf of naar cassette geschreven moet worden.

Als het antwoord 'C' is, regel 7070, dan krijgt D de waarde 1, S de waarde 1 en D$ wordt de bestandsnaam zoals opgegeven, waarna naar regel 7120 gesprongen wordt.

Als het antwoord 'S' is, regel 7080, maken we een sprong naar 7100 omdat als er geen 'C' of 'S' opgegeven is regel 7090 een foutboodschap geeft en naar regel 7060 springt.

In regel 7100 wordt, als D groter is dan twee, aan D$ de bestandsnaam toegekend met als toevoegingen de replace funktie van de diskcommando's en de woorden 'SEQ, WRITE'. Deze laatste twee zijn nodig om de diskdrive te vertellen dat er een bestand geopend wordt voor schrijven.

Tevens wordt aan S de waarde drie toegekend.

In regel 7110 wordt, als D gelijk is aan 1, aan S de waarde 1 toegekend en aan D$ de waarde van F$. In F$ staat de bestandsnaam zoals reeds eerder opgegeven is.

Vervolgens wordt in regel 7120 een file geopend naar, afhankelijk van de waarde van D, cassette of floppy.

In regel 7130 wordt op het scherm gezet hoeveel adressen er bewaard moeten worden.

In regel 7140 wordt eerst aan N$ de waarde van N toegekend. In N staat het totaal aantal adressen.

Het omzetten gebeurt met behulp van de BASIC omzetroutine. Deze omzetroutine zorgt ervoor dat een nummerieke variabele omgezet wordt naar een stringvariabele.

Vervolgens krijgt A$ de waarde van de 'RETURN' waarna vervolgens N$ en de bestandsnaam worden weggeschreven. Dat na elke variabele die weggeschreven wordt een 'RETURN', via A$, weggeschreven wordt, gebeurt om te zorgen dat bij het teruglezen altijd het juiste gegeven gelezen wordt.

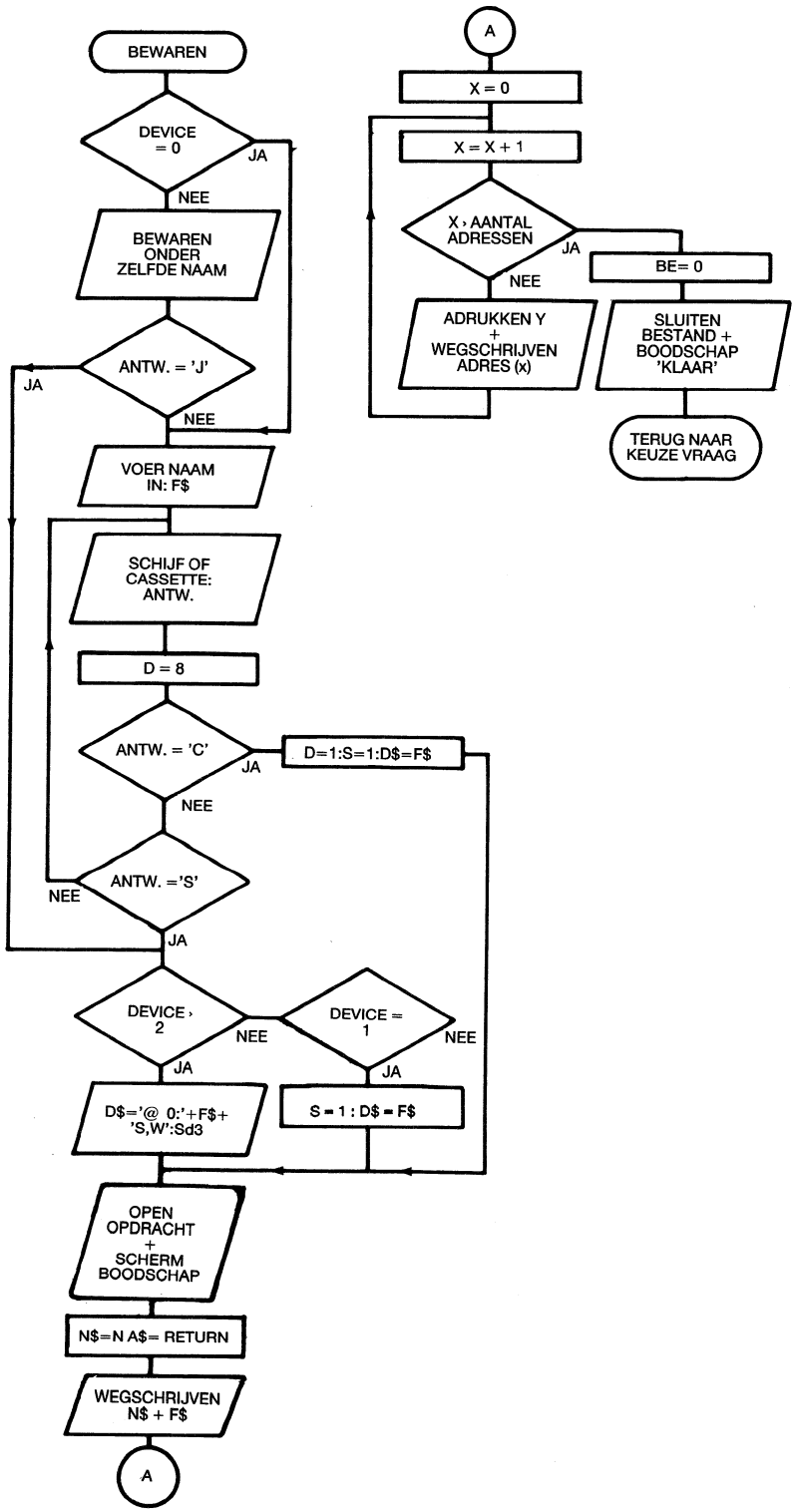
Regel 7150 start een lus met de variabele X waarna de waarde van X op het scherm wordt gezet.

De regels 7160 en 7170 schrijven achtereenvolgens alle adressen weg.

Regel 7180 sluit de lus af, sluit het bestand en stelt de variabele BE op nul.

Regel7190 geeft vervolgens de boodschap dat het bestand opgeborgen is en springt dan naar de keuzeregel.

```
7000 print"■*** ■bewaren■ ***":print
7010 ifd=0then7040
7020 print"bewaren onder dezelfde naam? (j/n)";:gosub60
7030 ifc$="j"then7100
7040 print"wat wordt de naam van het te bewaren"
7050 input"bestand ";f$:d=8
7060 print"naar schijf of cassette (s/c)";:gosub60
7070 ifc$="c"thend=1:s=1:d$=f$:goto7120
7080 ifc$="s"then7100
7090 print"foute keuze":goto7060
7100 ifd>2thend$="@0:"+f$+",seq,write":s=3
7110 ifd=1thens=1:d$=f$
7120 f=2:openf,d,s,d$
7130 print"er worden < ";n;" > adressen bewaard":print:print:print
7140 n$=str$(n):a$=chr$(13):print#f,n$;a$:print#f,f$;a$
7150 forx=1ton:print"■";tab(15);x
7160 print#f,n$(x);a$:print#f,a$(x);a$:print#f,p$(x);a$
7170 print#f,w$(x);a$:print#f,t$(x);a$:print#f,k$(x);a$
7180 nextx:closef:be=0
7190 print:print"bestand opgeborgen":print:goto210

ready.
```


BEWAREN
DEVICE = 0
JA
NEE
BEWAREN ONDER ZELFDE NAAM
ANTW. = 'J'
JA
NEE
VOER NAAM IN: F$
SCHIJF OF CASSETTE: ANTW.
D = 8
ANTW. = 'C'
JA
D=1:S=1:D$=F$
NEE
ANTW. ='S'
NEE
JA
DEVICE › 2
NEE
JA
D$='@ 0:'+F$+ 'S,W':Sd3
DEVICE = 1
NEE
JA
S = 1 : D$ = F$
OPEN OPDRACHT + SCHERM BOODSCHAP
N$=N A$= RETURN
WEGSCHRIJVEN N$ + F$
A
A
X = 0
X = X + 1
X › AANTAL ADRESSEN
JA
NEE
BE= 0
ADRUKKEN Y + WEGSCHRIJVEN ADRES (x)
SLUITEN BESTAND + BOODSCHAP 'KLAAR'
TERUG NAAR KEUZE VRAAG

Printen
De volgende module is de afdrukmodule en bestaat eigenlijk uit drie aparte delen. Een schematisch overzicht van de delen ziet er als volgt uit:

De module begint weer met bovenin het scherm 'PRINTEN' te zetten: regel 8000.
Vervolgens zet regel 8010 de boodschap op het scherm om de printer aan te zetten en daarna een toets in te drukken. Let wel op dat het programma verder niet controleert of de printer werkelijk aanstaat. Als, om welke reden dan ook, uit het programma gesprongen wordt kan er opnieuw gestart worden met GOTO 80. Dan komt het keuze menu op het scherm en zijn alle adressen nog aanwezig.
Als er een toets is ingedrukt bij regel 8010 vraagt regel 8020 wat voor een lijst er afgedrukt moet worden.
Typ nu een 'A' voor een adreslijst, een 'T' voor een telefoonlijst en een 'E' voor etiketten.
Regel 8030 controleert of de juiste toets is ingedrukt en als dit zo is wordt er naar regel 8050 gesprongen.
Bij een foute toets indruk, dus geen A, T, of E verschijnt er een foutboodschap en maakt het programma een sprong naar de keuzeregel.
Dit gebeurt in regel 8040.
In regel 8050 krijgt de variabele Z$ de waarde van C$, A, T of E, waarna na de subroutine op 280 gesprongen wordt. Deze subroutine zorgt voor de selectie van de adressen.
De regels 8060 en 8070 zorgen ervoor dat het goede deel van de module aangesproken wordt.

Telefoonlijst
Als eerste, in de module, komt de telefoonlijst aan de beurt. Om de verschillende delen makkelijk uit elkaar te houden is hier een REM ingevoerd om aan te geven dat nu de telefoonlijst komt (regel 8080).
Regel 8090 opent een file naar de printer en print vervolgens de kop van de lijst.

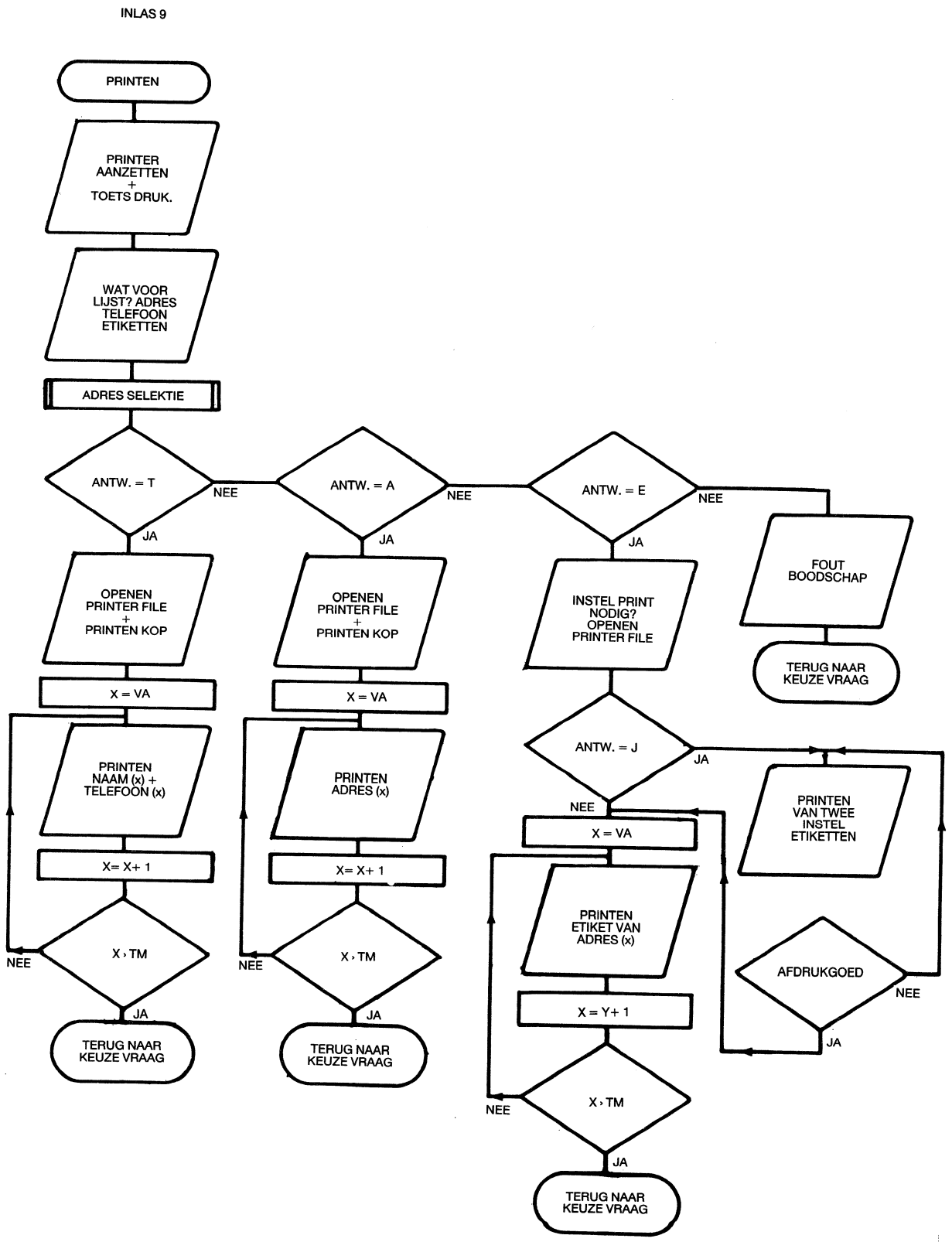
Om alles netjes onder elkaar te krijgen wordt met behulp van de variabele SP$, die aan het begin van het programma 30 spaties groot gesteld is, en de BASIC instruktie LEFT$ de totale lengte van de naam op 30 karakters gesteld.
Deze truuk zal bij de andere printdelen ook gebruikt worden.
Via regel 8100 wordt de hele printlijst aangemaakt.
De eerste opdracht is een lus te starten, met de X variabele, met als begin adres de waarde in variabele VA en als eindadres de waarde in variabele TM.
Deze twee variabelen krijgen hun waarde in de subroutine die op regel 280 begint.
Vervolgens wordt de naam en telefoonnummer geprint waarna door middel van NEXTX de lus afgesloten wordt .
Nu wordt de file naar de printer gesloten en naar de keuzeregel gesprongen.

Adreslijst
Regel 8110 geeft de start aan van het tweede deel van het printprogramma namelijk de adreslijst.
Deze adreslijst is een volledig verslag van de geselecteerde adressen en beslaat per adres twee regels.
Ook hier zal met behulp van de LEFT$-opdracht alles netjes onder elkaar gezet worden.
De regels 8120 en 8130 printen, nadat in 8120 een file naar de printer geopend is, de beide kopregels.
In de eerste kolom van de eerste regel komt de naam te staan waarna, op positie

vijfentwintig, het adres komt. Vervolgens komt op positie vijftig de postcode en woonplaats.
Op de tweede regel komt, op positie vier, het telefoonnummer en op positie vierendertig het kenmerk.
Vervolgens start regel 8140 met een lus, met variabele X, met als start de waarde van variabele VA en als eind de waarde van variabele TM.
Vervolgens worden de gegevens van de adressen op de juiste positie geprint. Dit wordt gedaan in de regels 8140, 8150 en 8160.
Regel 8160 print het kenmerk waarna de lus afgesloten wordt en daarna de file. Vervolgens wordt er naar de keuzeregel gesprongen.

Etiketten
Regel 8170 geeft aan dat hier de start is van het derde en laatste deel van de module, namelijk het printen van etiketten.
In regel 8180 wordt gevraagd of er een instelprint nodig is waarna een file naar de printer geopend wordt.
Als het antwoord op de vraag niet gelijk is aan 'J' wordt er naar regel 8220 gesprongen.
Met behulp van een lus met variabele X wordt in regels 8190 en 8200 twee etiketten geprint met sterren. Deze kunnen gebruikt worden voor het instellen van de etiketten zodat er niet op de verkeerde plaats geprint wordt.
De etiketten waar het programma mee werkt zijn enkelbaans etiketten waarbij totaal zeven regels geprint moet worden totdat het volgende etiket bereikt wordt.
Als er grotere etiketten gebruikt worden dan kan in regel 8200 voor de NEXT een aantal PRINT# 1 opdrachten ingebracht worden. Dit moet dan ook in regel 8230 gebeuren voor de NEXT opdracht.
In regel 8210 wordt gevraagd of de instelling nu wel goed is en als het antwoord niet gelijk is aan 'J' wordt, door een sprong naar regel 8190, weer twee etiketten geprint.
Als de instelling wel goed is, of als er geen instellingsafdruk nodig is, worden vanaf regel 8220 de adressen geprint. Er wordt per etiket de naam, adres, postcode en woonplaats gegevens geprint.
Het printen gebeurt weer onder controle van een lus, met de variabele X. De start- en eindwaarden van deze lus worden weer bepaald door de variabele VA en TM die uit de subroutine op regel 280 komen.
In regel 8230 wordt aan het eind de lus afgesloten, waarna de file gesloten wordt en naar de keuze regel gegaan wordt.

Verwijderen (remove)
De volgende module regelt het verwijderen van adressen uit het bestand. Via de keuze REMOVE uit het menu komt men in deze module. Er is speciaal voor het engelse woord gekozen omdat de 'V' al reeds als eerste letter voorkomt. Namelijk in VERANDEREN en dan ook in VERWIJDEREN.
Regel 9000 start met het schoonmaken van het scherm en het afdrukken van het woord 'VERWIJDEREN' bovenin het scherm.
Het verwijderen van meer dan 1 adres kan alleen gebeuren als er eerst met het hoogste adresnummer begonnen wordt. Vervolgens komt de een na hoogste en zo verder. Dit moet zo gebeuren omdat de adressen die boven het te verwijderen adres zitten naar beneden opgeschoven worden. Dit wordt ook in het kort aangegeven in regel 9010.
In regel 9020 wordt het te verwijderen adresnummer gevraagd, waarna gecontroleerd wordt of deze groter is dan nul en


INLAS 9
PRINTEN
PRINTER AANZETTEN + TOETS DRUK.
WAT VOOR LIJST? ADRES TELEFOON ETIKETTEN
ADRES SELEKTIE
ANTW. = T
NEE
ANTW. = A
NEE
ANTW. = E
NEE
JA
JA
JA
OPENEN PRINTER FILE + PRINTEN KOP
OPENEN PRINTER FILE + PRINTEN KOP
INSTEL PRINT NODIG? OPENEN PRINTER FILE
FOUT BOODSCHAP
TERUG NAAR KEUZE VRAAG
X = VA
X = VA
PRINTEN NAAM (x) + TELEFOON (x)
PRINTEN ADRES (x)
ANTW. = J
JA
NEE
X = VA
PRINTEN VAN TWEE INSTEL ETIKETTEN
X= X+ 1
X= X+ 1
PRINTEN ETIKET VAN ADRES (x)
AFDRUKGOED
NEE
JA
X = Y+ 1
X › TM
NEE
X › TM
NEE
X › TM
JA
JA
JA
TERUG NAAR KEUZE VRAAG
TERUG NAAR KEUZE VRAAG
TERUG NAAR KEUZE VRAAG

```
8000 print:print"*** printen ***":print
8010 print"zet de printer aan en druk een toets";:gosub60
8020 print"wat voor een lijst? (adres/telefoon of  etiketten)";:gosub60
8030 ifc$="a"orc$="t"orc$="e"then8050
8040 print"foute keuze":goto210
8050 z$=c$:gosub280
8060 ifz$="e"then8170
8070 ifz$="a"then8120
8080 rem start telefoonlijst
8090 open1,4:print#1,left$("naam"+sp$,30);"telefoon"
8100 forx=vatotm:print#1,left$(n$(x)+sp$,30);t$(x):nextx:close1:goto210
8110 rem start adreslijst
8120 open1,4:print#1,left$("naam"+sp$,25);left$("adres"+sp$,25);
8130 print#1,"postcode woonplaats":print#1,left$("   telefoon"+sp$,30);"kenmerk"
8140 forx=vatotm:print#1,left$(n$(x)+sp$,25);left$(a$(x)+sp$,25);
8150 print#1,left$(p$(x)+"  "+w$(x)+sp$,30):print#1,left$("   "+t$(x)+sp$,30);
8160 print#1,k$(x):nextx:close1:goto210
8170 rem start etiketten
8180 print"instel print nodig?";:gosub60:open1,4:ifc$<>"j"then8220
8190 forx=1to2:print#1:print#1:print#1,tab(5);"****":print#1,tab(5);"******* **"
8200 print#1,tab(5);"**** **  *******":print#1:print#1:nextx
8210 print"instelling goed?";:gosub60:ifc$<>"j"then8190
8220 forx=vatotm:print#1:print#1:print#1,tab(5);n$(x):print#1,tab(5);a$(x)
8230 print#1,tab(5);p$(x);"  ";w$(x):print#1:print#1:nextx:close1:goto210

ready.
```

kleiner of gelijk aan het hoogste adres. Als dit zo is wordt er naar regel 9040 gesprongen en anders geeft regel 9030 een foutboodschap en springt dan naar de keuze regel.
In regel 9040 wordt gevraagd of het zeker is dat het adres verwijderd moet worden en wordt tevens de variabele BE op 1 gezet.
Regel 9050 test of het antwoord ongelijk is aan 'J' en indien dit zo is, wordt er direkt naar de keuze regel gesprongen.
Regel 9060 test of het adres gelijk is aan het hoogste adresnummer, er hoeft dan geen adres opgeschoven te worden en als dit het geval is wordt er naar regel 9100 gesprongen.
In de regel wordt er een lus met de variabele X gestart, met als start het te verwijderen adresnummer en als eind het hoogste adres minus 1.
Vervolgens wordt de waarde van X op het scherm gezet.

In de regels 9080 en 9090 worden de adressen opgeschoven.
Dit gebeurt door elke keer het tabelelement X de waarde te geven van het tabel-element wat 1 hoger ligt.
In regel 9090 wordt de lus afgesloten.
In regel 9100 wordt de variabele met het hoogste adresnummer met 1 verlaagd waarna de module eindigt met een sprong naar de keuzeregel.

Kleur instellen

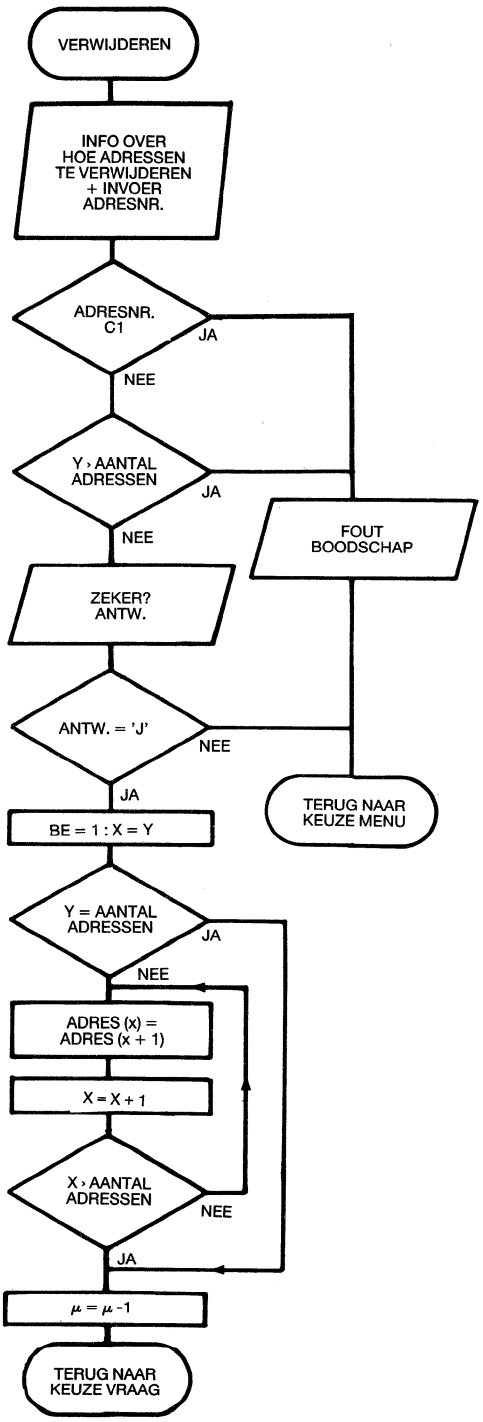
De volgende module kan alleen gebruikt worden bij de CBM 64 en 128. Dit is namelijk de routine om de schermkleuren aan te passen In regel 10000 wordt ditmaal alleen het scherm schoongemaakt.
In regel 10010 wordt een cursorhome opdracht uitgevoerd en de tekst 'SCHERM KLEUREN AANPASSEN' neergezet.
In regel 10020 komt een tekst met de mogelijkheden waarna in regel 10030 de gewenste keuze gemaakt kan worden.

```
GAK HOOFDKANTOOR
POSTBUS 8300
1005 CA  AMSTERDAM


S.C.N. HOT-NEWS
POSTBUS 6999
1009 AR  AMSTERDAM


WEKA UITGEVERIJ
D. CURTIUSSTR. 7
1051 JL  AMSTERDAM
```

Regel 10040 controleert eerst of de letter 'E' ingetoetst is en sprint naar de keuzeregel als dit zo is.
In regel 10050 vindt de controle plaats op de letter 'R' waarna, als deze letter ingetoetst is, met behulp van een POKE opdracht de randkleur aangepast wordt.
Vervolgens wordt de variabele die de randkleur aangeeft met 1 verhoogd waarna er gecontroleerd wordt of deze de waarde 16 heeft. Als dit zo is krijgt de variabele de waarde nul.
In regel 10060 vindt de controle plaats op de letter 'A' waarna, als deze letter ingetoetst is, met behulp van een POKE opdracht de achtergrondkleur aangepast wordt. Vervolgens wordt de variabele die de achtergrondkleur aangeeft met 1 verhoogd waarna er gecontroleerd wordt of deze de waarde 16 heeft.
Als dit zo is krijgt de variabele de waarde nul.
In regel 10070 wordt gekeken of de letter 'T' ingetoetst is en als dit niet zo is wordt er naar regel 10010 gesprongen.
Vanaf regel 10080 tot en met 10170 wordt de tekstkleur gecontroleerd. Dit controleren is tamelijk ingewikkeld daar de tekstkleuren niet bij elkaar liggen.
Regel 10180 geeft een sprongopdracht naar regel 10010.

Programma-einde en Hulpmenu

De laatste twee modulen zijn voor het einde van het programma en het hulp menu.
In regel 11010 wordt gecontroleerd of BE de waarde nul heeft. Dit betekent dan dat er geen wjizigingen in het bestand aangebracht zijn of dat het bestand reeds weggeschreven is.
Als BE niet gelijk is aan nul wordt de boodschap op het scherm gezet dat de wijzigingen niet bewaard zijn. dit gebeurt in regel 11020.
Vervolgens wordt in regel 11030 gevraagd of dit goed is.
In regel 11040 wordt gecontroleerd of er iets anders als 'J' ingetoetst is en indien dit zo is wordt er naar de keuzeregel gesprongen.
Regel 11050 heeft de END opdracht.
Regel 12000 print de kopregel van het hulpmenu waarna naar het menu gesprongen wordt op regel 80.



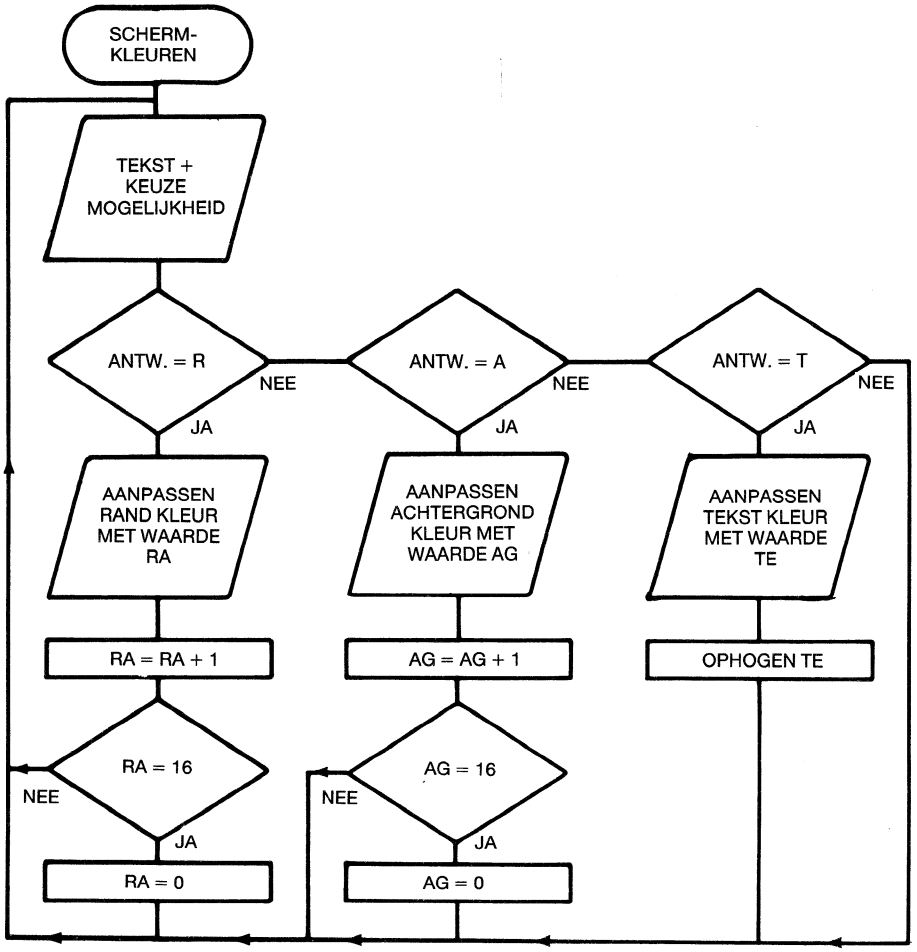
Beschrijving van de subroutine's

Ophalen Karakter
Zoals reeds eerder gezegd bevatten de regels 60 en 70 de meest gebruikte subroutine.
Deze subroutine haalt telkens een (1) teken uit de toetsenbord buffer met behulp van de GET-routine. Vervolgens wordt gecontroleerd of het wel een teken is of een zogenaamde lege string. Als het geen teken is wordt er weer teruggesprongen op dezelfde regel. Dit alles gebeurt op regel 60.
Regel 70 print eerst een (1) spatie en daarna het karakter waarna de RETURN opdracht volgt.

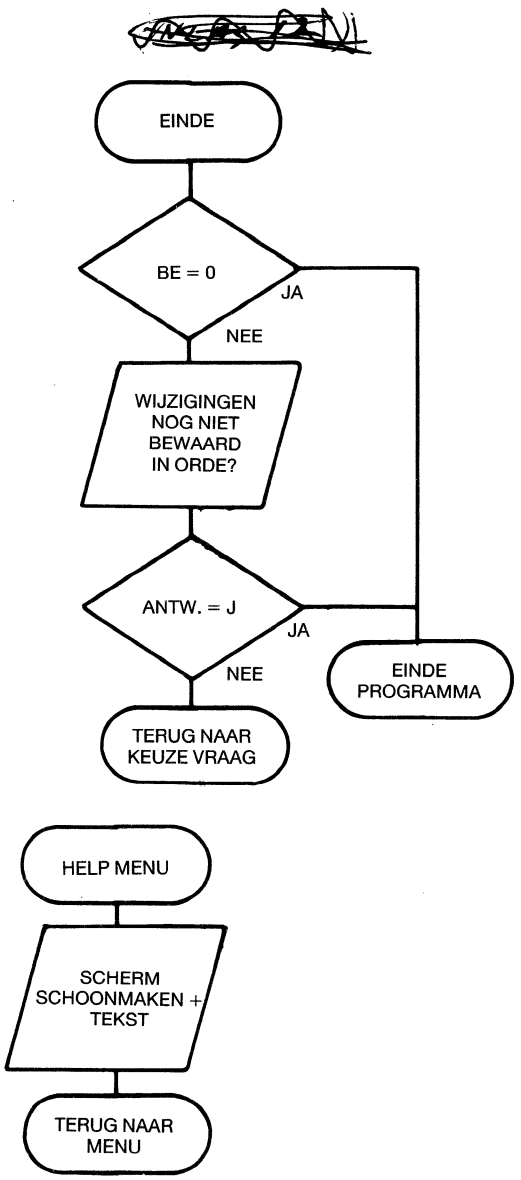
Selekteren Adressen
Regels 280 tot en met 340 bevatten de subroutine voor de slectie van adressen.
Tevens hoort de subroutine op regels 350 en 360 hier ook bij.
Regel 280 zet de vraag op het scherm welke adressen er nodig zijn en stelt de variabelen VA en TM op nul.
Vervolgens print regel 290 de keuzemogelijkheden. Regel 300 roept de subroutine op regel 60 aan en controleert dan meteen of het antwoord gelijk is aan 'A'. Als dit zo is krijgt variabele VA de waarde 1 en variabele TM de waarde van variabele N.
Regel 310 controleert of het antwoord gelijk is aan 'E'. Indien dit zo is wordt het adresnummer gevraagd via variabele VA. Hierna krijgt variabele TM de waade van VA waarna gecontroleerd wordt of VA groter is dan N of kleiner is dan 1. Als dit zo is wordt naar de subroutine op regel 360 gesprongen. Deze subroutine zet een regel op het scherm dat aangeeft hoeveel adressen er aanwezig zijn. De laatste opdracht van regel 310 is een sprong naar dezelfde regel.

```
9000 print"■*** ■verwijderen■ ***":print
9010 print"indien u meer dan 1 adres wilt verwijde-ren moet u met de hoogste";
9020 print"beginnen":input"welk adresnr verwijdern";y:ify>0andy<=nthen9040
9030 print"item niet aanwezig":goto210
9040 print"weet u het zeker?";:gosub60:print:print:be=1
9050 ifc$<>"j"then210
9060 ify=nthen9100
9070 forx=yton-1:print"■";tab(15)"■";x;"■"
9080 n$(x)=n$(x+1):a$(x)=a$(x+1):p$(x)=p$(x+1):w$(x)=w$(x+1):t$(x)=t$(x+1)
9090 k$(x)=k$(x+1):nextx
9100 n=n-1:goto210

ready.
```

```
10000 print"■"
10010 print"■*** ■scherm kleuren aanpassen■ ***":print
10020 print"rand(r), achtergrond(a), tekst(t),":print"einde(e) ";
10030 print"type de gewenste letter";:gosub60
10040 ifc$="e"then210
10050 ifc$="r"thenpoke53280,ra:ra=ra+1:ifra=16thenra=0
10060 ifc$="a"thenpoke53281,ag:ag=ag+1:ifag=16thenag=0
10070 ifc$<>"t"then10010
10080 printchr$(te):ifte=159thente=5:goto10010
10090 ifte=158thente=159
10100 ifte=156thente=158
10110 ifte>144andte<156thente=te+1
10120 ifte=144thente=149
10130 ifte=129thente=144
10140 ifte=31thente=129
10150 ifte=30thente=31
10160 ifte=28thente=30
10170 ifte=5thente=28
10180 goto10010

ready.
```



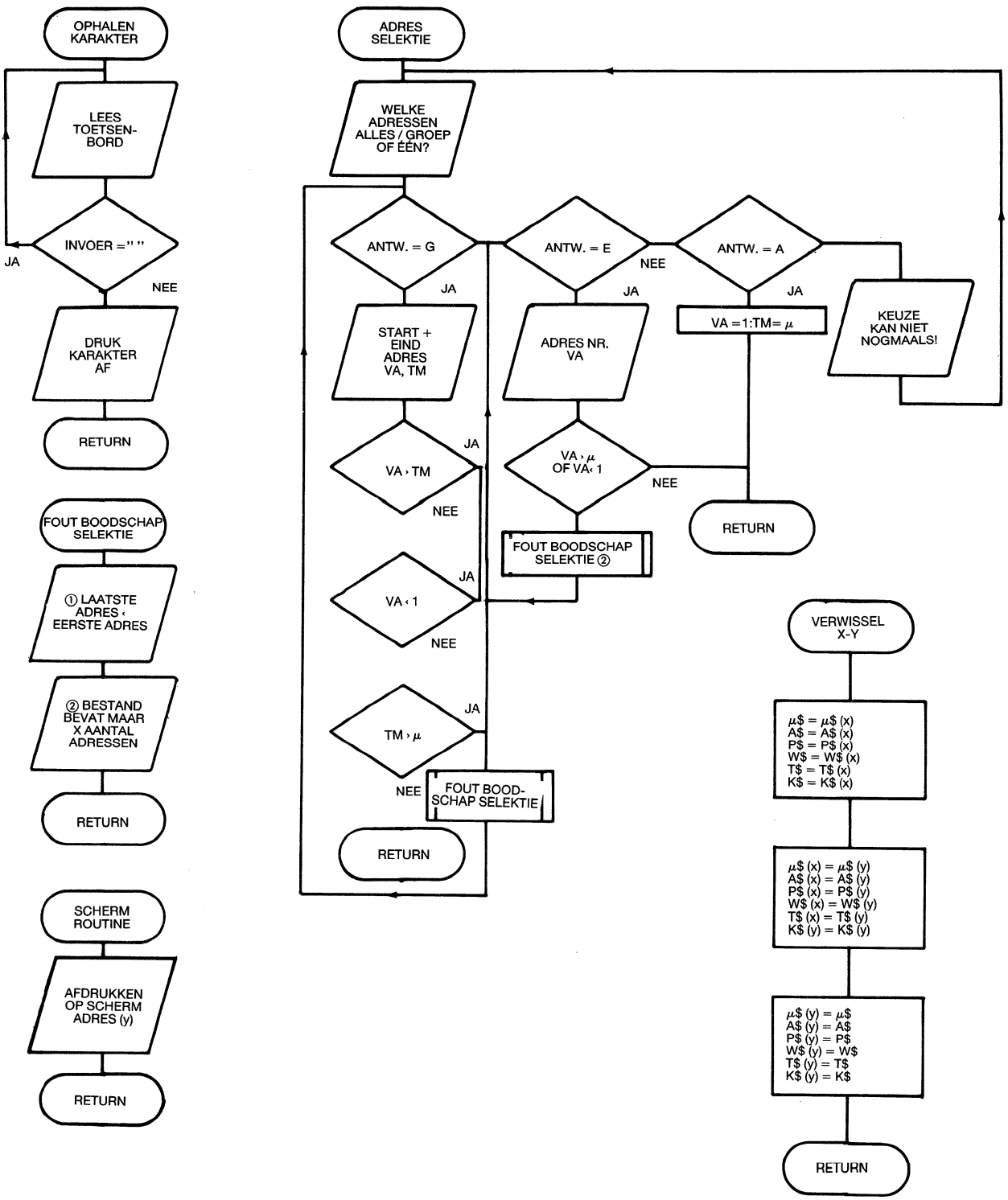

```
11000 rem eind van het programma
11010 ifbe=0then11050
11020 print"wijzigingen nog niet bewaard"
11030 print:print"is dit ok? (j/n)";:gosub60
11040 ifc$<>"j"then210
11050 end
12000 print"***  help menu  ***":print:print:goto80

ready.
```



Regel 320 controleert of het antwoord gelijk is aan 'J' waarna, als dit zo is, de vraag gesteld wordt om de start- en eindadressen aan te geven. Bij de invoer van deze twee getallen dienen deze met een komma gescheiden te worden.
Vervolgens vindt er een controle plaats op de variabelen VA en TM. Deze controle bestaat uit de controle of VA groter is dan TM en of VA kleiner is dan 1 en of TM groter is dan N. Als een van deze controles waar is, wordt er naar regel 350 gesprongen. De laatste opdracht op regel 320 is een sprong naar dezelfde regel. Deze sprong wordt alleen uitgevoerd als de sprong naar regel 350 uitgevoerd wordt.
Regel 330 controleert of de variabele VA nog gelijk is aan nul, dit kan alleen als er geen 'A', 'E' of 'G' ingevoerd was en als dit zo is wordt er een foutboodschap afgedrukt en teruggesprongen naar regel 270.
Regel 340 heeft de 'RETURN' opdracht.

