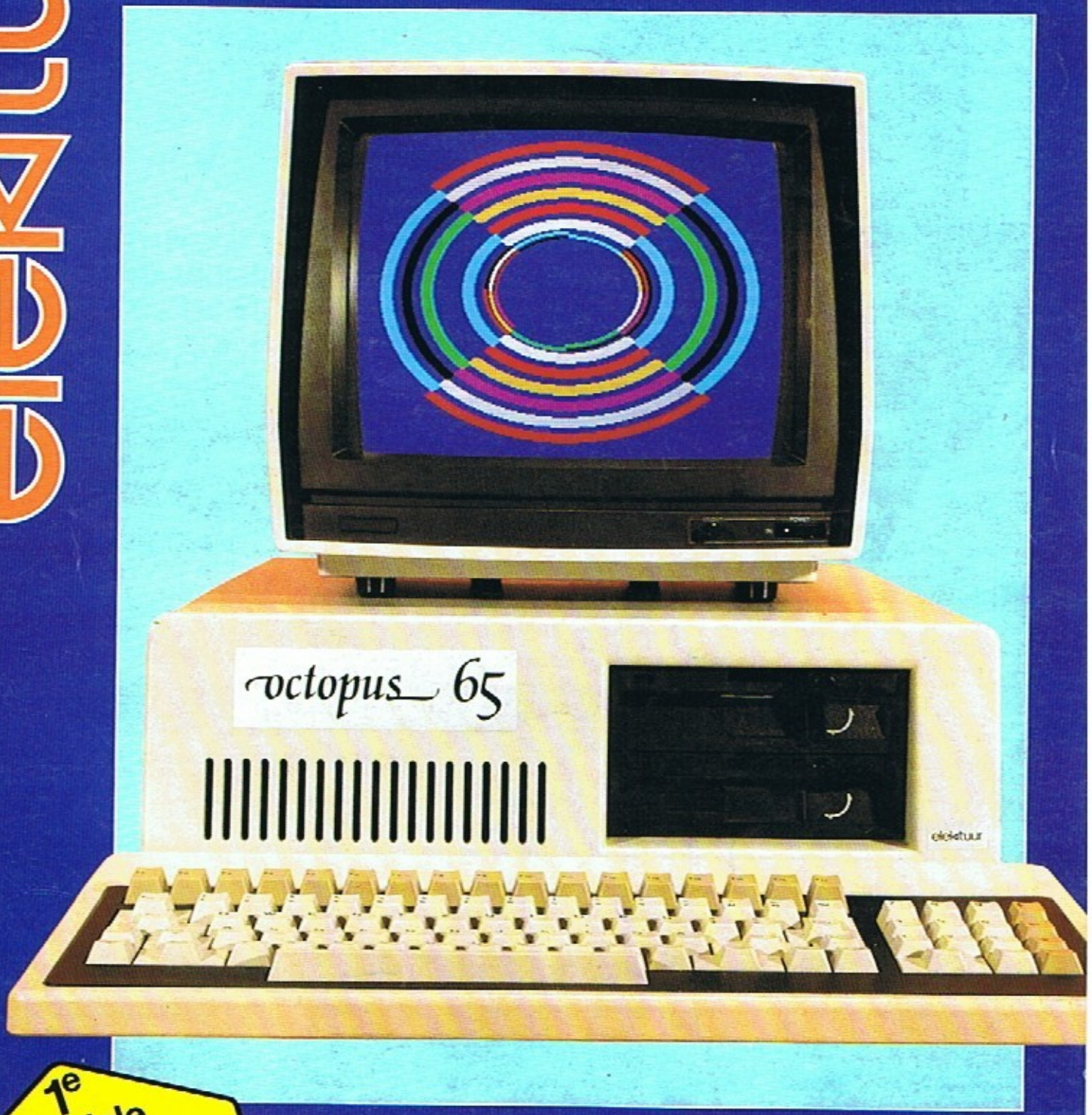
f 18,75 / Bfrs. 370

elektuur extra

COMPUTING

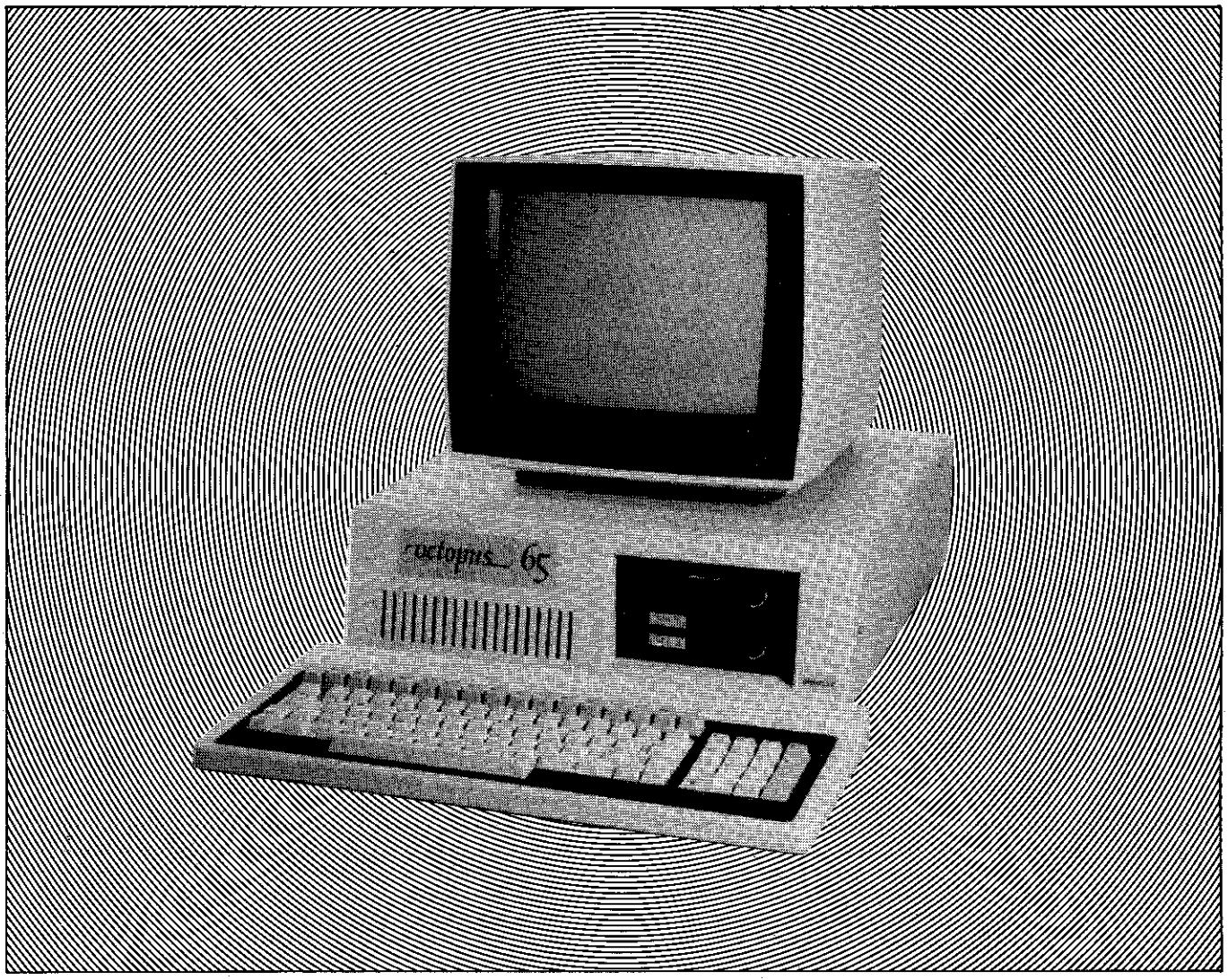


1e speciale computer-uitgave

de Octopus 65:
een veelzijdige zelfbouw-computer met een prima software-pakket

Elektuur Computing

een extra uitgave voor de aktieve computer-hobbyist



De computer-hobby is een van de meest verbreide technische hobby's van deze tijd. Op dit gebied verschijnt dan ook bijzonder veel lektuur. Toch ontbrak het tot nu toe aan een uitgave die zich volledig richt op de aktieve computer-hobbyist. Daarmee bedoelen we degene die niet alleen speelt met het toetsenbord van zijn computer, maar zich ook verdiept in de hardware, zelf de soldeerbout in zijn computer durft te steken en misschien wel een komplete computer zou willen bouwen aan de hand van een goed zelfbouwontwerp. Bovendien houdt hij zich ook kwa software bezig met allerlei programmeertalen: BASIC, Pascal, Forth, machinetaal, enz. Voor die aktieve hobbyist is er nu Elektuur Computing. Deze extra Elektuur-uitgave zal zo'n vier maal per jaar verschijnen.

In deze eerste uitgave van Elektuur Computing beginnen we direkt met de beschrijving van een komplete zelfbouwcomputer, de OCTOPUS 65. Een goed doordacht ontwerp met zeer veel toepassingsmogelijkheden. Betaalbaar, geschikt voor toekomstige uitbreidingen en tevens bruikbaar als ontwikkelingssysteem. De OCTOPUS 65 is een systeem met een of meer disk-drives, dat werkt met het snelle Ohio-disk-operating-system OS-65D. In deze uitgave vindt u een beschrijving van alle hardware voor deze computer, plus uitgebreide beschrijvingen van de bijbehorende software.

We wensen alle "aktieve computer-hobbyisten" veel bouw- en programmeerplezier.

de redaktie van het
elektronica-maandblad
Elektuur

Elektuur Computing

inhoud

Deze Elektuur Computing is een uitgave van:
Uitgeversmij. Elektuur B.V.,
Peter Treckpoelstraat 2-4, Beek (L)
Telefoon: 04402-74200, Telex 56617
Korrespondentie-adres: Postbus 75,
6190 AB Beek (L)
Kantoortijden: 8.30-12.00 en 12.-16.00 uur
Direkteur: J.W. Ridder,
Bourgognestraat 13a, Beek (L)

Deze uitgave is samengesteld door de redaktie van het elektronicamaandblad Elektuur.
Medewerkers: A. Nachtmann,
P. v.d. Linden, R. Krings, D. Meyer,
P. Theunissen, J. Schaap (vert.)

De hieronder vermelde printen en frontplaten kunnen worden besteld via de handel en rechtstreeks bij Elektuur B.V., Beek (L).
(E)PROM's kunt u door Elektuur B.V. laten programmeren.
Uitsluitsel over de bestelwijze, verkrijgbaarheid en prijs van onderstaande produkten geeft het overzicht in de laatste uitgave van het elektronicamaandblad Elektuur (pag. 6).

PRINT SERVICE

bestelnr.	print
82017	dynamische RAM-kaart
82159	floppy-disk-interface
83082	VDU-kaart
83102	omnibus CPU-kaart
83108-1	basisprint
83108-2	opzetprint

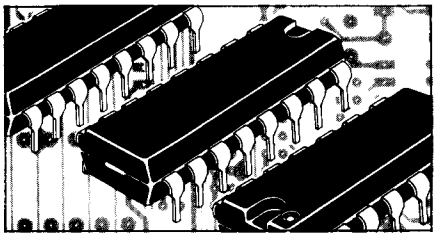
PAPERWARE SERVICE

bestelnr.	programma
PWS-3	aanvullende informatie universele terminal
PWS-4	aanvullende informatie VDU-kaart + source-listings

PROGRAMMEER SERVICE

bestelnr.	programma
523	karaktergenerator in 1 × 2732 EPROM
533-NL	Octopus-monitorprogramma in 1 × 2732 EPROM

HARDWARE



Introduktie

Wij hebben het genoegen u voor te stellen:

— De Octopus 65 —

De Octopus 65 is een werkelijk volledige microcomputer, bedoeld om zelf te bouwen en om er zelf veel, zeer veel mee te doen.

De delen waaruit de Octopus 65 is samengesteld, werden in de afgelopen jaren ontwikkeld in het Elektuur-laboratorium en in een aantal artikelen in Elektuur gepubliceerd. Brengen we dan niets nieuws? Toch wel, want we tonen u hier het totale koncept van een computer, waarvan de verschillende delen inmiddels door vele lezers werden beproefd in enthousiaste nabouw.

We gaan u precies vertellen hoe de delen van de Octopus 65 moeten worden gebouwd. Weliswaar kunt u dat ook terugzoeken in Elektuur, maar om de totale computer-opzet goed te doorgronden willen we hier toch een zo volledig mogelijk beeld geven van het koncept. Daarna gaan we verder met de samenbouw, de afregeling en de ingebruikname.

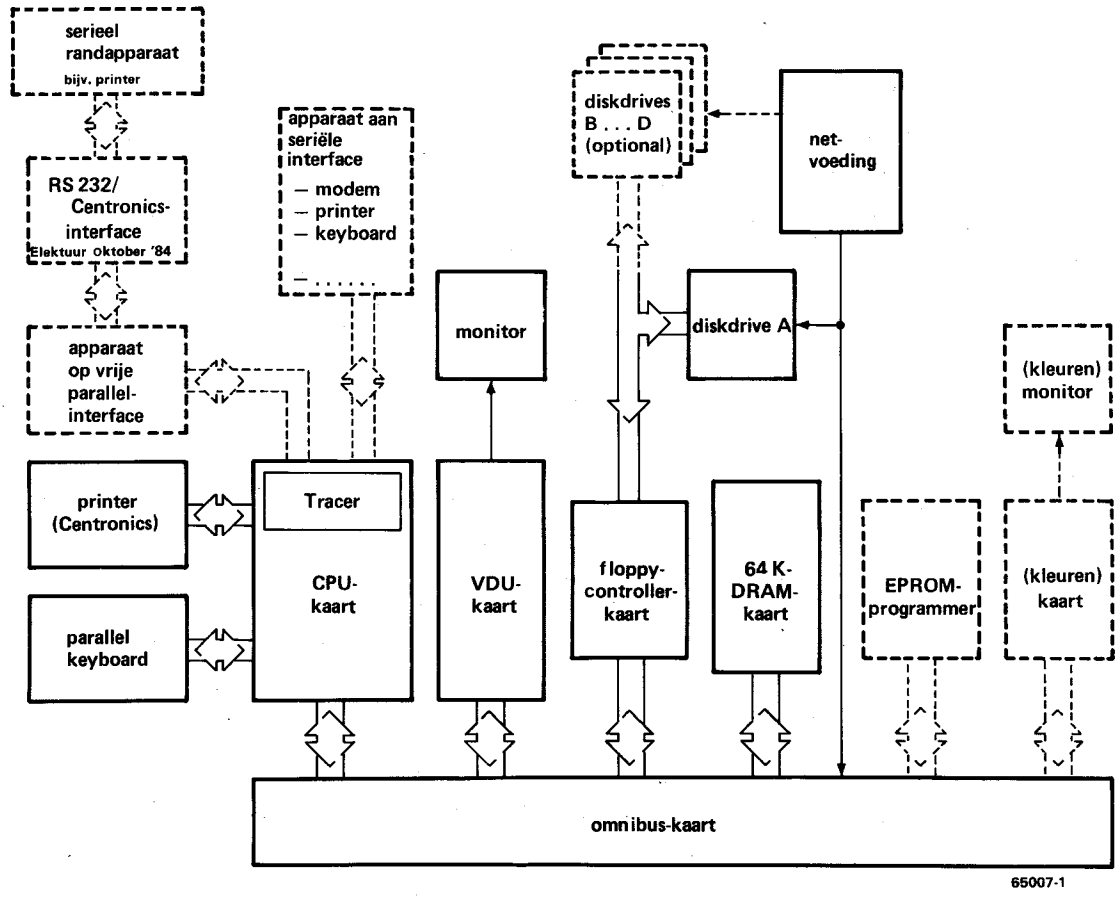
Om u maar meteen met de Octopus 65 vertrouwd te maken, geven we eerst het blokschema in figuur 1.