Scherm Routine
De regels 370 tot en met 410 bevatten de print opdrachten om een (1) adres op het scherm te zetten. Eerst wordt het adresnummer op het scherm gezet. Hierna wordt op de volgende regel de naam gezet en op positie 28 het telefoonnummer.
De volgende regel zet het adres op het scherm waarna, op de daarop volgende regel, de postcode en woonplaats afgedrukt worden. Als laatste wordt het kenmerk op het scherm gezet waarna een 'RETURN' opdracht volgt.

Verwisselen
De verwisselroutine op regels 430 tot en met 610 is de meest eenvoudige routine die er is. Aangezien we niet zondermeer kunnen zeggen dat A gelijk moet worden aan B en dat B gelijk moet worden aan A moeten we een truuk uithalen.
Het verwisselen van twee adressen gebeurt nu door de waarden van het eerste adres aan hulpvariabelen toe te kennen.
Vervolgens worden de waarden van het tweede adres aan het eerste adres toegekend waarna het tweede adres de waarden van de hulpvariabelen krijgt.


OPHALEN KARAKTER
LEES TOETSEN-BORD
INVOER ="" ""
JA
NEE
DRUK KARAKTER AF
RETURN
FOUT BOODSCHAP SELEKTIE
① LAATSTE ADRES < EERSTE ADRES
② BESTAND BEVAT MAAR X AANTAL ADRESSEN
RETURN
SCHERM ROUTINE
AFDRUKKEN OP SCHERM ADRES (y)
RETURN
ADRES SELEKTIE
WELKE ADRESSEN ALLES / GROEP OF ÉÉN?
ANTW. = G
JA
START + EIND ADRES VA, TM
VA › TM
JA
NEE
VA ‹ 1
JA
NEE
TM › μ
JA
NEE
FOUT BOOD-SCHAP SELEKTIE
RETURN
ANTW. = E
NEE
JA
ADRES NR. VA
VA › μ OF VA ‹ 1
NEE
FOUT BOODSCHAP SELEKTIE ②
ANTW. = A
JA
VA =1:TM= μ
RETURN
KEUZE KAN NIET NOGMAALS!
VERWISSEL X-Y
μ$ = μ$ (x)
A$ = A$ (x)
P$ = P$ (x)
W$ = W$ (x)
T$ = T$ (x)
K$ = K$ (x)
μ$ (x) = μ$ (y)
A$ (x) = A$ (y)
P$ (x) = P$ (y)
W$ (x) = W$ (y)
T$ (x) = T$ (y)
K$ (y) = K$ (y)
μ$ (y) = μ$
A$ (y) = A$
P$ (y) = P$
W$ (y) = W$
T$ (y) = T$
K$ (y) = K$
RETURN

7/20

BASIC-spelletjes

Inhoud

7/20.1

Galgje

Galgje is één van de meest gespeelde spelletjes als het gaat om simpel tijdverdrijf. Toch is het ook mogelijk om Galgje een spannend spel te maken dat tegen de computer gespeeld kan worden, of dat via de computer met z'n tweëen gespeeld kan worden.

Spelregels
Voor diegenen die nog nooit van het spel gehoord hebben (bestaan die?) volgen hier de spelregels:
Een speler (de computer in dit geval) neemt een woord in zijn gedachten en zet evenveel puntjes op papier (het scherm in dit geval) als er letters in het woord aanwezig zijn. De tegenspeler gaat dan letters raden. Raadt deze een aanwezige letter dan wordt deze op alle voorkomende plaatsen ingevuld. Is de geraden letter niet in het woord aanwezig dan wordt een deel van een galg getekend. Deze galg bestaat uit 11 delen. Heeft de tegenstander dus 11 keer fout geraden en is het woord nog niet bekend dan is de galg compleet en verliest hij. Wordt het woord echter geraden voordat de galg af is dan wint de speler.

Programma
Het programma bevat zelf een aantal woorden, deze staan in de dataregels 3000 e.v. Uiteraard kunnen deze naar believen uitgebreid worden, zolang de variabele AW (aantal woorden) in regel 420 maar mee-veranderd wordt. Een andere mogelijkheid is om de computer een stuk tekst van een of andere tekstverwerker in te laten lezen en hieruit de woorden te laten selecteren. Aan deze mogelijkheid gaan we hier echter voorbij.

Om een modulaire opzet te verkrijgen is het programma opgedeeld in subroutines. Deze subroutines verzorgen de spelregels, de initialisatie, het selecteren van een woord, de controle, de invoer en de schermopbouw. Door deze subroutines één voor één aan te roepen blijft het overzicht gehandhaafd en is het programma gemakkelijk te begrijpen. Het programma is geschreven in WEKA-BASIC om eenvoudig gebruik te kunnen maken van sprites en schermsturing. Uiteraard laat het programma zich gemakkelijk naar BASIC 2.0 vertalen. Wellicht kunnen de sprites dan door standaard 64-tekens vervangen worden.

Subroutines
De gebruikte subroutines zijn:
300: initialisatie
450: spritedefenitie
700: woordselectie
800: schermopbouw
1000: invoer
1100: check op woord
1300: check op letter

20.1 Galgje

Deze subroutines zijn allen vrij eenvoudig van opzet en hier en daar van commentaar voorzien zodat het geen probleem moet zijn om deze te doorgronden. Een ondoorzichtige regel is misschien regel 1170. Hier wordt gecheckt of alle losse letters samen overeenkomen met het onbekende woord. Als dit klopt dan bevat de variabele ZZ het getal -WL, daar de vergelijking per letter steeds -1 oplevert (TRUE=-1). Op die manier detecteert het programma steeds of alle letters al bekend zijn.

Het hoofdprogramma is door het gebruik van subroutines heel eenvoudig geworden en beslaat de regels 2000 t/m 2170. De variabele H bevat het resultaat van de vergelijking. Is H=3 dan is het woord geraden en is het programma ten einde. De variabele T houdt het aantal beurten bij en de variabele L is 1 of 2, afhankelijk van het antwoord van de speler. Is L=1 dan wordt er één letter gecheckt, is L=2 dan wordt het hele woord gecontroleerd.

Aanpassing
In deze versie van galgje wordt een ingetypte letter overal ingevuld. Dat wil zeggen, dat als een woord bijvoorbeeld drie maal de letter 't' bevat, deze slechts één maal geraden hoeft te worden. Door de eerste REM in regel 1150 weg te laten kan dit veranderd worden. Staat deze REM er namelijk niet meer dan moet de letter 't' nu drie maal geraden worden!

Bij de listing bevinden zich ook 3 schermafdrukken om een indruk te krijgen hoe het spelverloop is.

20.1 Galgje

```
10 print"■n":color6,6:pen1:poke56,62:clr
20 cursor13,10:print"G A L G J E"
30 dim g$(11),b$(25),d$(25)
40 pause1
90 goto 2000
100 rem spelregels
110 print"■■nDe computer kiest een woord. De bedoe-■"
120 print"ling is om dit woord te raden. Daartoe■"
130 print"moet telkens een letter geraden worden.■"
140 print"Is deze letter goed dan verschijnt■"
150 print"die op het scherm. Is de letter fout■"
160 print"dan wordt de galg uitgebreid.■"
170 print"Lukt het niet om het woord te raden■"
180 print"voordat de galg af is dan verlies je,■"
190 print"raad je het wel dan win je.■"
200 print"■Veel plezier"
210 printtab(30)"■KTOETS>"
220 key
230 return
240 :
300 rem initialisatie
310 g$(1)="de stelling"
320 g$(2)="de paal"
330 g$(3)="de balk"
340 g$(4)="de steunbalk"
350 g$(5)="de strop"
360 g$(6)="het hoofd"
370 g$(7)="het lichaam"
380 g$(8)="de linkerarm"
390 g$(9)="de rechterarm"
400 g$(10)="het linkerbeen"
410 g$(11)="het rechterbeen"
420 g$="":c$="":t=0:h=0:f=0:aw=60:rem aw=aantal woorden
430 return
440 :
450 sproff:sprdef254
451 >..................***....
452 >.................*****...
453 >................**..**...
454 >...............**..***...
455 >.............*******....
456 >............*******.....
457 >...........*******......
458 >..........*******.......
459 >.........*******........
460 >........*******.........
461 >.......*******..........
462 >......*******...........
463 >.....*******............
464 >....*******.............
465 >...*******..............
466 >..*******...............
467 >.*******................
```

20.1 Galgje

```
468 >***..**................
469 >**..**.................
470 >*****..................
471 >****...................
475 sprite6,254,0,1,0:sprpos6,69,110
500 sprdef248
501 >.......................
502 >........*******........
503 >......**********.......
504 >.....********.****.....
505 >....*******....****....
506 >...*.............*....
507 >..*...............*...
508 >.**..*****...*****..**..
509 >*.*..*...*...*...*..*.*.
510 >*.*...***.....***...*.*.
511 >.**.......***.......**..
512 >...*......*.*......*....
513 >...*.....*...*.....*....
514 >....*....*****....*.....
515 >.....*.*.....*.*.......
516 >.....*..*******..*.....
517 >......*...***...*......
518 >.......**.....**.......
519 >.........*****.........
520 >..........*.*..........
521 >..........*.*..........
525 sprite0,248,0,1,0:sprpos0,120,120
530 sprdef249
531 >........*******........
532 >.......**.....**.......
533 >......*..*...*..*......
534 >.....*....***....*.....
535 >.....*.....*.....*.....
536 >....*......*......*....
537 >....*.....*.*.....*....
538 >....*.....*.*.....*....
539 >....*......*......*....
540 >....*......*......*....
541 >....*............*....
542 >....*............*....
543 >....*............*....
544 >....*............*....
545 >....*............*....
546 >....*............*....
547 >....*............*....
548 >....*............*....
549 >.....*..........*.....
550 >......*........*......
551 >......**********......
555 sprite1,249,0,1,0:sprpos1,120,140
560 sprdef250
561 >........................
```

20.1 Galgje

```
562 >........................
563 >........................
564 >........................
565 >........................
566 >........................
567 >........................
568 >....********............
569 >...***.....**************
570 >..***.......*............
571 >...***......*............
572 >....**.....**************
573 >....********............
574 >........................
575 >........................
576 >........................
577 >........................
578 >........................
579 >........................
580 >........................
581 >........................
585 sprite2,250,0,1,0:sprpos2,100,137
600 sprdef251
601 >........................
602 >........................
603 >........................
604 >........................
605 >........................
606 >........................
607 >........................
608 >............********....
609 >**************.....***...
610 >...........*.......***..
611 >...........*......***...
612 >**************.....**....
613 >............********....
614 >........................
615 >........................
616 >........................
617 >........................
618 >........................
619 >........................
620 >........................
621 >........................
625 sprite3,251,0,1,0:sprpos3,139,137
630 sprdef252
631 >..................*.....
632 >..................*.....*
633 >..................*.....*
634 >..................*.....*.
635 >..................*.....*.
636 >..............*.....*..
637 >..............*.....*..
638 >..............*.....*...
```

20.1 Galgje

```
639 >.............*......*...
640 >.............*......*....
641 >.............*......*....
642 >............*......*.....
643 >............*......*.....
644 >...........****...*......
645 >...........*...****......
646 >...********.....*.......
647 >..*...*****.....*.......
648 >..*.......*.....*.......
649 >..*.......*.....*.......
650 >...***************.......
651 >.........................
655 sprite4,252,0,1,0:sprpos4,108,160
660 sprdef253
661 >......*...................
662 >*......*..................
663 >*......*..................
664 >.*......*.................
665 >.*......*.................
666 >..*......*................
667 >..*......*................
668 >...*......*...............
669 >...*......*...............
670 >....*......*..............
671 >....*......*..............
672 >.....*......*.............
673 >.....*......*.............
674 >......*...****............
675 >......****...*............
676 >.......*.....********...
677 >.......*.....*****...*..
678 >.......*.....*.......*..
679 >.......*.....*.......*..
680 >.......**************....
681 >.........................
685 sprite5,253,0,1,0:sprpos5,131,160
690 return
695 :
700 rem keuze van een woord
710 x=int(rnd(0)*aw+1)
720 restore
730 for i=1 to x:read w$:next
740 wl=len(w$)
750 for i=1 to wl
760 b$(i)="."
770 d$(i)=mid$(w$,i,1)
780 next
790 return
795 :
800 rem schermopbouw
810 print"■■ G A L G J E"tab(30)"Beurt"t
820 print"■■ woord: ";:for i=1 to wl:printb$(i);:next
```

20.1 Galgje

```
825 printtab(30)"lengte";wl
830 on f goto 860,859,858,857,856,855,854,853,852,851,850
840 goto 870:rem nog geen galg
850 spron 5:rem rechterbeen
851 spron 4:rem linkerbeen
852 spron 3:rem rechterarm
853 spron 2:rem linkerarm
854 spron 1:rem lichaam
855 spron 0:rem hoofd
856 cursor 13,8:print"!":rem strop
857 spron 6:rem steunbalk
858 cursor 5,7:print"──────────┐":rem balk
859 cursor 5,7:print"┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┴":rem paal
860 cursor2,18:print"┌──┴──────────┐"
870 cursor20,7:print"Foute letters:"
880 cursor20,8:printg$
890 cursor20,18:printc$
900 cursor20,19:printe$
980 return
990 :
1000 rem invoer
1010 z$="":cursor 1,22:input"Geef letter of woord:";z$
1020 if z$="" then 1010
1030 q$=z$:l=len(z$)
1040 if l=1 then return
1050 l=2:return
1090 :
1100 rem check letter
1110 h=0:e$="":zz=0
1120 for i=1 to wl:if d$(i) <> q$ then 1160
1130 h=1:if b$(i)=q$ then 1160
1140 h=2:b$(i)=q$
1150 rem i=wl:rem weghalen voor dubbele letters
1160 next
1170 for i=1 to wl:zz=zz+(mid$(w$,i,1)=b$(i)):next
1180 if zz=-wl then q$=w$:goto 1300
1190 if h<>0 then 1220
1200 c$=q$+": Fout! Daar is"
1210 g$=g$+q$:f=f+1:e$=g$(f)
1220 if h<>1 then 1240
1230 c$=q$+": Al gebruikt!"
1240 if h<>2 then 1260
1250 c$=q$+": Goed!"
1260 return
1270 :
1300 rem check woord
1310 h=0:f=f+1:c$="Mis! Dat is fout."
1320 if q$<>w$ then return
1330 h=3:f=f-1:c$="Geraden!"
1340 for i=1 to wl:b$(i)=d$(i):next
1350 return
1360 :
2000 if cc=0 then gosub 100:rem spelregels
```

20.1 Galgje

```
2010 gosub 300:rem init
2015 if cc=0 then gosub 450:rem spritedefinition
2020 gosub 700:rem keuze
2030 gosub 800:rem schermopbouw
2040 gosub 1000:rem invoer
2050 t=t+1
2060 on l gosub 1100,1300
2070 if (h<>3) and (f<11) then 2030
2080 gosub 800
2090 c$="je antwoord is goed: "
2100 if h=3 then 2120
2110 c$="Je hangt! Het woord was "
2120 print:printc$;w$
2130 cursor 0,23:print"Nog eens (j/n)? ";
2140 get d$
2150 if d$="j" thenprint"ja":cc=1:sproff:goto 90
2160 if d$="n" thenprint"nee":sproff:end
2170 goto 2140
3000 data tomaat,nagellak,plakband,toilet,balkon
3010 data emmer,olifant,pluisje,zeemeeuw,fabrikant
3020 data automaat,computer,woordenboek,rekenmachine,fietspad
3030 data kettingzaag,projector,dierentuin,wasmachine,voordeur
3040 data monitor,televisie,videorecorder,afstandbediening,radio
3050 data printer,toetsenbord,schuifdak,venster,keukenvloer
3060 data tapijt,klapstoel,uitgeverij,dobbelsteen,galgje
3070 data biefstuk,karnemelk,kaasschaaf,angstschreeuw,zeeegel
3080 data trompet,klarinet,saxofoon,piano,pianostemmer
3090 data kastanje,denneboom,bospaadje,eikeboom,beukenootje
3100 data behang,plamuurmes,hoedenplank,wasmand,ijskast
3110 data aanrecht,fornuis,gasoven,pannelap,taartvorm
3120 data dekbed,dekbedovertrek,matras,spiraal,nachtkastje
3130 data asbak,sigaar,sigaret,aansteker,vuurtje
3140 data buitenland,binnenland,hoofdstad,kerk,dorpsgek

ready.
```

20.1 Galgje

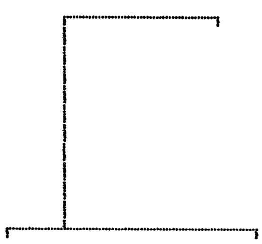
De computer kiest een woord. De bedoe-
ling is om dit woord te raden. Daartoe
moet telkens een letter geraden worden.
Is deze letter goed dan verschijnt
die op het scherm. Is de letter fout
dan wordt de galg uitgebreid.
Lukt het niet om het woord te raden
voordat de galg af is dan verlies je,
raad je het wel dan win je.

Veel plezier

<TOETS>

G A L G J E   Beurt 6

woord: au.o.aa.   lengte 8



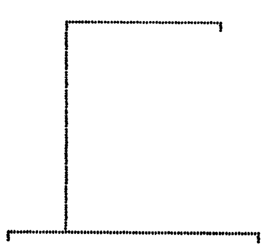
Foute letters:
eiy

y: Fout! Daar is
de balk

Geef letter of woord:? z■

G A L G J E   Beurt 8

woord: automaat   lengte 8



Foute letters:
eiy

Geraden!

je antwoord is goed: automaat
Nog eens (j/n)?

7/20.2

Reversi

Reversi (of Othello) is een bordspel voor twee spelers. De bedoeling is om zoveel mogelijk stenen van de eigen kleur op het bord te krijgen. Daar echter met elke beurt de kleurindeling van het spel wijzigt is Reversi een zeer dynamisch en verassend spel.

De attributen die nodig zijn om Reversi te kunnen spelen zijn een bord met 64 vakjes en 64 tweekleurige stenen. De stenen zijn wit aan de ene kant en zwart aan de andere kant. Bij de start van het spel liggen er vier stenen midden op het bord, te weten twee witte en twee zwarte.
De speler met wit begint met een steen met zijn kleur naar boven op het bord te leggen. Dit moet zo gebeuren dat er één of meer zwarte stenen ingesloten worden. Dit mag zowel horizontaal, vertikaal en/ of diagonaal zijn. De ingesloten stenen worden dan omgedraaid zodat ze wit worden. Zwart doet hetzelfde en probeert op die manier zoveel mogelijk zwarte stenen op het bord te krijgen.

Het spel eindigt wanneer een speler geen stenen meer op het bord heeft liggen, of wanneer alle stenen op zijn. Degene met op dat moment de meeste stenen is winnaar.

Het is duidelijk, de opzet van het spel is zeer eenvoudig. Toch is dit een spel dat uren plezier kan leveren, temeer omdat de computer een geduldige tegenspeler is.

Eenvoudige versie
De hier gepresenteerde versie van Reversi is zeer eenvoudig van opzet. De computer heeft een lijstje van locaties in aflopende volgorde van gunstigheid. Het zal duidelijk zijn dat de hoekpunten de gunstigste locaties zijn, dit omdat een steen op een hoekpunt nooit meer omgedraaid kan worden.
De spelstrategie van de computer is gebaseerd op het aflopen van dit lijstje. De eerste plaats die genomen kan worden neemt-ie dan ook. Op die manier zal het spel voor de gevorderde speler niet zo'n uitdaging zijn, de beginner zal er echter veel van kunnen leren. Het is zonder veel problemen mogelijk om het programma moeilijker en 'intelligenter' te maken, daar komen we zo op terug.

Het programma is modulair opgezet, het hoofdprogramma bestaat uit 5 regels die allen subroutines aanroepen. Deze subroutines zijn:

1920: Initialisatie
1300: Speelveld afdrukken
 840: Computer laten zetten
1060: Speler zet
1460: Spel beëindigen
1620: Zet uitvoeren

20.2 Reversi

Behalve de laatstgenoemde subroutine spreken de anderen voor zich. Zelfs de routine 'Computer laten zetten' moet met bovenstaand verhaal niet al te veel problemen op leveren. De subroutine op 1620 echter is vrij complex omdat alle mogelijke 'omdraaiingen' uitgevoerd moeten worden. Dit omdraaien gebeurt door het hele veld sequentieel te doorlopen met behulp van de richtingvectoren RX() en RY(). Deze vectoren worden voor alle 8 richtingen ingesteld en dan gebruikt om alle diagonalen, horizontalen en vertikalen te doorlopen. Is een aangrenzend veld leeg dan gebeurt er uiteraard niets (1730), is een veld van de eigen kleur dan wordt gekeken of de afstand naar het 'zet-veld' groter dan (2,2) is (regel 1750). Is dit het geval dan zorgt regel 1790 voor de kleurenwissel. De variabele Z bevat het aantal omgedraaide stenen, is dit uiteindelijk gelijk aan 0 dan is de zet niet mogelijk geweest. De subroutines 840 en 1060 maken hier gebruik van, voor de computerzet is er zelfs geen andere mogelijkheid om te zien of de zet correct is.

Aanpassingen

Zoals gezegd is het spel-algorithme van de computer zeer eenvoudig. De eerst mogelijke zet wordt altijd uitgevoerd. Het ligt uiteraard voor de hand dat deze zet niet altijd de beste hoeft te zijn. Een eenvoudige uitbreiding is om de computer alle mogelijke zetten te laten uitrekenen en prioriteiten aan echte belangrijke zetten (zoals de hoekpunten) te geven. De tweede prioriteit kan dan de zet met het grootste netto resultaat zijn (te vinden in de variabele Z). Zijn er nu meer zetten met het zelfde resultaat dan kan de toevalsgenerator (RND) uitkomst bieden: geen spel zal meer hetzelfde verlopen, iets wat nu wel het geval is.

Bij de listing bevinden zich ook 3 schermafdrukken om een indruk te krijgen hoe het spelverloop is.

20.2 Reversi

```
100 REM REVERSI
110 :
120 REM VARIABELEN:
130 REM  VELD$   :TABEL MET GEZETTE STENEN (Z OF W)
140 REM  RX,RY   :TABEL MET INCREMENTEN PER RICHTING
150 REM  R       :ONDERZOCHTE RICHTING
160 REM  KSP$    :KLEUR VAN DE SPELER
170 REM  KCO$    :KLEUR VAN DE COMPUTER
180 REM  K       :GEEFT AAN WIE AAN ZET IS:0=SPELER, 1=COMP
190 REM  AS      :AANTAL STENEN OP HET 'BORD'
200 REM  I,J     :LUSVARIABELEN
210 :
220 REM DIMENSIONERING
230 :
240 DIM VELD$(8,8)
250 DIM RX(8),RY(8)
260 :
270 REM START VAN HET SPEL
280 :
290 PRINT"■■[s][s][s][s]                R E V E R S I"
300 PRINT"[s][s]   WILT U SPELREGELS ZIEN (J/N)? ";
310 GET A$
320 IF A$="J" THEN PRINT"JA":GOTO 360
330 IF A$="N" THEN PRINT"NEE":GOTO 640
340 GOTO 310
350 :
360 REM SPELREGELS
370 :
380 PRINT"■[s] REVERSI IS EEN SPEL VOOR 2 SPELERS."
385 PRINT" DE STENEN (32 PER PERSOON) HEBBEN 2"
390 PRINT" TWEE KLEUREN:WIT AAN EEN KANT, ZWART"
395 PRINT" AAN DE ANDERE. WINNAAR IS DEGENE DIE"
400 PRINT" AAN HET EIND VAN HET SPEL DE MEESTE"
405 PRINT" STENEN VAN ZIJN KLEUR OP HET BORD"
410 PRINT" HEEFT.[s]"
415 PRINT" BIJ DE START STAAN ER REEDS 4 STENEN"
420 PRINT" OP HET BORD. EEN VOLGENDE STEEN MAG"
425 PRINT" ALLEEN ZO NEERGEZET WORDEN DAT DAAR-"
430 PRINT" DOOR EEN OF MEER STENEN VAN DE TEGEN-"
435 PRINT" SPELER INGESLOTEN WORDEN. DIT GELDT"
440 PRINT" HORIZONTAAL, VERTIKAAL EN DIAGONAAL."
445 PRINT" LUKT DIT INSLUITEN DAN WORDEN ALLE IN-"
450 PRINT" GESLOTEN STENEN OMGEDRAAID EN KRIJGEN"
455 PRINT" DUS DE KLEUR VAN DE SPELER. DE PLAATS"
460 PRINT" WAAR GEZET WORDT WORDT AANGEDUID MET"
465 PRINT" 'KOLOM,RIJ', BIJV. A,3. KAN GEEN ZET"
470 PRINT" GEDAAN WORDEN DAN MOET '0,0' INGETYPT"
475 PRINT" WORDEN..."
480 PRINTTAB(30)"<TOETS>"
485 WAIT198,1
490 PRINT"■"
640 PRINT"[s][s][s][s]
```

20.2 Reversi

```
                    SPEELT U WIT OF ZWART (W/Z)? ";
650 GETA$
660 IFA$="W"THENKSP$="W":KCO$="Z":K=1:PRINT"WIT":GOTO690
670 IFA$="Z"THENKSP$="Z":KCO$="W":K=0:PRINT"ZWART":GOTO690
680 GOTO650
690 FORI=1TO200:NEXT:PRINT"□"
700 :
710 REM HOOFDPROGRAMMA
720 :
730 GOSUB 1920:REM INITIALISATIE
740 GOSUB 1300:REM SPEELVELD
750 IF K=0 THEN GOSUB 840:REM COMP.ZET
760 IF K=1 THEN GOSUB 1060:REM U ZET
770 IF AS=64 THEN GOSUB 1460:REM EINDE
780 K=1-K:GOTO 740
790 :
800 REM SUBROUTINES
810 :
820 REM COMPUTERZET
830 :
840 PRINT
850 RESTORE
870 REM DATA VOOR SPEELSTRATEGIE
880 DATA H,1,H,8,A,1,A,8,H,3,H,6,F,1,F,8,C,1,C,8
890 DATA A,3,A,6,F,3,F,6,C,3,C,6,H,4,H,5,E,1,E,8
900 DATA D,1,D,8,A,4,A,5,F,4,F,5,E,3,E,6,D,3,D,6
910 DATA C,4,C,5,G,4,G,5,E,2,E,7,D,2,D,7,B,4,B,5
920 DATA G,3,G,6,F,2,F,7,C,2,C,7,B,3,B,6,H,2,H,7
930 DATA G,1,G,8,B,1,B,8,A,2,A,7,G,2,G,7,B,2,B,7,0,0
940 :
950 READ KOL$,RIJ
960 IF KOL$="0" AND RIJ=0 THEN 1010
970 KOL=ASC(KOL$)-64
980 IF VELD$(KOL,RIJ)<>"." THEN 950
990 GOSUB 1630:REM UITVOEREN
1000 IF Z=0 THEN 950
1010 PRINT"DE COMPUTER ZET OP ";KOL$;" ,";RIJ"■■              "
1020 RETURN
1030 :
1040 REM ZET VAN DE SPELER
1050 :
1060 PRINT:KOL$="0":RIJ=0
1070 INPUT"GEEF EEN ZET (KOLOM,RIJ)";KOL$,RIJ
1072 PRINT"                                          "
1073 PRINT"                                          "
1080 IF KOL$="0" AND RIJ=0 THEN 1260
1090 KOL=ASC(KOL$)-64
1100 :
1110 IF KOL>0 AND KOL<9 AND RIJ>0 AND RIJ<9 THEN 1150
1120 PRINT"INVOER FOUT! OVERNIEUW...□□□"
1130 GOTO 1060
1140 :
```

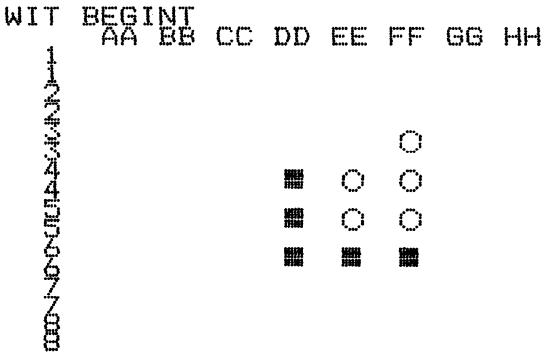
20.2 Reversi

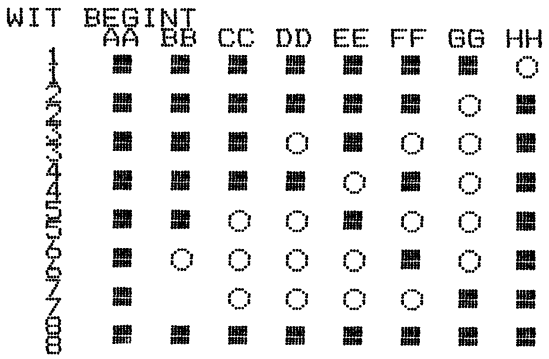
```
1150 IF VELD$(KOL,RIJ)="."THEN 1190
1160 PRINT"VELD AL BEZET! OVERNIEUW...▒▒▒▒"
1170 GOTO 1060
1180 :
1190 GOSUB 1630:REM UITVOEREN
1200 :
1210 IF Z<>0 THEN 1260
1220 PRINT"DEZE ZET IS NIET MOGELIJK! ER WORDEN"
1230 PRINT"GEEN STENEN INGESLOTEN...▒▒▒▒"
1240 GOTO 1060
1250 :
1260 RETURN
1270 :
1280 REM SPEELVELD AFDRUKKEN
1290 :
1300 PRINT"▒▒▒▒"
1310 PRINT"▒▒     AA BB CC DD EE FF GG HH "
1320 FOR J=1 TO 8
1330 : PRINT" ▒▒"J;
1340 : FOR I=1 TO 8
1350 : IFVE$(I,J)="W"THENPRINT" ▒▒▒▒▒▒▒▒";
1351 : IFVE$(I,J)="Z"THENPRINT" ▒▒▒▒▒▒▒▒";
1352 : IFVE$(I,J)="."THENPRINT"    ";
1360 : NEXT
1370 : PRINT:PRINT"▒▒ "J
1380 NEXT
1390 :
1400 RETURN
1440 :
1450 REM EINDE SPEL
1460 :
1470 Z=0
1480 REM AANTAL STENEN TELLEN
1490 FOR J=1 TO 8
1500 : FOR I=1 TO 8
1510 : IF VE$(I,J)=KC$ THEN Z=Z+1
1520 : NEXT
1530 NEXT
1540 :
1550 PRINT
1560 IF Z>32 THEN PRINT"U HEEFT MET"64-Z":"Z"STENEN VERLOREN
1570 IF Z=32 THEN PRINT"GELIJKSPEL!"
1580 IF Z<32 THEN PRINT"U HEEFT MET"64-Z":"Z"STENEN GEWONNEN"
1590 PRINT
1600 END
1610 :
1620 REM ZET UITVOEREN
1630 :
1640 Z=0
1650 IF K=0 THEN FS$=KCO$
1660 IF K=1 THEN FS$=KSP$
1670 REM ALLE RICHTINGEN ONDERZOEKEN
```

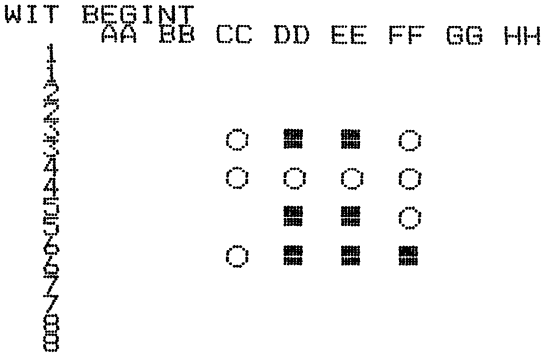
20.2 Reversi

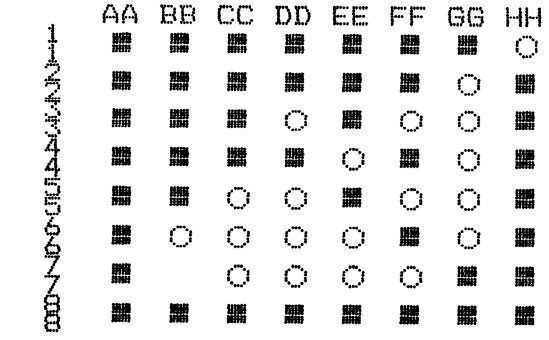
```
1680 FOR R=1 TO 8
1690 I=KO:J=RI
1700 I=I+RX(R):J=J+RY(R)
1710 IF I>8 OR I<1 OR J>8 OR J<1 THEN 1820
1720 IF VE$(I,J)=FS$ THEN 1750
1730 IF VE$(I,J)="." THEN 1820
1740 GOTO 1700
1750 IF ABS(I-KO)<2 AND ABS(J-RI)<2 THEN 1820
1760 I=KO:J=RI
1770 I=I+RX(R):J=J+RY(R)
1780 IF VE$(I,J)=FS$ THEN 1820
1790 VE$(I,J)=FS$
1800 Z=Z+1
1810 GOTO 1770
1820 NEXT
1830 :
1840 IF Z=0 THEN 1880
1850 REM Z=0: GEEN ZET MOGELIJK
1860 VE$(KO,RI)=FS$
1870 AS=AS+1
1880 RETURN
1890 :
1900 REM INITIALISATIE
1910 :
1920 REM SPEELVELD MET "." VULLEN
1930 FOR J=1 TO 8
1940 FOR I=1 TO 8
1950 VE$(I,J)="."
1960 NEXT
1970 NEXT
1980 :
1990 REM VELD VAN 4 STENEN VOORZIEN
2000 VE$(4,4)="Z":VE$(5,5)="Z"
2010 VE$(4,5)="W":VE$(5,4)="W"
2020 AS=4
2030 :
2040 REM INCREMENTEN VOOR 8 RICHTINGEN
2050 RX(1)= 1:RY(1)= 0
2060 RX(2)= 1:RY(2)= 1
2070 RX(3)= 0:RY(3)= 1
2080 RX(4)=-1:RY(4)= 1
2090 RX(5)=-1:RY(5)= 0
2100 RX(6)=-1:RY(6)=-1
2110 RX(7)= 0:RY(7)=-1
2120 RX(8)= 1:RY(8)=-1
2130 :
2140 PRINT"WIT BEGINT"
2150 RETURN

READY.
```

20.2 Reversi


WIT BEGINT
AA BB CC DD EE FF GG HH
GEEF EEN ZET (KOLOM,RIJ)? D,6


WIT BEGINT
AA BB CC DD EE FF GG HH
GEEF EEN ZET (KOLOM,RIJ)? B,7


WIT BEGINT
AA BB CC DD EE FF GG HH
DE COMPUTER ZET OP C , 4


AA BB CC DD EE FF GG HH
GEEF EEN ZET (KOLOM,RIJ)? B,7
U HEEFT MET 49 : 15 STENEN GEWONNEN
READY.

7/20.3

Getallenraadsel

Behalve dat het met de 64 goed spelletjes spelen is, is de machine ook uitstekend geschikt om spelletjes te 'bedenken'. Een goed voorbeeld van een spel dat door de computer uitgedokterd kan worden is een getallenraadsel: een rekenopgave waarbij cijfers door letters vervangen zijn. De bedoeling is dan om uit te zoeken welke letters welke cijfers voorstellen.

Typen
Het aantal typen raadsels dat ons programma kan genereren is vier. Deze vier typen zien er als volgt uit:

```
Type 1: A1 + A2 = A5
        +   +   +
        A3 + A4 = A6
        =   =   =
        A7 + A8 = A9

Type 2: A1 + A2 = A5
        +   -   +
        A3 - A4 = A6
        =   =   =
        A7 + A8 = A9

Type 3: A1 - A2 = A5
        -   -   -
        A3 - A4 = A6
        =   =   =
        A7 - A8 = A9

Type 4: A1 - A2 = A5
        -   +   -
        A3 + A4 = A6
        =   =   =
        A7 - A8 = A9
```

Bij elk type raadsel worden de getallen A1 t/m A4 door de computer gegenereerd (met behulp van de random-generator). De overige waardes worden dan berekend en gecontroleerd. De controle omvat allereerst het checken op te kleine getallen. Het kan namelijk voorkomen dat er vaak 1-of 2-cijferige getallen verschijnen. Dit wordt o.a. opgelost door te controleren of A1 kleiner dan 31 is. Zo ja dan wordt A1 gekwadrateerd.
Een tweede rekenkundige controle is nodig voor de typen 2, 3 en 4. Het mag namelijk niet voorkomen dat een van de uitkomsten negatief is. Eisen zijn dan:
2: A4<A2 en A4<A3
3: A2<A1 en A4<A3 en A3<A1 en A4<A2 en A3-A4<A1-A2
4: A2+A3+A4<A1

Het programma
Na bovenstaand verhaal is het programma vrij snel te doorgronden. Alle commando's zijn rechttoe-rechtaan, er zijn nergens 'vieze' truuks o.i.d gebruikt. Regel 330 wordt aangeroepen om de random generator te initialiseren, de array B houdt bij welke letters voor de cijfers ge-

20.3 Getallenraadsel

bruikt worden. Dit om er voor te zorgen dat een letter niet voor twee verschillende cijfers gebruikt wordt.