Zoals u ziet, bestaat de "harde" kern van de Octopus 65 uit vier units die de trouwe lezers van Elektuur erg bekend zullen voorkomen, te weten:

1. de CPU-kaart (Central Processing Unit), die is uitgerust met een oude bekende, de 6502-microprocessor (velen van u zullen nu weer terugdenken aan de junior computer, waar ook alles om een 6502 draaide).

Figuur 1. De Octopus 65 bestaat grofweg gezien uit vier kaarten: CPU-, VDU-, RAM- en FCU-kaart. Ze zijn met elkaar verbonden via de Elektuurbus. Bij de periferie, zoals keyboard, monitor, floppy drives, printer en modem, kan elk willekeurig apparaat worden genomen, zolang dit maar een standaard-aansluiting bezit en met standaard-nivo's werkt.

1



Technische gegevens CPU-kaart
* CPU type 6502 of 65C02
* 2 VIA's type 6522 of 65C22
* ACIA type 6551 of 65C51
* 2-Kbyte- of 8-Kbyte-RAM
* 2-, 4-, 8- of 16-Kbyte-EPROM
* volledige adresdekodering
* volledig gebufferde adresbus
* volledig gebufferde databus
* 64-polige Elektuurbus
* DMA-mogelijkheid
* klokfrekwentie 1, 2 of 4 MHz
* 4 8-bit-ports
* 4 16-bit-timers
* 2 seriële data-ports
* 8 handshake-lijnen
* parallel-keyboard-aansluiting (ASCII)
* Centronics-aansluiting
* RS232-V24-aansluiting
* alle I/O-lijnen uitgevoerd op konnektor-pennen

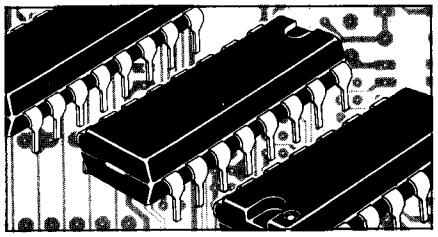
2. de VDU-kaart (Video Display Unit), programmeerbaar tot max. 80 × 25 karakters, lichtpenaansluiting, standaard ASCII-karakters.
3. de FDU-kaart (Floppy Disk Unit).

Technische gegevens VDU-kaart
* CRTC type 6845
* 2-Kbyte-video-RAM type 6116
* karaktergenerator-EPROM type 2716
* keuze tussen kwarts- en LC-oscillator
* volledig gebufferde databus
* volledig gebufferde adresbus
* volledige adresdekodering
* 64-polige Elektuurbus
* aansluitmogelijkheid voor lichtpen
* 75-ohm-monitor-uitgang
* alle CRTC-registers toegankelijk via systeembus

4. de DRAM-kaart. Deze bevat 64 Kbyte dynamisch RAM-geheugen. In plaats van deze kaart kan men ook een statische RAM-kaart nemen.

Deze vier prints zijn op Eurokaart-formaat en worden simpelweg in de eveneens bekende Elektuurbus geplaatst, die zorgt voor de juiste doorverbindingen. In het blokschema is verder te zien welke randapparatuur aan de Octopus 65 kan worden aangesloten en welke uitbreidingen we nog voor ogen hebben. Het zal u duidelijk zijn dat de Octopus 65 bedoeld is om intensief met floppy disks samen te werken. Daarom besloten we het ontwerp zodanig in te richten, dat één of meer disk drives in de Octopus 65 kunnen worden ingebouwd.

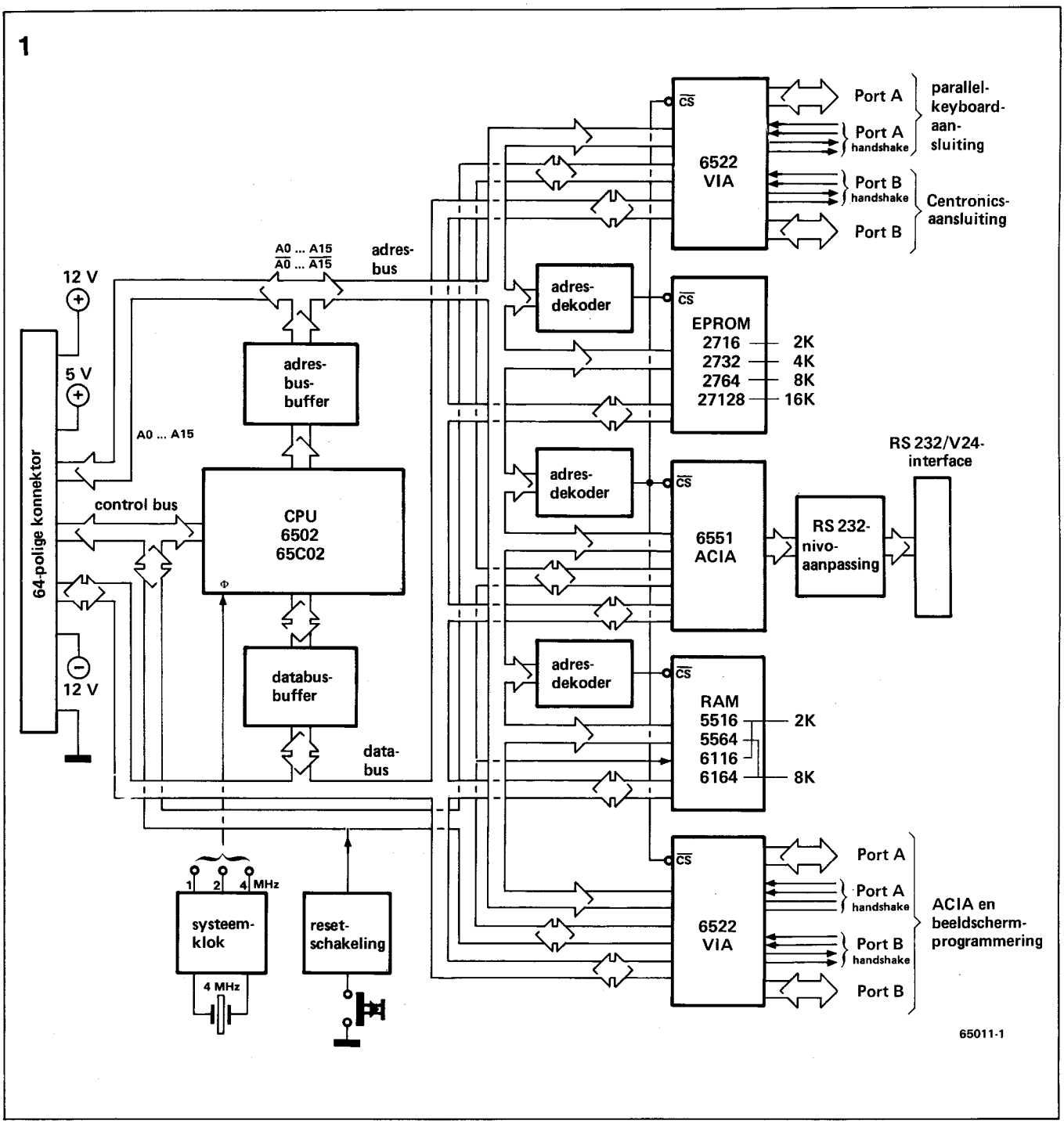
In de hierna volgende hoofdstukken worden de vier reeds genoemde hoofdbestanddelen van de Octopus 65 behandeld. Daarna bespreken we de Elektuurbus, de benodigde voeding, de disk drives, de samenbouw in een kast, de afregeling en de ingebruikname. Tenslotte willen we nog enkele opmerkingen geven over het bij de Octopus 65 te gebruiken keyboard, de monitor, een eventuele printer, enz. In het daarna volgende tweede gedeelte van deze uitgave wordt de software behandeld, waarbij we u volledig gaan inlichten omtrent de gebruiksmogelijkheden van de Octopus 65. Verder geven we u enkele speciaal voor de Octopus 65 ontwikkelde toepassingsprogramma's, alsmede informatie over het verkrijgen van meer programma's. En tenslotte is er nog een literatuuroverzicht.



Octopus' schakelcentrale

De CPU-kaart

Laten we eerst het blokschema van de CPU-kaart (CPU = Central Processing Unit) eens bekijken (zie figuur 1). Alle hoofdbestanddelen die zich op deze kaart bevinden, vinden we hier terug. Het blokschema is eigenlijk zo eenvoudig dat een korte beschrijving al voldoende is om dit schema goed te kunnen begrijpen.
Geheel links zien we de 64-polige konnektor voor de aansluiting van de kaart op de Elektuurbus. Op deze konnektor zijn aangesloten: control bus, gebufferde adres- en databus, +12 V, +5 V en −12 V. Daarnaast de microprocessor, de CPU, het eigenlijke hart van onze Octopus 65 computer. Men kan hier kiezen tussen een "normale" 6502 of een CMOS-low-power-versie, een 65C02. De klokgenerator (links onderaan) levert de frekwenties 1, 2 en 4 MHz, waarvan men er eenvoudig één kan kiezen door het leggen van een draadbrug. Alle adreslijnen van de CPU zijn zowel gewoon als geïnverteerd beschikbaar en zijn, evenals de datalijnen, gebufferd. De lijnen van de control bus zijn niet gebufferd, maar dat is ook niet nodig.
Aan de rechterzijde van het blokschema vinden we twee VIA's van het type 6522 (VIA = Versatile Interface Adaptor). Voor nadere bijzonderheden over de opbouw en werking van deze komplexe IC's verwijzen we naar het VIA-boek, dat bij uitgeversmij Elektuur B.V. is verschenen. We vermelden hier alleen dat elke VIA beschikt over twee bidirektionele poorten A en B, vier handshake-lijnen per poort, twee zestien-bits counters/timers en een acht-bits I/O-schuifregister. Poort A van de eerste VIA wordt gebruikt voor een parallel-keyboard-aansluiting en poort B voor een Centronics-aansluiting. De poorten A en B van de tweede VIA worden gebruikt voor het programmeren van de ACIA (via kortsluitstekkertjes) en het beeldformaat (alleen in kombinatie met de VDU-kaart), en nog enkele andere zaken. De ACIA 6551 (ACIA = Asynchronous Communication Interface Adaptor) is eveneens een zeer veelzijdig IC (ook rechts in het blokschema). De ACIA wordt hier toegepast voor de RS 232/V24-interface. De ACIA zorgt voor de baudrate, parallel-serie-omzetting, foutdetektie, enz. De ACIA realiseert dus de seriële data-overdracht tussen de CPU en de buitenwereld. Tussen de ACIA en de RS 232-konnektor zijn nog wat poorten opgenomen die zorgen voor de noodzakelijke nivoaanpassingen (RS 232 werkt met een positieve èn een negatieve voedingsspanning).
Op de CPU-kaart is verder nog plaats voor één RAM-IC en één EPROM-IC. Voor de Octopus 65 is een RAM-IC van 2-Kbyte-CMOS-geheugen voldoende en een EPROM-IC van 4 Kbyte (2732).
De VIA's en de ACIA hebben een gemeenschappelijke adresdekoder, de geheugen-IC's echter hebben elk een eigen adresdekoder. Verder zijn alle IC's gekoppeld met de adres- en databus en (met uitzondering van de EPROM) ook met de control bus.
De resetschakeling zorgt er voor dat de processor automatisch gereset wordt bij het inschakelen van de voedingsspanning. Door middel van een drukknop kan men ook een reset met de hand bewerkstelligen (manual reset). Het verdient aanbeveling deze mogelijkheid beschikbaar te hebben.
Na de beschrijving van het blokschema zal het echte schema in figuur 2 weinig vragen meer oproepen.
Tussen de CPU (IC1) en de data- en adresbus zijn de buffers IC10...IC14 geschakeld (zie fig. 3). De klokgenerator (weer links onderaan in het schema) levert een basisfrekwentie van 4 MHz, waarvan flipflops FF1 en FF2 2 en 1 MHz afleiden. Een van deze drie frekwenties kan d.m.v. konnektor PL14 worden gekozen en aan de processor worden toegevoerd. Door het doorverbinden van de punten M en N kunnen de flipflops buiten bedrijf worden gesteld, in het geval dat men van een externe klok gebruik wil maken.
De reset-schakeling (rechts van de klokgenerator) zorgt er met de RC-kombinatie R17 en C1 voor dat de reset-ingang van de CPU pas 0,5 sekonde na het inschakelen van de voedingsspanning wordt vrijgegeven. Sluit men de punten P en Q aan op een drukknopschakelaar met verbreekkontakt, dan kan men daarmee "manual reset" toepassen. Maakt men hiervan geen gebruik, dan moeten P en Q perma-



Figuur 1. Het blokschema van de 6502-CPU-kaart. Door de vele aansluitmogelijkheden heeft deze een werkelijk universeel karakter.