Zoals de voorbeelden in deze paragraaf tonen is het mogelijk om de opgave inclusief oplossing op de printer af te drukken. Om een en ander netjes onder elkaar te krijgen wordt er gebruik gemaakt van de printer-tab, CTRL-P. Deze tab is specifiek voor Commodore-printers. Het kan dus noodzakelijk zijn om de regels 1370 t/m 1410 aan te passen aan de tab-instructie van uw printer. De Epsons en compatibles bijvoorbeeld gebruiken CHR$(9) (CTRL-I) om te tabben.

```
GETALLENRAADSEL:

      GEC + EAZ = VOC
        +   +     +
      YGC + FZC = CVV
        =   =     =
     EZCV + GAC = EVAG

OPLOSSING:

      719 + 140 = 859
        +   +     +
      379 + 609 = 988
        =   =     =
     1098 + 749 = 1847
```

```
GETALLENRAADSEL:

      XSQ - IU = IY
        -   -    -
       MO - SX = XI
        =   =    =
       QX - YM = MU

OPLOSSING:

      149 - 76 = 73
        -   -    -
       58 - 41 = 17
        =   =    =
       91 - 35 = 56
```

```
GETALLENRAADSEL:

      WNF + IN = WDM
        +   -    +
      NMF - M  = NHH
        =   =    =
     IFIH + M  = IFKF

OPLOSSING:

      843 + 14 = 857
        +   -    +
      473 - 7  = 466
        =   =    =
     1316 + 7  = 1323
```

```
GETALLENRAADSEL:

      IOM - MXF = MXF
        -   +     -
      BFI + WH  = OMF
        =   =     =
      UUW - MIW =  FV

OPLOSSING:

      834 - 417 = 417
        -   +     -
      278 + 69  = 347
        =   =     =
      556 - 486 =  70
```

20.3 Getallenraadsel

```
100 REM CIJFERRAADSELS MAKEN
110 :
120 REM VARIABELE:
130 REM  Z1$ :TABEL MET OPERATOREN
140 REM  A   :TABEL MET GETALLEN
150 REM  A$  :TABEL MET RAADSELSYMBOLEN
160 REM  B$  :SYMBOOLTABEL
170 REM  TY  :TYPE OPGAVE
180 REM  B   :GEBRUIKTE LETTERS
190 :
200 DIM Z1$(4,6),A$(9),A(9),B$(10),B(26):SP$="    "
210 :
230 Z1$(1,1)="+":Z1$(1,2)="+":Z1$(1,3)="+"
240 Z1$(1,4)="+":Z1$(1,5)="+":Z1$(1,6)="+"
250 Z1$(2,1)="+":Z1$(2,2)="+":Z1$(2,3)="-"
260 Z1$(2,4)="+":Z1$(2,5)="-":Z1$(2,6)="+"
270 Z1$(3,1)="-":Z1$(3,2)="-":Z1$(3,3)="-"
280 Z1$(3,4)="-":Z1$(3,5)="-":Z1$(3,6)="-"
290 Z1$(4,1)="-":Z1$(4,2)="-":Z1$(4,3)="+"
300 Z1$(4,4)="-":Z1$(4,5)="+":Z1$(4,6)="-"
310 :
320 PRINT"■■■    G E T A L L E N R A A D S E L■■■"
330 X=RND(-TI):FOR I=1 TO 26:B(I)=0:NEXT
340 :
350 REM KEUZE VAN DE LETTERS
360 FOR I=1 TO 10
370 : X=INT(RND(1)*26)+65:IF B(X-64) THEN 370
380 : B$(I)=CHR$(X):B(X-64)=1
390 NEXT
400 :
410 REM KEUZE VAN OPGAVE-TYPE
420 TY=INT(RND(1)*4)+1
430 :
440 REM GETALLEN GENEREREN
450 A(1)=INT(RND(1)*998)+1
460 IF A(1)<31 THEN A(1)=A(1)*A(1):GOTO460
470 A(2)=INT(RND(1)*998)+1
480 IF A(2)<10 THEN A(2)=INT(A(1)/2)
490 A(3)=INT(RND(1)*998)+1
500 IF A(3)<10 THEN A(3)=INT(A(1)/3)
510 A(4)=INT(RND(1)*998)+1
520 :
530 ON TY GOSUB 580,650,780,920
540 :
550 REM OPGAVE SAMENSTELLEN EN TESTEN
560 REM OPGAVE-TYPE 1
570 :
580 A(5)=A(1)+A(2)
590 A(6)=A(3)+A(4)
600 A(7)=A(1)+A(3)
610 A(8)=A(2)+A(4)
620 A(9)=A(7)+A(8)
630 GOTO 1030
```

20.3 Getallenraadsel

```
640 :
650 REM OPGAVE-TYPE 2
660 :
670 IF A(4)>=A(3) THEN A(4)=INT(A(3)/2)
680 IF A(4)>=A(2) THEN A(4)=INT(A(2)/2)
690 A(5)=A(1)+A(2)
700 A(6)=A(3)-A(4)
710 A(7)=A(1)+A(3)
720 A(8)=A(2)-A(4)
730 A(9)=A(7)+A(8)
740 GOTO 1030
750 :
760 REM OPGAVE-TYPE 3
770 :
780 IF A(2)>=A(1) THEN A(2)=A(2)-INT(A(2)/A(1))*A(1)
790 IF A(3)>=A(1) THEN A(3)=A(3)-INT(A(3)/A(1))*A(1)
800 AH=A(2):IF A(3)<A(2) THEN AH=A(3)
810 IF A(4)>=AH THEN A(4)=A(4)-INT(A(4)/AH)*AH
820 IF A(3)-A(4)>=A(1)-A(2) THEN 420
830 A(5)=A(1)-A(2)
840 A(6)=A(3)-A(4)
850 A(7)=A(1)-A(3)
860 A(8)=A(2)-A(4)
870 A(9)=A(7)-A(8)
880 GOTO 1030
890 :
900 REM OPGAVE-TYPE 4
910 :
920 IF A(4)+A(3)+A(2)<A(1) THEN 960
930 A(2)=INT(A(1)/2)
940 A(3)=INT(A(1)/3)
950 A(4)=INT((A(1)-A(2)-A(3))/2)
960 A(5)=A(1)-A(2)
970 A(6)=A(3)+A(4)
980 A(7)=A(1)-A(3)
990 A(8)=A(2)+A(4)
1000 A(9)=A(7)-A(8)
1010 :
1020 REM VERTALEN LAAR LETTERS
1030 FOR I=1 TO 9
1040 : H$=MID$(STR$(A(I)),2):A$(I)=""
1050 : FOR J=1 TO LEN(H$)
1060 :  K=VAL(MID$(H$,J,1))
1070 :  IF K=0 AND A$(I)=LEFT$("    ",J-1) THEN A$(I)=A$(I)+" ":GOTO 1100
1080 :  IF K=0 THEN K=10
1090 :  A$(I)=A$(I)+B$(K)
1100 : NEXT
1110 : A$(I)=RIGHT$(SP$+A$(I),4)
1120 NEXT
1130 :
1140 REM RAADSEL AFDRUKKEN
1150 GOSUB 1450
1160 PRINT
```

20.3 Getallenraadsel

```
1170 PRINT
1180 PRINT
1190 PRINT"■N WILT U EEN AFDRUK OP DE PRINTER?";
1200 GET A$
1210 IF A$="N" THEN PRINT" NEE":GOTO 1320
1220 IF A$="J" THEN PRINT" JA":GOTO 1240
1230 GOTO 1200
1240 CLOSE4:OPEN 4,4:PRINT#4:PRINT#4,"GETALLENRAADSEL:":PRINT#4
1250 GOSUB 1370
1260 FOR I=1 TO 9
1270 : A$(I)=MID$(STR$(A(I)),2)
1280 : A$(I)=RIGHT$(SP$+A$(I),4)
1290 NEXT
1300 PRINT#4::PRINT#4:PRINT#4,"OPLOSSING:":PRINT#4
1310 GOSUB 1370
1320 PRINT" ■NOG EENS (J/N)? ";
1330 GET A$:IF A$<>"J" AND A$<>"N" THEN 1330
1340 IF A$="J" THEN 320
1350 END
1360 REM AFDRUK OP DE PRINTER
1370 PRINT#4,"■05"A$(1)"■10"Z1$(TY,1)"■12"A$(2)"■17= "A$(5)
1380 PRINT#4,"■08"Z1$(TY,2)"■15"Z1$(TY,3)"■22"Z1$(TY,4)
1390 PRINT#4,"■05"A$(3)"■10"Z1$(TY,5)"■12"A$(4)"■17= "A$(6)
1400 PRINT#4,"■08=■15=■22="
1410 PRINT#4,"■05"A$(7)"■10"Z1$(TY,6)"■12"A$(8)"■17= "A$(9)
1420 RETURN
1430 :
1440 REM UITVOER OP HET SCHERM
1450 PRINTTAB(5)A$(1)TAB(10)Z1$(TY,1)TAB(12)A$(2)TAB(17)"= "A$(5)
1460 PRINTTAB(8)Z1$(TY,2)TAB(15)Z1$(TY,3)TAB(22)Z1$(TY,4)
1470 PRINTTAB(5)A$(3)TAB(10)Z1$(TY,5)TAB(12)A$(4)TAB(17)"= "A$(6)
1480 PRINTTAB(8)"="TAB(15)"="TAB(22)"="
1490 PRINTTAB(5)A$(7)TAB(10)Z1$(TY,6)TAB(12)A$(8)TAB(17)"= "A$(9)
1500 RETURN

READY.
```

7/20.4

Mens erger je niet

Hier hebben we nu een spel dat niet nader hoeft te worden toegelicht. Een spel voor het hele gezin (maximaal 4 spelers), maar ook voor één enkele speler die een uurtje verveling wil verdrijven omdat ook de computer als speler kan worden ingezet. De computer kan gelukkig niet vals spelen, tijdens het testen was hij meerdere malen de verliezer! Het heerlijke pesten van tegenstanders door dobbelstenen bij het werpen door de kamer te smijten is niet meer mogelijk. Eindelijk kan nu ook bij dit spel niemand meer vals spelen.

Nodig is:

- C64 / C128
- een tot vier spelers
- een goed humeur

Omdat dit spel zo bekend is leent het programma ervan zich uitstekend voor het opfrissen van de kennis van BASIC.

Handleiding voor het spel
Zodra het programma is gestart verschijnt de naam van het spel en die van de programmeur op het scherm. Men kan dan RETURN indrukken of even 15 seconden wachten alvorens verder te gaan.

De computer vraagt nu hoeveel spelers meedoen. Dit moet beantwoord worden door een van de toetsen 1 tot 4 in te toetsen. Is het aantal spelers kleiner dan 4, dan vraagt de computer of hij zelf ook mee moet spelen. Met j of n beantwoorden.

Het schermbeeld wisselt weer en de computer verwacht invoer van de namen van de spelers (elke naam afsluiten met een RETURN). Na het invoeren wordt gevraagd of alle namen correct zijn. Dit moet worden beantwoord met j of n. Wordt n ingetoetst dan springt de computer weer terug naar het invoeren van de spelersnamen. Wordt j ingetoetst dan gaat het programma verder met het tonen van de kleuren waarmee de spelers gaan spelen.

Nu begint het eigenlijke spel. Op het scherm verschijnt het speelbord zoals we dan van het gewone spel kennen. De naam van de speler die aan de beurt is verschijnt in zijn eigen speelkleur met daar achter de zich voortdurend snel wisselende getallen 1 tot en met 6. Wordt nu RETURN ingedrukt, dan blijft het laatste getal staan. Dat getal geeft dan de waarde die men heeft "gegooid". Staat er nu nog precies één enkele pion van de speler op het bord of kan er nog maar met één enkele pion in het spel worden gebracht dan wordt met die pion verder gespeeld. Kan geen pion worden ingebracht (alleen mogelijk bij het gooien van "6") en kan ook niet verder worden gespeeld dan geeft de computer dat aan.

Na het lezen van het bericht moet een RETURN worden gegeven. Als meerdere pionnen van een speler verder kunnen, vraagt de computer met welke pion de zet moet worden uitgevoerd. Als antwoord op die vraag moet het nummer van de gekozen pion worden ingevoerd. Bij de vraag: 'met welke pion verder (1-4)?' verschijnt achter het vraagteken het aantal gegooide ogen.

Wordt een pion door een andere van het bord gegooid, dan wordt dat aangegeven. Het wegnemen van de pion wordt uitgevoerd door het indrukken van de RETURN-toets.

Speelt de computer zelf ook mee en is hij aan zet, dan hoeven de meldingen niet met een RETURN te worden beantwoord. Het spel wordt verder gespeeld met de gewone regels zoals we die van "Mens erger je niet" kennen. Er is één uitzondering: de speler die eenmaal af is mag niet nog drie keer werpen, en dan bij een "6" gooien, weer aan het spel meedoen.

Beschrijving van het programma

90 - 200	beginscherm en instellen toongenerator
210 - 400	invoeren van spelersnamen
405 - 430	instellen parameters voor kleuren
440 - 630	speelveld-setup
640 - 840	data voor speelveldposities
850 - 990	inlezen van data
1000-1100	melding wie aan de beurt is
1105-1710	uitvoeren van een worp
1800-1870	subroutine voor posities
2000-2090	subroutine voor werpen
2100-2150	"pion die aan de beurt is"
2200-2230	"werpen met deze pion mogelijk"
3000-3190	"speler is uit"
3195-3450	volgorde winnaars opgeven
4000-4210	stuurroutine voor computerspeler

Test u uw kennis van BASIC nog!

Overzicht van de variabelen

veldvariabelen

AP(4,4)	coördinaten van het startpunt
EP(4,4)	coördinaten van het eindpunt
F(4)	pokewaarden van de speelkleuren
F$(4)	speelkleuren van strings
FE$(4)	kleurstring in volgorde van de winnaars
G(4)	vlagvariabele
N$(4)	namen van de spelers
P(40)	beeldschermcoördinaten van de punten op het speelbord
PP(4,4)	posities van de pionnen van de spelers
S$(4)	namen van de spelers in volgorde van de winnaars
TH(4)	vlagvariabele voor spelers die al uit zijn
Z(4)	pionnen die nog moeten beginnen
ZE(4)	pion op eindpunt

gewone variabelen

A	algemene lusvariabele
B	secundaire lusvariabele
C	lusvariabele
F	startadres voor kleur
K	lusvariabele voor wachttijd
I	peekwaarde van een kleurpositie
J	peekwaarde van een schermbeeldpositie
L$	lege string voor regelwissen
MS	aantal spelers
N	lusvariabele
Q	teller voor de pionnen waarmee kan worden gespeeld
S	startadres van de SID

20. 4 Mens erger je niet

SG	vlag voor de computer als speler	W	pion die aan zet is
		X	aantal gegooide ogen
T	nummer van de posities van een pion	Y	teller voor spelers die klaar zijn
		Z	startadres van het beeldscherm-geheugen
T$	"get"-variabele		
UL	aantal spelers	ZZ	lusvariabele voor spelverloop
V	teller voor "3 maal werpen"		

```
90 print chr$(142);chr$(8):poke 53280,6:poke 53281,0
100 print"■■■":poke 788,52
110 print"■■*****************************************";
120 print"■*****       mens erger je niet       *****";
130 print"■■*****************************************"
140 s=54272
150 poke s+1,30:poke s+5,15
160 poke s+6,255:poke s+24,15
170 print"■■■■■■■een spel van: ■thomas roth■"
180 print tab(28);"■UCI■■■■B■■■■■■CK"
190 print"■";tab(33);"■1984"
195 for a=1 to 1500:get t$:if t$="" then next a
200 dim ap(4,4),ep(4,4),p(40)
210 print"■■■■■ **      mens erger je niet     ** ■■■■"
220 print"■■■■hoeveel spelers (1-4) ?";
230 get t$:if t$<"1" or t$>"4" then 230
240 ms=val(t$)
250 print"■■■";t$;"■■"
260 if ms=4 then 300
270 print"■■■■moet ik ook meespelen (j/n) ?"
280 get t$:if t$<>"j" and t$<>"n" then 280
290 if t$="j" then ms=ms+1:sg=1
300 print"■■■■■ **      mens erger je niet     **■■■■"
310 print"■■■nu de namen van de spelers invoeren.■■"
320 for a=1 to ms
330 print"■";a;"■■  speler ";
340 if ms=a and sg=1 then print"  dat ben ik":n$(ms)="cbm 64":goto 370
350 input n$(a)
360 if n$(a)=" " or n$(a)="" or n$(a)="cbm 64" then print"■■■":goto 330
370 next a
380 print"■■■alle namen ok (j/n) ?"
390 get t$:if t$<>"j" and t$<>"n" then 390
400 if t$="n" then run 200
405 ul=ms
410 f$(1)="■";f$(2)="■"
420 f$(3)="■";f$(4)="■"
430 f(1)=7:f(2)=1:f(3)=4:f(4)=8
440 print"■■■■■ **      mens erger je niet     ** ■■■■"
450 print"■■"
460 for a=1to ms
470 print"■";f$(a);"■ ■ ■■ ";n$(a)
```

20.4 Mens erger je niet

```
480 next a
490 print"■■■■druk 'return' om te beginnen"
500 get t$:if t$<>chr$(13) then 500
510 print"■■■■■■ **      mens erger je niet     **■■■■"
520 print"Q Q                Q-Q-Q                Q Q"
525 print"                   B   B"
530 print"Q Q                Q W Q                Q Q"
535 print"                   B   B"
540 print"                   Q W Q"
545 print"         ■mens■■      B   B      ■erger■■"
550 print"                   Q W Q"
555 print"                   B   B"
560 print"         Q-Q-Q-Q-Q W Q-Q-Q-Q-Q"
565 print"         B                   B"
570 print"         Q W W W W   W W W W Q"
575 print"         B                   B"
580 print"         Q-Q-Q-Q-Q W Q-Q-Q-Q-Q"
585 print"                   B   B"
590 print"                   Q W Q"
595 print"         ■je■■        B   B     ■niet■■"
600 print"                   Q W Q"
605 print"                   B   B"
610 print"Q Q                Q W Q                Q Q"
615 print"                   B   B"
620 print"Q Q                Q-Q-Q                Q Q■■"
630 :
640 z=1024:f=55296
700 data 40,42,120,122
710 data 75,77,155,157
720 data 795,797,875,877
730 data 760,762,840,842
740 :
750 data 60,140,220,300,380,382,384,386,388,468
760 data 548,546,544,542,540,620,700,780,860,858
770 data 856,776,696,616,536,534,532,530,528,448
780 data 368,370,372,374,376,296,216,136,56,58
790 :
800 data 450,452,454,456
810 data 138,218,298,378
820 data 466,464,462,460
830 data 778,698,618,538
840 :
850 for a=1 to 4:for b=1 to 4
860 read ap(a,b):next b,a
870 for a=1 to 40:read p(a):next a
880 for a=1 to 4:for b=1 to 4
890 read ep(a,b):next b,a
900 for a=1 to ms:for b=1 to 4
910 poke f+ap(a,b),f(a):poke z+ap(a,b),176+b:next b,a
920 for a=1 to ms:for b=1 to 4
```

20. 4 Mens erger je niet

```
930 poke f+ep(a,b),f(a):next b,a
940 for a=1 to 40:poke f+p(a),12:next a
950 s=54272
960 poke s+4,33:for a=1 to 100:next
970 poke s+4,0:for a=1 to 20:next
980 poke s+4,33:for a=1 to 400:next:poke s+4,0
985 for a=1 to 4:z(a)=4:next a
990 l$="                                                    ":poke s+1,60
1000 for zz=1 to ms:v=0
1005 if th(zz)=1 then 1350
1010 print l$:print f$(zz);n$(zz);"  is aan de beurt     ";
1015 if n$(zz)="cbm 64" then poke 198,1:poke 631,13
1020 x=int(rnd(0)*6+1)
1030 print"    ";x;
1040 get t$:if t$<>chr$(13) then 1020
1045 if n$(zz)="cbm 64" then for a=1 to 500:next
1050 poke s+4,33:for a=1 to 50:next:poke s+4,0:print
1060 if x=6 then 1600
1065 if g(zz)=1 then 1105
1070 if n$(zz)="cbm 64" then 1074
1072 print l$:print"je kan nog niet beginnen ";x:goto 1080
1074 print l$:print"ik kan nog niet beginnen ";x
1077 for a=1 to 500:next:goto 1090
1080 get t$:if t$<>chr$(13) then 1080
1090 if v<2 then v=v+1:goto 1010
1095 if v=2 then 1350
1100 :
1105 q=0:w=0:g(zz)=1
1110 for a=1 to 4:gosub 1800
1111 if pp(zz,a)=0 or pp(zz,a)+x>44 then 1130
1112 if pp(zz,a)=1 then 2200
1113 pp(zz,a)=pp(zz,a)+x:gosub 1800:pp(zz,a)=pp(zz,a)-x
1115 if pp(zz,a)+x<41 then if ((peek (f+p(t))) and 15)=f(zz) then 1130
1116 if x+pp(zz,a)<41 then 1125
1119 if (peek(z+ep(zz,pp(zz,a)+x-40)))>90 then 1130
1125 q=q+1:w=a
1130 next a
1135 if n$(zz)="cbm 64" then 4000
1140 if q<>0 then 1180
1150 print l$:print"je kan niet verder!"
1160 get t$:if t$<>chr$(13) then 1160
1170 goto 1310
1180 if q=1 then 1230
1190 print l$:print"met welke pion verder (1-4) ? ";x
1200 get t$:if t$<"1" or t$>"4" then 1200
1210 w=val(t$)
1220 if pp(zz,w)=0 or pp(zz,w)+x>44 then 1200
1222 a=w:pp(zz,w)=x+pp(zz,w):gosub 1800
1223 pp(zz,w)=pp(zz,w)-x
1225 if pp(zz,w)+x<41 then if (peek(f+p(t)) and 15)=f(zz) then 1200
```

20.4 Mens erger je niet

```
1227 if pp(zz,w)+x>40 then if peek(z+ep(zz,pp(zz,w)+x-40))>90 then 1200
1230 if pp(zz,w)<41 then 1260
1240 poke z+ep(zz,pp(zz,w)-40),87
1250 goto 1280
1260 a=w:gosub 1800:poke z+p(t),81
1270 poke f+p(t),12
1280 pp(zz,w)=pp(zz,w)+x:a=w:gosub 1800
1282 if pp(zz,w)>40 then 2100
1285 if peek(z+p(t))>90 then gosub 2000
1290 poke z+p(t),w+176
1300 poke f+p(t),f(zz)
1310 if x=6 then 1010
1350 next zz:goto 1000
1600 g(zz)=1
1602 for a=1 to 4
1605 if pp(zz,a)=1 then 1105
1610 next a
1612 for a=1 to 4
1615 if pp(zz,a)=0 then 1635
1620 next a
1630 goto 1105
1635 pp(zz,a)=1
1640 gosub 1800
1650 poke z+ap(zz,a),81
1670 if peek(z+p(t))<>81 then w=a:gosub 2000
1680 poke z+p(t),176+a
1690 poke f+p(t),f(zz)
1700 goto 1010
1710 :
1800 if zz=1 then t=pp(zz,a)-10
1810 if zz=2 then t=pp(zz,a):return
1820 if zz=3 then t=pp(zz,a)+10
1830 if zz=4 then t=pp(zz,a)+20
1840 if t>40 then t=t-40
1850 if t<1 then t=t+40
1860 return
1870 :
2000 j=peek(z+p(t))-176
2005 i=(peek(f+p(t))) and 15
2010 for n=1 to 4
2020 if f(n)=i then 2035
2030 next n
2035 poke s+1,20:poke s+4,33
2040 print l$:print f$(zz);"■";w;"■■ werpt  ";f$(n);"■";j;"■■ !"
2045 for k=1 to 500:next:poke s+1,60:poke s+4,0
2050 poke z+ap(n,j),176+j
2060 get t$:if t$<>chr$(13) then 2060
2070 z(n)=z(n)+1
2075 pp(n,j)=0
2080 return
```

20. 4 Mens erger je niet

```
2090 :
2100 c=pp(zz,w)-40
2110 poke z+ep(zz,c),w+176
2115 if pp(zz,w)-x>40 then 1310
2120 ze(zz)=ze(zz)+1:z(zz)=z(zz)+1
2130 if ze(zz)<4 then 1310
2140 goto 3000
2150 :
2200 pp(zz,a)=pp(zz,a)+x:gosub 1800
2205 pp(zz,a)=pp(zz,a)-x
2210 if ((peek(f+p(t))) and 15)=f(zz) then 1130
2220 q=1:w=a
2230 goto 1140
3000 y=y+1:s$(y)=n$(zz):fe$(y)=f$(zz):th(zz)=1
3020 poke s+4,33:for b=1 to 20
3025 print l$:print s$(y);" is";y"■ !"
3030 for a=1 to 100:next:poke s+1,30
3040 print l$:print "■";s$(y);" is";y"■ !"
3050 for a=1 to 100:next:poke s+1,40
3060 next b:poke s+4,0
3100 if y+1<ms then poke s+1,60:goto 1000
3110 print l$:print" - en -"
3120 for a=1 to 1200:next
3125 for n=1 to ms:if th(n)=0 then 3130
3127 next n
3130 y=y+1:s$(y)=n$(n):fe$(y)=f$(n)
3140 poke s+4,33:for b=1 to 20
3150 print l$:print n$(n);" is";y"■ !"
3160 for a=1 to 100:next:poke s+1,30
3170 print l$:print "■";n$(n);" is";y"■ !"
3180 for a=1 to 100:next:poke s+1,40
3190 next b
3195 poke s+4,0:for a=1 to 100:next
3200 print"■":poke s+24,1:poke s+4,33
3210 print"■■▒**********************************************▒";
3220 print"■*****      mens erger je niet      *****";
3230 print"■■■**********************************************▒■■■■"
3260 for b=1 to y:poke s+1,(y-b+2)*10
3265 print"■■■■■■":for c=1 to b:print"■":next c
3270 print b;"■ ■";fe$(b);s$(b)
3280 print"■■■■■■■■■■■■■■■■■■■■nogmaals   (j/n)?"
3290 print"■";tab(12);"■■j■/n"
3300 for a=1 to 20:get t$
3310 if t$="j" or t$="n" then 3380
3320 next a
3330 print"■";tab(12);"j/■■n■"
3340 for a=1 to 20:get t$
3350 if t$="j" or t$="n" then 3380
3360 next a
3365 print"■■■■■■":for c=1 to b:print"■":next c
```

20.4 Mens erger je niet

```
3370 print"■";b;"■. ";s$(b):next b:goto 3260
3380 print"■■■■■■■":for a=1 to y
3390 print"■";a;"■. ";s$(a)
3400 next:poke s+4,0:poke s+24,15
3410 print"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■nogmaals   (j/n)?"
3420 if t$="j" then run 200
3430 print"■dat was het";chr$(9)
3440 poke 788,49:end
3450 :
4000 if q<>0 then 4030
4010 print l$:print"ik kan niet verder!"
4020 for a=1 to 700:next:goto 1310
4030 if q>1 then 4070
4040 print l$:print"ik ga verder met pion: ";w
4050 for a=1 to 1000:next
4060 goto 1230
4070 for a=4 to 1 step-1:gosub 1800
4080 if pp(zz,a)=0 or pp(zz,a)+x>44 then 4110
4090 pp(zz,a)=pp(zz,a)+x:gosub 1800:pp(zz,a)=pp(zz,a)-x
4100 if pp(zz,a)<41 and pp(zz,a)+x>40 then 4200
4110 if pp(zz,a)+x<41 then if (peek(f+p(t))) and 15=f(zz) then 4140
4120 if x+pp(zz,a)<41 then 4135
4130 if (peek(z+ep(zz,pp(zz,a)+x-40)) and 15)>90 then 4140
4132 goto 4140
4135 if peek(z+p(t))<90 then w=a:goto 4140
4137 if (peek(f+p(t))  and 15)=f(zz) then 4140
4138 w=a:goto 4040
4140 next a
4150 goto 4040
4200 if peek(z+ep(zz,pp(zz,a)+x-40))<90 then 4138
4210 goto 4140

ready.
```

7/20.5

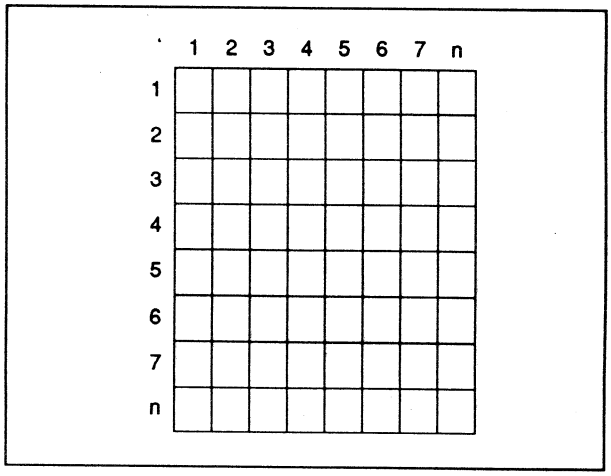
n-Koninginnenprobleem

Inleiding

Het koninginnenprobleem of n-koninginnenprobleem is een afgeleide van het schaakspel. De daarbij gebruikte oplossingsmethode die hier toegepast wordt is daarentegen vrij algemeen en heet 'backtracking' hetgeen in het Nederlands te vertalen valt als 'doorzoeken met terugvolgen' (letterlijk vertaald met 'terugvolgen'). Bij het koninginnenprobleem is het meestal de bedoeling om één oplossing te vinden. Indien echter met de computer een algoritme wordt opgesteld maakt het in feite niet uit of slechts één of alle oplossingen worden gevonden (alleen qua tijd wel natuurlijk). De oplossingen worden in de vorm van een oplossings-n-voud op het scherm gezet (bijv. 2 4 1 3). Elk element van dit n-voud legt een veld van het schaakbord vast, waarop een koningin gezet wordt. Daarnaast wordt het aantal mogelijke oplossingen gegeven.

Definitie van het probleem

Een n-schaakbord is een vierkant bord bestaande uit n bij n velden. Het totaal aantal velden is dan n^2 (n*n). Indien voor n de waarde 8 wordt genomen ontstaat een bord met 8 bij 8 velden, oftewel het 'gewone' schaakbord. Een veld wordt gekenmerkt (aangewezen) door een veld- en een kolomnummer. Bij het schaken wordt gewerkt met cijfers en letters, maar voor de computer is het eenvoudiger om voor zowel de kolommen als de rijen getallen te gebruiken. De kolommen lopen van links naar rechts en de rijen van boven naar beneden, elk genummerd van 1 tot en met n, zie fig. 1



Figuur 7/20.5-1: n-schaakbord

In het schaakspel heeft ieder van de twee strijdende partijen slechts één koningin. Bij het n-koninginnen probleem wordt, zoals de naam al zegt, uitgegaan van n koninginnen (en geen andere stukken). Bij een normaal schaakbord gaat het dan om in totaal 8 koninginnen (n=8).

Het gebied dat een koningin bestrijkt beslaat niet alleen de kolom en de rij waarop de koningin staat (zoals bij een toren uit het schaakspel), maar ook nog de diagonalen die

door haar veld lopen (lijnen onder 45 graden t.o.v. een kolom/rij).

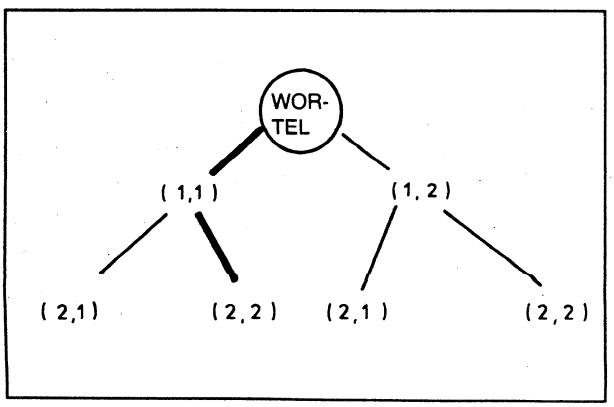
Een koningin bedreigt een andere koningin of, anders gezegd, kan een andere koningin slaan (of natuurlijk geslagen worden, maar dat onderscheid is hier niet essentieel), indien zo'n andere koningin binnen het bereik van de éne koningin valt.

Doel van het 'spel'

Bij het koninginnenprobleem gaat het erom de n koninginnen zo te plaatsen dat ze elkaar niet kunnen slaan. Het huidige programma zal alle oplossingen vinden en weergeven. Ook oplossingen die kunnen ontstaan door draaiing van het (schaak)bord worden als verschillend aangemerkt, daar het programma het schaakbord maar vanuit één richting bekijkt.

Oplossingsmethode

Zoals reeds is vermeld, zal van 'backtracking' gebruikt worden gemaakt. Dit kan met behulp van een zogenaamde oplossingsboom, zoals weergegeven in figuur 2 voor het geval n=2. Elke knoop van de boom (met uitzondering van de wortel) geeft de plaats van een koningin weer. De plaats wordt aangegeven door een rij- en kolomnummer. Elk 'niveau' in de boom gaat zodoende over een rij. Waarbij dan elk knooppunt van een vorige rij steeds een hele huidige rij bevat.



Figuur 7/20.5-2: Oplossingsboom, N=2

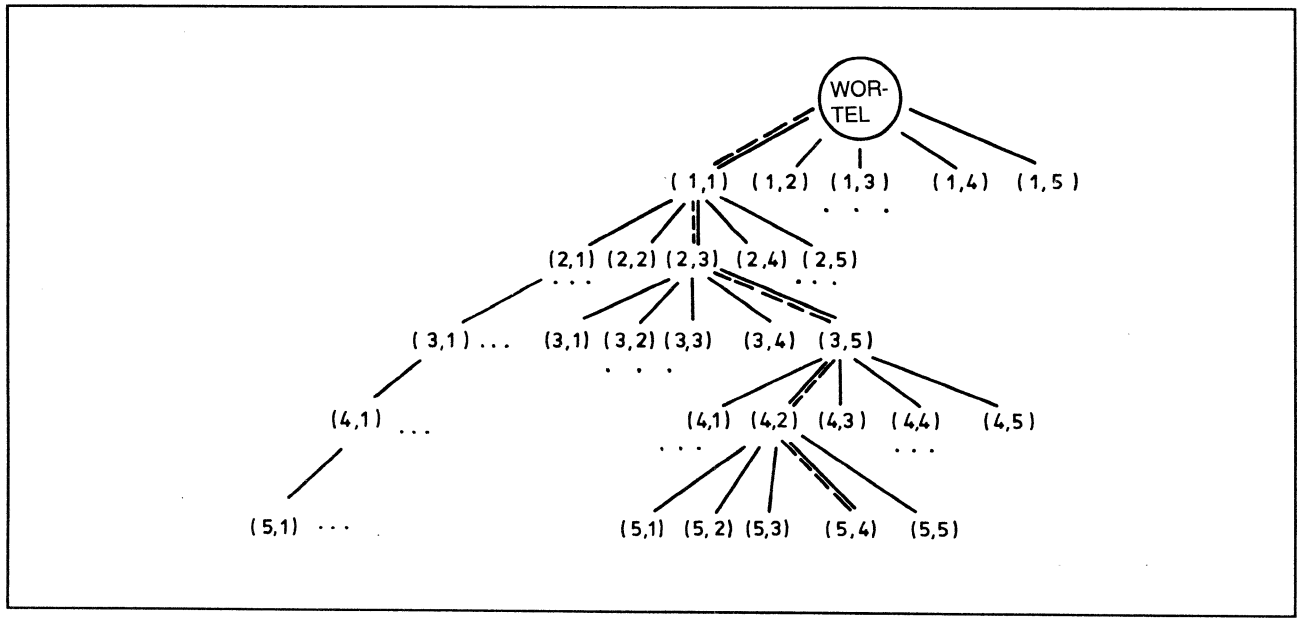
Bij het n-koninginnen mag slechts één koningin per rij voorkomen. Een oplossing is zodoende een tak (met lengte n) uit de oplossingsboom. In figuur 2 is een tak vet aangegeven. Een oplossing heeft dus de vorm van een tak, maar omgekeerd is niet iedere tak een oplossing. Merk op dat bij het 2-schaakbord geen oplossing mogelijk is.

Doordat zo'n oplossingsboom alle mogelijkheden bevat met op elke rij een koningin, is het voldoende om alle takken van de boom na te lopen om te kijken of het een oplossing is. Door alle takken, die allemaal een lengte n hebben, van links naar rechts af te lopen, worden alle mogelijke oplossingen gevonden. Het is natuurlijk mogelijk bij iedere tak elke knoop van zo'n tak te bekijken, maar indien in het begin (bijvoorbeeld bij de eerste en de derde rij) al een conflict optreedt, of anders gezegd dat in het begin al geconstateerd wordt dat een koningin binnen het gebied van een andere koningin valt, is het niet nodig onderliggende knopen te bekijken. Hiermee worden in zo'n geval gelijk een aantal takken uitgeschakeld, hetgeen de zoektijd (aanzienlijk) verkort (zie figuur 3).

Op het moment dat bij het bekijken van een volgende knoop uit een tak die tak of takken die daarbij horen niet langer een oplossing zijn, dan moet een volgende knoop (uit dezelfde rij) worden genomen. Is zo'n knoop niet meer aanwezig moet een niveau naar 'boven' worden gegaan. DIt moet net zolang gebeuren tot alle takken bekeken zijn en er onder de wortel dus geen nieuwe knoop meer is.

Indien alle knopen van een boom zouden moeten worden onderzocht zou dit (n*n+n) stappen vergen. Doordat bij het koninginnenprobleem steeds (grote) delen wegvallen is er een beduidend kleiner aantal stappen

20.5 n-Koninginnenprobleem



Figuur 7/20.5-3: Deel van de oplossingsboom voor N=5

nodig. Om een voorbeeld te geven, de 5-tak (1,1) - (2,1) - (3,1) - (4,1) - (5,1) in figuur 3 hoeft na de eerste twee knopen al niet verder te worden bekeken, daar de eerste twee koninginnen al binnen elkaars gebied vallen. Ook de andere takken die mogelijk zijn vanuit de knopen (1,1) - (2,1) hoeven niet verder bekeken te worden. Na dit geconstateerd te hebben kan worden verder gegaan met de 5-tak vanaf knoop (2,2) (die dus ook niet mogelijk is). Zo wordt verder gegaan tot bij knoop (2,3) een oplossing wordt gevonden, zie figuur 3. En daarna verder zoeken naar meerdere oplossingen.

Programma

In het programma worden de knopen van de oplossingsboom voorgesteld door een ééndimensionale array KONINGIN(N). De index K in KONINGIN(K) staat voor het nummer van de rij, de waarde KONINGIN(K) staat voor het nummer van de kolom. Deze array representeert dus precies een knoop-n-voud en dus een opstelling van n koninginnen op een n-schaakbord.

Wederzijdse bedreiging

Op wederzijdse bedreiging oftewel het binnen elkaars gebied vallen van twee koninginnen kan door de volgende uitdrukking worden getest:

```
(KO(I) = KO(K)) or
(ABS(KO(I)-KO(K)) = K-I)
```

De eerste regel (het deel voor de logische of) heeft betrekking op het testen of twee koninginnen in dezelfde kolom staan. De tweede regel (het deel na de of) verzorgt het controleren of twee koninginnen op eenzelfde diagonaal staan. Een diagonaal houdt in dat eenzelfde aantal plaatsen naar links of rechts moet worden gegaan als naar boven of naar beneden. Om de K-I hoeft geen ABS() te staan omdat K altijd groter of gelijk aan I is. Er hoeft niet getest te worden of er meerdere koninginnen op een rij staan, door het controleren per tak. Een tak heeft steeds maar één knoop per niveau en aangezien een niveau overeen komt met een rij, dus ook maar één koningin per rij.