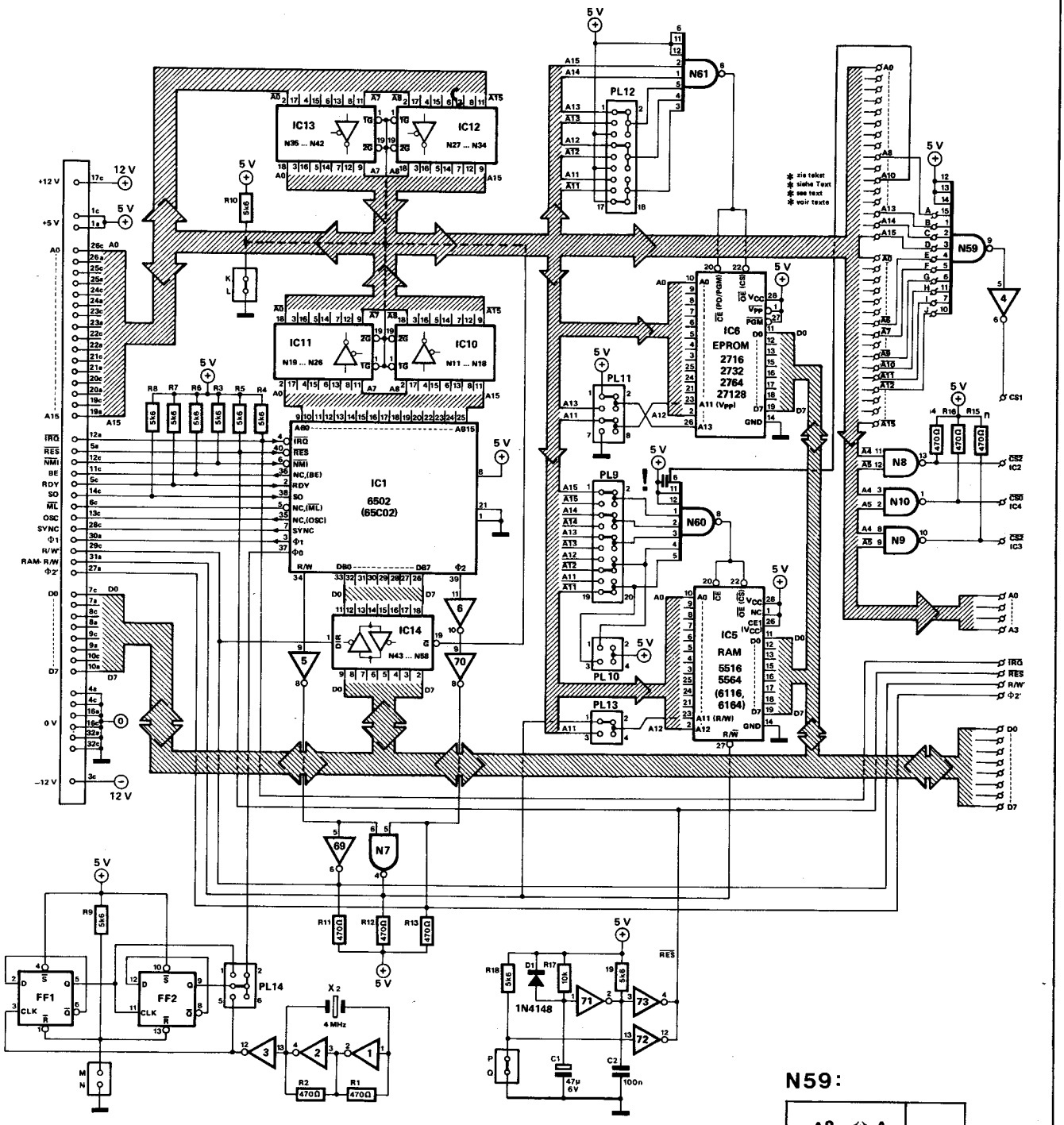
nent worden doorverbonden. Ook de brug K-L (links boven) moet doorverbonden worden, waardoor de adres- en databusversterkers permanent aktief zijn. Bij latere uitbreidingen is het mogelijk via aansluiting K de adres- en databus apart te sturen. De adresdekoder voor de VIA's IC2 en IC3 en de ACIA IC4 wordt gevormd door N59. De adresdekoder voor de RAM IC5 is N60 en de dekoder voor de EPROM IC6 is N61. Op de ACIA is een kristal aangesloten dat dient voor het opwekken van de verschillende baudrates. Wie de RS 232-interface niet nodig denkt te hebben, kan de ACIA IC4, samen met de IC's 18 en 19 (nivo-aanpassers) en de bijbehorende onderdelen (figuur 2, geheel rechts) weglaten. De schakeling bevat overigens nog enige konnektoren die onze Octopus 65 op dit moment eigenlijk nog niet nodig heeft, maar die de CPU-kaart echter reeds voor latere uitbreidingen geschikt maakt.

Het stroomverbruik van de CPU-kaart met "normale" IC's is 100 mA bij +12 V en —12 V en 1...1,5 A bij +5 V. Het is ook mogelijk, zoals eerder opgemerkt, een CMOS CPU (type 65C02) toe te passen, terwijl ook voor de perifere delen CMOS-exemplaren kunnen worden gebruikt. De vervangende typenummers staan tussen haakjes aangegeven in de onderdelenlijst. Het totale stroomverbruik wordt daarmee tot 100 mA beperkt, zodat voeding uit batterijen of akku's mogelijk wordt.

In figuur 3 zien we de komponentenopstelling voor een deel van de CPU-kaart. Door de talrijke konnektoren die zijn toegepast en die de kaart zeer universeel maken, passen helaas niet alle onderdelen van deze kaart op één Eurokaart. Om tóch het Euroformaat aan te kunnen houden is een sandwichkonstruktie toegepast door op de grote kaart een kleinere te monteren (zie figuur 4). Beide printplaten zijn dubbelzijdig, dus men dient vóór de montage van de onderdelen met een ohmmeter de doorgemetalliseerde gaten te kontroleren. Is dit geheel in orde bevonden, dan kunnen de onderdelen op de prints worden gesoldeerd.

Bij de PL-konnektoren kan men de gewenste doorverbindingen gewoon aan elkaar solderen (met draadbrugjes), of de in de handel verkrijgbare konnektoren met

2a



N59:

A8	↔ A	E100 ... E13F
A13	↔ B	
A14	↔ C	
A15	↔ D	
$\overline{A6}$	↔ E	
$\overline{A7}$	↔ F	
$\overline{A9}$	↔ G	
$\overline{A10}$	↔ H	
$\overline{A11}$	↔ I	
$\overline{A12}$	↔ J	