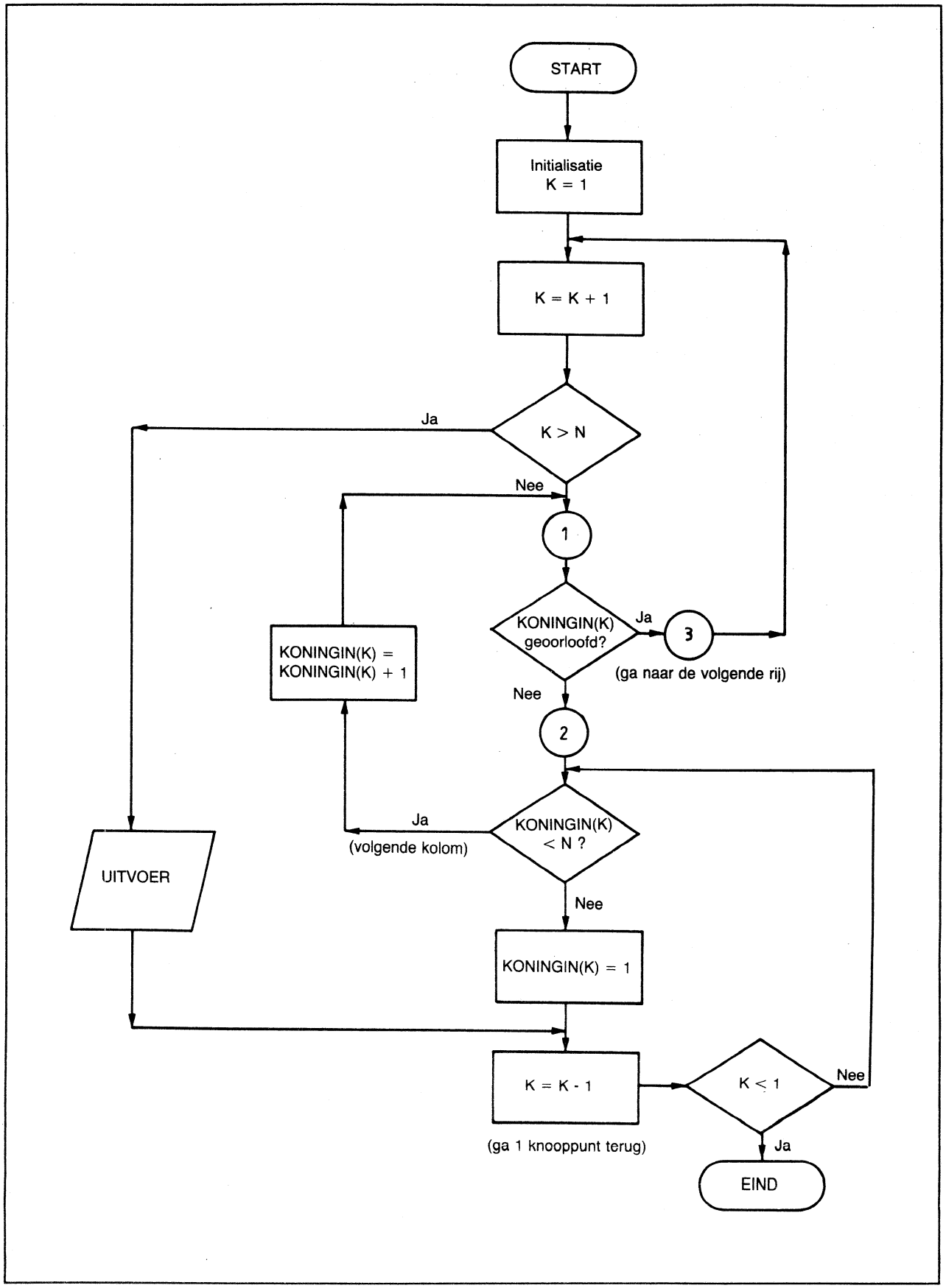
20.5 n-Koninginnenprobleem

De rekentijd kan, afhankelijk van n, enige minuten bedragen. Dit komt omdat het programma niet qua rekentijd is geoptimaliseerd, misschien iets om zelf uit te zoeken? Er is getracht het programma zo goed mogelijk overeen te laten komen met hetgeen behandeld is, zonder op de programmastructuur te letten. Wanneer u voldoende inzicht heeft gekregen in het programma, is het natuurlijk op snelheid en/of programmatechnisch te optimaliseren, maar het is daarbij niet uitgesloten dat dat de duidelijkheid schaadt. De besproken werkwijze komt ook uitgebreid naar voren in de stroomschema's.

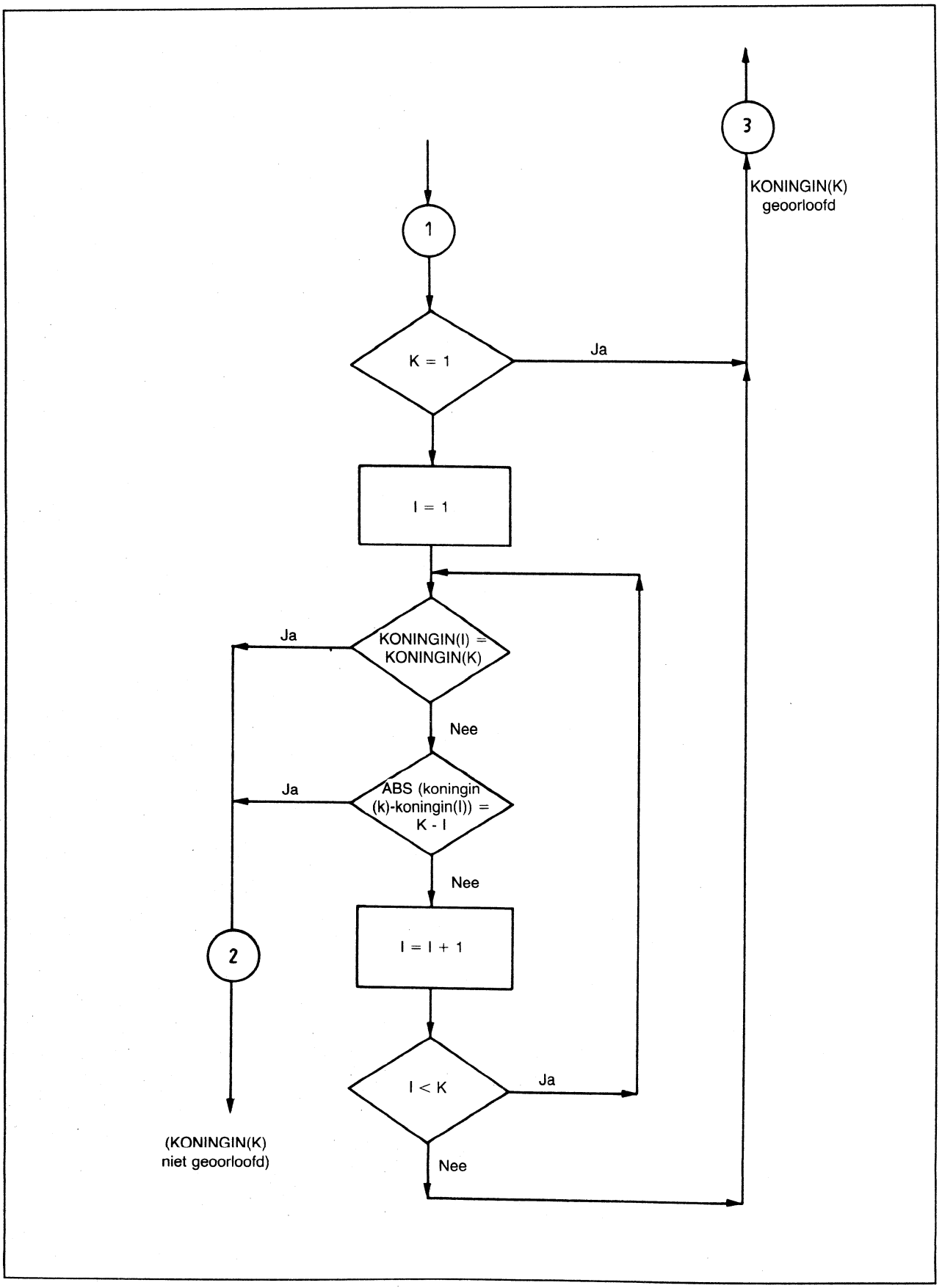
```
10 REM ------------------------------------------------------
20 REM                    N-KONINGINNEPROBLEEM
30 REM ------------------------------------------------------
40 REM
50 T=0:CLS$=CHR$(147)
60 PRINTCLS$
70 PRINT"N-KONINGINNENPROBLEEM"
80 PRINT"----------------------"
90 PRINT:PRINT
100 INPUT"GEEF DE GROOTTE VAN HET SCHAAKBORD";N
110 DIM KO(N)        :REM POSITIES IN KO()
120 REM INITIALISATIE
130 REM ALLE KONINGINNEN WORDEN IN KOLOM 1 GEZET
140 FOR I=1 TO N
150 :  KO(I)=1
160 NEXT
170 K=1
180 K=K+1            :REM VOLGENDE RIJ
190 IF K>N THEN 310 :REM OPLOSSING GEVONDEN, DAN PRINTEN
200 REM IS KO(K) GEOORLOOFD?
210 IF K=1 THEN 180 :REM UITBREIDING KOLOM 1 IS ALTIJD TOEGESTAAN
220 I=1              :REM LOKAAL
230 IF (KO(I)=KO(K)) OR (ABS(KO(K)-KO(I))=K-I) THEN 270
240 I=I+1
250 IF I<K THEN 230
260 GOTO 180         :REM KO(K) TOEGESTAAN
270 REM KO(K) NIET TOEGESTAAN
280 IF KO(K)>=N THEN KO(K)=1:GOTO 370
290 KO(K)=KO(K)+1    :REM IN VOLGENDE KOLOM ZETTEN
300 GOTO 200
310 REM UITVOERGEDEELTE EN RAND TESTEN
320 FOR I=1 TO N
330 :  PRINT KO(I);
340 NEXT
350 T=T+1
360 PRINT
370 K=K-1            :REM EEN KNOOPPUNT TERUG
380 IF K>=1 THEN 280
390 PRINT"AANTAL OPLOSSINGEN:"T
```

Listing 7/20.5-1: n-koninginneprobleem

20.5 n-Koninginnenprobleem



20.5 n-Koninginnenprobleem



7/21

Berekeningen in BASIC

Inhoud

7/21.1

Fouriercoëfficiënten

Probleembeschrijving
Elke functie f(x) is te schrijven als een trigoniometrische rij van de vorm:

$$f(x) = \sum_{n=0}^{\infty} (a_n \cos nx + b_n \sin nx)$$

Het is dan wel nodig dat de functie eindig is en niet een oneindig aantal sprongen bevat. Uit de parameters a_n en b_n zijn de volgende grootheden te verkrijgen:

$$A_n = \sqrt{a_n + b_n} \quad \text{en} \quad \varphi_n = arctan\frac{a_n}{b_n}$$

Van deze grootheden stelt A_n de amplitude van de grondgolf en de harmonische voor en φ de faseverschuiving. De waarden a_n en b_n kunnen met behulp integralen worden berekend, maar dit is vaak vrij bewerkelijk. Soms zijn de functies niet gegeven, maar zijn er slechts enkele meetpunten bekend. Dan moeten a_n en b_n numeriek worden berekend.

Oplossingsmethode
In dit programma zullen slechts een beperkt aantal Fouriercoëfficiënten worden berekend. Indien N het aantal meetpunten is, worden de volgende formules gebruikt om M Fouriercoëfficienten te berekenen. a_0 tot en met a_{N-1} en b_1 tot en met b_{N-1}.

$$a_0 = \frac{2}{N} \sum_{i=0}^{N-1} y_i$$

$$a_k = \frac{2}{N} \sum_{i=0}^{N-1} y_i \cos\frac{k \cdot 2\pi i}{N}$$

voor k = 1,2,...,(M-1)/2

$$b_k = \frac{2}{N} \sum_{i=0}^{N-1} y_i \sin\frac{k \cdot 2\pi i}{N}$$

voor k = 1,2,...,(M-1)/2

$$y_i = f(\frac{2\pi i}{N})$$

voor i = 0,1,...,N-1

In het programma wordt de lusvariabele/teller zo gekozen dat steeds een stapgrootte van 2*PI/N ontstaat.

21.1 Fouriercoëfficiënten

Structuurdiagram

Lees in:	Aantal punten N Aantal gewenste coëfficiënten M
Indien M even is, dan M=M+1	
S=2*PI/N (stapgrootte)	
Voor I=0 to 2*PI met stapgrootte S doe	
	bereken de functiewaarde F=f(I)
	a(0)=a(0)+F
	voor J=1 tot j=(M-1)/2 doe
	a(J)=a(J)+F*cos(I*J)
	b(J)=b(J)+F*sin(I*J)
a(0)=a(0)/N	
geef a(0) weer	
voor j=1 tot(M-1)/2 doe	
	a(J)=a(J)*2/N
	b(J)=b(J)*2/N
	geef a(J) en b(J) weer.

De functie zelf

Voor de meetpunten wordt van de functie zelf uitgegaan. Het moet namelijk een functie zijn met een grondgolf met een lengte van 2*PI (/K), met K = 1, 2, 3, ... Indien u zelf een andere functie wilt benaderen moet u bij regel 590 zijn, want daar worden in een subroutine de (meet)punten van de functie berekend.

```
GEEF AANTAL STEUNPUNTEN 8
GEEF AANTAL FOURIERCOEFFICIENTEN 6


A( 0 )=-4.80578514E-04
A( 1 )= .99960279
                    B( 1 )= 3.99429409E-04
A( 2 )=-7.25526336E-04
                    B( 2 )= 7.98267406E-04
A( 3 )=-3.92770878E-04
                    B( 3 )= 1.19573852E-03
```

21.1 Fouriercoëfficiënten

Indien bij regel 590 de volgende functie wordt ingevoerd:

F = COS(SI) + SIN(SI) + SIN(2*SI) + .5 * COS(2*SI)

geeft het programma de volgende resultaten.

```
GEEF AANTAL STEUNPUNTEN 10
GEEF AANTAL FOURIERCOEFFICIENTEN 4


A( 0 )=-2.29055062E-04
A( 1 )= 1.00010403
                    B( 1 )= 1.00077234
A( 2 )= .500020699
                    B( 2 )= 1.00101708
```

Zoals op te merken valt komen de waarden bijna overeen met de echte waarden:

$a_0=0$
$a_1=1$
$a_2=0.5$
$b_1=1$
$b_2=1$

21.1 Fouriercoëfficiënten

```
100 REM FOURIERCOEFFICIENTEN
110 REM
120 REM VOOR DE COMMODORE 64
130 REM
140 REM VARIABELEN:
150 REM
160 REM F     : FUNCTIEWAARDE
170 REM AO    : FOURIERCOEFFICIENT A(0)
180 REM A     : MATRIX VOOR FOURIERCOEFFICIENTEN A(I)
190 REM B     : MATRIX VOOR FOURIERCOEFFICIENTEN B(I)
200 REM N     : AANTAL STEUNPUNTEN
210 REM M     : GEWENST AANTAL FOURIERCOEFFICIENTEN
220 REM S     : STAPGROOTTE
230 REM I,J   : INDEXTELLERS
240 REM H$    : INVOERVARIABELE
250 REM
280 DIM A(101),B(101)
290 PRINTCHR$(147)
300 INPUT"GEEF AANTAL STEUNPUNTEN";N
310 INPUT"GEEF AANTAL FOURIERCOEFFICIENTEN";M
320 IF M<1 OR M>100 GOTO 300
330 IF M/2=INT(M/2) THEN M=M+1
340 S=6.28/N:SI=0
350 FOR I=0 TO N-1
360 SI=I*S
370 GOSUB 590
380 AO=AO+F
390 FOR J=1 TO (M-1)/2
400 A(J)=A(J)+F*COS(SI*J)
410 B(J)=B(J)+F*SIN(SI*J)
420 NEXT
430 NEXT
440 :
450 AO=AO/N
460 PRINT CHR$(147)
470 PRINT"A( 0 )=";AO
480 FOR J=1 TO (M-1)/2
490 A(J)=A(J)*2/N
500 B(J)=B(J)*2/N
510 PRINT"A(";J;")="A(J);CHR$(13);TAB(17);"B(";J")=";B(J)
520 IF INT(J/20)=J/20 THEN INPUT "DRUK OP RETURN...";H$
530 NEXT
540 :
550 END
560 :
570 REM SUBROUTINE VOOR BEREKENING FUNCTIEWAARDE
580 REM
590 F=COS(SI)
600 RETURN
```

7/21.2

Figuren van Lissajous

Inleiding

Mocht u ooit eens bij iemand op bezoek zijn geweest die in het bezit is van een oscilloscoop, dan is u het een en ander aan mooie plaatjes voorgeschoteld. Op zich misschien wel een aardig idee om eens te proberen een zelfde soort plaatje op het beeldscherm van de Commodore 64 te toveren.

Eerst even enige begrippen uitleggen. Een oscilloscoop is een apparaat om elektrische signalen zichtbaar te maken. Dus bijvoorbeeld om de netspanning te laten zien (al kan dat niet direct) of om makkelijker een versterker door te kunnen meten. Vervolgens wordt gebruik gemaakt van de sinus en de cosinus, die misschien nog enigszins bekend zijn van gonio of meetkunde. Hier worden ze gebruikt als tijdfuncties die een beeld te zien geven als van een golvende zee.

Op zich is het misschien wel aardig om enkel een sinus of een cosinus op een beeldscherm weer te geven, maar in dit verhaal gaat het om een combinatie. Niet alleen op de signaalingang van een oscilloscoop wordt een sinusoïde (sinus of cosinus) aangeboden maar ook op de tijds-ingang. Om het anders te zeggen zowel de X- als de Y-as krijgen een sinusoïde te verwerken. In de elektronica gaat dat door twee al dan niet gekoppelde sinusgeneratoren op een oscilloscoop aan te sluiten.

Theorie

We gaan nu wat dieper in op de sinus- en cosinusfunctie. In het platte vlak, dus met X- en Y-coördinaten gaan we uit van de volgende functies

$X_t = A_X \sin(360.f_X.t)$

$Y_t = A_Y \cos(360.f_Y.t)$

De variabelen A_X en A_Y staan voor de amplituden ofwel de hoogte van de golf. f_t en f_y staan voor de frequentie, het aantal golven per seconde. X_t en Y_t bepalen de plaats op tijdstip t. Indien A en f in beide regels gelijk worden gekozen, dus $A_X=A_Y$ en $f_t=f_y$, dan ontstaat een 'gewone' cirkel. Het wordt echter interessanter indien bij de cosinus-functie een fasehoek (φ) phi wordt opgeteld. Zo ontstaan de meest algemene formules voor het verkrijgen van lissajous figuren:

$X_t = A_X \sin(360.f_X.t)$

$Y_t = A_Y \cos(360.f_Y.t + ...)$

Programma

1. De grafische mode wordt ingesteld in regel 1120.

2. Als gebruiker moet u de amplitude (A), de frequentie (f) en het faseverschil (φ) opgeven en hier vraagt het programma

21.2 Figuren van Lissajous

dan ook om. De amplitude wordt in pixels opgegeven, met een maximum van 100 en het faseverschil in graden. Zie regel 1180-1240.

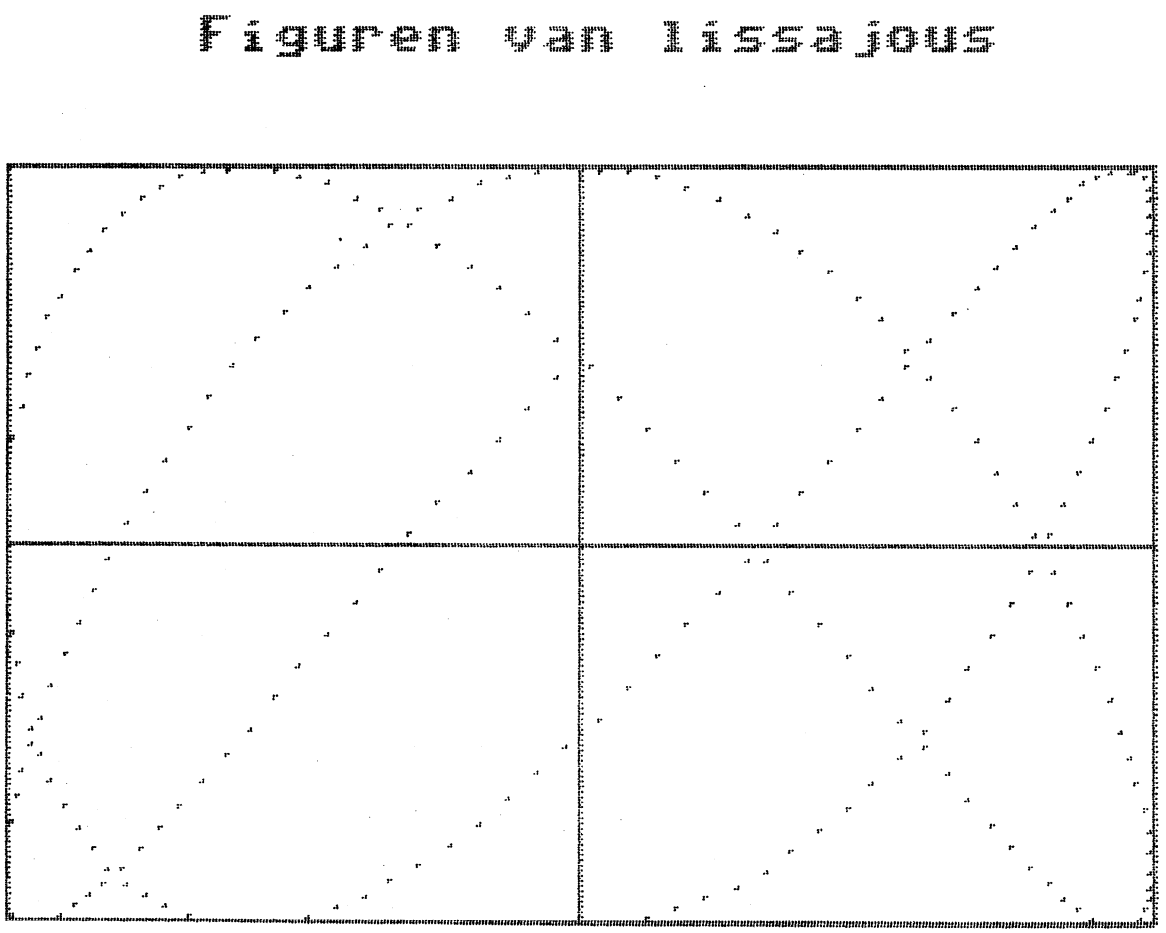
3. Het programma vraagt ook hoeveel trillingen op het scherm moeten worden weergegeven, zie regel 1270

4. Het corrigeren voor de schermafmetingen en het verschillend aantal punten in de X- en Y-richting gebeurt in de regels 1290-1350

5. Het eigenlijke tekenen vindt plaats in de regels 1380-1410. Per 3 graden worden de X- en Y-coördinaten berekend.

Vanwege de eenvoud van het programma lijkt een structuurschema niet echt noodzakelijk. Ter illustratie volgen twee figuren van Lissajous. Nu kunt u dus ook andere mensen van deze mooie plaatjes laten genieten...

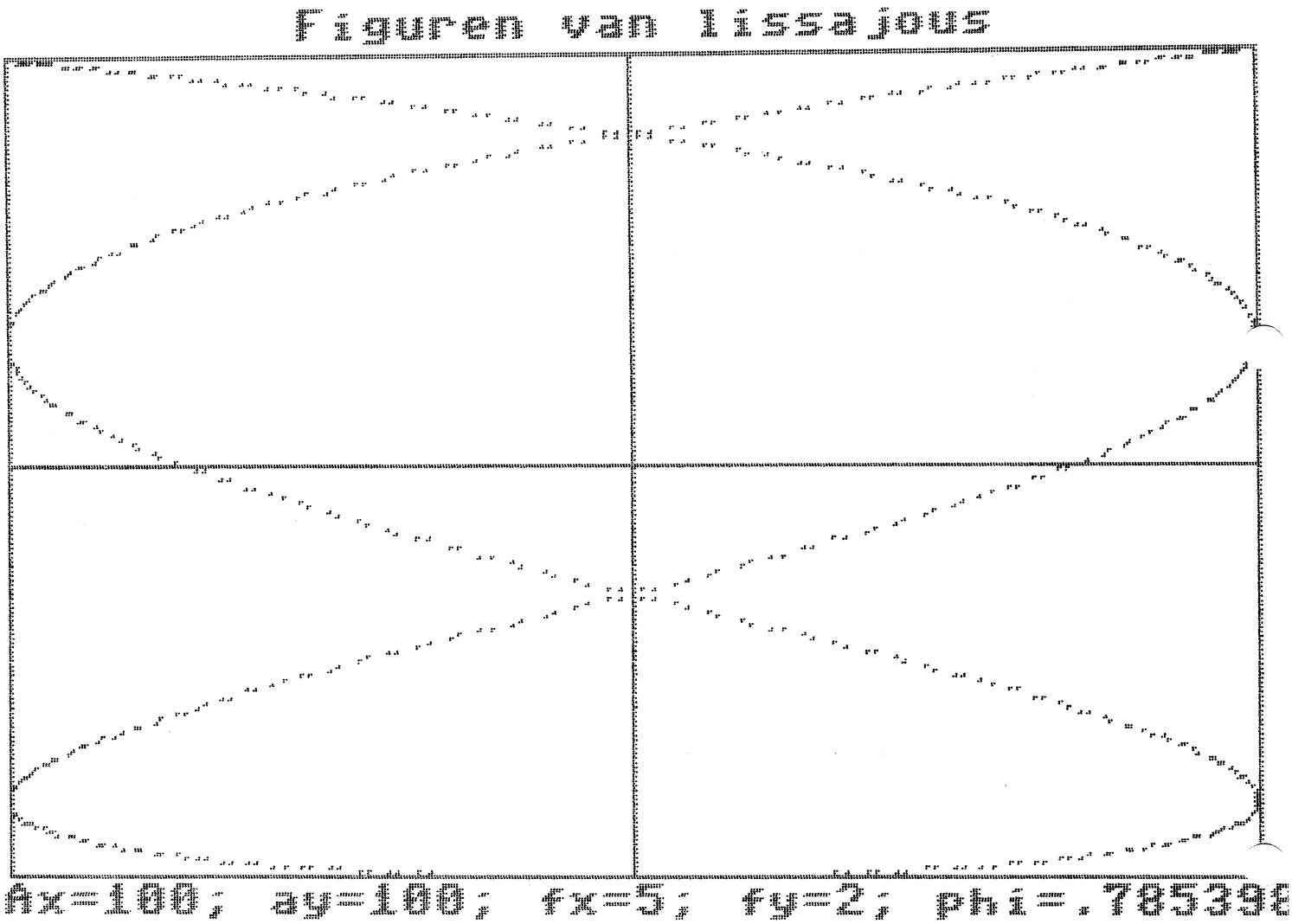
Let op: Omdat het programma gebruik maakt van het grafische scherm, is gekozen voor een programmering in WEKA-BASIC. Met kleine aanpassingen draait het programma ook onder Simons' Basic e.d.



21.2 Figuren van Lissajous

```
10 REM ------------------------------------------------------
20 REM                  FIGUREN VAN LISSAJOUS
30 REM ------------------------------------------------------
40 REM
50 REM DRAAIT ALLEEN ONDER WEKA-BASIC!
60 REM
70 CLS$=CHR$(147)
80 PRINTCLS$+CHR$(142)
90 PRINT"FIGUREN VAN LISSAJOUS"
100 PRINT
110 INPUT"X-AMPLITUDE (HOR.), MAX=100";AX
120 INPUT"Y-AMPLITUDE (VERT), MAX=100";AY
130 INPUT"FREQUENTIE VAN X            ";FX
140 INPUT"FREQUENTIE VAN Y            ";FY
150 INPUT"VERSCHIL IN FASE            ";FASE
160 INPUT"AANTAL X-TRILLINGSTIJDEN    ";AANTAL
170 PHI=FASE*π/180
180 HO=AX*1.5:VE=AY*.9
200 CLEAR
210 HIRES6,1
220 WRITE10,0,"FIGUREN VAN LISSAJOUS"
250 DRAW 160-HO,100-VE TO 160+HO,100-VE
260 DRAW 160-HO,100 TO 160+HO,100
270 DRAW 160-HO,100+VE TO 160+HO,100+VE
280 DRAW 160-HO,100-VE TO 160-HO,100+VE
290 DRAW 160,100-VE TO 160,100+VE
300 DRAW 160+HO,100-VE TO 160+HO,100+VE
310 REM
320 PU=360*AANTAL
330 FOR P=1 TO PU STEP 3
340 I=P*π/180
350 PLOT AX*SIN(I)*1.5+160,(AY*COS(FY/FX*I)+PHI)*.9+100
360 NEXT
370 S$="AX="+MID$(STR$(AX),2)+"; AY="+MID$(STR$(AY),2)+"; FX="+
                                                    MID$(STR$(FX),2)
380 S$=S$+"; FY="+MID$(STR$(FY),2)+"; PHI="+MID$(STR$(PHI),2)
390 WRITE1,24,S$
400 KEY

READY.
```

21.2 Figuren van Lissajous



7/21.3

Functieonderzoek

7/21.3.1
Hoofdprogramma

Probleembeschrijving

Dit programma verzorgt het onderzoeken (analyseren) van een functievergelijking. In principe kan een functie op alle willekeurige karakteristieke eigenschappen onderzocht worden. In de praktijk zal het neerkomen op het onderzoeken van de volgende eigenschappen:

- Nulpunten bepalen, het gelijk aan nul (0) stellen van de functie
- De extremen bekijken (minima en maxima)
- De afgeleiden (eerste en tweede) in een bepaald punt
- Bepaalde integraal (oppervlak tussen de kromme en de X-as)
- grafische weergave
- Waardetabellen

Het programma moet in staat zijn willekeurige functies te onderzoeken (zowel algebraïsche als transcendente functies).

Oplossingsmethode

Voor het oplossen van wiskundige vraagstukken bestaan er twee verschillende oplossingsmethoden: De analytische methode die een gesloten (exacte) oplossing geeft en een numerieke methode die met concrete getallen werkt en normaal gesproken een benadering oplevert. De laatste methode werkt meestal op een iteratieve manier. Dit houdt in dat de benadering stap voor stap verbeterd wordt. Het is mogelijk een grote nauwkeurigheid te behalen, echter wel vaak ten koste van veel rekentijd.

Aangezien het programma een willekeurige functie moet kunnen oplossen, wordt voor de numerieke methode gekozen. De analytische methode is bij gecompliceerde functies vaak niet bruikbaar.

De gebruiker heeft de keuze uit twee te volgen wegen:

1. Alle berekeningen worden aan het programma overgelaten.

2. Soms is het nodig dat een gesloten oplossing in de vorm van een vergelijking aanwezig is. In dat geval lost de gebruiker het probleem zelf op (zonder programma). Daar dit zeer veel werk met zich mee kan brengen, zijn fouten natuurlijk niet uitgesloten. Dan is het erg makkelijk om met het programma de zelf gevonden resultaten te controleren. Voorbeeld: Men wil beschikken over een onbepaalde integraal (niet gespecificeerde grenzen). Deze dient dan met de hand te worden gevonden. Zijn er rekenfouten gemaakt, dan komen

21.3 Functieonderzoek

deze naar voren bij een oppervlakteberekening (bepaalde integraal, gespecificeerde grenzen). Eerst wordt zelf een oppervlakte bepaald 'met de hand', en daarna wordt de berekening met behulp van het programma gecontroleerd.

Onderzoeken naar bovengenoemde eigenschappen (nulpunten, extremen enz.) zijn in de beroepspraktijk zeer vaak nodig. Het is dan normaal gesproken echter geen doel op zich, maar onderdeel van een groter geheel. Om die reden is de volgende opzet gekozen:

Voor elke eigenschap van een functie wordt een algemeen bruikbare subroutine gemaakt. De subroutines dienen zo ontworpen te worden dat ze zonder wijziging in een willekeurig gebruikersprogramma kunnen worden geplaatst.

Het hoofdprogramma dat in deze paragraaf behandeld wordt roept elke subroutine afzonderlijk aan. Hiertoe wordt op het scherm een menu getoond waaruit de gebruiker kan kiezen. Dit kiezen geschiedt door het intypen van het juiste nummer. Op deze manier kunnen de subroutines direct gebruikt worden zonder dat de gebruiker eerst zelf een hoofdprogramma dient te maken.

Algemeen bruikbare subroutines maken geen gebruik van PRINT-opdrachten. Er worden dus geen foutmeldingen op het scherm getoond. Daarentegen wordt alle informatie aan het aanroepende (hoofd)programma doorgegeven. Het aanroepende programma dient de informatie, indien nodig, op het scherm weer te geven. Ook het optreden van fouten dient aan het aanroepende programma te worden doorgegeven. Het hoofdprogramma dient hierop dan adequaat te reageren.

Bij het ontbreken van lokale variabelen dient een eigen strategie te worden gebruikt. In dit programma gebeurt dat door het reserveren van bepaalde namen van variabelen. REAL- en STRING-variabelen die alleen in een bepaalde subroutine voorkomen krijgen als beginletter een Q; Lokale integer-variabelen krijgen als beginletter een J. In het hoofdprogramma worden geen variabelen gebruikt waarvan de namen beginnen met een Q of een J. Op deze manier is zeker gesteld dat lokale variabelen het programmaverloop van het aanroepende (hoofd)programma niet kunnen beïnvloeden.

Bij de Commodore 64-BASIC zijn slechts de eerste twee tekens van variabelen-namen van belang. Om hiermee (enigszins) compatibel te zijn wordt hiermee in dit programma rekening gehouden. Het zal meestal niet voorkomen dat twee variabelen de eerste twee tekens hetzelfde zijn.

De beschreven strategie (namen van lokale variabelen laten beginnen met een Q of een J en namen bestaande uit twee tekens) creëren de mogelijkheid om de subroutines op verschillende plaatsen zonder wijzigingen in te zetten. Nadeel is echter dat de bedoeling van variabelen niet langer echt duidelijk is, met als gevolg dat de subroutines niet zo heel makkelijk te doorgronden zijn. Om enigszins voor dit gebrek te compenseren is er een royale uitleg per subroutine aanwezig.

Zoals reeds is vermeld, maken de subroutines gebruik van de interactieve methode. Dit houdt in dat de benadering steeds verbeterd wordt door het herhaald toepassen van dezelfde rekenmethodes/formules. De nieuw berekende waarde wordt aan de variabele W_{nieuw} toegekend. De variabele W_{oud} bevat de laatst berekende waarde. Een subroutine itereert (herhaalt de gewenste handeling) net

21.3 Functieonderzoek

zo lang tot de gewenste nauwkeurigheid is gehaald. Er zal nu worden uitgelegd wat onder nauwkeurigheid wordt verstaan. Hierbij wordt uitgegaan van een nauwkeurigheid van 0,0001. Wat betekent dit? Er wordt onderscheid gemaakt tussen twee soorten nauwkeurigheid: absolute en relatieve. De absolute nauwkeurigheid is bereikt, indien geldt:

$|W_{nieuw} - W_{oud}| < 0.0001$

Dit betekent dat het verschil tussen de huidige en vorige waarde zich pas na de vierde decimaal openbaart. Men kan er van uit gaan dat de eerste drie decimalen in orde zijn.

Bij de absolute nauwkeurigheid ontstaan er echter problemen. Als de gezochte waarde erg groot is (bijv. 100000) dan zijn er wel erg veel berekeningen nodig om de gewenste nauwkeurigheid te behalen. Zodoende is het beter om van de relatieve nauwkeurigheid gebruik te maken. Indien als voorbeeld weer van een nauwkeurigheid van 0,0001 wordt uitgegaan houdt dat in dat bij een benadering van 3951.274 de eerste drie cijfers (dus 3, 9 en 5) in orde zijn. Bij de relatieve nauwkeurigheid dient aan de volgende voorwaarde voldaan te zijn:

$|(W_{nieuw} - W_{oud})/W_{nieuw}| < 0{,}0001$

Voorbeeld:

W_{nieuw}=1235.67	W_{oud}=1234.56
abs. nauwkeurigheid:	1235.67 - 1234.56 = 1.11
rel. nauwkeurigheid:	1.11/1235.67 = 0.0009
proc. nauwkeurigheid:	0.0009·100 = 0.09%

In dit voorbeeld is de gewenste nauwkeurigheid nog niet helemaal bereikt.

In de subroutines wordt de gewenste nauwkeurigheid opgeslagen in de variabele QN (als vaste constante). QG bevat een waarde die voor een aantal BASIC-versies bruikbaar is, die met zes of zeven significante cijfers werken (zoals de Commodore 64-BASIC).

Het komt soms voor dat de gewenste nauwkeurigheid niet wordt gehaald. Daarom zijn in de subroutines twee extra testen aanwezig die kunnen zorgen voor een voortijdige beëindiging van de iteratie:

- Indien het aantal iteratiestappen een maximum heeft bereikt wordt de subroutine verlaten.
- Bij enkele subroutines (bijv. bij de numerieke integratie) neemt het aantal berekeningen bij elke iteratiestap aanzienlijk toe. Door het vele rekenwerk, kunnen afrondingsfouten hun invloed doen gelden. Dit houdt in dat de laatste cijfers van de benadering ten onrechte worden aangepast. Het is mogelijk dat op een gegeven moment de nieuwe waarde een minder goede benadering is dan de vorige waarde. Op dat moment wordt de iteratie in de subroutine onderbroken en wordt de op één na laatste waarde geretourneerd.

In beide gevallen meldt de subroutine (uiteraard) dat de gewenste nauwkeurigheid niet is gehaald.

Zoals reeds is vermeld, is het de bedoeling dat een willekeurige functie moet kunnen worden onderzocht. Zou het programma zijn toegespitst op een bepaald functie-type, dan zou het mogelijk zijn om door ingave van getallen de functie vast te leggen. Als het programma bijvoorbeeld alleen met polynomen zou moeten werken, dan zou het mogelijk zijn enkel de graad en de coëfficiënten in te voeren. Intern zou dan de functie kunnen worden opgebouwd. Echter in het algemene geval is het niet mogelijk om door

middel van ingevoerde getallen een functie vast te leggen. Daarentegen moet de functie bij regel 200 worden ingevoerd. Dit geschiedt door middel van de opdracht DEF FN (define function, definieer functie), gevolgd door de functie zelf. Hoe het precies moet wordt duidelijk in de voorbeelden. Zoals daar naar voren komt, is de te getroosten moeite slechts gering. Indien echter de opdracht DEF FN niet aanwezig is, moet de functie met behulp van een GOSUB worden verwezenlijkt. Om enigszins compatibel te blijven wordt er niet getracht de functie in machinetaal te schrijven.

Er is een groot aantal functies dat in bepaalde punten of bepaalde gebieden niet gedefinieerd zijn. Bijvoorbeeld, een logaritme is alleen te berekenen als x (de operand) groter is dan 0 (nul). Normaal gesproken is een foutmelding het resultaat, wanneer een gebruiker een gebied opgeeft waarin definitie problemen optreden. De gebruiker zal daar terdege rekening mee moeten houden, omdat het de 64 helaas aan een ON ERROR GOTO-constructie ontbreekt.

Aangezien voor het opstellen van de (waarde)tabellen slechts enkele programmaregels nodig zijn, is afgezien van het maken van een subroutine voor deze taak. De benodigde regels zijn in het hoofdprogramma geïmplementeerd.

Bij elke subroutine is verder een vrij gedetailleerd structuurdiagram aanwezig. Uit ruimte-overwegingen zijn de volgende 'inkortingen' toegepast:

Bij een enkele IF-test met alleen een THEN en dus geen ELSE, wordt niet de gebruikelijke tekening gegeven. Daarentegen wordt een regel in normaal nederlands geschreven, beginnend met 'Indien'.

Structuurdiagram
Het hoofdprogramma is vrij overzichtelijk opgezet. Eerst wordt een menu getoond waaruit de gebruiker door middel van een cijfer kan kiezen. Na een keuze te hebben gemaakt wordt om de benodigde gegevens gevraagd (bijv. de grenzen van het interval) en vervolgens wordt de juiste subroutine aangeroepen. Nadat de subroutine is uitgerekend verzorgt het hoofdprogramma het op het scherm zetten van de resultaten. De opbouw (van het hoofdprogramma) is zo duidelijk dat er geen structuurdiagram nodig is. Het is eenvoudiger om meteen de listing te bekijken.

Voorbeeld
Er wordt een functie gekozen die alle te kiezen eigenschappen bezit en die op een groot gedeelte niet gedefinieerd is (alle x-waarden moeten positief zijn).

$y = x \cdot \ln(x)$

Alle eigenschappen worden, in de volgorde van het menu, onderzocht. Steeds nadat de resultaten van een onderzoek (subroutine) op het scherm zijn getoond verschijnt op het scherm wederom het keuzemenu. Het spreekt voor zich dat dat hier slechts éénmaal wordt getoond. Bijzonderheden van de verschillende methoden worden in de voorbeelden van de diverse subroutines belicht.

Zoals duidelijk moge zijn is de juiste functie reeds bij regel 200 ingevoerd (bij Commodore 64-BASIC wordt de natuurlijke logaritme ln voorgesteld door LOG). Verandering van regel 200 vindt bij het voorbeeld in de volgende paragraaf plaats.

Op de pagina's hierna treft u wat voorbeelden en het hoofdprogramma aan. De subroutines volgen in een volgende aanvulling.

21.3 Functieonderzoek

```
100 REM ------------------------------------
105 REM
110 REM FUNCTIE-ANALYSE MET DE COMMODORE 64
115 REM
120 REM ------------------------------------
130 PRINT"■"
140 INPUT "■WILT U EEN NIEUWE FUNKTIE INVOEREN  N■■■";A$
150 PRINT
160 A$=LEFT$(A$,1):IF A$<>"J"THEN GOTO 200
170 PRINT"VOER FUNCTIE ALS VOLGT IN EN GEEF RUN:■■■■"
175 PRINT"200 DEF FN F(X)=...■■■■■"
180 LIST 200
190 END
200 DEF FN F(X)=LOG(X)
210 MAX=40:REM MAXIMUM AANTAL EXTREMEN
220 DIM WAARDE(MAX),BWAARDE(MAX)
230 FF$=CHR$(13)+">>>    F O U T ! !    <<<"+CHR$(13)
240 PRINT"■■■   MENU FUNKTIEANALYSE■"
241 PRINT"    1:  NULPUNTEN"
242 PRINT"    2:  EXTREMEN"
243 PRINT"    3:  AFGELEIDEN"
244 PRINT"    4:  BEPAALDE INTEGRAAL"
245 PRINT"    5:  TEKENING"
246 PRINT"    6:  WAARDETABEL"
247 PRINT"    7:  EINDE"
250 PRINT:PRINT
260 PRINT"MAAK UW KEUZE: ";
265 GET A$:IF A$<"1" OR A$>"7" THEN 265
270 PRINTA$:KZ=VAL(A$)
280 ON KZ GOTO 300,400,500,600,700,800,900
290 GOTO 240
300 REM ------------------------------------
305 REM NULPUNTEN BEREKENEN
310 REM ------------------------------------
320 PRINT"■■NULPUNTENBEREKENING M.B.V. REGULA FALSI":PRINT
330 GOSUB 950     : REM INVOEREN INTERVAL
340 GOSUB 21300   : REM SUBROUTINE NULPUNTEN
350 IF NW=0 THEN PRINT"TUSSEN"XL"EN"XR"ZIJN GEEN NULPUNTEN
                                          GEVONDEN":GOTO 395
355 PRINT"NULPUNTEN (SNIJPUNTEN MET X-AS):":PRINT
360 FOR IW=1 TO NW:PRINT"X="WA(IW);TAB(20)"Y="FNF(WA(IW)):NEXT
395 INPUT"DRUK OP RETURN...";W$
397 GOTO 240
400 REM ------------------------------------
405 REM EXTREMEN BEREKENEN
410 REM ------------------------------------
420 PRINT"■■BEREKENING VAN EXTREMEN":PRINT
430 GOSUB 950     : REM INVOEREN INTERVAL
440 GOSUB 22300   : REM SUBROUTINE EXTREMEN
```

21.3 Functieonderzoek

```
 450 IF NW=0 THEN PRINT"TUSSEN"XL"EN"XR"ZIJN GEEN NULPUNTEN
                                                  GEVONDEN":GOTO 495
455 PRINT"EXTREMEN":PRINT
460 FOR IW=1 TO NW:PRINT"X="WA(IW);TAB(20)"Y="FNF(WA(IW));
465 IF BW(IW)=1 THEN PRINT TAB(34);"(MAX)"
470 IF BW(IW)<>1 THEN PRINT TAB(34);"(MIN)"
475 NEXT
495 INPUT"DRUK OP RETURN...";W$
497 GOTO 240
500 REM ------------------------------------
505 REM AFGELEIDEN (DIFFERENTIEREN)
510 REM ------------------------------------
520 PRINT"■■BEREKENING VAN 1E EN 2E AFGELEIDE":PRINT
530 PRINT"DE AFGELEIDE WORDT IN EEN PUNT BEREKEND":PRINT
535 INPUT"GEEF HET PUNT X";XO:PRINT
540 GOSUB 23300  : REM SUBROUTINE AFGELEIDE
550 IF NW=0 THEN PRINT"DE AFGELEIDE IS NIET TE BEREKENEN":GOTO 595
555 PRINT"Y  ="YO:PRINT"Y' ="Y1;
560 IFK1=1 THEN PRINT"(EVT. ONNAUWKEURIG)";
565 IF(K1<>1)AND(K1<>0)THEN PRINT"(EVT. FOUT)";
570 PRINT:IF K2=2 THEN 595:REM IF K1=2 (??)
575 PRINT"Y''="Y2;
580 IFK2<>0 THEN PRINT"(EVT. ONNAUWKEURIG)";
595 PRINT:INPUT"DRUK OP RETURN...";W$
597 GOTO 240
600 REM ------------------------------------
605 REM INTEGREREN
610 REM ------------------------------------
620 PRINT"■■BEREKENING VAN BEPAALDE INTEGRAAL":PRINT
630 GOSUB 950    : REM INVOEREN INTERVAL
640 GOSUB 24300  : REM SUBROUTINE EXTREMEN
650 IF NW=0 THEN PRINT"DE INTEGRAAL IS NIET TE BEREKENEN":GOTO 695
655 PRINT"INTEGRAALWAARDE:";WA;
660 IFKW<>1 THEN PRINT"(EVT. ONNAUWKEURIG)"
665 IFKW=1 THEN PRINT
695 INPUT"DRUK OP RETURN...";W$
697 GOTO 240
700 REM ------------------------------------
705 REM TEKENEN (KROMME, GRAFIEK)
710 REM ------------------------------------
720 PRINT"■■GRAFISCHE VOORSTELLING VAN DE FUNCTIE":PRINT
730 GOSUB 950    : REM INVOEREN INTERVAL
735 PRINT:PRINT"NORMAAL WORDT HET HELE SCHERM GEBRUIKT. OF WILT
                                                  U EEN GELIJKE ";
740 INPUT"X/Y VERDELING  N■■■■";A$:K1=1:A$=LEFT$(A$,1):IF A$="J"
                                                  THEN K1=2
745 YZ=20:Y=0:PRINT
750 IF K1=2 THEN 765
755 PRINT"Y-WAARDEN GROTER DAN (ABSOLUUT) 20 WOR- DEN NIET
                                          AFGEDRUKT. GEEF EEN";
```

21.3 Functieonderzoek

```
760 INPUT" EVENTUELE  ANDERE GRENS";Y:PRINT:IF Y<>0 THEN YZ=Y
765 PRINT"■GEEF HET TYPE VAN DE GRAFIEK:■"
766 PRINT"  1:   PUNTEN"
767 PRINT"  2:   LIJNEN"
768 PRINT"  3:   DICHT"
770 GET A$:IF A$<"1" OR A$>"3" THEN 770
775 K2=VAL(A$)
780 GOSUB 25300
790 GOTO240
800 REM ------------------------------------
805 REM FUNCTIEWAARDENTABEL
810 REM ------------------------------------
820 PRINT"■TABEL MET FUNCTIEWAARDEN":PRINT
830 GOSUB 950     : REM INVOEREN INTERVAL
835 INPUT"STAPGROOTTE";XS:IFXS<=0 THEN 835
840 NZ=23:IZ=0:REM AANTAL REGELS-2; REGELTELLER
845 PRINT
850 FOR X = XL TO (XR*1.0001) STEP XS
855 IZ=IZ+1:Y=FNF(X):PRINT"X="X;TAB(20)"Y="Y
860 IF IZ>NZ YTHEN IZ=0:INPUT"     DRUK OP RETURN...";W$
870 IFW$="E" THEN X=1.0002*XR
875 NEXT
880 PRINT:PRINT:INPUT"EINDE TABEL. DRUK OP RETURN";W$
890 GOTO240
900 PRINT:END
950 PRINT"■BEREIK:■"
960 INPUT"LINKER X-WAARDE";XL
970 INPUT"RECHTER X-WAARDE";XR
980 IF XR<=XL THEN PRINTFF$:GOTO950
990 RETURN


READY.
```

21.3 Functieonderzoek

```
wilt u een nieuwe funktie invoeren  n
VOER FUNCTIE ALS VOLGT IN EN GEEF RUN:
200 DEF FN F(X)=...

200 DEF FN F(X)=LOG(X)


READY.


RUN


   MENU FUNKTIEANALYSE

   1:  NULPUNTEN
   2:  EXTREMEN
   3:  AFGELEIDEN
   4:  BEPAALDE INTEGRAAL
   5:  TEKENING
   6:  WAARDETABEL
   7:  EINDE


MAAK UW KEUZE:


TABEL MET FUNCTIEWAARDEN


BEREIK:

LINKER X-WAARDE 2
RECHTER X-WAARDE 5
STAPGROOTTE .2

X= 2                        Y= .693147181
X= 2.2                        Y= .788457361
X= 2.4                        Y= .875468738
X= 2.6                        Y= .955511446
X= 2.8                        Y= 1.02961942
...
X= 4.4                        Y= 1.48160454
X= 4.6                        Y= 1.5260563
X= 4.8                        Y= 1.56861592
X= 5                        Y= 1.60943791


EINDE TABEL.  DRUK OP RETURN
```

7/21.3.2
Subroutine: Het bepalen van de nulpunten

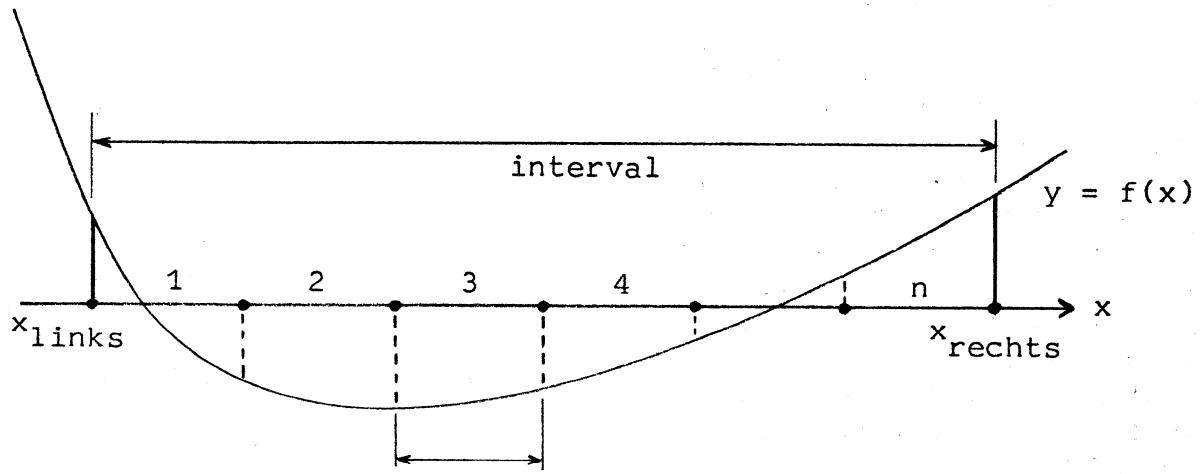
Onder een nulpunt wordt verstaan elke x-waarde met een bijbehorende functiewaarde gelijk aan 0 (nul). Op dat punt snijdt of raakt de functie de x-as.

In de praktijk is het zeer vaak nodig de nulpunten van een functie te kunnen berekenen. Vaak zal het echter een onderdeel zijn van andere doelstellingen. Bijvoorbeeld als de extremen (hoogste en laagste waarden) van een functie berekend moeten worden, is het zaak de nulpunten van de eerste afgeleide te berekenen. Het snijpunt van twee functies wordt verkregen door het gelijkstellen van beide functies: f(x)=g(x). Hetgeen neerkomt op f(x)-g(x)=0. Dit kan op de volgende manier geïnterpreteerd worden: het zoeken van de nulpunten van y=f(x)-g(x).

De subroutine 'nulpunten' moet in staat zijn de nulpunten van een willekeurige functie te bepalen. Dat houdt in dat het mogelijk moet zijn van zowel algebraïsche vergelijkingen als van transcendente vergelijkingen (vergelijkingen die niet analytisch zijn op te lossen), de nulpunten te bepalen.

Voor een nulpunt geldt dat de y-waarde nul is. Dat houdt in dat in de functievergelijking y=f(x) de waarde van y door nul kan worden vervangen. Op deze manier ontstaat er een zogenaamde bepaalde vergelijking van de vorm f(x)=0; hierbij is x de onbekende. Hiervoor moeten dan oplossingen worden gevonden, dus de waarde van de onbekende x.

De gebruiker wordt om een bereik (interval) gevraagd waarbinnen de nulpunten dienen te worden gevonden. Er dienen dus twee punten te worden gegeven, één voor de linkergrens en één voor de rechtergrens. Vervolgens gaat het programma aan de slag om de positie van het meest links gelegen nulpunt te benaderen. Hiertoe wordt het interval in kleine stukjes opgesplitst (deelintervallen). Deze deelintervallen krijgen de nummers 1 tot en met n:



21.3 Functieonderzoek

De subroutine werkt de deelintervallen van links naar rechts af. Op het moment dat geconstateerd wordt dat een deelinterval een nulpunt heeft, wordt het nulpunt met behulp van de 'regula falsi' methode benaderd. De uitleg van deze methode volgt later in dit betoog. Is het nulpunt bepaald, dan wordt met het volgende deelinterval verder gegaan, net zolang totdat alle deelintervallen zijn behandeld.

De linkergrens van een deelinterval wordt aangegeven met a, de rechtergrens met b. Het huidige deelinterval bevat een nulpunt als de waarden (van de functies) f(a) en f(b) een verschillend teken hebben. Dit kan worden getest door één van de volgende twee regels:

Indien $f(a) \cdot f(b) \leq 0 \rightarrow$ Nulpunt aanwezig.

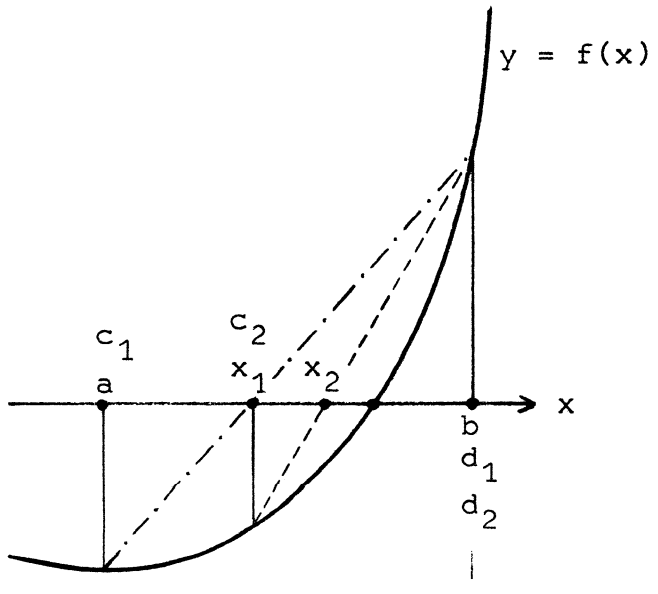
Indien $\text{sgn } f(a) \neq \text{sgn } f(b) \rightarrow$ Nulpunt aanwezig.

Het is echter wel noodzakelijk dat een deelinterval zo klein is dat er maximaal één nulpunt aanwezig is. Indien er namelijk twee nulpunten aanwezig zijn, wordt met bovenstaande formules geconstateerd dat er geen nulpunt aanwezig is. In het geval van één nulpunt is y van teken veranderd. Het feit dat y van teken is veranderd houdt in dat y op een gegeven moment de waarde nul heeft aangenomen. Zijn er twee nulpunten en was y bij a bijvoorbeeld positief, dan is y na het eerste nulpunt negatief. Na passeren van het tweede nulpunt echter, is y weer positief geworden, zodat f(a) en f(b) niet langer van teken verschillen, met als ongewenste (foute) conclusie dat er geen nulpunt in het huidige deelinterval aanwezig is.

In het onderhavige programma krijgt een deelinterval standaard de waarde 0.1, maar deze waarde kan natuurlijk naar believen gewijzigd worden. Het is aan te raden de waarde te verkleinen en een langere rekentijd voor lief te nemen. Op deze manier ontstaat er namelijk een grotere kans dat alle gewenste nulpunten gevonden worden.

Bovengenoemde methode heeft echter een nadeel. Daar in feite alleen naar snijpunten wordt gezocht, worden raakpunten niet gevonden, tenzij zo'n raakpunt toevallig op één van de grenzen ligt. Bij het berekenen van de extremen kan zo'n raakpunt echter wel naar voren komen.

Op het moment dat geconstateerd wordt dat een deelinterval een nulpunt bevat wordt van de 'regula falsi' methode gebruikt gemaakt. Deze methode staat ook wel bekend als secansmethode (secans is 1/cos) of interpolatie methode. De methode werkt als volgt: in het huidige deelinterval wordt de kromme door een rechte lijn benaderd (in de tekening aangegeven door de streep-stippellijn):



21.3 Functieonderzoek

Het snijpunt van de rechte lijn met de x-as wordt aangegeven met x1. Doordat f(a) en f(b) een verschillend teken hebben geldt ook dat de rechte lijn een nulpunt in het huidige deelinterval heeft. Het snijpunt x1 is een eerste benadering van het gezochte nulpunt van de functie. Vervolgens wordt de breedte van het deelinterval verkleind (de grenzen worden naar binnen getrokken). Om echter de waarden a en b niet aan te tasten worden er twee nieuwe variabelen c en d gebruikt. Bij het begin heeft c1 de waarde van a en d1 de waarde van b. Nu wordt het deelinterval in twee stukken verdeeld. Het eerste stuk loopt van c1 tot x1 en het tweede stuk van x1 tot d1. Deze beide stukken worden weer als deelinterval beschouwd. Met het deelinterval dat het nulpunt bevat wordt verder gewerkt. Aangezien in de figuur het tweede (rechtse) stuk het nulpunt bevat krijgt c2 de waarde van x1. In dit deelinterval wordt de curve wederom door een rechte lijn benaderd. Het gaat nu om de rechte door f(c2) en f(d2). De waarde van d verandert niet, dus d1 = d2. Het snijpunt van de nieuwe rechte lijn met wordt aangeduid met x2. Het snijpunt x2 is dan weer een verbeterde benadering van het gezochte nulpunt. Dit gaat net zo lang door totdat (indien mogelijk, hier dus wel) de gewenst nauwkeurigheid is bereikt. Elke volgende stap/iteratie is een betere benadering – van het gezochte nulpunt – dan de vorige. Het resultaat wordt in een variabeleveld geplaatst. Dit variabeleveld wordt later ter 'beschikking' gesteld aan het aanroepende programma. Vervolgens wordt het volgende deelinterval behandeld, indien voorhanden.

De kromme in een deelinterval wordt benaderd door een rechte. Het snijpunt van deze rechte met de x-as kan aan de hand van één van beide volgende formules worden berekend (lineaire interpolatie):

$$x_i = d_i - f(d_i) \cdot \frac{d_i - c_i}{f(d_i) - f(c_i)}$$

$$x_i = c_i - f(c_i) \cdot \frac{d_i - c_i}{f(d_i) - f(c_i)}$$

In het programma wordt geen gebruik gemaakt van een teller i. Voor de grenzen worden, na elke verschuiving/verplaatsing, steeds weer dezelfde variabelen gebruikt.

Aangezien het om een algemeen bruikbare subroutine gaat beginnen de namen van lokale variabelen met een Q of een J (zie vorige paragraaf). De variabelen hebben de volgende betekenis:

XL:linkergrens van het totale interval.

XR:rechtergrens van het totale interval.

NW:aantal gevonden nulpunten.

MAX:maximum aantal toegestane nulpunten (bijv. MAX=40 houdt in dat er maximaal 40 nulpunten in het veld waarde opgeslagen kunnen worden).

BWtest (boolean) waarde die aangeeft of er meer dan MAX nulpunten gevonden zijn. (0=nee, 1=ja).

JF:a) geeft aan of er bij de berekening een fout is opgetreden (0=nee, 1=ja).

b) geeft aan of het volgende deelinterval overgeslagen moet worden (0=nee, 1=ja).

QH:breedte van een deelinterval: benaming in fig.: h

QN:gewenste nauwkeurigheid, bijv. 0,0001.

21.3 Functieonderzoek

QA:linker x-waarde van het huidige deelinterval. In fig.: a

QB:rechter x-waarde van het huidige deelinterval. In fig.: b

Q1:f(QA), d.w.z. de functiewaarde bij x=QA; in fig.:f(a).

Q2:f(QB), d.w.z. de functiewaarde bij x=QB; in fig. :f(b).

JN:aantal (n) deelintervallen.

WAARDE(i):i-de nulpunt (x-waarde).

QC:linker x-waarde van het huidige deelinterval; in fig.: c

QD:rechter x-waarde van het huidige deelinterval; in fig.: d.

Q3:f(QC), d.w.z. de functiewaarde bij x=QC; in fig.:f(c).

Q4:f(QD), d.w.z. de functiewaarde bij x=QD; in fig. :f(d).

QX:resultaat van de lineaire interpolatie; in fig.: x.

Q5:f(QX), d.w.z. de functiewaarde bij x=QX.

Toelichting op de nauwkeurigheid
Er zijn twee gevallen te onderscheiden:

- Bij een vlakke kromme is de waarde van Q5 al vrij klein, terwijl de x-waarde nog relatief onbekend is
- Bij een zeer steile kromme is de waarde van Q5 nog vrij groot, wanneer de x-waarde al vrij precies bekend is

Om in die gevallen toch een indruk van de nauwkeurigheid te krijgen geeft het hoofdprogramma niet alleen de nulpunten maar ook de bijbehorende functiewaarden weer (zie voorbeeld)

Voorbeeld
We gaan nu uit van een functie, waarvan de resultaten/antwoorden makkelijk te controleren zijn. We kiezen:

$y=\sin(5x)$

De x-waarden van de nulpunten zijn:

$0,\pm\frac{1}{5}\cdot\pi,\pm\frac{2}{5}\cdot\pi,\pm\frac{3}{5}\cdot\pi,\pm\frac{4}{5}\cdot\pi,\pm\pi,\ldots$

Natuurlijk kunnen ook functievergelijkingen gebruikt worden, die gecompliceerder zijn en waarvan de waarden niet met de hand te bepalen zijn. Maar aangezien het hier een voorbeeld betreft is het nuttig om de resultaten met de 'echte' waarden te kunnen vergelijken.

De in de vorige paragraaf gebruikte vergelijking dient nu in regel 200 vervangen te worden door de nieuwe functie. Hoe dat in zijn werk gaat wordt nu één keer getoond en wordt in het vervolg achterwege gelaten. Daarna wordt het keuzemenu getoond, waarvan de weergave hier ook achterwege blijft.

21.3 Functieonderzoek

```
wilt u een nieuwe funktie invoeren  n
VOER FUNCTIE ALS VOLGT IN EN GEEF RUN:
200 DEF FN F(X)=...

200 DEF FN F(X)=5*EXP(-.2*X)*SIN(X)


READY.



200 DEF FN F(X)=SIN(5*X)


   menu funktieanalyse
   1:  NULPUNTEN
   2:  EXTREMEN
   3:  AFGELEIDEN
   4:  BEPAALDE INTEGRAAL
   5:  TEKENING
   6:  WAARDETABEL
   7:  EINDE


MAAK UW KEUZE:



bereik:
LINKER X-WAARDE-3.2
RECHTER X-WAARDE 3.2
NULPUNTEN (SNIJPUNTEN MET X-AS):

X=-3.14159142                    Y=-6.1735143E-06
X=-2.51327387                    Y= 1.28444207E-06
X=-1.88495594                    Y= 1.74672419E-06
X=-1.25663813                    Y=-5.36013184E-06
X=-.628317307                    Y=-6.11572903E-06
X=-3.31783667E-09                    Y=-1.65835479E-08
X= .628317307                    Y= 6.11572903E-06
X= 1.25663813                    Y= 5.36013184E-06
X= 1.88495594                    Y=-1.74672419E-06
X= 2.51327387                    Y=-1.2785904E-06
X= 3.14159142                    Y= 6.16181095E-06
DRUK OP RETURN...
```

21.3 Functieonderzoek

```
21000 REM ------------------------------------------------------
21010 REM            SUBROUTINE : NULPUNTEN
21020 REM ------------------------------------------------------
21290 :
21300 NW=0:KW=0:JF=0
21310 QH=.1:QG=0.0001
21320 QA=XL:QB=XL
21330 JN=(XR-XL)/QH+1
21340 :
21350 REM LUS VOOR ALLE DELEN
21360 :
21370 FOR J=1 TO JN
21380 QB=QB+QH
21390 IF J=1 THEN Q1=FN F(QA)
21400 Q2=FN F(QB)
21410 IF ABS(Q2)<=QG THEN NW=NW+1:IF NW<=MAX THEN WA(NW)=QB
21420 IF ABS(Q2)<=QG OR ABS(Q2)>1E3 THEN QB=QB+QH:J=J+1:JF=1:
21430 IF JF=1 OR SGN(Q1)=SGN(Q2) THEN 21610                Q2=FN F(QB)
21440 :
21450 REM "REGULA FALSI"
21460 QC=QA:Q3=Q1
21470 QD=QB:Q4=Q2
21480 :
21490 REM BEGIN VAN DE HERHALINGSLUS
21500 QX=QC-Q3*(QD-QC)/(Q4-Q3)
21510 Q5=FN F(QX): IF ABS(Q5)>1E3 THEN 21620
21520 IF SGN(Q5) = SGN(Q3) THEN QC=QX:Q3=Q5:GOTO 21530
21525 QD=QX:Q4=Q5
21530 REM EINDE HERHALINGSLUS
21540 :
21550 IF ABS(Q5) > QG AND ABS(QD-QC)>QG THEN 21500
21560 :
21570 IF ABS(Q5)<0.1 THEN NW=NW+1:IF NW<=MAX THEN WA(NW)=QX
21580 REM INDIEN ABS(Q5)>.1: GEEN NULPUNT
21590 REM EINDE "REGULA FALSI"
21600 :
21610 JF=0:IF NW>MA THEN KW=1:J=JN
21620 QA=QB:Q1=Q2
21630 NEXT
21640 :
21650 :
21660 RETURN
21670 :
```

7/21.3.3
Subroutine: Extremen

Probleembeschrijving
De subroutine 'Extremen' dient alle extreme waarden (hoogste/laagste waarden) van de functie binnen een bepaald gebied te vinden. Hierbij dient niet alleen naar de absolute extreme waarden gezocht te worden (dus de grootste en kleinste waarde van de functie binnen het bepaalde gebied), maar ook de relatieve extreme waarden. Er is een lokaal (of relatief) maximum, indien een monotone stijging van de functie gevolgd wordt door een monotone daling (eerst nemen de functiewaarden toe bij toenemende x, daarna nemen de functiewaarden af bij toenemende x). Een lokaal minimum is het tegenovergestelde, dus een monotone daling gevolg door een monotone stijging. De bepaling van een lokaal extreem hangt dus alleen af van de directe omgeving. Het is dus mogelijk dat een lokaal maximum 'dieper' (kleinere y-waarde) ligt dan een lokaal minimum. In het vervolg wordt de term relatief of lokaal weggelaten.

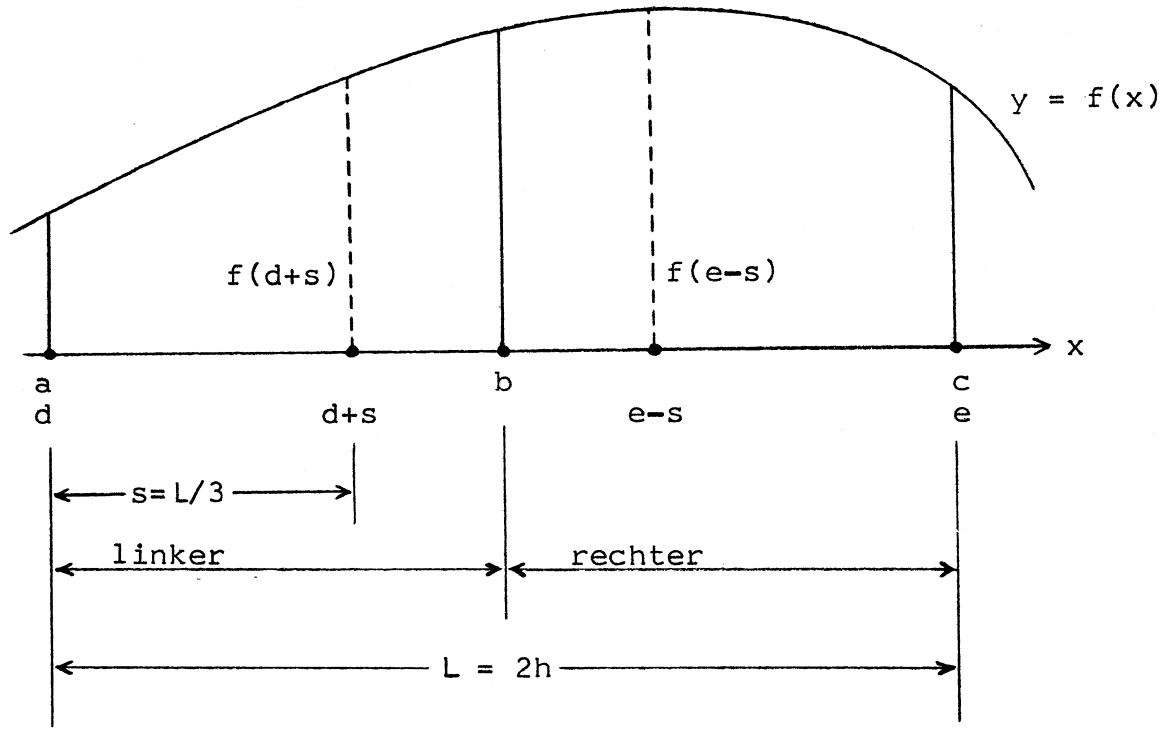
Methode voor het bepalen van extremen
Doordat in een extreem de raaklijn horizontaal loopt, kan een extreem worden bepaald door de afgeleide gelijk aan 0 (nul) te stellen. Indien de gebruiker zelf (analytisch) de afgeleide van de gewenste functie kan bepalen, kan deze afgeleide in regel 200 als de nieuwe functie worden gebruikt. Vervolgens kunnen dan de nulpunten van de afgeleide worden bepaald. Deze subroutine gaat er echter van uit dat er geen afgeleide voorhanden is.

Het programma gaat als volgt te werk: de gebruiker moet een bereik opgeven waarbinnen de extremen dienen te worden gevonden. Om vervolgens de extremen te kunnen bepalen wordt het totale bereik ook hier in 'kleine' deelintervallen opgesplitst. De deelintervallen worden voorzien van de nummers 1 tot en met n.

Ook hier werkt het programma de deelintervallen van links naar rechts af. Hierbij worden echter het huidige en het volgende deelinterval samengenomen tot een dubbeldeelinterval. Dit in tegenstelling tot de vorige paragraaf waar steeds elk deelinterval afzonderlijk wordt bekeken. De lus (de iteratie) begint dus bij het tweede deelinterval. Hierbij worden dan de coördinaten van het eerste en tweede deelinterval gebruikt. Bij het derde deelinterval worden het tweede en derde deelinterval gebruikt. Er zijn dus de volgende dubbele deelintervallen: 1/2, 2/3, 3/4, 4/5, 5/6, ... Elk dubbeldeel bevat dus drie coördinaten: een linker plaats (a), een middenpositie (b) en een rechter plaats (c). Een dubbeldeel bevat een maximum als f(a) en f(c) beide kleiner zijn dan f(b). Let er wederom op dat zo'n dubbel deelinterval voldoende klein is. Een minimum is aanwezig, indien geldt dat zowel f(a) als f(c) groter zijn dan f(b). Indien geconstateerd wordt dat een dubbeldeel een extreem bevat, wordt, door het herhaald verkorten het dubbeldeel met een derde deel, ingezoemd op het extreem. Nadat de positie en de waarde van een extreem is bepaald, wordt overgegaan naar het volgende dubbele deelinterval. Zo wordt het gehele gebied afgelopen.

Als voorbeeld is deelinterval 3 eerst het rechterdeel van het tweede dubbeldeel en daarna het linker deel van het derde dubbeldeel. Dit is gedaan om ook extremen die zich op de deelintervalgrenzen bevinden te kunnen opsporen. Er is echter nog een andere belangrijke reden. Sommige extremen kunnen alleen gelokaliseerd worden indien het bijbehorende deelinterval óf het rechterdeel óf het linkerdeel van een dubbeldeel is.

21.3 Functieonderzoek



De breedte van een dubbeldeel moet steeds zo klein zijn dat er steeds maar één extreem aanwezig is. Een deelinterval moet echter ook niet te klein zijn, vanwege de horizontale afgeleide. In de buurt van het extreem hebben de y-waarden (functiewaarden) ongeveer dezelfde waarden. Bijvoorbeeld: bij een normale nauwkeurigheid heeft zowel COS(3.141) als COS(1.142) de waarde -1.00000

Het vaststellen van het extreem gaat door middel van het herhaald in drieën delen van het dubbelinterval. In de tekening hierboven begint het dubbeldeel bij x=a en eindigt het bij x=c en heeft het de lengte van L=2h. Opdat de waarde van a en c niet aangepast worden, wordt gebruik gemaakt van de variabelen d en e. Bij het begin heeft d de waarde a en e de waarde c. Nu wordt het dubbeldeel in drieën gedeeld, waarbij elk deel dezelfde lengte s=L/3 heeft. De beide nieuwe middelpunten liggen op x=d+s en x=e-s. Nu wordt bepaald of f(d+s) kleiner is dan f(e-s). Is dat het geval, dan ligt maximum tussen d+s en e. In dit geval wordt d over een afstand s naar rechts geschoven. ($d_{nieuw}=d_{oud}+s$). Is dit niet het geval dan ligt het maximum tussen d_{oud} en e-s. in dat geval wordt e over een afstand s naar links verschoven ($e_{nieuw}=e_{oud}+s$). De grenzen van het met een derde deel verkorte dubbeldeel worden wederom aangegeven met d en e. Het nieuwe (dubbele) deelinterval (van d tot e) wordt ook weer in drieën opgedeeld. Dit gaat net zo lang door tot de afstand tussen d en e kleiner is dan de gewenste nauwkeurigheid (zie listing).

21.3 Functieonderzoek

In het geval van een minimum wordt bekeken of f(d+s) groter is dan f(e-s). Verder wordt dezelfde methode gehanteerd als bij een maximum.

Variabelen
Aangezien het om een algemeen bruikbare subroutine gaat, beginnen de namen van lokale variabelen met een Q of een J (zie de eerste paragraaf). De variabelen hebben de volgende betekenis:

XL:linkergrens van het totale interval.

XR:rechtergrens van het totale interval.

NW:aantal gevonden nulpunten.

MAX:maximamum aantal toegestane extremen (bijv. MAX=40 houdt in dat er maximaal 40 extremen in het veld waarde opgeslagen kunnen worden).

BWtest (boolean) waarde die aangeeft of er meer dan MAX extremen gevonden zijn. (0=nee, 1=ja).

JF:a) geeft aan of er bij de berekening een fout is opgetreden (0=nee, 1=ja).

b) geeft aan welke deelintervallen overgeslagen moeten worden (0=geen, 1=de volgende, 2=de volgende en daaropvolgende).

QH:breedte van een deelinterval: benaming in fig.: h

QN:gewenste nauwkeurigheid, bijv. 0,0001.

QA:linker x-waarde van het vorige deelinterval. In fig.: a

QB:linker x-waarde van het huidige deelinterval. In fig.: b

QC:rechter x-waarde van het huidige deelinterval. In fig.: c

Q1:f(QA), d.w.z. de functiewaarde bij x=QA; in fig.:f(a).

Q2:f(QB), d.w.z. de functiewaarde bij x=QB; in fig. :f(b).

Q3:f(QC), d.w.z. de functiewaarde bij x=QC; in fig. :f(c).

JN:aantal (n) deelintervallen.

JM:test (boolean) variabele die aangeeft of in het huidige dubbele deelinterval een maximum of een minimum wordt gezocht (1=maximum, 2=minimum)

WAARDE(i):i-de nulpunt (x-waarde).

BW(i):test (boolean) waarde die aangeeft of het i-de extreem een maximum of een minimum is (1=maximum, 2=minimum)

QD:linker x-waarde van het huidige deel van het dubbeldeel; in fig.: d

QE:rechter x-waarde van het huidige deel van het dubbeldeel; in fig.: e

Q4:f(QD), d.w.z. de functiewaarde bij x=QD; in fig.:f(d).

Q5:f(QE), d.w.z. de functiewaarde bij x=QE; in fig. :f(e).

QL:lengte van het huidige deel van het dubbeldeel; in fig.: L (beginwaarde L=2h).