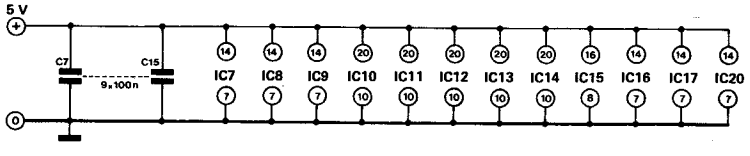
12 V ⊕ —— +12 V

5 V ⊕ —— +5 V

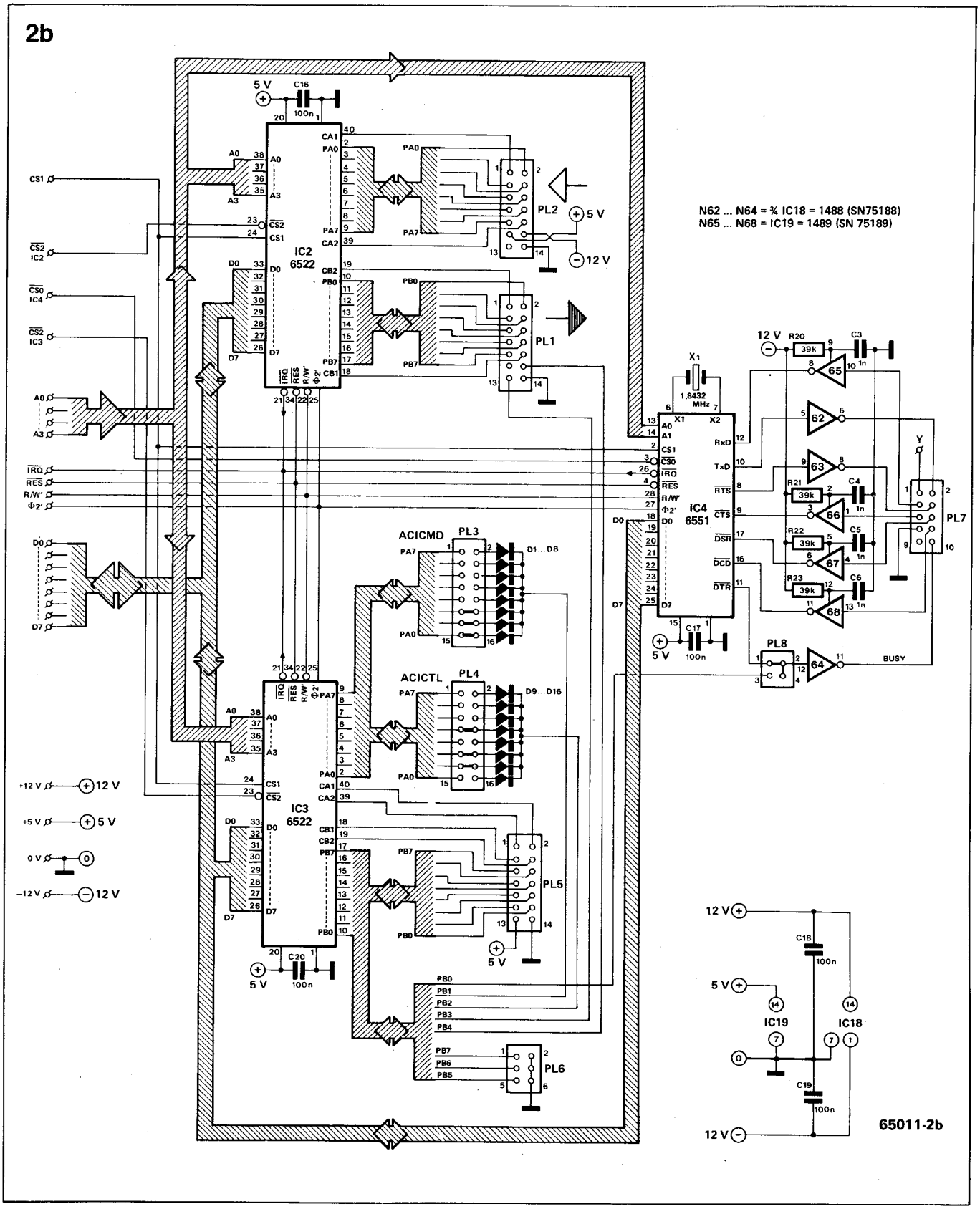
0 —— 0 V

12 V ⊖ —— −12 V

N1 ... N6 = IC7 = 74LS04
FF1,FF2 = IC8 = 74LS74
N7 ... N10 = IC9 = 74LS01
N11 ... N18 = IC10 = 74LS240
N19 ... N26 = IC11 = 74LS240
N27 ... N34 = IC12 = 74LS240
N35 ... N42 = IC13 = 74LS240
N43 ... N58 = IC14 = 74LS245
N59 = IC15 = 74S133 (74ALS133)
N60 = IC16 = 74LS30
N61 = IC17 = 74LS30
N69 ... N73 = ⅙ IC20 = 74LS06 (74LS05)



65011-2a

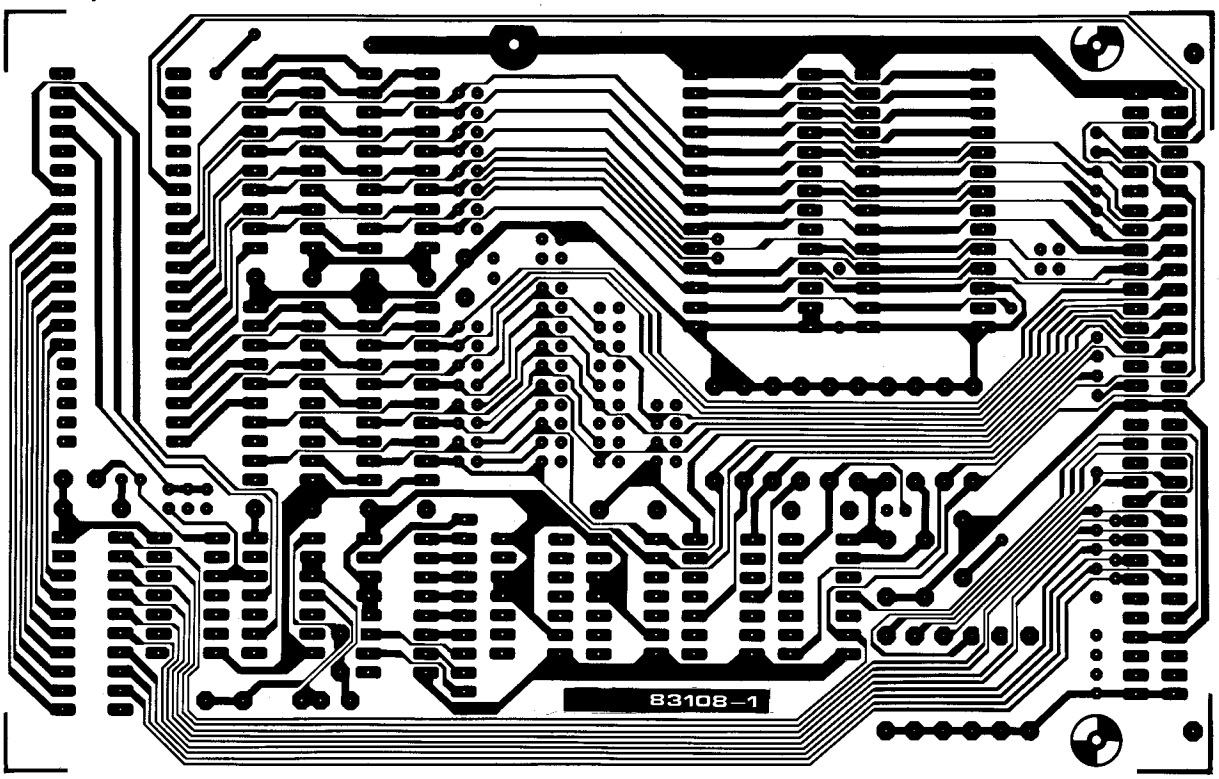


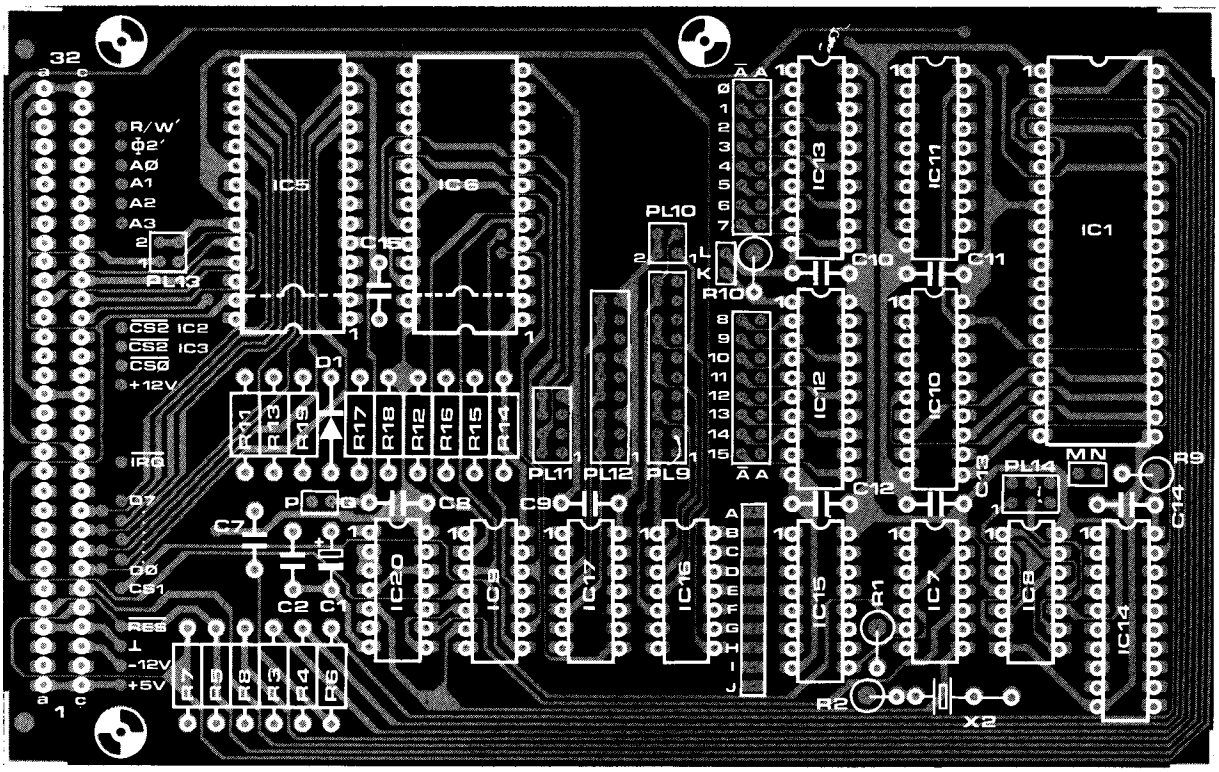
Figuur 2. Als we de blokken van figuur 1 vervangen door de IC's, dan ontstaat al bijna het hier afgebeelde "echte" schema. Door de vele verbindingslijnen lijkt het alleen wat ingewikkelder.

kortsluitbrugjes gebruiken. Dit kan natuurlijk niet voor de I/O-konnektor PL7, waarvoor men in elk geval een echte konnektor dient te gebruiken waarop een bandkabel kan worden aangesloten. Hoe de konnektoren moeten worden doorverbonden, staat in tabel 1 aangegeven. Figuur 5 laat de aansluitingen aan PL1 (Centronics interface) en PL2 (toetsenbord-parallel-ingang) zien, terwijl figuur 6 de aansluitingen van PL7 toont (RS 232-interface). Door middel van PL3, PL4 en PL8 wordt de ACIA geprogrammeerd ten behoeve van de RS 232-interface. Omdat modems zich momenteel in een grote mate van belangstelling mogen verheugen hebben we in het schema de kortsluitbrugjes voor de aansluiting van een V21-modem aangegeven (full duplex, 300 Bd). Deze seriële interface kan echter voor zeer vele toepassingen worden gebruikt; zie hiervoor de in tabel 2 gegeven

3

soldeerzijde

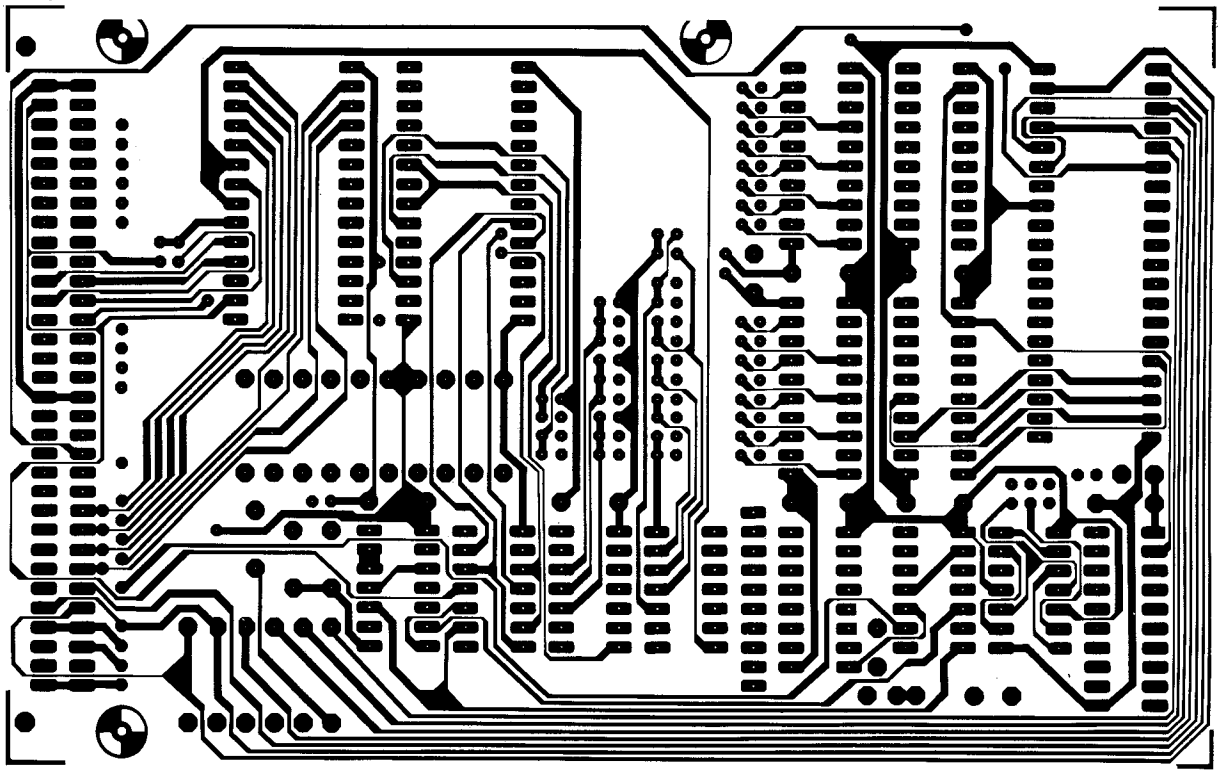




informatie.
De port- en control-lijnen op konnektor PL5 zijn op dit moment (nog) niet in gebruik, zodat men deze naar eigen inzicht kan toepassen. Met behulp van de artikelen in Elektuur oktober 1984 ("RS 232--Centronics interface"), november 1984 ("RS 232/V24 besturingssignalen") en het eerder genoemde VIA-boek kan men hier naar verkiezing nog een extra parallel- of serie-interface realiseren.
PL6 dient voor het programmeren van het beeldformaat van de CPU-kaart. Voor onze Octopus 65 mogen hier géén kortsluitbrugjes geplaatst worden, omdat het beeldformaat na elke inschakeling en elk reset-kommando door het monitorprogramma SAMMON wordt ingesteld. Men

3

komponentenzijde



Figuur 3. De grote print bevat onder andere CPU, RAM, EPROM, klokgenerator en resetlogika.

Onderdelenlijst

Weerstanden:
R1,R2,R11...R16 = 470 Ω
R3...R10,R18,R19 = 5k6
R17 = 10 k
R20...R23 = 39 k

Kondensatoren:
C1 = 47 μ/6 V
C2,C7...C20 = 100 n
C3...C6 = 1 n

Halfgeleiders:
D1...D16 = 1N4148
IC1 = 6502 (65C02)
IC2,IC3 = 6522 (65C22)
IC4 = 6551 (65C51)
IC5 = 5516, 5564 (6116, 6164)
IC6 = 2716, 2732, 2764, 27128
IC7 = 74LS04
IC8 = 74LS74
IC9 = 74LS01
IC10...IC13 = 74LS240
IC14 = 74LS245
IC15 = 74S133 (74ALS133)
IC16,IC17 = 74LS130
IC18 = 1488 (SN 75188)
IC19 = 1489 (SN 75189)
IC20 = 74LS06 (74LS05)

Diversen:
X1 = kristal 1,8432 MHz
X2 = kristal 4 MHz
64-polige konnektor volgens DIN 41612, male konnektorstrips (zelf op maat te maken)
bijvoorbeeld Molex
2× 8624A-102 (10-89-1801) (40× 2 kontakten)
1× 8624A-102 (10-83-1321) (16× 2 kontakten)
±25 kortsluitstekkertjes nummer 7589