QS:QL/3; in fig.: s=L/3

Q6:f(QD+QS), d.w.z. de functiewaarde bij x=QD+QS; in fig.: f(d+s)

21.3 Functieonderzoek

Q7:f(QE-QS), d.w.z de functiewaarde bij x=QE-QS; in fig.: f(e-s)

QX:de benaderde positie van het extreem, het middelpunt tussen QD en QE

Voorbeeld
Gevraagd: Bepaal de extremen van de functie

$y=f(x)=5e^{-0,2x}\sin x$

tussen x=0 en x=10

De antwoorden worden als volgt gecontroleerd: Eerst wordt de afgeleide van de functie (analytisch) bepaald. Deze is:

$y' = f'(x)= 5e^{-0,2x}(-0,2\sin x + \cos x)$

Om de nulpunten van de afgeleide te bepalen is het voldoende om slechts het tweede gedeelte bij regel 200 in te voeren. De reden hiervoor is dat een produkt de waarde nul heeft, indien minstens één van de factoren nul is en de eerste factor ($5e^{-0,2x}$) is steeds ongelijk aan nul. Vervolgens worden de extremen dan bepaald door gebruik te maken van de subroutine nulpunten.

```
200 DEF FN F(X)=5*EXP(-0.2*X)*SIN(X)

READY.




bereik:
LINKER X-WAARDE 0
RECHTER X-WAARDE 10
EXTREMEN

X= 1.37340662                    Y= 3.72530006
  (MAX)
X= 4.51498386                    Y=-1.98740322
  (MIN)
X= 7.65659893                    Y= 1.06025595
  (MAX)



22000 REM ------------------------------------------------
22010 REM          SUBROUTINE : EXTREMEN
22020 REM ------------------------------------------------
22290 :
```

21.3 Functieonderzoek

```
22300 NW=0:KW=0:JF=0
22310 QH=0.1:QG=0.0001
22320 QA=XL:QB=QA+QH:QC=QB
22330 JN=(XR-XL)/QH+1
22340 :
22350 :
22360 REM LUS VOOR 2..N
22370 FOR J=2 TO JN
22380 QC=QC+QH
22390 :
22400 IF J=2 THEN Q1=FN F(QA):Q2=FN F(QB)
22410 Q3=FN F(QC)
22420 IF JF>0 THEN 22700
22430 IF ABS(Q1)>1E4 OR ABS(Q2)>1E4 OR ABS(Q3)>1E4 THEN 22700
22440 IF Q3>=Q2 AND Q2>=Q1 THEN 22700
22450 IF Q3<=Q2 AND Q2<=Q1 THEN 22700
22460 JM=2:IF Q2>Q1 THEN JM=1
22470 :
22480 REM METHODE DER DRIEDELING
22490 QD=QA:Q4=Q1
22500 QE=QC:Q5=Q3
22510 QL=QE-QD
22520 :
22530 REM BEGIN VAN DE HERHALINSLUS
22540 QS=QL/3
22550 Q6=FN F(QD+QS):Q7=FN F(QE-QS)
22560 IF ABS(Q6)>1E4 OR ABS(Q7)>1E4 THEN 22720
22570 ON JM GOTO 22580, 22600
22580 IF Q6<Q7 THEN QD=QD+QS:GOTO 22610
22590 QE=QE-QS:GOTO22610
22600 IF Q6>Q7 THEN QD=QD+QS:GOTO 22610
22605 QE=QE-QS
22610 QL=QE-QD
22620 REM EINDE VAN DE HERHALING
22630 :
22640 IF QL>QG THEN 22540
22650 :
22660 NW=NW+1:QX=(QD+QE)/2:IF NW<=MA THEN WA(NW)=QX:BW(NW)=JM
22670 IF QX>QB THEN JF=2
22680 REM EINDE VAN "DRIEDELING"
22690 :
22700 IF JF>0 THEN JF=JF-1
22710 IF NW>MA THEN KW=1:J=JN
22720 QA=QB:QB=QC:Q1=Q2:Q2=Q3
22730 NEXT
22740 :
22750 :
22760 RETURN
```

7/21.3.4
Subroutine: Afgeleiden

Probleembeschrijving
Deze subroutine dient afgeleiden te bepalen. De moeilijkheid ligt in het bepalen van de eerste afgeleide. Bijkomend geeft het programma zonder veel extra moeite de tweede afgeleide. Er wordt niet analytisch gedifferentieerd (er wordt geen vergelijking voor de afgeleide bepaald), maar slechts de waarde van de afgeleide in een bepaald punt, dat de gebruiker moet specificeren.

De eerste afgeleide stelt grafisch gezien de stijging (of daling) van de functiekromme in het punt x voor.

Indien met behulp van de subroutine extremen zijn bepaald, is het nu mogelijk om deze te controleren met behulp van de huidige subroutine (afgeleiden). Voor een extreem moet aan de volgende voorwaarde zijn voldaan:

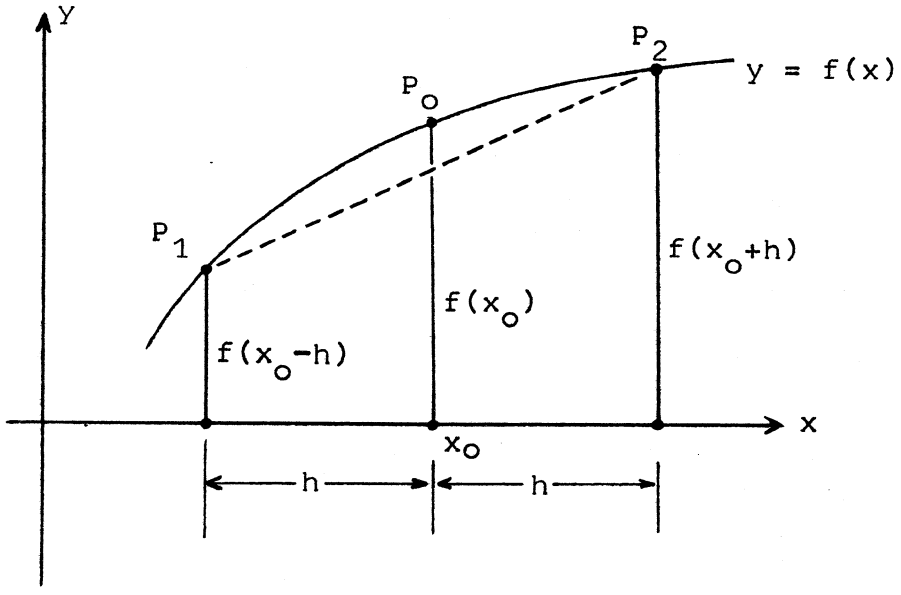
lokaal minimum: $y'(x) = 0, y''(x) < 0$
lokaal minimum: $y'(x) = 0, y''(x) > 0$

Oplossingsmethode
Gevraagd wordt de eerste en tweede afgeleide in het punt $x=x_0$ te bepalen. De eerste afgeleide stelt de stijging (of daling) van de functie in het punt P_0 voor.

Volgens de definitie verkrijgt men de afgeleide door berekening van de volgende limietwaarde:

$$y'_0 = f'(x_0) = \lim_{h \to 0} \frac{f(x0+h) - f(x0)}{h}$$

Deze formule voor het berekenen van de limietwaarde wordt het rechter differentiaal-



21.3 Functieonderzoek

quotiënt van de eerste orde genoemd. De limietberekening laat zich als volgt grafisch verduidelijken: het punt P_2 verplaatst zich op de kromme in de richting van P_0; daarbij draait de lijn door P_2 en P_0 naar een limiet waarde/richting. Deze limietwaarde is de raaklijn in het punt P_0.

Op dezelfde manier is het linker differentiaalquotient te definiëren (die hetzelfde resultaat geeft):

$$y'_0 = f'(x_0) = \lim_{h \to 0} \frac{f(x0) - f(x0-h)}{h}$$

Het wiskundige midden tussen de twee differentiaalquotienten is het centrale differentiaalquotient van de eerste orde:

$$y'_0 = f'(x_0) = \lim_{h \to 0} \frac{f(x0+h) - f(x0-h)}{2h}$$

Hierbij verplaatsen de punten P_1 en P_2 zich tegelijkertijd in de richting van P_0. De lijn door P_1 en P_2 heeft dan (zeer) de richting van de afgeleide in P_0.

In de subroutine wordt de limietwaarde niet analytisch maar numeriek bepaald. de afgeleide word daarom slechts benaderd. Deze benadering komt stapsgewijs tot stand met behulp van het centrale differentiaalquotient.

De breuk $\frac{f(x0+h) - f(x0-h)}{2h}$

geeft de richting van de lijn door P_1 en P_2 aan. Het programma begint met een waarde van h=0,4. h is de afstand tussen de x-waarden van de punten P_1 en P_2. Vervolgens wordt h gehalveerd, en wordt de nieuwe stijging/richting bepaald. Dit wordt meerdere malen herhaald en op deze manier verschuiven de punten P_1 en P_2 in de richting van P_0. De gewenste nauwkeurigheid is bereikt, indien het verschil tussen de nieuwe stijging en de laatst berekende stijging erg klein is.

Voor de berekening van de benadering van de tweede afgeleide wordt de volgende formule gebruikt:

$$\frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}$$

De resultaten zijn soms grote waarden: bijvoorbeeld bij een steil verlopende kromme heeft de eerste afgeleide een (zeer) grote waarde. Indien met een gewone/eenvoudige nauwkeurigheid wordt gerekend, zijn eigenlijk alleen de eerste vier cijfers correct. Een getal 12345,67 wordt dan geconverteerd naar 12350,00.

Variabelen

Aangezien het om een algemeen bruikbare subroutine gaat beginnen de namen van lokale variabelen met een Q of een J (zie vorige paragraaf). De toevoeging nieuwe waarde geeft aan dat het om de laatst berekende waarde gaat, de toevoeging oude waarde duidt op de waarde berekend bij de vorige iteratiestap. De variabelen hebben de volgende betekenis:

Btest (boolean) waarde die aangeeft of de differentiatie mogelijk was. (0=nee, bijv. door het niet continu zijn van een functie, 1=resultaat beschikbaar).

B1:test (boolean) variabele die aangeeft of de gewenste nauwkeurigheid bij de eerste afgeleide is behaald (0=gehaald, 1=eventueel onnauwkeurig, 2=eventueel fout).

21.3 Functieonderzoek

B2:test (boolean) variabele die aangeeft of de gewenste nauwkeurigheid bij de tweede afgeleide is behaald (0=gehaald, 1=eventueel onnauwkeurig, 2=eventueel fout).

QH:breedte van een deelinterval: benaming in fig.: h

QN:gewenste nauwkeurigheid, bijv. 0,0005.

X0:x-waarde, plaats waar de afgeleiden moeten worden berekend, in fig. x_0

Y0:f(X0),d.w.z. de functiewaarde van X0.

Q1:eerste afgeleide in x=X0 (oude waarde van de variabele Y1).

Q2:tweede afgeleide in x=X0 (oude waarde van de variabele Y2).

QE:relatieve nauwkeurigheid van de eerste afgeleide (nieuwe waarde).

QF:relatieve nauwkeurigheid van de tweede afgeleide (nieuwe waarde).

QC:relatieve nauwkeurigheid van de eerste afgeleide (oude waarde van de variabele QE).

QD:relatieve nauwkeurigheid van de tweede afgeleide (oude waarde van de variabele QE).

J1:geeft aan of er nog meer iteratiestappen nodig zijn voor de berekening van de eerste afgeleide (0=nee, 1=ja)

J2:geeft aan of er nog meer iteratiestappen nodig zijn voor de berekening van de tweede afgeleide (0=nee, 1=ja)

QR:f(X0+QH), d.w.z. de functiewaarde in x=X0+QH

QL:f(X0-QH, d.w.z. de functiewaarde in x=X0-QH

Voorbeeld

De extremen die in de vorige paragraaf zijn berekend worden met behulp van de huidige subroutine gecontroleerd. In elk punt waar zich een maximum bevindt moet de eerste afgeleide nul zijn en de tweede kleiner dan nul. Bij de minima moet de eerste afgeleide eveneens de waarde nul hebben, maar de tweede afgeleide moet positief zijn.

Het programma geeft als afgeleide niet altijd exact de waarde nul. Dit komt doordat:

- er met een niet al te grote nauwkeurigheid gerekend wordt
- de subroutine de x-waarden van de extremen slechts met drie cijfers na de komma geeft

21.3 Functieonderzoek

```
200 DEF FN F(X)=5*EXP(-0.2*X)*SIN(X)

READY.



BEREKENING VAN 1E EN 2E AFGELEIDE

DE AFGELEIDE WORDT IN EEN PUNT BEREKEND

GEEF HET PUNT X 4.515
Y  =-1.98740322
Y' =-7.97212124E-06 (EVT. ONNAUWKEURIG)
Y''= 2.065
DRUK OP RETURN...


BEREKENING VAN 1E EN 2E AFGELEIDE

DE AFGELEIDE WORDT IN EEN PUNT BEREKEND

GEEF HET PUNT X 7.656
Y  = 1.06025576
Y' = 6.4920634E-04
Y''=-1.102
DRUK OP RETURN...
```

```
23000 REM ------------------------------------------------
23010 REM           SUBROUTINE : AFGELEIDEN
23020 REM ------------------------------------------------
23290 :
23300 NW=1:K1=0:K2=0
23310 QH=0.4:QG=0.0005
23320 Q1=1E30:Q2=1E30:JN=0
23330 QE=10:QC=1E30:J1=1
23340 QF=10:QD=1E30:J2=1
23350 :
23360 REM EINDE INITIALISATIE
23370 :
```

21.3 Functieonderzoek

```
23380 YO=FN F(XO)
23390 :
23400 REM BEGIN VAN DE HERHALINGSLUS
23410 QR=FN F(XO+QH):QL=FN F(XO-QH)
23420 :
23430 IF J1=0 THEN 23500
23440 Y1=(QR-QL)/(2*QH)
23450 QE=ABS((Y1-Q1)/Q1)
23460 IF QE>2*QC THEN Y1=Q1:QE=QC:J1=0
23470 IF QE<=QG THEN J1=0
23480 IF ABS(Y1)<1E-16 THEN Y1=0:J1=0
23490 :
23500 IF J2=0 THEN 23570
23510 Y2=(QR-2*YO+QL)/(QH*QH)
23520 QF=ABS((Y2-Q2)/Q2)
23530 IF QF>2*QD THEN Y2=Q2:QF=QD:J2=0
23540 IF QF<=10*QG THEN J2=0
23550 IF ABS(Y2)<1E-16 THEN Y2=0:J2=0
23560 :
23570 Q1=Y1:Q2=Y2
23580 QC=QE:QD=QF
23590 QH=0.5*QH
23600 REM EINDE HERHALINGSLUS
23610 :
23620 IF J1+J2>0 AND QH>0.005 THEN 23410
23630 :
23640 IF QE>2*QG THEN K1=1
23650 IF ABS(Y1)<0.01 AND QE<0.1 THEN K1=0
23660 IF QE>200*QG THEN K1=2
23670 IF K1=2 AND ABS(Y1)<0.01 AND QE<1 THEN K1=1
23680 IF QF>10*QG THEN K2=1
23690 IF ABS(Y2)<0.01 AND QF<0.1 THEN K2=0
23700 IF QF>1000*QG THEN K2=2
23710 IF K2=2 AND ABS(Y2)<0.01 AND QF<1 THEN K2=1
23720 :
23730 REM AFRONDEN OP 4 CIJFERS
23740 IF ABS(Y1)<=0.01 OR K1=2 THEN 23780
23750 JN=INT(LOG(ANS(Y1))/LOG(10))
23760 JM=4-1-JN
23770 Y1=SGN(Y1)*INT(10↑JM*ABS(Y1)+.5)/10↑JM
23780 IF ABS(Y2)<=0.01 OR K2=2 THEN 23830
23790 JN=INT(LOG(ABS(Y2))/LOG(10))
23800 JM=4-1-JN
23810 Y2=SGN(Y2)*INT(10↑JM*ABS(Y2)+.5)/10↑JM
23820 :
23830 :
23840 :RETURN
23850 :
```

7/21.3.5
Subroutine: Bepaalde integraal

Probleembeschrijving
Gegeven: een willekeurige functie y=f(x).
Gevraagd: de bepaalde integraal

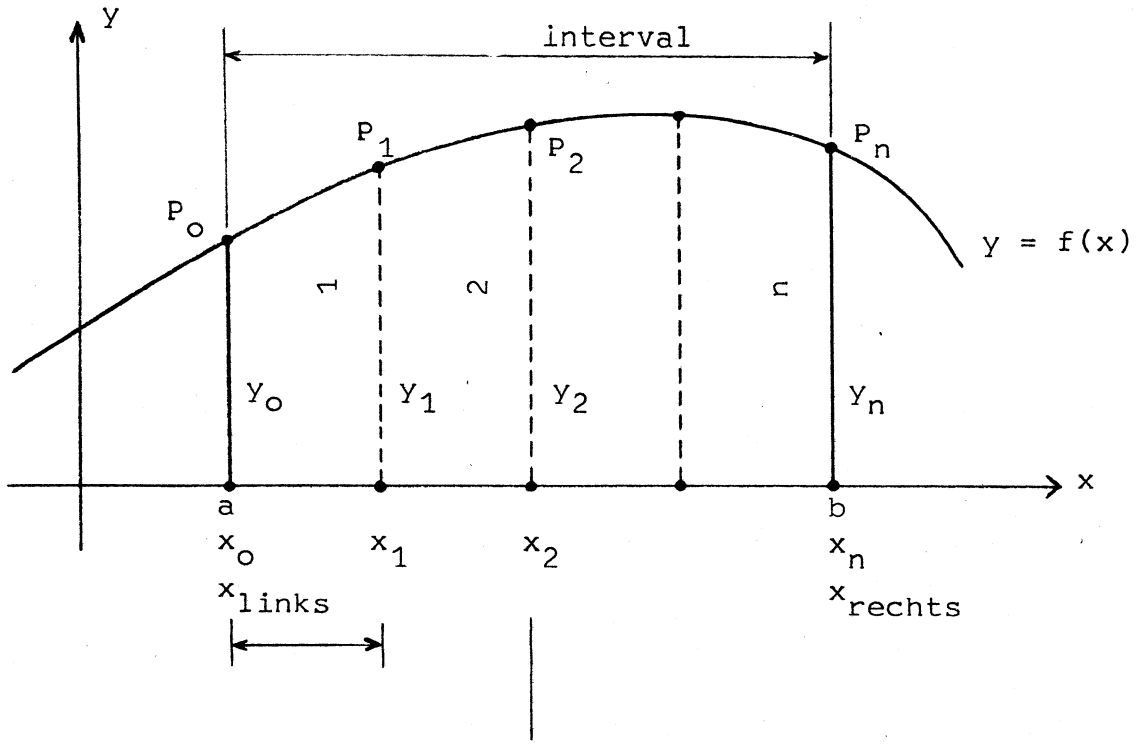
$$I = \int_a^b f(x)\,dx$$

Hierbij zijn a en b de grenzen van het integratie-interval (linker en rechter x-waarde). De bepaalde integraal kan grafisch worden voorgesteld als het oppervlak tussen de krommen en de s-as. x=a en x=b zijn de rechten die het vlak A respectievelijk links en rechts begrenzen.

Het resultaat van de bepaalde integraal is geen functievergelijking maar slechts een getal.

Oplossingsmethode
Er wordt een numerieke methode gebruikt. Hierbij moet het gebied worden opgedeeld in deelintervallen met gelijke lengte. Deze deelintervallen worden in vervolg als stroken betiteld.

De integraal wordt berekend door de deelintegralen van de stroken bij elkaar op te tellen (de oppervlakte van elk van de stroken wordt berekend en deze worden bij elkaar opgeteld). Elk deeloppervlak wordt met behulp van een benaderingsmethode bepaald. De functie f(x) wordt in elke strook door een polynoom vervangen. Het zou het makkelijkste zijn hier een polynoom van de eerste graad – een rechte lijn – voor te nemen. Dus bij strook één komt er dan een rechte lijn door P_0 en P_1. Dit is de toepassing van de zogenaamde trapeziumregel. Deze benadering heeft echter als nadeel dat het aantal stroken zeer groot (en dus de breedte zeer



21.3 Functieonderzoek

klein) moet zijn om een goede benadering te krijgen. Er is dus zeer veel rekenwerk voor nodig.

Een wezenlijk betere methode is de Simpson-regel. Hierbij wordt de functie/kromme in een dubbele strook vervangen door een kwadratische vergelijking (parabool). Als voorbeeld kan worden opgemerkt dat de parabool van de eerste dubbele strook door de punten P_0, P_1 en P_2 loopt. Het vlak tussen de parabool en de x-as is exact te berekenen. In de volgende tekst wordt onder een deelvlak een dubbele strook verstaan.

De gezochte benadering, ofwel de som van alle deelvlakken, is direct met behulp van de volgende formule te berekenen (Simpson-formule):

$$I = h/3 \, (y_0 + 4\, y_1 + 2\, y_2 + 4\, y_3 + \ldots + 2\, y_{n-2} + 4\, y_{n-1} + y_n)$$

Let er wel op dat deze formule alleen dan een goed resultaat oplevert, indien n even is (opdat een opdeling in dubbele stroken mogelijk is).

De nauwkeurigheid van de benadering is hoger naarmate de verdeling fijner is. Om vast te kunnen stellen of de gewenste nauwkeurigheid is behaald, vinden de volgende handelingen plaats:

Om te beginnen wordt het gehele interval in vier stroken opgedeeld (twee dubbele stroken), waarna de integraal I met behulp van bovenstaande Simpson-regel wordt berekend. Vervolgens wordt het aantal stroken n verdubbeld en wordt de integraal I opnieuw uitgerekend. Hierna wordt de nieuwe I met de oude I vergeleken. Is het verschil zeer klein, dan wordt de nieuwe I als resultaat gegeven. Is het onderscheid echter nog relatief groot, dan wordt het aantal n nogmaals verdubbeld en I opnieuw berekend. Deze handelingen, het verdubbelen van n en berekenen van I, wordt net zolang herhaald tot de gewenste nauwkeurigheid is behaald, dus tot de nieuwe I vrijwel gelijk is aan de in de vorige iteratiestap berekende I. In het algemeen kan worden gezegd dat er een rij van benaderingen ontstaat die naar de echte integraal convergeert.

De iteratie wordt in de volgende drie gevallen beëindigd:

- indien de gewenste nauwkeurigheid is behaald
- indien n >= 1024 (dus n > 512 dubbele stroken)
- indien de relatieve nauwkeurigheid slechter wordt, dus als het verschil tussen twee opeenvolgende waarden van I weer toeneemt. De oorzaak hiervan is dat bij een groot aantal stroken de berekening erg omvangrijk wordt, zodat afrondingsfouten het verkrijgen van een grotere nauwkeurigheid in de weg staan

Het programma berekent na elke verdubbeling van het aantal stroken alle functiewaarden opnieuw. Het is natuurlijk mogelijk om reeds berekende waarden opnieuw te gebruiken, maar om het programma overzichtelijk te houden is daar niet voor gekozen. De reducering van de rekentijd is gering, daar deze hoofdzakelijk bepaald wordt door het aantal stroken dat gebruikt wordt.

Variabelen

Aangezien het om een algemeen bruikbare subroutine gaat beginnen de namen van lokale variabelen met een Q of een J (zie vorige paragraaf). De toevoeging nieuwe waarde geeft aan dat het om de laatst berekende waarde gaat, de toevoeging oude waarde

21.3 Functieonderzoek

duidt op de waarde berekend bij de vorige iteratiestap. De variabelen hebben de volgende betekenis:

XL:linkergrens van het integratieinterval; in fig. a of x_0 of x_{links}

XR:rechtergrens van het integratieinterval; in fig. b of x_n of x_{rechts}

NWAARDE:test (boolean) variabele die aangeeft of de integratie mogelijk was (0=niet mogelijk, bijv. functie niet continu, 1=resultaat beschikbaar)

BWtest (boolean) waarde die aangeeft of de integraal de gewenste nauwkeurigheid heeft. (0=ja, 1=eventueel onnauwkeurig).

JN:huidige stroken aantal; in fig.: h

QH:breedte van een deelinterval: benaming in fig.: h

QG:gewenste nauwkeurigheid, bijv. 0,001.

QNIEUW:bepaalde integraal (nieuwe waarde)

QA:bepaalde integraal (oude waarde)

Q1relatieve nauwkeurigheid (nieuwe waarde).

Q2relatieve nauwkeurigheid (oude waarde).

QR:som van de functiewaarden van de randpunten, d.w.z. f(XLINKS)+f(RECHTS)

QX:huidige x-waarde (rechterrand van de huidige strook); in fig.: x_1 tot x_n

QFAC:factor, heeft afwisselend de waarde 4 en de waarde 2

WAARDE:bepaalde integraal, eindantwoord.

Voorbeeld
Wederom wordt de functie $y=f(x)=5e^{-0,2x}$ sin x gebruikt. Nu worden eerst de nulpunten berekend:

```
200 DEF FN F(X)=5*EXP(-0.2*X)*SIN(X)

READY.

bereik:
LINKER X-WAARDE 0
RECHTER X-WAARDE 10
NULPUNTEN (SNIJPUNTEN MET X-AS):

X= 0                          Y= 0
X= 3.1415966                          Y=-1.0532156E-05
X= 6.28318934                          Y= 5.74160593E-06
X= 9.42477985                          Y=-1.43491623E-06
DRUK OP RETURN...
```

21.3 Functieonderzoek

Nu de nulpunten dan bekend zijn kan het oppervlak van het positieve vlak (tussen de eerste en het tweede nulpunt) en van het negatieve vlak (tussen het tweede en het derde nulpunt) worden berekend:

```
BEREKENING VAN BEPAALDE INTEGRAAL


bereik:
LINKER X-WAARDE 0
RECHTER X-WAARDE 3.142
INTEGRAALWAARDE: 7.37254217
DRUK OP RETURN...
```

```
BEREKENING VAN BEPAALDE INTEGRAAL


bereik:
LINKER X-WAARDE 3.142
RECHTER X-WAARDE 6.283
INTEGRAALWAARDE:-3.93316331
DRUK OP RETURN...
```

Ter controle wordt nu de totale integraal (oppervlak) tussen de eerste en het derde nulpunt berekend:

```
BEREKENING VAN BEPAALDE INTEGRAAL


bereik:
LINKER X-WAARDE 0
RECHTER X-WAARDE 6.283
INTEGRAALWAARDE: 3.43937884
DRUK OP RETURN...
```

In dit geval is de uitkomst in orde. Men moet echter wel bedenken, dat bij een functie met veel extremen en nulpunten (een 'onrustige functie' om het zo maar te zeggen) het integratie-interval niet al te groot moet zijn. Het programma begint zoals beschreven met vier stroken. Geven deze vier stroken een kleine benaderingswaarde, dan is een verbetering niet altijd mogelijk door verdere iteratiestappen. In zo'n geval geeft het hoofdprogramma 'onnauwkeurig' of 'fout' weer. De gebruiker moet in zo'n geval het integratieinterval in kleinere delen opsplitsen en na afloop optellen. Of de aanvangswaarde JN moet in regel 24300 verhoogd worden (bijv. van 4 naar 32).

21.3 Functieonderzoek

```
24000 REM ------------------------------------------------------
24010 REM          SUBROUTINE : BEPAALDE INTEGRAAL
24020 REM ------------------------------------------------------
24290 :
24300 NW=1:KW=1:JN=4
24310 QG=0.0001:QA=0:Q2=1E30
24320 :
24330 REM EINDE INITIALISATIE
24340 :
24350 QR=FN F(XL)+FN F(XR)
24360 :
24370 REM BEGIN VAN DE HERHALINGSLUS
24380 QN=QR
24390 QH=(XR-XL)/JN
24400 QF=4
24410 :
24420 FOR J=1 TO JN-1
24430 QX=XL+J*QH
24440 QN=QN+QF*FN F(QX)
24450 QF=6-QF
24460 NEXT
24470 :
24480 QN=QN*QH/3
24490 Q1=ABS((QN-QA)/QN)
24500 IF Q1>=Q2 THEN QN=QA:Q1=Q2:JN=1024
24510 IF ABS(QN)<1E-16 THEN QN=0:JN=1024
24520 JN=JN+JN
24530 QA=QN
24540 Q2=Q1
24550 REM EINDE HERHALINGSLUS
24560 :
24570 IF Q1>QG AND JN<1024 THEN 24380
24580 :
24590 IF Q1>QG THEN KW=0
24600 WA=QN
24610 :
24620 :
24630 RETURN
24640 :
```

7/21.3.6
Subroutine: Kromme

Probleembeschrijving
De gegeven functievergelijking y=f(x) moet in een rechthoekig coördinatenstelsel worden weergegeven. De linker en rechtergrens van het weer te geven deel wordt door de gebruiker opgegeven.

Werking
De afstand tussen twee punten op de x-as kan als deelinterval Δx worden beschouwd. Hiermee wordt niet de lengte op het beeldscherm bedoeld, maar op de x-as. Heeft bijvoorbeeld de gebruiker de grenzen 2 en 6 opgegeven en zijn er op het beeldscherm horizontaal 128 punten (0 tot en met 127), dan heeft elk deelinterval de volgende waarde/lengte:

Δx=(6-2)/127=0.03149606

In dit voorbeeld heeft het eerste punt (punt 0) de x-waarde 2, het tweede punt (punt 1) de x-waarde 2.03149606, het derde punt (punt 2) de x-waarde 2.06299212 en het laatste punt (punt 127) de x-waarde 2+127·0.03149606=6.

De kromme kan nu getekend woorden door alle punten van de x-as af te lopen. Voor elk beeldpunt wordt de bijbehorende x-waarde berekend, zoals zojuist is vermeld, en vervolgens de functiewaarde. Deze waarden worden in een variabelenveld opgeslagen. Het is niet mogelijk om direct de informatie op het scherm te zetten. Het is namelijk de bedoeling het gehele beeldscherm te benutten en daartoe moeten eerst de maximale en minimale y-waarde (functiewaarde) bekend zijn.

Nadat de extreme waarden bekend zijn, kan de verdeling van de y-as worden bepaald. Om het volle beeldscherm te kunnen benutten komt dan (uiteraard) de maximale y-waarde boven in beeld en de minimale y-waarde wordt onderaan gepositioneerd. Nu wordt nogmaals langs alle punten van het gewenste deel van de x-as gegaan, elk punt van de horizontale verdeling. De bijbehorende functiewaarde wordt uit het variabelenveld en met behulp van de verdeling van de y-as wordt de verticale positie op het scherm bepaald. Nu dus zowel de horizontale als de verticale positie van een punt bekend zijn kan het punt 'aan' worden gezet.

Als alternatief kan de gebruiker opgeven dat verticaal dezelfde verdeling wordt gebruikt als horizontaal.

De onderverdeling van de assen wordt niet weergegeven en er worden geen getallen bij geplaatst. Bij weergave van de functie door losse punten kan het beeld wordt verstoord. Bovendien geeft het, bij een fijne verdeling, een onrustig beeld. Daar de gebruiker zelf het interval dient op te geven en het programma de extreme waarden weergeeft, is het niet echt nodig om de assen van getallen te voorzien. De grafische weergave heeft alleen maar tot doel een indruk van het functieverloop te geven. Is men op zoek naar exacte waarden, dan moet een andere subroutine worden gebruikt.

De gebruiker kan een willekeurig interval opgeven. Het kan echter voorkomen dat een functie op bepaalde plaatsen niet is gedefinieerd. In dat geval zal het programma met een foutmelding afbreken.

Variabelen
Aangezien het om een algemeen bruikbare subroutine gaat beginnen de namen van lokale variabelen met een Q of een J (zie de

eerste paragraaf). De variabelen hebben de volgende betekenis:

XL:linkergrens van het interval (begin van de kromme).

XR:rechtergrens van het interval (einde van de kromme).

YTOE:toelaatbare y-waarde (er worden alleen y-waarden getekend met een absolute waarde kleiner dan YTOE, dus |Y| <= YTOE).

T1:test variabele die aangeeft of de x- en y-as dezelfde verdeling moeten krijgen (1=nee, dus volle beeldscherm benutten, 2=ja).

T2:test variabele voor de keuze van weergave (1=losse punten, 2=verbinden van punten door een lijn, 3=opvullen van de vlakken).

JH:horizontale resolutie. Een regel bestaat uit JH+1 punten, van 0 tot en met JH.

JV:verticale resolutie. Een kolom bestaat uit JV+1 punten, van 0 tot en met JV.

QV:verhouding van de hoogte tot de breedte van een punt (zie werking)

Q1:minimum, kleinste weer te geven y-waarde

Q2:maximum, grootste weer te geven y-waarde

Q3:plaats (x-waarde) van het minimum.

Q4:plaats (x-waarde) van het maximum.

J9:test, geeft aan of de DIM-opdracht reeds is uitgevoerd (0=nee, 1=ja). De reden hiervoor is dat deze subroutine meerdere malen moet kunnen worden aangeroepen. De Commodore-64 BASIC geeft een foutmelding als dezelfde DIM-opdracht nog een keer wordt uitgevoerd.

QX:x-waarde van het huidige beeldschermpunt/rasterpunt.

QY(i):y-waarde van het huidige (i-de) punt.

QS:horizontale stapgrootte (delta) Δx.

QD:verticale stapgrootte (delta) DELTAy.

JT:test, die aangeeft of bij het tekenen van de kromme de bij de vorige x-waarde behorende punt weggelaten werd (0=nee, d.w.z. bij weergave mogelijkheid 2 is een lijn tussen het huidige punt en het vorige punt; 1=ja, d.w.z er wordt geen lijn getekend.

JY(i):test die aangeeft of de y-waarde van een punt te berekenen is

JX:verticale positie van de x-as.

J1:horizontale positie van het beginpunt van een lijn.

J2: verticale positie van het beginpunt van een lijn.

J3:a) horizontale positie van een punt.

b) horizontale positie van het eindpunt van een lijn.

j4:a) verticale positie van een punt.

b) verticale positie van het eindpunt van een lijn.

21.3 Functieonderzoek

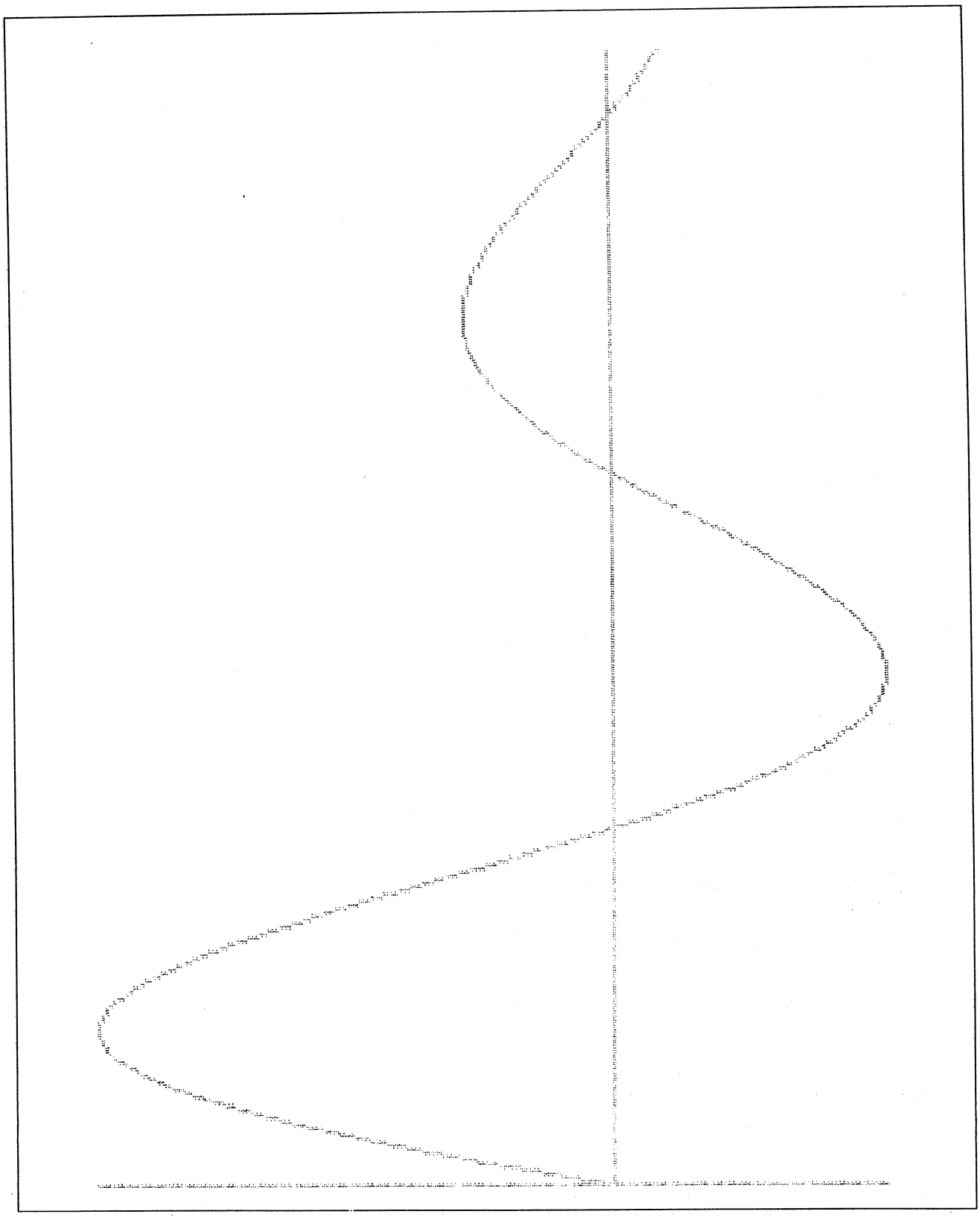
Het programma zoekt naar de hoogste en laagste y-waarde. Deze waarden zijn niet zo nauwkeurig als bij de subroutine extremen. Bovendien gaat het hier om absolute en niet om relatie (lokale) extremen.

```
25000 REM ------------------------------------------------
25010 REM          SUBROUTINE : GRAFIEK TEKENEN
25020 REM ------------------------------------------------
25030 :
25040 REM LET OP: DEZE ROUTINE WERKT ALLEEN MET
25050 REM WEKA-BASIC, SIMONS' BASIC O.I.D.
25060 :
25300 JH=319:JV=199:QV=1.2
25305 Q1=1E30:Q2=-1E30
25310 :
25315 IF J9=0 THEN DIM QY(JH),JY(JH)
25320 J9=1
25330 :
25340 PRINT"■"
25345 PRINT"MOMENTJE GRAAG, FUNCTIEWAARDEN WORDEN   BEREKEND...■"
25350 QS=(XR-XL)/JH
25355 :
25360 FOR J=0 TO JH
25365 QX=XL+J*QS
25370 QY(J)=FN F(QX)
25375 PRINT"*";
25380 JY(J)=1
25385 IF ABS(QY(J))>YZ THEN 25400
25390 IF QY(J)<Q1 THEN Q1=QY(J):Q3=QX
25395 IF QY(J)>Q2 THEN Q2=QY(J):Q4=QX
25400 NEXT
25410 :
25420 IFQ1=0ANDQ2=0THENPRINT"ALLE FUNCTIEWAARDEN ZIJN 0.":
                                                INPUTQ$:GOTO25725
25425 IFQ1=1E30ORQ2=-1E30THENPRINT"FUNCTIEWAARDEN TE GROOT.":
25430 PRINT"■"                                  INPUTQ$:GOTO25725
25435 IF K1=2 THEN 25475
25440 PRINT"DE GRAFIEK HEEFT DE VOLGENDE GRENZEN:"
25445 PRINT:PRINT"KLEINSTE Y-WAARDE:"Q1
25446 PRINT"  BIJBEHORENDE X-WAARDE:"Q3
25450 PRINT:PRINT"GROOTSTE Y-WAARDE:"Q2
25451 PRINT"  BIJBEHORENDE X-WAARDE:"Q4
25455 PRINT
25460 PRINT:INPUTQ$
25465 PRINT
25470 HIRES6,1:CLEAR
25475 IF Q1>0 THEN Q1=0
25480 IF Q2<0 THEN Q2=0
25485 QD=(Q2-Q1)/JV
```

21.3 Functieonderzoek

```
25495 REM X-AS
25500 JX=INT(Q2/QD+.5):J2=JX:J4=JX
25505 J1=0:J3=JH
25510 GOSUB 25800:REM LIJN TEKENEN
25515 :
25520 REM Y-AS
25525 IF XL>0 OR XR<0 THEN 25550
25530 J1=INT(-XL/QS+.5):J3=J1
25535 J2=0:J4=JV
25540 :GOSUB 25800
25550 REM KROMME
25555 IF K1=2 THEN QD=QS*QV:Q2=JX*QD
25560 JK=1:J2=0
25570 :
25580 FOR J=0 TO JH
25585 J4=INT((Q2-QY(J))/QD+.5)
25590 J3=J
25595 ON K2 GOTO 25610,25635,25680
25600 :
25605 REM TEKENMODE 1: PUNTEN
25610 IF J4>JV OR J4<0 THEN 25705
25615 GOSUB 25780:REM PUNT ZETTEN
25620 GOTO25705
25625 :
25630 REM TEKENMODE 2: LIJNEN
25635 IF J4>JV OR J4<0 THEN JK=1:GOTO 25705
25640 IF JY(J)=0 THEN JK=1:GOTO 25705
25645 IF JK=1 THEN J1=J3:J2=J4
25650 JK=0
25655 GOSUB 25800
25660 J1=J3:J2=J4
25665 GOTO 25705
25670 :
25675 REM TEKENMODE 3: VLAKKEN
25680 J1=J3:J2=JX
25685 IF J4>JV THEN J4=JV
25690 IF J4<0 THEN J4=0
25695 GOSUB 25800
25700 :
25705 NEXT
25710 :
25715 KEY:REM WACHT OP EEN TOETSDRUK
25720 TEXT
25725 RETURN
25730 :
25770 REM PUNT ZETTEN
25780 PLOT J3,J4:RETURN
25785 :
25790 REM LIJN TEKENEN
25800 DRAW J1,J2 TO J3,J4:RETURN
```

21.3 Functieonderzoek



7/22

Statistiek

Inhoud

7/22.1

Gemiddelde waarde, variantie en standaardafwijking

Inleiding

In de statistiek verkrijgt men een groot aantal gegevens door tellen, meten of ondervragen. Om uit een grote hoeveelheid gegevens zinnige informatie te kunnen halen moeten de gegevens – ook wel data genoemd – gecomprimeerd worden, daar met een berg informatie op zich weinig aan te vangen valt. Hiertoe wordt van *kentallen* gebruik gemaakt, om de informatie te karakteriseren. De gemiddelde waarde geeft, zoals de naam al zegt, de gemiddelde waarde van metingen of gegevens aan. Op grond van bijvoorbeeld snelheidsmetingen kan men dan zeggen dat op een bepaalde weg gemiddeld 120 kilometer per uur wordt gereden. Daartoe zijn misschien wel zo'n 2000 metingen verricht. De gemiddelde waarde is dan wat handelbaarder dan die 2000 metingen.

Omdat echter de gemiddelde waarde niet alles zegt over de metingen, wordt ook een kental gebruikt die de afwijkingen of schommelingen rond de gemiddelde waarde aangeeft, de variantie. Daar de variantie met kwadratische waarden werkt ontstaat er een verschil in dimensie tussen de gemiddelde waarde en de variantie. Indien de gemiddelde waarde over de lengte van staven gaat is de dimensie meters; de variantie heeft dan de dimensie vierkante meter. Om nu te zorgen dat de schommelingen rond de gemiddelde waarde de zelfde dimensie hebben als laatstgenoemde grootheid wordt een derde kental gebruikt, namelijk de standaardafwijking. De standaardafwijking wordt verkregen door de wortel van de variantie te nemen.

Definities

In het programma worden de gegevens in X(i) ingelezen. Uit deze gegevens wordt de gemiddelde waarde als volgt berekend:

$$XM = \frac{1}{n} \sum_{i=1}^{n} X(i)$$

De formule voor de variantie ziet er als volgt uit:

$$S = \frac{1}{n-1} \sum_{i=1}^{\infty} (X(i) - XM)^2$$

De standaardafwijking tenslotte is als volgt gedefinieerd:

$$SA = \sqrt{S}$$

22.1 Gemiddelde waarde, variantie en standaardafwijking

Structuurdiagram

I = aantal (meet)waarden
lees voor j=1 tot de waarde X(J)
XM = 0
bereken voor J=1 tot I: XM=XM+X(J)
bereken:XM=XM/I
S=0
bereken voor J=1 tot I:S=S+(X(J)-XM)^2
$SA = \sqrt{S}$
geef XM,S en SA weer.

```
HOEVEEL MEETGEGEVENS ZIJN ER 10

MEETGEGEVEN INVOEREN 48
MEETGEGEVEN INVOEREN 49
MEETGEGEVEN INVOEREN 51
MEETGEGEVEN INVOEREN 52
MEETGEGEVEN INVOEREN 50
MEETGEGEVEN INVOEREN 48
MEETGEGEVEN INVOEREN 50
MEETGEGEVEN INVOEREN 49
MEETGEGEVEN INVOEREN 51
MEETGEGEVEN INVOEREN 48

GEMIDDELDE          : 49.6
STANDAARDDEVIATIE   : 1.42984071
VARIANTIE           : 2.04444445
```

```
100 REM GEMMIDDELDE, STANDAARDDEVIATIE EN VARIANTIE
110 REM
120 REM VOOR DE COMMODORE 64
130 REM
140 REM VARIABELEN
150 REM
160 REM X(I)    : MEETWAARDEN
170 REM XM      : GEMMIDDELDE
180 REM SA      : STANDAARDDEVIATIE
190 REM S       : VARIANTIE
200 REM I       : AANTAL MEETWAARDEN
210 REM J       : TELLER
220 REM
230 DIM X(1000)
240 PRINT"INVOER VAN MEETGEGEVENS":PRINT
250 INPUT"HOEVEEL MEETGEGEVENS ZIJN ER";I
260 IF I>1000 THEN PRINT"TE VEEL...":GOTO 470
270 PRINT:FOR J=1 TO I
280 INPUT"MEETGEGEVEN INVOEREN";X(J)
290 NEXT
300 REM
310 XM=0
320 FOR J=1 TO I:XM=XM+X(J):NEXT
330 XM=XM/I
340 REM
350 S=0
360 FOR J=1 TO I
370 S=S+(X(J)-XM)↑2
380 NEXT
390 REM
400 PRINT
410 PRINT"GEMIDDELDE          :";XM
420 IFI=1 THEN GOTO 460
430 S=S/(I-1):SA=SQR(S)
440 PRINT"STANDAARDDEVIATIE :";SA
450 IF I<>1 THEN PRINT "VARIANTIE          :";S:END
460 PRINT"DEVIATIE EN STANDAARDAFWIJKING KUNNEN   NIET WORDEN BEREKEND"
```

7/22.2

Covariantie, correlatiecoëfficiënt en regressierechte

Inleiding

In de vorige paragraaf zijn kentallen geïntroduceerd voor bijvoorbeeld meetresultaten (snelheid, lengte). Vaak bestaat er enige samenhang (correlatie) tussen verschillende gegevens. Hoe langer een staaf des te groter het gewicht. Dit is natuurlijk voor de hand liggend. Een fabrikant van merkkleding zal waarschijnlijk geïnteresseerd zijn, in de interesse van leeftijdsgroepen voor merkkleding, een onderzoek waarvan de uitkomsten wat minder voor de hand liggen. Om enigszins informatie te verkrijgen over afhankelijkheden of verbanden, wordt gebruik gemaakt van de covariantie, correlatiecoëfficiënten en regressierechten. Geldt bijvoorbeeld voor de correlatiecoëfficiënt KK dat 0<KK<1 dan zegt men dat twee soorten elementen positief gecorreleerd zijn. Anders gezegd er bestaat een zekere positieve samenhang, neemt de een toe, dan is het waarschijnlijk dat de ander ook toeneemt.

Definities

Nu zijn er dus twee meetwaarden of twee soorten gegevens. Deze worden ingelezen in X(i) en Y(i). Vervolgens gaan we eerst te werk zoals in de vorige paragraaf. De gemiddelde waarde berekenen geeft XM en YM. Voor de standaardafwijkingen verkrijgen we SX en SY. Voor de covariantie (VS) geldt dan:

$$VS = \frac{1}{n-1} \sum_{i=1}^{n} (X(i) - XM) * (Y(i) - YM)$$

n staat wederom voor het aantal gegevens.

De correlatiecoëfficiënt KK wordt verkregen met behulp van de formule

KK = VS / (SX*SY)

Voor de parameters van de regressierechte geldt tenslotte:

y = AR + BR*x

met BR = VS / SX en AR = YM - BR * XM

22.2 Covariantie, correlatiecoëfficiënt en regressierechte

Structuurdiagram

I=aantal (meet)waarden	
XM=0 en YM=0	
voor J=1 tot I doe	
	lees X(J) en Y(J)
	XM=XM+X(J) en YM=YM+Y(J)
XM=XM/I en YM=YM/I	
S=0 en SY=0 en VS=0	
voor J=1 tot I doe	
	S=S+(X(J)-XM) end SY=SY+(Y(J)-YM) en VS=VS+(X(J)-XM)*(Y(J)-YM)
BR=VS/S en AR=YM-BR*XM	
SX=S/(I-1) en SY=SY/(I-1)	
VS=VS/(i-1) en KK=VS/(SX*SY)	
geef VS, KK, AR en BR met tekst weer.	

Enige programmauitleg

Om te voorkomen dat door nul wordt gedeeld, wordt I, het aantal meetwaarden, gecontroleerd op de grenzen 2 en 1000. De controle op 1000 is nodig omdat 1000 de dimensie van de array's X en Y is.

BR wordt niet volgens de gegeven formule berekend, maar volgens de volgende formule:

$$BR = \frac{\sum_{i=1}^{n} (X(I) - XM)(Y(I) - YM)}{\sum_{i=1}^{n} (X(I) - XM)^2}$$

Dit is gedaan om afrondingsfouten en overbodige berekeningen te voorkomen.

22.2 Covariantie, correlatiecoëfficiënt en regressierechte

```
100 REM COVARIANTIE, CORRELATIE EN REGRESSIE
110 REM
120 REM VOOR DE COMMODORE 64
130 REM
140 REM VARIABELEN:
150 REM
160 REM X(I),Y(I) : MEETWAARDEN
170 REM XM,YM     : GEMIDDELDEN
175 REM SY,SX     : STANDAARDAFWIJKINGEN
180 REM VS        : COVARIANTIE
190 REM KK        : CORRELATIECOEFFICIENT
200 REM AR,BR     : PARAMETERS VAN DE REGRESSIERECHTE Y=AR+BR*X
210 REM I         : AANTAL MEETWAARDEN
220 REM S         : HULPSOMMATIE-VARIABELE
230 REM J         : TELLER
240 REM
250 DIM X(1000),Y(1000)
260 :
270 PRINT"INVOER VAN MEETGEGEVENS":PRINT
280 INPUT"HOEVEEL MEETGEGEVENS ZIJN ER";I
290 IF I>1000 THEN PRINT "TE GROOT":GOTO 270
300 IF I<2 THEN PRINT "DAT HEEFT GEEN ZIN!":GOTO 270
310 XM=0:YM=0
320 FOR J=1 TO I
330 PRINT"MEETGEGEVEN INVOEREN"
340 INPUT"1E WAARDE";X(J):XM=XM+X(J)
350 INPUT"2E WAARDE";Y(J):YM=YM+Y(J)
360 NEXT
370 :
380 XM=XM/I:YM=YM/I
390 :
400 S=0:SY=0:VS=0
410 FOR J=1 TO I
420 S=S+(X(J)-XM)↑2
430 SY=SY+(Y(J)-YM)↑2
440 VS=VS+(X(J)-XM)*(Y(J)-YM)
450 NEXT
460 REM
470 BR=VS/S
480 AR=YM-BR*XM
490 SX=SQR(S/(I-1))
500 SY=SQR(SY/(I-1))
510 VS=VS/(I-1)
520 KK=VS/(SX*SY)
530 REM
540 PRINT
550 PRINT"COVARIANTIE              :";VS
560 PRINT"CORRELATIECOEFFICIENT :";KK
570 PRINT"REGRESSIERECHTE Y =";AR;"+";BR;"* X"
```

22.2 Covariantie, correlatiecoëfficiënt en regressierechte

```
INVOER VAN MEETGEGEVENS

HOEVEEL MEETGEGEVENS ZIJN ER 5
MEETGEGEVEN INVOEREN

1E WAARDE 162
2E WAARDE 50
MEETGEGEVEN INVOEREN

1E WAARDE 155
2E WAARDE 49
MEETGEGEVEN INVOEREN

1E WAARDE 172
2E WAARDE 68
MEETGEGEVEN INVOEREN

1E WAARDE 163
2E WAARDE 45
MEETGEGEVEN INVOEREN

1E WAARDE 163
2E WAARDE 50

COVARIANTIE            : 42.5
CORRELATIECOEFFICIENT : .785027236
REGRESSIERECHTE Y =-137.394521 + 1.16438356 * X
```

7/22.3

Histogram

Inleiding
Staafdiagrammen worden gebruikt voor het weergeven van het voorkomen van bepaalde waarden. Als voor een bepaald schoolwerk maximaal 50 punten te behalen valt, kan worden aangegeven welke punten door hoeveel leerlingen wordt gehaald. Hierbij wordt dan het absolute aantal weergegeven.

Ook met het relatieve aantal kan worden gewerkt. Hiervoor geldt de vergelijking:

Relatieve aantal = absolute aantal / totaal aantal leerlingen

In het algemeen is dit het quotiënt van het absolute aantal van voorkomen van een bepaalde waarde/gegeven en de som van voorkomen van alle gegevens samen.

Als men een aantal gegevens samen voegt tot klassen, bijvoorbeeld mensen met een leeftijd tussen de 15 en 20 jaar en mensen met een leeftijd tussen de 20 en 25 jaar in plaats van alle leeftijden afzonderlijk, dan wordt het staafdiagram vervangen door een histogram. Een staaf stelt dan niet meer het voorkomen van een gegeven afzonderlijk (bijv. 50 punten) voor maar van een gegevensklasse (tussen 45 en 50 punten). Deze diagrammen geven een aanschouwelijk beeld van statistische waarden of meetresultaten.

Programmawerking
Het programma verzorgt het weergeven van waarden op het scherm in de vorm van een staafdiagram. Het is echter zeer eenvoudig om het programma aan te passen voor een histogram. De reden dat dit niet meteen gedaan is, is dat bij een staafdiagram een foutmelding moet worden gegeven bij invoer van een verkeerde (meet)waarde, terwijl bij een histogram alle waarden mogen, zolang ze maar binnen de aanwezige klassen vallen.

Eerst wordt om de gegevens gevraagd die moeten worden uitgezet, dus bijvoorbeeld de snelheden van auto's. Dan wordt om de gegevens gevraagd omtrent hoeveelheid van voorkomen. Aan de hand hiervan worden de x- en y-verdelingen bepaald. Als laatste wordt de schermweergave verzorgd.

Voor histogrammen geeft KB de klassegrootte aan. In regel 360 dient dan het =-teken door het <-teken vervangen te worden.

In plaats van het invoeren van de minimale meetwaarde en de afstand tussen twee waarden, is het ook mogelijk om de kleinste en grootste waarde in te voeren en daaruit de x-as verdeling te bepalen.

Een meetwaarde nul mag niet voorkomen, daar dit getal gebruikt wordt om het einde van het invoeren te signaleren. Indien ge-

22.3 Histogram

Lees de kleinste meetwaarde in (MIN)
lees de afstand tussen twee meetwaarden (KB)
voor J=1 tot 71 doe
bereken de mogelijke meetwaarde (KL(J)=MIN+(J-1)*KB)
maak de absolute aantallen (RH(J)) van de meetwaarden 0 (RH(J) geeft aan, hoeveel keer de waarde KL(J) voorkomt)
lees de meetwaarde
doe tot meetwaarde = 0
RH(J)=RH(J)+1, indien meetwaarde = KL(J)
lees meetwaarde
Bereken het maximum van de absolute aantallen (MAX=max(KL(J) met 1<=J<=71))
MAX > 20
ja / nee
Eenheid van de y-as RE=MAX/20 / Eenheid van de y-as RE=1
Plaats beeldschermrand in B$ (1. Verticale streep "|", regel 21 "-")
voor j=1 tot 71 doe
Bereken de stapgrootte voor de meetwaarde J (MAX=aantal meetwaarde in kolom J / Eenheid van de y-as, MAX=KL(J)/RE
voor K=1 tot 20-MAX doe: B$(K,J+1)=" "
voor k=21-MAX tot 20: B$(K,J+1)="X"
Geef regel voor regel op het scherm weer.

wenst kan dit opgeheven worden door regel 330 aan te passen. In het geval dat niet met de absolute aantallen moet worden gewerkt, maar met de relatieve, dan moeten na regel 400 alle absolute waarden (RH(J)) door het totaal aantal ingevoerde waarden (I) te worden gedeeld; daarna moet het nog met 20 vermenigvuldigd worden om het volle beeldscherm te kunnen benutten.

Voorbeeld van een run

In dit voorbeeld moeten de meetwaarden (gehaald puntenaantal bij een schoolwerkstuk) tussen de 1 en 50 liggen. De meetwaarde is het behaalde puntenaantal van elke scholier afzonderlijk. Na invoering van de gegevens worden de resultaten op het scherm weergegeven.

22.3 Histogram

```
100 REM HISTOGRAM
110 REM
120 REM VOOR DE COMMODORE 64
130 REM
140 REM VARIABELEN:
150 REM
160 REM T          : TABEL VOOR OPNAME MEETGEGEVENS
170 REM KL         : TABEL VOOR DE MOGELIJKE MEETGEGEVENS
175 REM RH         : TABEL VOOR HET AANTAL MEETGEG. MET DE WAARDE KL
180 REM B$         : TABEL VOOR DE BEELDSCHERMINHOUD
190 REM MIN        : KLEINSTE MEETGEGEVEN
200 REM KB         : AFSTAND TUSSEN TWEE MEETGEGEVENS
210 REM RE         : EENHEID VOOR DE Y-AS
220 REM I          : TELLER VOOR DE MEETGEGEVENS
230 REM J,K,MAX    : TELLERS EN HULPVARIABELEN
240 P=39 : REM SCHERMBREEDTE
250 DIM T(1000),B$(22,P),KL(P),RH(P):I=1
260 INPUT"GEEF DE KLEINSTE MEETWAARDE";MIN
270 INPUT"AFSTAND TUSSEN TWEE GEGEVENS";KB
280 FOR J=1 TO P-1
290 KL(J)=MIN+(J-1)*KB:RH(J)=0
300 NEXT
310 PRINT
320 INPUT"GEGEVENS INVOEREN, 0=STOP";T(I)
330 IF T(I)=0 THEN I=I-1:GOTO 400
340 REM GELDIG?
350 FOR J=1 TO P-1
360 IF T(I)=KL(J) THEN RH(J)=RH(J)+1:GOTO 390
370 NEXT
380 PRINT"MEETWAARDE ONGELDIG":GOTO 320
390 IF I<1000 THEN I=I+1:GOTO 320
400 MAX=0 : REM EENHEID Y-AS BEREKENEN
410 FOR J=1 TO P-1
420 IF RH(J)>MAX THEN MAX=RH(J)
430 NEXT
440 RE=1:IF MAX>20 THEN RE=MAX/20
450 REM BEELDSCHERMSTRINGS DEFINIEREN
460 FOR J=1 TO 20:B$(J,1)="|":NEXT
470 FOR J=2 TO P:B$(21,J)="-":NEXT
480 B$(21,1)="└":B$(21,11)="┬":B$(21,21)="┬":B$(21,31)="┬"
490 FOR J=1 TO P-1
500 MAX=RH(J):IF RE<>1 THEN MAX=INT(RH(J)/RE+.5)
510 FOR K=1 TO 20-MAX:B$(K,J+1)=" ":NEXT
520 FOR K=21-MAX TO 20
522 IF K<>21 THEN B$(K,J+1)="X"
524 NEXT
530 NEXT
540 PRINT
550 FOR J=1 TO 21
560 FOR K=1 TO P:PRINTB$(J,K);:NEXT
570 PRINT:NEXT
580 PRINTMIN;TAB(9);MIN+10*KB-1;TAB(19);MIN+20*KB-1;TAB(29);MIN+30*KB-1
590 PRINT:PRINT"DE EENHEID VAN DE Y-AS IS";RE
600 PRINT"DE EENHEID VAN DE X-AS IS";KB;
610 GET A$:IFA$="" THEN 610
```

22.3 Histogram

```
GEEF DE KLEINSTE MEETWAARDE 1
AFSTAND TUSSEN TWEE GEGEVENS 1

GEGEVENS INVOEREN, 0=STOP 15
GEGEVENS INVOEREN, 0=STOP 17
GEGEVENS INVOEREN, 0=STOP 22
GEGEVENS INVOEREN, 0=STOP 19
GEGEVENS INVOEREN, 0=STOP 17
GEGEVENS INVOEREN, 0=STOP 19
GEGEVENS INVOEREN, 0=STOP 20
GEGEVENS INVOEREN, 0=STOP 19
GEGEVENS INVOEREN, 0=STOP 24
GEGEVENS INVOEREN, 0=STOP 20
GEGEVENS INVOEREN, 0=STOP 26
GEGEVENS INVOEREN, 0=STOP 20
GEGEVENS INVOEREN, 0=STOP 19
GEGEVENS INVOEREN, 0=STOP 13
GEGEVENS INVOEREN, 0=STOP 22
GEGEVENS INVOEREN, 0=STOP 19
GEGEVENS INVOEREN, 0=STOP 19
GEGEVENS INVOEREN, 0=STOP 21
GEGEVENS INVOEREN, 0=STOP 20
GEGEVENS INVOEREN, 0=STOP 0

|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|                    X
|                    X
|                    XX
|                    XX
|                  X XX X
|              X X X XXXX X X
└──────────┴──────────┴──────────┴──────────
 1           10

DE EENHEID VAN DE Y-AS IS 1
DE EENHEID VAN DE X-AS IS 1
```

7/22.4

Toevalsgenerator

Inleiding

Onder simulatie wordt verstaan het zo goed mogelijk nabootsen van de werkelijkheid door middel van een goed model. Bij zeer veel simulaties moet gebruik worden gemaakt van een toevalsgenerator. Ook bij een aantal spelletjes wordt een toevalsgenerator toegepast. Basic heeft daarvoor de RND-functie tot de beschikking. Hiermee worden toevalsgetallen gegenereerd, uniform (gelijkelijk) verdeeld over het interval (0,1). Soms zijn echter toevalsgetallen nodig met een andere waarschijnlijkheidsverdeling. Ook is het misschien zinvol om eens te zien hoe nu toevalsgetallen worden gegenereerd en hoe toevallig ze feitelijk zijn.

Het programma in deze paragraaf genereert toevalsgetallen in het gebied (0,1), die uniform verdeeld zijn. Hetzelfde geldt voor een gebied (a,b) net zoals n discrete waarden die ieder een waarschijnlijkheid van 1/n hebben. Verder kunnen er nog getallen met een exponentiële verdeling, een normaalverdeling, een geometrische verdeling, een binominaal- en een poissonverdeling worden gegenereerd.

Voor spelen die gebruik maken van een dobbelsteen, is het nodig toevalsgetallen met een waarde 1, 2, 3, 4, 5 en 6 te kunnen genereren, waarbij de kansen uniform over de waarden zijn verdeeld. Hiertoe dient de discrete verdeling te worden gekozen met als ondergrens 1 en als bovengrens 6.

Programmawerking

Net zoals bij het BASIC-commando RND(-TI), dat op de 64 wordt gebruikt om de random-generator in te stellen, moet de gebruiker een startwaarde Y0 opgeven. Deze beginwaarde wordt gebruikt om in het bereik (0,1) uniform verdeelde toevalsgetallen te genereren. Met behulp van de startwaarde Y0 worden 100 getallen berekend volgens de volgende formule:
Y0 = (13 * Y0 + 177) mod 283

Deze worden dan opgeslagen in de array S. Deze tabel wordt dan later geïndiceerd met behulp van een toevalsgetal. De startwaarde voor de hiervoor te gebruiken 'toevalsgetallen', wordt als volgt berekend:
Z0 = (17 * Y0 + 179) mod 281.

De op het interval (0,1) uniform verdeelde toevalsgetallen verkrijgt men dan dus met behulp van de volgende formules:

```
Z0 = (17 * Z0 + 179) mod 281
I = INT([Z0 / 281 * 100]) + 1
TOEVAL = S(i)/283
Y0 = (13 * Y0 + 177) mod 283
S(I) = Y0
```

22.4 Toevalsgenerator

Waarbij dan, uiteraard, TOEVAL het gewenste toevalsgetal is. De generator zorgt voor een groot aantal verschillende toevalsgetallen, aangezien de gebruikte getallen (281, 283, ...) priemgetallen zijn.

Voor de berekening van de andere kansverdelingen gelden de volgende formules.

Exponentiële verdeling

X0*exp(-X0*t) voor t >= 0
0 voor t < 0

Eerst moet natuurlijk een toevalsgetal (TOEVAL) worden gegenereerd uit het gebied (0,1) met gelijke kansverdeling. Daarna kan het gewenste toevalsgetal (TOEV) worden berekend voor de exponentiële verdeling. Die berekening ziet er als volgt uit:

TOEV = -LOG(TOEVAL)/X0

(0,1) Normaalverdeling
Eerst moeten twee (0,1)-uniform verdeelde toevalsgetallen TOEVAL en X0 worden gegenereerd. Twee toevalsgetallen TOEV1 en TOEV2 volgens de normaalverdeling zijn dan als volgt te berekenen:

TOEV1 = SQRT(-2*LOG(X0)) * SIN(2* PI*TOEVAL)

en

TOEV2 = SQRT(-2*LOG(X0)) * COS(2* PI*TOEVAL)

Gelijke kansverdeling (uniforme verdeling) op het interval (X0, X1)
Wederom een getal (TOEVAL) uit het (0,1) interval met gelijke kansverdeling en hieruit als volgt het gewenste getal (TOEV) berekenen:

TOEV = (X1-X0) * TOEVAL + X0

Uniforme verdeling
die 'garandeert' dat de waarde X0, X0+1, ..., X1 een gelijke kans van optreden hebben. Uit het getal TOEVAL ((0,1)-uniforme verdeling) wordt op de volgende manier het gewenste getal berekend:

TOEV = INT(X1-X0+1)*TOEVAL + X0)

Binomiaalverdeling
Bij de binomiaalverdeling nemen de toevalsgetallen een van de waarden van 0, 1, ..., n aan, met de volgende waarschijnlijkheid:

$$p_i = \binom{n}{i} p^i (1-p)^{n-i}$$

Een toevalsgetal uit deze kansverdeling is op de volgende manier te berekenen (X0=p, X1=n):

TOEV=0	
voor J=1 tot X1 doe	
	bereken een (0,1)-uniform verdeeld toevalsgetal TOEVAL
	Indien TOEVAL <=X0, verhoog TOEV met 1

Aan het eind is TOEV dan het gewenste binomiaalverdeelde toevalsgetal.

Poissonverdeling
De toevalsgetallen nemen in de poissonverdeling een waarde aan van 0, 1, ... met de waarschijnlijkheid

$$p_i = \frac{(X0)^i * EXP(-X0)}{i!}$$

Met het volgende algoritme verkrijgt men een getal met deze verdeling

22.4 Toevalsgenerator

```
TOEV=0
R=1
bereken een (0,1)-uniform verdeeld toevalsgetal TOEVAL
vermenigvuldig R met TOEVAL
doe zolang R <= EXP(-X0)
    verhoog TOEV met 1
    bereken een (0,1)-uniform verdeeld toevalsgetal TOEVAL
    vermenigvuldig R met TOEVAL
```

Geometrische verdeling
Bij deze verdeling nemen de toevalsgetallen één van de waarde 1, 2, ... aan, met de volgende waarschijnlijkheid:

$$p_i = p*(1-p)^{i-1}$$

Met een getal (TOEVAL) uit het uniform verdeelde (0,1) interval wordt als volgt het gewenste getal (TOEV) berekend (X0=p):

TOEV = INT(LOG(TOEVAL)/LOG(1-X0)) + 1

Structuurdiagram
Het structuurdiagram geeft slechts een overzicht van de programmawerking. Met behulp van de hiervoor behandelde formules en de duidelijke programmaindeling is het mogelijk een meer verfijnd structuurdiagram op te stellen.

Enige opmerkingen
De meeste subroutines worden door middel van een GOSUB aangeroepen, hierdoor is het mogelijk het programma zelf als subroutine te gebruiken in een ander programma. Er moet echter wel op gelet worden dat de variabelen de juiste waarde krijgen, daar er in de subroutines niet op getest wordt. Bij het toekennen van verkeerde getallen aan parameters kunnen er dan ook fouten ontstaan.

Het gehele beeldschermmenu komt alleen in de eerste schermweergave naar voren. Daarna wordt alleen nog maar om de verdelingsfunctie gevraagd, waarna de gevraagde toevalsgetallen op het scherm worden weergegeven.

```
Lees de startwaarde Y0
lees het gewenste aantal toevalsgetallen (AANTAL)
voor I=1 tot 100 doe
    bereken een (0,1)-uniform verdeeld toevalsgetal TOEVAL
    Plaats TOEVAL in S(I)
Kies de verdeling
lees de eventuele parameters behorende bij de gekozen verdeling
voor I=1 tot AANTAL doe
    bereken het toevalsgetal TOEV
    geef TOEV weer
```

22.4 Toevalsgenerator

```
100 REM TOEVALSGENERATOR
110 REM
120 REM VOOR DE COMMODORE 64
130 REM
140 REM VARIABELEN:
150 REM
160 REM Y,Z       : VARIABELE VAN DE GENERATOR
170 REM AANTAL    : AANTAL GEWENSTE TOEVALSGETALLEN
175 REM T,T1,T2   : TOEVALSGETALLEN
180 REM X0,X1     : PARAMETERS VOOR DE GEKOZEN VERDELING
190 REM TV        : (0,1)-VERDEELDE TOEVALSGETAL
200 REM H$        : HULPSTRING
210 REM S         : TABEL MET INITIELE WAARDEN
220 REM I,J,K,A   : INDEXEN EN LUSTELLERS
230 REM
240 PRINT
250 PRINT"TOEVALSGENERATOR":PRINT
260 INPUT"STARTWAARDE (1-283)";Y0
270 IF Y0<1 OR Y0>283 THEN PRINT"FOUT...":GOTO 260
280 INPUT"AANTAL GEWENSTE TOEVALSGETALLEN";AANTAL
290 GOSUB 830
300 PRINT
310 PRINT"VERDELING OPGEVEN"
320 PRINT"=================="
330 PRINT"(E)XPONENTIELE VERDELING"
340 PRINT"(N)ORMAALVERDELING (0,1)"
350 PRINT"(A)NDERE INTERV. VOOR GELIJKE VERD."
360 PRINT"(D)ISCRETE VERDELING"
370 PRINT"(B)INOMIAALVERDELING"
380 PRINT"(P)OISSONVERDELING"
390 PRINT"(G)EOMETRISCHE VERDELING"
400 PRINT"RETURN = EINDE"
410 PRINT"-----------------------------------"
420 H$="":INPUT"MAAK EEN KEUS";H$:PRINT
430 IF H$="" THEN END
440 H$=LEFT$(H$,1)
450 REM
460 IF H$<>"E" THEN GOTO 520
470 INPUT"GEEF X0 VOOR EXP. VERDELING";X0
480 IF X0=0 THEN PRINT"FOUT...":GOTO 470
490 FOR A=1 TO AA:GOSUB 1110:PRINTT:NEXT
500 INPUTH$:PRINT:GOTO 300
510 REM
520 IF H$<>"N" THEN GOTO 560
530 FOR A=1 TO AA/2:GOSUB 1170:PRINTT1;T2:NEXT
540 INPUTH$:PRINT:GOTO 300
550 REM
560 IF H$<>"A" THEN GOTO 610
570 INPUT"GEEF ONDERGRENS INTERVAL";X0
580 INPUT"GEEF BOVENGRENS INTERVAL";X1
590 FOR A=1 TO AA:GOSUB 1260:PRINTT:NEXT
600 INPUTH$:PRINT:GOTO 300
605 REM
610 IF H$<>"D" THEN GOTO 670
620 INPUT"KLEINSTE GEHELE GETAL";X0
630 INPUT"GROOTSTE GEHELE GETAL";X1
```

22.4 Toevalsgenerator

```
640 IFXO>=X1 THEN PRINT"FOUT...":GOTO 620
650 FOR A=1 TO AA:GOSUB 1330:PRINTT:NEXT
660 INPUTH$:PRINT:GOTO 300
670 REM
680 IF H$<>"B" THEN GOTO 730
690 INPUT"GEEF WAARSCHIJNLIJKHEID P";XO:PRINTXO
700 INPUT"GEEF AANTAL N";X1
710 FOR A=1 TO AA:GOSUB 1400:PRINTT:NEXT
720 INPUTH$:PRINT:GOTO 300
730 REM
740 IF H$<>"P" THEN GOTO 780
750 INPUT"GEEF POISSONVERD.-PARAMETER";XO
760 FOR A=1 TO AA:GOSUB 1490:PRINTT:NEXT
770 INPUTH$:PRINT:GOTO 300
780 REM
790 IF H$<>"G" THEN PRINT"FOUTE INVOER...":GOTO 300
800 INPUT"GEEF WAARSCH. VOOR GEOM. VERD.";XO
810 FOR A=1 TO AA:GOSUB 1570:PRINTT:NEXT
820 INPUTH$:PRINT:GOTO 300
825 END
830 REM
850 REM SUBROUTINE VOOR INITIALISATIE
900 DIM S(100)
910 FOR I=1 TO 100
920 YO=(13*YO+177)-INT((13*YO+177)/283)*283
930 S(I)=YO/283
940 NEXT
950 ZO=(17*YO+179)-INT((17*YO+179)/281)*281
960 RETURN
970 REM
1000 REM SUBROUTINE OM VOLGEND TOEVALSGETAL TE BEREKENEN
1010 REM  UITVOER STAAT IN DE VARIABELE TV
1020 REM
1040 ZO=(17*ZO+179)-INT((17*ZO+179)/281)*281
1050 I=INT(ZO/281*100)+1
1060 TV=S(I)
1070 YO=(13*YO+177)-INT((13*YO+177)/283)*283
1080 S(I)=YO/283
1090 RETURN
1100 REM
1110 REM BEREKENING VAN EEN TOEVALSGETAL VOOR DE EXP. VERDELING
1120 REM
1130 GOSUB 970
1140 T=-LOG(TV)/XO
1150 RETURN
1160 REM
1170 REM BEREKENING VAN TWEE NORMAAL VERDEELDE TOEVALSGETALLEN
1180 REM
1190 GOSUB 970
1200 XO=TV
1210 GOSUB 970
1220 X1=SQR(-2*LOG(XO))
1230 T1=SIN(6.2832*TV)*X1
1240 T2=COS(6.2832*TV)*X1
1250 RETURN
```

22.4 Toevalsgenerator

```
1260 REM
1270 REM BEREKENING VAN TOEVALSGETAL IN (X0,X1)
1280 REM
1300 GOSUB 970
1310 T=(X1-X0)*TV+X0
1320 RETURN
1330 REM
1340 REM BEREKENING VAN EEN DISCREET TOEVALSGETAL
1350 REM
1370 GOSUB 970
1380 T=INT((X1-X0+1)*TV+X0)
1390 RETURN
1400 REM
1410 REM BEREKENING VAN TOEVALSGETALLEN VOOR DE BINOMIAALVERDELING
1420 REM
1440 T=0
1450 FOR J=1 TO X1
1460 GOSUB 970:IF TV<=X0 THEN T=T+1
1470 NEXT
1480 RETURN
1490 REM
1500 REM BEREKENING VAN EEN TOEVALSGETAL VOOR DE POISSONVERDELING
1510 REM
1530 T=0:T1=1:T2=EXP(-X0)
1540 GOSUB 970:T1=T1*TV
1550 IF T1>T2 THEN T=T+1:GOTO 1540
1560 RETURN
1570 REM
1580 REM BEREKENING VAN EEN TOEVALSGETAL VOOR DE GEOMETRISCHE VERDELING
1590 REM
1600 GOSUB 970
1610 T=INT(LOG(TV)/LOG(1-X0))+1
1630 RETURN

READY.
```

22.4 Toevalsgenerator

```
TOEVALSGENERATOR

STARTWAARDE (1-283) 127
AANTAL GEWENSTE TOEVALSGETALLEN 10

VERDELING OPGEVEN
==================
(E)XPONENTIELE VERDELING
(N)ORMAALVERDELING (0,1)
(A)NDERE INTERV. VOOR GELIJKE VERD.
(D)ISCRETE VERDELING
(B)INOMIAALVERDELING
(P)OISSONVERDELING
(G)EOMETRISCHE VERDELING
RETURN = EINDE
-----------------------------------
MAAK EEN KEUSE

GEEF XO VOOR EXP. VERDELING 1
 .521482918
 1.07073592
 1.98188525
 2.09009884
 .377588738
 .11601781
 2.50995268
 .900514769
 1.60239563
 .233800846


VERDELING OPGEVEN
==================
(E)XPONENTIELE VERDELING
(N)ORMAALVERDELING (0,1)
(A)NDERE INTERV. VOOR GELIJKE VERD.
(D)ISCRETE VERDELING
(B)INOMIAALVERDELING
(P)OISSONVERDELING
(G)EOMETRISCHE VERDELING
RETURN = EINDE
-----------------------------------
MAAK EEN KEUSA

GEEF ONDERGRENS INTERVAL 1
GEEF BOVENGRENS INTERVAL 1000
 68.0706714
 516.385159
 590.515901
 657.586573
 682.29682
 837.618375
 795.257951
 918.809187
 735.24735
 654.056537
```

7/22.5

Combinatoriek

Bij veel problemen in de statistiek wordt bij de berekening van relatieve aantallen (aantallen van voorkomen) en waarschijnlijkheden, methoden uit de mathematische combinatoriek gebruikt. Met het relatieve aantal (van voorkomen) bedoelt men het quotiënt van het aantal dingen, die een bepaalde eigenschap bezitten, en het totale aantal van de op deze eigenschap onderzochte dingen.

Bij het werpen van een munt zijn er twee mogelijkheden: kop of munt. Beide mogelijkheden hebben volgens bovenstaande definitie een relatief aantal (van voorkomen) van 0,5.

De mathematische waarschijnlijkheid is een functie met een waardebereik van 0 to 1, die aangeeft met welke waarschijnlijkheid een bepaalde mogelijkheid kan optreden. Er bestaat een nauwe samenhang tussen de begrippen relatieve aantal (van voorkomen) en waarschijnlijkheid. Dit is duidelijk te zien aan het voorbeeld van het werpen met een munt. De waarschijnlijkheid van het gooien van kop is getalsmatig gelijk aan het bijbehorende relatieve aantal, dus 0,5.

De basis van de combinatoriek is het kiezen van m elementen uit een verzameling van n elementen. Dit is van groot praktisch belang voor het berekenen van relatieve aantallen en waarschijnlijkheden. Als voorbeeld kan misschien een (getallen)lotto dienst doen: Uit 41 getallen van 1 tot en met 41 worden 6 getallen getrokken. Het aantal verschillende mogelijkheden is met behulp van de combinatoriek te berekenen.

In het algemeen moet onderscheid gemaakt worden bij het kiezen van elementen uit een verzameling of de volgorde van belang is of niet. Bij een lotto is het niet van belang in welke volgorde de getallen getrokken worden; alleen de waarde van een getal is van belang. Verder moet bekeken worden of elementen al dan niet meerdere malen mogen voorkomen. Bovenstaande in acht nemende wordt de volgende indeling gemaakt:

Onder *Variatie* V(m,n) verstaat men het aantal manieren waarop m elementen uit een verzameling van n elementen worden gekozen, waarbij de volgorde van belang is. Mag een element meerdere malen voorkomen, dus trekking met teruglegging, dan geldt:

$V(m,n)=n^m$

Indien een element slechts eenmaal mag voorkomen, dus zonder teruglegging, dan geldt:

$$V(m,n) = \frac{n!}{(n-m)!}$$

22.5 Combinatoriek

Indien n=m, treedt er een bijzonder geval van variatie, zonder teruglegging op, men spreekt dan van permutatie en er geldt:

$V(m,n) = P(n) = n!$

Onder *combinaties* C(m,n) verstaat men het kiezen van m uit n elementen waarbij de volgorde *niet* van belang is. Wordt met teruglegging getrokken dan geldt:

$$C(m,n) = \binom{n+m-1}{n-1}$$

Zonder teruglegging geldt:

$$C(m,n) = \binom{n}{m}$$

Het onderscheid tussen permutaties, variaties en combinaties kan het beste duidelijk gemaakt worden aan de hand van een voorbeeld. Stel dat er een verzameling van drie letters (x,y,x) gegeven is, dan ontstaan de volgende berekeningen:

Permutatie van alle drie de elementen: 3!=6
xyz, xzy, yxz, yzx, zxy, zyx

Variatie van twee elementen met teruglegging: $3^2 = 9$
xx, xy, xz, yx, yy, yz, zx, zy, zz

Variatie van twee elementen zonder teruglegging: $\frac{3!}{(3-2)!} = 6$
xy, xz, yx, yz, zx, zy

Combinaties van 2 Elementen met teruglegging: $\binom{3+2-1}{3-1} = 6$
xx, yy, zz, xy, zx, yz

Combinatie van 2 elementen zonder teruglegging: $\binom{3}{2} = 3$
xy, xz, yz

Bij de berekening van variaties en combinaties wordt van faculteiten en binomiaalcoëfficiënten gebruik gemaakt. Deze zijn voor natuurlijke getallen (gehele getallen) respectievelijk als volgt gedefinieerd:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

Nu nog enige formules/vergelijkingen die praktisch zijn bij berekeningen:

$$0! = 1$$

$$\binom{n}{n} = \binom{n}{0} = 1$$

$$\binom{n}{1} = \binom{n}{n-1} = n$$

$$\binom{n}{m} = \frac{n-m+1}{m}\binom{n}{m-1}$$

$$\binom{n}{m} = \frac{n}{m}\binom{n-1}{m-1}$$

$$\binom{n}{m} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Faculteiten en binomiaalcoëfficiënten kunnen vrij grote waarden aannemen. Daardoor kan bij gebruik van de definitievergelijkingen het een en ander nog wel eens uit de hand

lopen. Om deze problemen te voorkomen kan de Stirling benadering voor de natuurlijke logaritme van de faculteit worden gebruikt. Het voordeel is dat deze slechts langzaam in waarde toeneemt.

$$ln(n!) = (N + 0.5)\ ln(N) - n + \ln\sqrt{2\pi}$$

Ook voor de binomiaalcoëfficiënten kan bij grote waarde van de Stirling benadering gebruik worden gemaakt. Echter ook bij kleine waarden is het raadzaam niet van de definitievergelijking uit te gaan. Het is beter van de recursieve definitie gebruik te maken, die uit bovenstaande formules is af te leiden zoals bijvoorbeeld n! = (n-1)!*n.

Het programma

In het programma worden subroutines gebruikt voor de berekening van faculteiten en binimiaalcoëfficiënten alsmede voor de stirling benadering. In het hoofdprogramma COMBINATORIEK wordt van deze subroutines gebruik gemaakt voor de berekening van permutaties, variaties en combinaties.

De logaritme met grondtal 10 en de natuurlijke logaritme bij gebruik in de stirling benadering, worden gedefinieerd als functies met de namen S10 en SL. Omdat in BASIC alleen de natuurlijke logaritme voorhanden is, wordt de functie voor de logaritme met grondtal 10 als volgt berekend:

L10(X) = LOG(X)/LOG(10)

De faculteit n! wordt in de subroutine FACULTEIT direct berekend voor waarden tot n=33. Voor grotere waarden wordt de Stirling benadering gebruikt.

Voor de berekening van de binomiaalcoëfficiënten in de subroutine BINOMIAAL-COEFFICIENTEN wordt de logaritme van de te bepalen coëfficiënten bepaald. Is de waarde kleiner dan 37, dan wordt de recursieve formule gebruikt:

$$\binom{n}{m} = \frac{n-m+1}{m}\binom{n}{m-1}$$

Wis het beeldscherm
Definiëer de functie L10(x) (Logaritme met grondtal 10)
Definiëer de functie SL(N) (Stirlingbenadering)
Defiëer de functie S10(N) (Stirlingbenadering met LOG10)
Lees N en M
Test of M<=N en N>1 en M>0 is. Herhaal anders de invoer van N en M
I1=M
Bereken PM=M! met behulp van de subroutine faculteit en vervolgens P2=log10(M!)
I1=N
Bereken PN=N! met behulp van de subroutine faculteit en vervolgens P2=log10(N!)
Bereken BI=(I1 over I2) met behulp van de subroutine BINOMIAAL-COEFFICIENTEN en vervolgens BL=L10(BI)
C=BI, CL=BL, VL=CL+P2, WL=M*L10(N)

22.5 Combinatoriek

VL>37?	
ja	nee
$V=10^{37}$	V=C*PM
WL > 37	
ja	nee
$W=10^{37}$	W=NM
I1=N+M+1, I2=N-1	
Bereken D=(I1 over I2) met behulp van de subroutine BINOMIAAL-COEFFICIENTEN en vervolgens DL=L10(D)	
Geef op het scherm weer:	
Permutaties van N:	PN en P1
Permutaties van M:	PM en P2
Variaties zonder teruglegging:	V en VI
Combinaties zonder teruglegging:	C en CL
Variaties met teruglegging:	W en WL
Combinaties met teruglegging:	D en DL

Structuurdiagram van het hoofdprogramma

FA=0, FL=0		
I1 < 0		
ja	nee	
Terug naar het aanroepende programma	I1 < 34	
	ja	nee
	FA=1	$FA=10^{37}$
	Voor I0=2 tot I1 doe FA=FA*10	Functieaanroep FL=L10(FA)
	Functieaanroep FL=S10(I1)	Terug naar het aanroepende programma
	Terug naar het aanroepende programma	

Structuur van de subroutine FACULTEIT

22.5 Combinatoriek

BI=0, BL=0	
Test of: I2>I1 of I1<0 of I2<0 zo ja, terug naar aanroepende programma.	
BI=1, I4=I1-I2	
Test of: I2=I1 of I1=0 of I2=0 zo ja, terug naar aanroepende programma.	
Functieaanroep: BL=S10(I1)-S10(I4)-S10(I2)	
BL < 37 ja	nee
BI=I1, I3=I1	$BI=10^{37}$
Voor I0=2 tot I4 doe	Terug naar het aanroepende programma
I3=I3-1, BI=BI/I0*I3	
Functieaanroep: BL=L10(BI)	
Terug naar het aanroepende programma	

Structuurdiagram van de subroutine BINOMIAAL-COEFFICIENTEN

Voorbeeld 1

Het aantal mogelijkheden van het trekken van 6 getallen uit de rij 1 tot en met 41 moet worden berekend. Het gaat hierbij om trekking zonder teruglegging. Dit betreft dus combinaties zonder teruglegging en het resultaat is

$$\binom{41}{6} = 4496388$$

```
COMBINATORIEK

N, M = 41 , 6
AANTAL MANIEREN ZONDER HERHALEN:
PERMUTATIES VAN N > 1E37,    LOG = 49.5235463
PERMUTATIES VAN M = 720 ,   LOG = 2.8573325
VARIATIES (N,M)    = 3.23739936E+09 ,   LOG = 9.51019628
COMBINATIES (N,M) = 4496388 ,   LOG = 6.65286378

AANTAL MANIEREN MET HERHALEN:
VARIATIES (N,M)    = 4.75010425E+09 ,   LOG = 9.67670314
COMBINATIES (N,M) = 9366819 ,   LOG = 6.97159213
```

22.5 Combinatoriek

Voorbeeld 2

Nu dient de waarschijnlijkheid berekend te worden dat bij het trekken van 6 getallen uit 41 er m goede zijn. Hiertoe dient de definitie van het relatieve aantal (van voorkomen) gebruikt te worden. Het aantal mogelijke gevallen is het aantal verschillende mogelijkheden 6 uit 41 getallen te trekken. Dit is reeds in voorbeeld één berekend. Nu dient nog de voor de speler (bij een lotto) gunstige mogelijkheden te worden berekend dat m aangegeven getallen een combinatie van de zes getrokken getallen zijn. Dit getal dient dan nog vermenigvuldigd te worden met het aantal mogelijkheden van 6-m getallen uit 35 getallen te trekken. Dit levert dan:

$$w(m) = \frac{\binom{6}{m}\binom{35}{6-m}}{\binom{41}{6}}$$

Deze waarschijnlijkheidsverdeling van het (lotto)voorbeeld is van groot algemeen belang. Het staat bekend onder de naam hypergeometrische verdeling, met als algemene vorm:

$$w(j) = \frac{\binom{k}{j}\binom{n-k}{m-j}}{\binom{n}{m}}$$

Een voorbeeld is een fabriek die onderdelen fabriceert. Van n delen moeten er k bruikbaar zijn en zodoende n-k onderdelen onbruikbaar. Als men een steekproef houdt onder n produkten waarbij m produkten worden bekeken, dan is de waarschijnlijkheid te berekenen dat j produkten niet in orde zijn. Deze berekening gebeurt dan door middel van bovengenoemde formule voor de hypergeometrische formule.

Maar nu weer terug naar het (lotto)voorbeeld. om de waarschijnlijkheid aan te kunnen geven dat 4 van de 6 getrokken getallen goed zijn wordt het volgende met het programma COMBINATORIEK berekend:

$$\binom{6}{4} = 15, \quad \binom{35}{2} = 595, \quad \binom{41}{6} = 4496388$$

Dit geeft dan w(4)= 15*595/4496388 = 0.0019849. De kans dat vier getallen goed zijn is ongeveer één op vijfhonderd.

Nu dan de kans dat alle zes de getallen goed zijn:

$$\binom{6}{6} = 1, \quad \binom{35}{0} = 1, \quad \binom{41}{6} = 4496388$$

Dit geeft dan w(6)= 1/4496388. Erg klein dus, ongeveer één op de vierenhalf miljoen.

```
N, M = 6 , 4
AANTAL MANIEREN ZONDER HERHALEN:
PERMUTATIES VAN N = 720 ,   LOG = 2.8573325
PERMUTATIES VAN M = 24 ,   LOG = 1.38021124
VARIATIES (N,M)   = 360 ,   LOG = 2.5563025
COMBINATIES (N,M) = 15 ,   LOG = 1.17609126

AANTAL MANIEREN MET HERHALEN:
VARIATIES (N,M)   = 1296 ,   LOG = 3.112605
COMBINATIES (N,M) = 126 ,   LOG = 2.10037055
```

22.5 Combinatoriek

```
N, M = 35 , 2
AANTAL MANIEREN ZONDER HERHALEN:
PERMUTATIES VAN N > 1E37,    LOG = 40.0131987
PERMUTATIES VAN M = 2 ,   LOG = .301029996
VARIATIES (N,M)    = 1190 ,   LOG = 3.07554696
COMBINATIES (N,M) = 595 ,   LOG = 2.77451697

AANTAL MANIEREN MET HERHALEN:
VARIATIES (N,M)    = 1225 ,   LOG = 3.08813609
COMBINATIES (N,M) = 630 ,   LOG = 2.79934055

N, M = 0 , 0


READY.
```

```
1000 REM ************************************************
1010 REM            C O M B I N A T O R I E K
1020 REM ************************************************
1030 REM
1040 REM C-64 VERSIE
1050 REM
1060 REM INVOER:
1070 REM - AANTAL ELEMENTEN                    : N
1080 REM - AANTAL TE KIEZEN ELEMENTEN          : M
1100 REM UITVOER
1110 REM - PERMUTATIE VAN N       = PN,   P1 = L10(PN)
1120 REM - PERMUTATIE VAN M       = PM,   P2 = L10(PM)
1130 REM - VARIATIE (N,M) MET
1140 REM   HERHALINGEN            = V,    VL = L10(V)
1150 REM - VARIATIE (N,M) ZONDER
1160 REM   HERHALINGEN            = W,    WL = L10(W)
1170 REM - KOMBINATIE (N,M) MET
1180 REM   HERHALINGEN            = C,    CL = L10(C)
1190 REM - KOMBINATIE (N,M) ZONDER
1200 REM   HERHALINGEN            = D,    DL = L10(D)
1210 REM
1220 REM ------------------------------------------------
1230 PRINT"■"
1240 PRINT"COMBINATORIEK"
1250 DEF FN L10(X) = LOG(X)/LOG(10)              :REM 10-LOG
1260 DEF FN SL(N)  = (N+.5)*LOG(N)-N+.9189386    :REM LN(N!)
1270 DEF FN S10(N) = FN SL(N)/LOG(10)            :REM LOG(N!)
1280 REM
1300 REM
1310 PRINT
1320 INPUT"N, M =";N,M
1325 IF M=0 AND N=0 THEN END
1330 IF M<=N AND N>1 AND M>0 THEN 1400
1340 PRINT"FOUT! N>M, M>0 EN N>1"
1350 PRINT
1360 GOTO 1320
```

22.5 Combinatoriek

```
1370 REM
1380 REM BEREKENING VAN PERMUTATIE, VARIATIE EN COMBINATIE
1390 REM
1400 I1=M
1410 GOSUB 1900 :REM FACULTEIT
1420 PM=FA
1430 P2=FL
1440 I1=N
1450 GOSUB 1900
1460 PN=FA
1470 P1=FL
1480 I2=M
1490 GOSUB 2100 :REM BINOMIAAL-COEFFICIENTEN
1500 C=BI
1510 CL=BL
1520 VL=CL+P2
1530 V=C*PM:IF VL>37 THEN V=1E37
1540 WL=M*FNL10(N)
1550 W=N↑M:IF WL>37 THEN W=1E37
1560 I1=N+M-1
1570 I2=N-1
1580 GOSUB 2100
1590 D=BI
1600 DL=BL
1610 REM
1620 REM UITVOER VAN RESULTATEN
1630 REM
1640 PRINT"AANTAL MANIEREN ZONDER HERHALEN:"
1650 IF P1<37 THEN PRINT"PERMUTATIES VAN N =";PN;",  LOG ="P1
1660 IF P1>=37 THEN PRINT"PERMUTATIES VAN N > 1E37,   LOG ="P1
1670 IF P2<37 THEN PRINT"PERMUTATIES VAN M =";PM;",  LOG ="P2
1680 IF P2>=37 THEN PRINT"PERMUTATIES VAN M > 1E37,   LOG ="P2
1690 IF VL<37 THEN PRINT"VARIATIES (N,M)   =";V;",  LOG ="VL
1700 IF VL>=37 THEN PRINT"VARIATIES (N,M)   > 1E37,   LOG ="VL
1710 IF CL<37 THEN PRINT"COMBINATIES (N,M) =";C;",  LOG ="CL
1720 IF CL>=37 THEN PRINT"COMBINATIES (N,M) > 1E37,   LOG ="CL
1730 PRINT
1740 PRINT"AANTAL MANIEREN MET HERHALEN:"
1750 IF WL<37 THEN PRINT"VARIATIES (N,M)   =";W;",  LOG ="WL
1760 IF WL>=37 THEN PRINT"VARIATIES (N,M)   > 1E37,   LOG ="WL
1770 IF DL<37 THEN PRINT"COMBINATIES (N,M) =";D;",  LOG ="DL
1780 IF DL>=37 THEN PRINT"COMBINATIES (N,M) > 1E37,   LOG ="DL
1790 PRINT
1800 GOTO 1320
1900 REM
```

22.5 Combinatoriek

```
1910 REM
1920 REM ************************************************
1930 REM SUBROUTINE      F A C U L T E I T
1940 REM ************************************************
1950 REM
1960 FA=0
1970 FL=0
1980 IF I1<0 THEN RETURN
1990 IF I1<34 THEN 2030
2000 FA=1E37
2010 FL=FNS10(I1)
2020 RETURN
2030 FA=1
2040 FOR I0=2 TO I1
2050 :    FA=FA*I0
2060 NEXT
2070 FL=FNL10(FA)
2080 RETURN
2090 REM
2100 REM ************************************************
2110 REM  B I N O M I A A L - C O E F F I C I E N T E N
2120 REM ************************************************
2130 REM
2140 BI=0
2150 BL=0
2160 IF I2>I1 OR I1<0 OR I2<0 THEN RETURN
2170 BI=1
2180 IF I2=0 OR I1=0 OR I2=I1 THEN RETURN
2190 I4=I1-I2
2200 BL=FNS10(I1)-FNS10(I4)-FNS10(I2)
2210 IF BL<37 THEN 2240
2220 BI=1E37
2230 RETURN
2240 BI=I1
2250 I3=I1
2260 FOR I0=2 TO I4
2270 :    I3=I3-1
2280 :    BI=BI/I0*I3
2290 NEXT
2300 BL=FNL10(BI)
2310 RETURN

READY.
```

7/23

Handige toepassingen

Inhoud

7/23.1

1 uit 7...

De kop slaat op het volgende: bij dit artikel horen 7 losse en afzonderlijk incomplete listings. Deze listings combineert u naar believen tot 1 programma, een bestandsprogramma. En dit bestandsprogramma is op zich ook weer te wijzigen. Nu is het (slechts) een adressenbestand, maar het is een middagje werk om er bijvoorbeeld een platen- of, nog beter, een software-bestand van te maken.

Even resumeren. De eerste listing is het complete, modulair geprogrammeerde, BASIC-raamwerk van het bestandsprogramma. De volgende drie kleine listings zijn alle drie zoek-modules. De laatste drie listings zijn sorteer-modules. Om het bestandsprogramma volledig te laten werken kiest u eenmalig uit zowel de zoek- alsmede de sorteer-routines (in principe geldt: hoe ingewikkelder hoe beter, en dus sneller).

Eerst bouwen
Voordat we ingaan op de werking van het programma dient u het eerst over te nemen. U begint natuurlijk met het intypen van de eerste listing. Heeft u dit gedaan, dan zal het u zijn opgevallen dat er GOSUB-routines zijn naar (nog) niet-bestaande adressen. De GOSUB 3000, wil naar een sorteer-routine. De GOSUB 4000, wil naar een zoek-routine.
Mocht u nu beschikken over een goede toolkit of zo maar een losse merge-utility, dan wordt het erg leuk. Typt u dan namelijk de afzonderlijke zoek- en sorteer-routines maar eens in. En omdat u kunt mergen, kunt u ook testen hoe de verschillende routines in combinatie met elkaar werken! Vaak zijn de resultaten opmerkelijk te noemen.

Werken met data
Na opstarten verschijnt er al gauw een menu waaruit u (nog) met de numerieke toetsen kunt kiezen. Het programma werkt heel eenvoudig. U toetst als er gevraagd wordt, de database doet de rest.

<1>: is om gegevens in te voeren. U voert in, de naam, het adres en de woonplaats, waarna er nog even een goedkeuring wordt gevraagd. Zo goed, dan gaat u terug naar het hoofdmenu. Zo fout, dan mag u de data opnieuw invoeren.
<2>: is om gegevens te bekijken; mits er gegevens aan boord zijn. De velden komen één voor één langs, en het wachten is op een toetsindruk.
<3>: sorteren. U zegt waarop er gesorteerd dient te worden en de routine, mits aanwezig, doet het werk.
<4>: zoeken. Wederom dient u in te geven op welk onderdeel u wilt zoeken. Waarna u ook een sleutel dient in te geven.
<5>: verwijderen. Eerst moet u een te

23.1 1 uit 7...

verwijderen veld met de zoek-routine opzoeken om daarna te kiezen voor het weghalen.
<6>: printen. Eenzelfde werking als het bekijken van de gegevens op het scherm, maar nu naar de printer.
<7>: gegevens saven. Alle data gaan naar een door u gekozen, en in het programma in te stellen, randapparaat. Later kunt u deze data weer inlezen met de load-mogelijkheid.
<8>:gegevens laden. Eerder opgeslagen data kunt u nu weer binnen lezen. Misschien om het bestand te onderzoeken, dan wel om gegevens toe te voegen of te verwijderen.
<9>: stoppen. Spreekt voor zich.

Sorteren...
We hebben drie sorteer-algoritmen ter beschikking. 'Lineair sorteren', 'Bubble sorteren' en 'Quick sorteren'. De eerste is het eenvoudigst te doorgronden, de laatste is het snelst. Het resultaat is altijd eender: een perfect gesorteerd bestand.

Lineair sorteren begint bij het begin en onthoud altijd de 'overige' kaart. Is de volgende kaart groter, dan wordt deze onthouden en wordt er verder gezocht. Is de kaart kleiner, dan worden de 'vorige' en 'deze' verwisseld. U gaat net zolang door met naar voren halen, totdat alle 'grotere' elementen elkaar opvolgen.

Bubble sorteren is erg populair onder de machinetaal-programmeurs, het is dan ook erg simpel en geheugen-vriendelijk. U begint achteraan te zoeken. Telkens als u een kleiner element vindt, verwisselt u deze met zijn voorganger. Als u dit doet komt u uiteindelijk vooraan met het kleinste element. Dit spelletje begint weer van achter af aan, net zolang totdat u lengte-1 keer geweest bent.

Quick sorteren is zo snel omdat er pas in een latere instantie wordt verplaatst. Met twee arrays. De originele array en een hulp-array worden gecombineerd. In de hulp-array staat informatie over de opeenvolging van de elementen, terwijl de originele array nog even ongewijzigd blijft. Pas in een latere instantie wordt de data verplaatst.
Zeker als een bestand erg groot is en over vele lange informatiedragers beschikt is dit veruit de snelste sorteer-routine.
U kunt natuurlijk ook besluiten om alle drie de sorteer-routines mee te mergen, en met behulp van een keyboard-indruk en een 'ON... GOSUB' te reageren.

Zoeken...
Een vervelende kwestie. Er zijn namelijk twee wezenlijk verschillende soorten zoek-routines: de zoek-routines die zoeken in een gesorteerd bestand, en de zoek-routines die zoeken in een niet-gesorteerd bestand.
Zeker die laatste keuze bepaalt de snelheid van een zoek-routine. Het gericht zoeken in een gesorteerd bestand gaat namelijk altijd sneller dan het één-voor-één vergelijken van alle mogelijke elementen. Ook drie routines 'Lineair zoeken', 'Half/half zoeken' en 'Binair zoeken'.

Lineair zoeken begint vooraan in de array en werkt één voor één alle elementen af. Pas aan het einde gekomen is bekend of een element wel of niet aanwezig is. Een langzame routine die echter geschikt is voor zoeken in een ongesorteerde array!

Half/Half zoeken prikt in het midden van het array en begint vanaf daar naar 'beneden' te zoeken als het gezochte element kleiner is dan het op die positie gevonden element. Anders zoekt de routine naar 'boven'.

23.1 1 uit 7...

Met deze routine kort u de zoektijd in een grotere array behoorlijk in. Zeker als het element niet wordt gevonden is de routine precies twee keer zo snel als de vorige. Helaas werkt de routine slechts dan perfect als het gaat om een gesorteerde array.

Binair zoeken, wederom een populaire routine, die alleen werkt op gesorteerde arrays. De routine begint evenals de vorige 'in het midden', er wordt besloten naar 'beneden' of naar 'boven' te zoeken. Na de beslissing gaat de routine op de helft-van-de-helft zitten en neemt weer zo'n zelfde richtings-beslissing. Dit gaat net zolang door totdat het element wel of niet is gevonden.

De snelste routine als het gaat om de wat grotere bestanden. Zeker als het element niet wordt gevonden, dan weet u dit in een oogwenk.

Aan te raden is het om in ieder geval de lineaire zoek-routine in te programmeren. Eventueel pikt u daar nog eentje van de andere twee bij. Als extra optie kunt u dan inbouwen op welke van de drie mogelijkheden het bestand is gesorteerd en of het op dit moment wel is gesorteerd. Er kan tenslotte een bestandsverandering plaats hebben gevonden. Wordt dan gezocht, dan besluit u voor een gesorteerde of een niet-gesorteerde mogelijkheid.

```
1000 REM ---- OPSTARTEN ----
1010 :
1020 CLR:GOSUB2420:GOTO1100
1030 :
1040 REM ---- WACHTEN NA MEDEDELING ----
1050 :
1060 WAIT197,64,64
1070 :
1080 REM ---- MENU VERZORGEN ----
1090 :
1100 GOSUB2500:GOSUB2710
1110 ONJPGOTO1110,1130,1250,1390,1530,1720,1900,2060,2200,2340
1120 :
1130 REM ---- INVOEREN ----
1140 :
1150 PRINT"{CLR} ┌──────────┐ "
1160 PRINT" │INVOEREN│ "
1170 PRINT" └──────────┘ {C/DN}"
1180 IFV>MTHENPRINT"{C/DN}{C/DN}*** HET BESTAND ZIT VOL!! ***":GOTO1060
1190 : FORA=1TO3
1200 :     PRINTNM$(A);:INPUTBS$(A,V)
1210 : NEXTA
1220 GOSUB2790:IFJP=2THEN1150
1230 V=V+1:GOTO1100
1240 :
1250 REM ---- BEKIJKEN ----
1260 :
1270 PRINT"{CLR} ┌──────────┐ "
1280 PRINT" │BEKIJKEN│ "
1290 PRINT" └──────────┘ {C/DN}"
1300 IFV=1THENPRINT"{C/DN}{C/DN}*** HET BESTAND IS LEEG!! ***":GOTO1060
```

23.1 1 uit 7...

```
1310 : FORA=1TOV-1
1320 :     FORB=1TO3
1330 :         PRINTNM$(B);:PRINTBS$(B,A)
1340 :     NEXTB
1350 :     PRINT"{C/DN}------{C/DN}":WAIT197,64,64
1360 : NEXTA
1370 GOTO1100
1380 :
1390 REM ---- SORTEREN ----
1400 :
1410 PRINT"{CLR} ┌──────────┐"
1420 PRINT"│SORTEREN│"
1430 PRINT"└──────────┘{C/DN}"
1440 IFV<3THENPRINT"{C/DN}{C/DN}*** NIETS TE SORTEREN!! ***":GOTO1060
1450 PRINT" ┌──────────────────────┐"
1460 PRINT"│  1 - NAAM          │"
1470 PRINT"│  2 - ADRES         │"
1480 PRINT"│  3 - WOONPLAATS    │"
1490 PRINT"└──────────────────────┘"
1500 GOSUB2730:IFJP=0ORJP>4THEN1500
1510 GOSUB3000:GOTO1100
1520 :
1530 REM ---- ZOEKEN ----
1540 :
1550 PRINT"{CLR} ┌──────────┐"
1560 PRINT"│OPZOEKEN│"
1570 PRINT"└──────────┘{C/DN}"
1580 IFV<2THENPRINT"{C/DN}{C/DN}*** ER IS NIETS OM TE ZOEKEN!! ***":GOTO1060
1590 PRINT" ┌──────────────────────┐"
1600 PRINT"│  1 - NAAM          │"
1610 PRINT"│  2 - ADRES         │"
1620 PRINT"│  3 - WOONPLAATS    │"
1630 PRINT"└──────────────────────┘{C/DN}"
1640 GOSUB2730:IFJP=0ORJP>4THEN1640
1650 INPUT"WAT ZOEKT U";ZO$
1660 GOSUB4000:IFIZ=0THENPRINT"{C/DN}*** NIET GEVONDEN!! ***":GOTO1060
1670 PRINT"{C/DN}"
1680 : FORA=1TO3
1690 :     PRINTNM$(A);BS$(A,IZ)
1700 : NEXTA
1710 GOTO1060
1720 :
1730 REM ---- WEGHALEN ----
1740 :
1750 PRINT"{CLR} ┌──────────┐"
1760 PRINT"│WEGHALEN│"
1770 PRINT"└──────────┘{C/DN}"
1780 IFIZ=0THENPRINT"{C/DN}{C/DN}*** NOG NIETS OPGEZOCHT!! ***":GOTO1060
1790 PRINT"{C/DN}"
1800 : FORA=1TO3
1810 :     PRINTNM$(A);BS$(A,IZ)
1820 : NEXTA
1830 PRINT"{C/DN}WILT U DIT VELD WEGHALEN?":GOSUB2770:IFJP=2THEN
1840 V=V-1
1850 : FORA=0TO3
1860 :     BS$(A,IZ)=BS$(A,V):BS$(A,V)=""
```

23.1 1 uit 7...

```
1870 : NEXTA
1880 IZ=0:GOTO1080
1890 :
1900 REM ---- PRINTEN ----
1910 :
1920 PRINT"{CLR} ┌──────────┐"
1930 PRINT"│PRINTEN│"
1940 PRINT"└──────────┘{C/DN}"
1950 IFV=1THENPRINT"{C/DN}{C/DN}*** ER VALT NIETS TE PRINTEN!! ***":GOTO1060
1960 PRINT"{C/DN}PRINTER STAAT AANGESLOTEN...":GOSUB2770:IFJP=2THEN2770
1970 OPEN4,4
1980 : FORA=1TOV-1
1990 :     FORB=1TO3
2000 :         PRINT#4,NM$(B);:PRINT#4,BS$(B,A)
2010 :     NEXTB
2020 :     PRINT#4,"------"
2030 : NEXTA
2040 CLOSE4:GOTO1100
2050 :
2060 REM ---- SAVEN ----
2070 :
2080 PRINT"{CLR} ┌────────┐"
2090 PRINT"│SAVEN│"
2100 PRINT"└────────┘{C/DN}"
2110 IFV=1THENPRINT"{C/DN}{C/DN}*** ER VALT NIETS TE SAVEN!! ***":GOTO1060
2120 INPUT"{C/DN}FILE-NAAM";A$:OPEN1,DV,2,A$+",S,W":PRINT#1,V
2130 : FORA=1TOV-1
2140 :     FORB=1TO3
2150 :         PRINT#1,BS$(B,A)
2160 :     NEXTB
2170 : NEXTA
2180 CLOSE1:GOTO1100
2190 :
2200 REM ---- LADEN ----
2210 :
2220 PRINT"{CLR} ┌────────┐"
2230 PRINT"│LADEN│"
2240 PRINT"└────────┘{C/DN}"
2250 IFV>1THENGOSUB2770:IFJP=2THEN1100
2260 CLR:GOSUB2420:INPUT"{C/DN}FILE-NAAM";A$:OPEN1,DV,2,A$+",S,R":INPUT#1,V
2270 : FORA=1TOV-1
2280 :     FORB=1TO3
2290 :         INPUT#1,BS$(B,A)
2300 :     NEXTB
2310 : NEXTA
2320 CLOSE1:GOTO1100
2330 :
2340 REM ---- STOPPEN ----
2350 :
2360 PRINT"{CLR} ┌──────────┐"
2370 PRINT"│STOPPEN│"
2380 PRINT"└──────────┘{C/DN}"
2390 GOSUB2770:IFJP=2THEN1100
2400 END
2410 :
2420 REM ---- INITIALISATIE ----
```

23.1 1 uit 7...

```
2430 :
2440 POKE53280,0:POKE53281,0:POKE53272,20:PRINTCHR$(8)
2450 V=1:M=50:DV=8:DIM BS$(3,M+1),NM$(3),BS(M+1)
2460 NM$(1)="NAAM       : "
2470 NM$(2)="ADRES      : "
2480 NM$(3)="WOONPLAATS: ":RETURN
2490 :
2500 REM ---- MENU PRINTEN ----
2510 :
2520 POKE53265,PEEK(53265)AND236
2530 PRINT"{CLR}{YELO}{C/DN}{C/DN}                    ┌──────────────────────┐"
2540 PRINT"              │DEMO-DATABASIS│"
2550 PRINT"              └──────────────────────┘{C/DN}"
2560 PRINT"       ┌──────────────────────────────────┐"
2570 PRINT"       │ 1 - GEGEVENS INVOEREN    │"
2580 PRINT"       │ 2 - GEGEVENS BEKIJKEN    │"
2590 PRINT"       │ 3 - GEGEVENS SORTEREN    │"
2600 PRINT"       │ 4 - GEGEVENS OPZOEKEN    │"
2610 PRINT"       │ 5 - GEGEVENS WEGHALEN    │"
2620 PRINT"       │ 6 - GEGEVENS PRINTEN     │"
2630 PRINT"       │ 7 - GEGEVENS SAVEN       │"
2640 PRINT"       │ 8 - GEGEVENS LADEN       │"
2650 PRINT"       │ 9 - STOPPEN              │"
2660 PRINT"       ├──────────────────────────────────┤"
2670 PRINT"       │DRUK OP DE GEWENSTE TOETS│"
2680 PRINT"       └──────────────────────────────────┘"
2690 POKE53265,PEEK(53265)OR16:RETURN
2700 :
2710 REM ---- CIJFER OPHALEN ----
2720 :
2730 GETA$:IFA$=""THEN2730
2740 IFASC(A$)<48ORASC(A$)>57THEN2730
2750 JP=ASC(A$)-47:RETURN
2760 :
2770 REM ---- (J)A OF (N)EE ----
2780 :
2790 PRINT"{C/DN}IS DIT IN ORDE (J/N)"
2800 GETA$:IFA$=""OR(A$<>"J"ANDA$<>"N")THEN2800
2810 JP=1:IFA$="N"THENJP=2
2820 RETURN

READY.
```

23.1 1 uit 7...

```
4000 :
4010 REM ----- LINEAIR ZOEKEN -----
4020 :
4030 IZ=1
4040 IFBS$(JP-1,IZ)=ZO$THEN:RETURN
4050 IZ=IZ+1:IFIZ=VTHENIZ=0:RETURN
4060 GOTO4040

READY.
```

```
4000 :
4010 REM ----- HALF/HALF ZOEKEN -----
4020 :
4030 IZ=INT(V/2)
4040 IFZO$>BS$(JP-1,IZ)THEN4090
4050 IFIZ=0ORZO$>BS$(JP-1,IZ)THENIZ=0:RETURN
4060 IFZO$=BS$(JP-1,IZ)THENRETURN
4070 IZ=IZ-1:GOTO4050
4080 :
4090 IFIZ=VORZO$<BS$(JP-1,IZ)THENIZ=0:RETURN
4100 IFZO$=BS$(JP-1,IZ)THENRETURN
4110 IZ=IZ+1:GOTO4090

READY.
```

23.1 1 uit 7...

```
4000 :
4010 REM ---- BINAIR ZOEKEN ----
4020 :
4030 LZ=0:HZ=V
4040 IZ=INT((LZ+HZ)/2)
4050 IFIZ=LZTHENIZ=0:RETURN
4060 IFZO$<BS$(JP-1,IZ)THENHZ=IZ:GOTO4040
4070 IFZO$=BS$(JP-1,IZ)THENRETURN
4080 LZ=IZ:GOTO4040

READY.
```

```
3000 :
3010 REM ---- LINEAIR SORTEREN ----
3020 :
3030 FORIS=1TOV-1
3040 :   FORA=1TO3
3050 :      BS$(A,0)=BS$(A,IS)
3060 :   NEXTA:JS=IS-1
3070 :   IFBS$(JP-1,0)>=BS$(JP-1,JS)THEN3120
3080 :   FORA=1TO3
3090 :      BS$(A,JS+1)=BS$(A,JS)
3100 :   NEXTA:JS=JS-1
3110 :   IFJS<>0THEN3070
3120 :   FORA=1TO3
3130 :      BS$(A,JS+1)=BS$(A,0)
3140 :   NEXTA
3150 NEXTIS
3160 RETURN

READY.
```

23.1 1 uit 7...

```
3000 :
3010 REM ----- BUBBLE SORTEREN -----
3020 :
3030 FORIS=1TOV-1
3040 :   FORJS=V-1TOISSTEP-1
3050 :       IFBS$(JP-1,JS-1)>BS$(JP-1,JS)THEN3070
3060 :       GOTO3120
3070 :       FORA=1TO3
3080 :           BS$(A,0)=BS$(A,JS)
3090 :           BS$(A,JS)=BS$(A,JS-1)
3100 :           BS$(A,JS-1)=BS$(A,0)
3110 :       NEXTA
3120 :   NEXTJS
3130 NEXTIS
3140 RETURN

READY.
```

23.1 1 uit 7...

```
3000 :
3010 REM ----- QUICK SORTEREN -----
3020 :
3030 ZS=1:BS(ZS)=V:MS=1
3040 JS=BS(ZS):IS=MS-1
3050 IFJS-MS<3THEN3290
3060 NS=INT((IS+JS)/2)
3070 IS=IS+1
3080 IFIS=JSTHEN3190
3090 IFBS$(JP-1,IS)<=BS$(JP-1,NS)THEN3070
3100 JS=JS-1
3110 IFIS=JSTHEN3190
3120 IFBS$(JP-1,JS)>=BS$(JP-1,NS)THEN3100
3130 : FORA=1TO3
3140 :     BS$(A,0)=BS$(A,IS)
3150 :     BS$(A,IS)=BS$(A,JS)
3160 :     BS$(A,JS)=BS$(A,0)
3170 : NEXTA
3180 GOTO3070
3190 IFIS>=NSTHENIS=IS-1
3200 IFJS=NSTHEN3260
3210 : FORA=1TO3
3220 :     BS$(A,0)=BS$(A,IS)
3230 :     BS$(A,IS)=BS$(A,NS)
3240 :     BS$(A,NS)=BS$(A,0)
3250 : NEXTA
3260 ZS=ZS+1
3270 BS(ZS)=IS
3280 GOTO3040
3290 IFJS-MS<2THEN3360
3300 IFBS$(JP-1,MS)<BS$(JP-1,MS+1)THEN3360
3310 : FORA=1TO3
3320 :     BS$(A,0)=BS$(A,MS)
3330 :     BS$(A,MS)=BS$(A,MS+1)
3340 :     BS$(A,MS+1)=BS$(A,0)
3350 : NEXTA
3360 MS=BS(ZS)+1
3370 ZS=ZS-1
3380 IFZS>0THEN3040
3390 RETURN

READY.
```

7/23.2

Heeft u ook stickers?

Veel Commodore 64-gebruikers beschikken over een printer. De meeste mensen gebruiken deze printer vervolgens voor tekstverwerkings-doeleinden. Een bijzonder vervelend klusje vanuit de tekstverwerker is het afdrukken van stickers met namen, adressen en woonplaatsen.
Speciaal voor dit doel een kleine BASIC-listing. De listing opent een kanaal naar de printer (device-nummer 4), vraagt om de juiste data ('XXX' is onderbreken), vraagt of de gegevens in orde zijn, en drukte deze vervolgens af.

Het spreekt voor zich dat deze routine simpel is te wijzigen. U kunt een extra INPUT-commando toevoegen voor nog een extra data-regel. Op de lengte van de sticker kunt u inspelen door het aantal losse PRINT-opdrachten, als in de regels 1180 en 1190, te laten variëren.

```
1000 OPEN4,4
1010 INPUT"      NAAM:";N$
1020 IF N$="XXX" THEN 1210
1030 INPUT"     ADRES:";A$
1040 INPUT" POSTCODE:";C$
1050 INPUT"    PLAATS:";P$
1060 PRINT
1070 PRINTN$
1080 PRINTA$
1090 PRINTC$;"   ";P$
1100 PRINT
1110 PRINT"IS DIT IN ORDE (J/N)"
1120 GETI$:IFI$<>"J"ANDI$<>"N"THEN1120
1130 IFI$="N"THEN1010
1140 PRINT#4,N$
1150 PRINT#4
1160 PRINT#4,A$
1170 PRINT#4,C$;"   ";P$
1180 PRINT#4
1190 PRINT#4
1200 GOTO 1010
1210 CLOSE4
1220 END

READY.
```