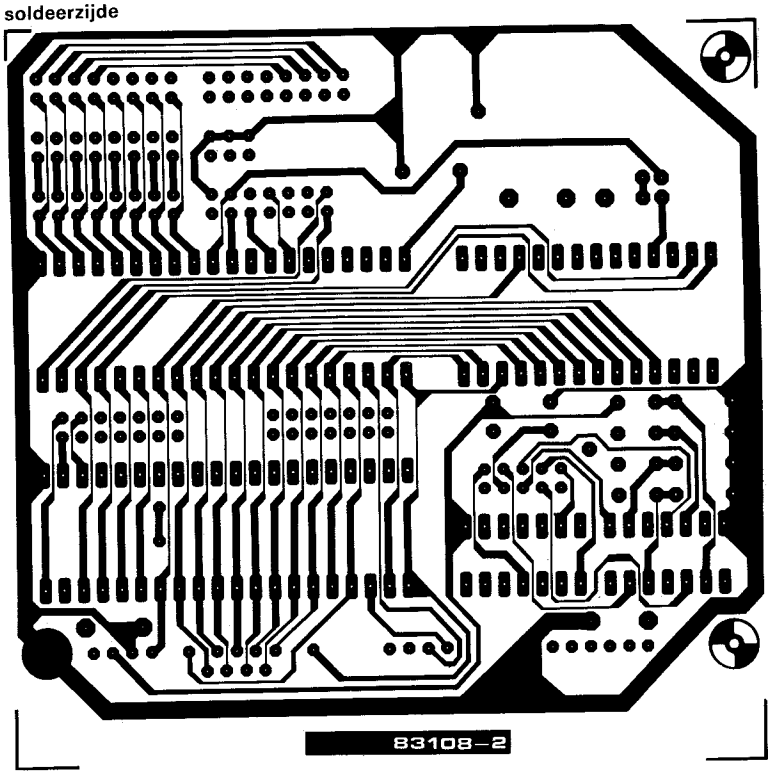
Tabel 1. De funktie van de verschillende PL-aansluitingen met daarnaast de doorverbindingen die hierop moeten worden aangebracht.

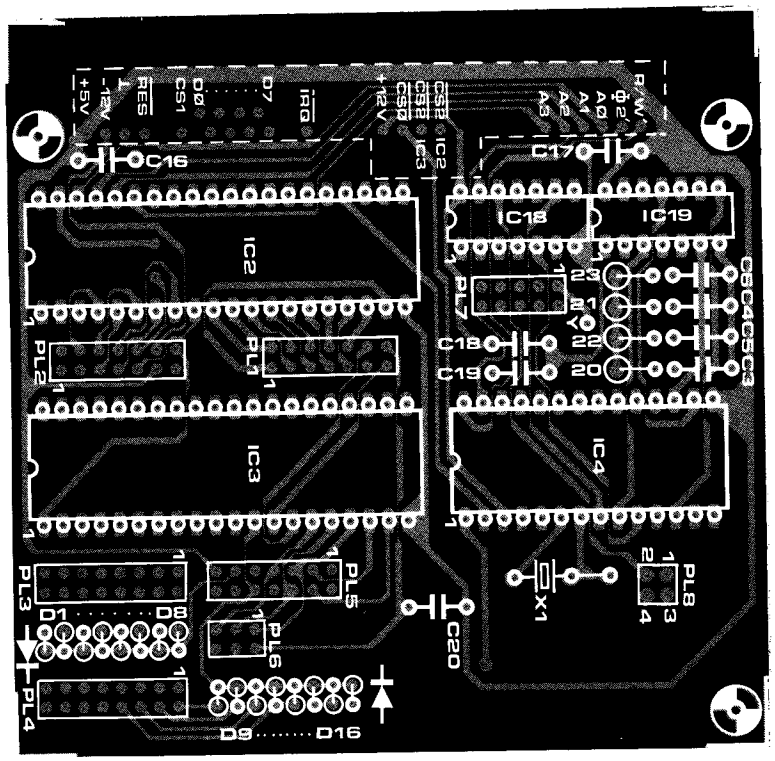
Tabel 1

Konnektor	Verbindingen Ø = open, 1 = gesloten
PL1	geen, Centronics-aansluiting
PL2	geen, parallel-keyboard-aansluiting
PL3	ACIA-programmering, zie tabel 2
PL4	ACIA-programmering, zie tabel 2
PL5	geen, port- en stuur-lijnen
PL6	beeldschermformaat, geen verbinding voor Octopus 65
PL7	geen, RS232-aansluiting
PL8	ACIA-programmering, zie tabel 2
PL9	1-2, 5-6, 9-10, 15-16, 19-20 = 1 alle andere = Ø
PL10	geen
PL11	1-2, 5-6 = 1 alle andere = Ø
PL12	1-2, 7-8, 17-18 = 1 alle andere = Ø
PL13	1-2 = 1 3-4 = Ø
PL14	3-4 = 1 alle andere = Ø

kan het beeldformaat met een BASIC-programma wijzigen, zie daartoe Elektuur oktober 1984 ("De 6845 geprogrammeerd"). PL9...PL13 dienen voor het adresseren van de RAM en de monitor-EPROM op deze CPU-kaart. De benodigde kortsluitbrugjes zijn zowel in het principeschema als in tabel 1 aangegeven.
Tenslotte moet men nog een kleine wijziging doorvoeren: pen 6 van IC16 moet vóór het insteken van het IC in de voet (of, als men geen voet toepast, vóór het insolderen van het IC) omgebogen worden, zodat die pen géén kontakt maakt met de voet (of met de printplaat) en dus vrij blijft. Met een stukje kabel wordt deze pen nu met adresbusaansluiting A10 verbonden, zoals in het schema is getekend. Deze konstruktie lijkt niet zo fraai, maar we hebben er wél 4 Kbyte werkgeheugen mee bespaard!
De adresdekoder N59 dient voor de adressering van de VIA's en de ACIA. De aansluitingen A...J, die niet met +5 V zijn verbonden, worden elk met één van de adresbusaansluitingen $\overline{A6}$, $\overline{A7}$, A8, $\overline{A9}$, $\overline{A10}$, $\overline{A11}$, $\overline{A12}$, A13, A14 en A15 verbonden en wel zodanig dat steeds de dichtstbijzijnde, dus meest praktische mogelijkheid wordt gekozen. De volgorde van deze doorverbindingen is onbelangrijk, als elk van de aansluitingen A...J maar met één van de genoemde adresbusaansluitingen wordt doorverbonden. Let daartoe maar niet op het principeschema, want dáár zijn de verbindingen zó gemaakt zoals het onze tekenaar het beste uitkwam!
Dan nog een belangrijke opmerking: in de opdruk van de grote print is een on-

4 soldeerzijde





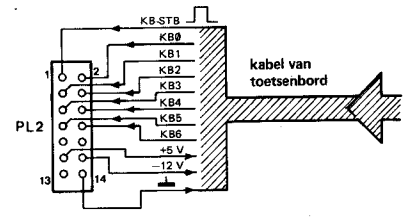
Figuur 4. De kleine print wordt boven op de grote print gemonteerd en bevat onder andere de twee VIA's en de ACIA.

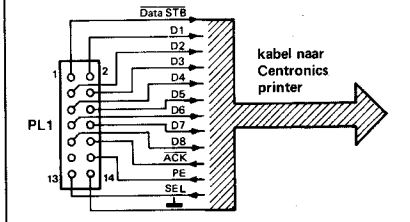
Figuur 5. De aansluitgegevens voor de konnektoren PL1 (Centronics-aansluiting) en PL2 (parallel-keyboard-ingang).

Figuur 6. De aansluitgegevens voor de seriële in/uitgang (PL7).

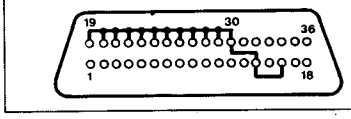
Tabel 2. Alle informatie voor de programmering van de seriële aansluiting (ACIA).

5





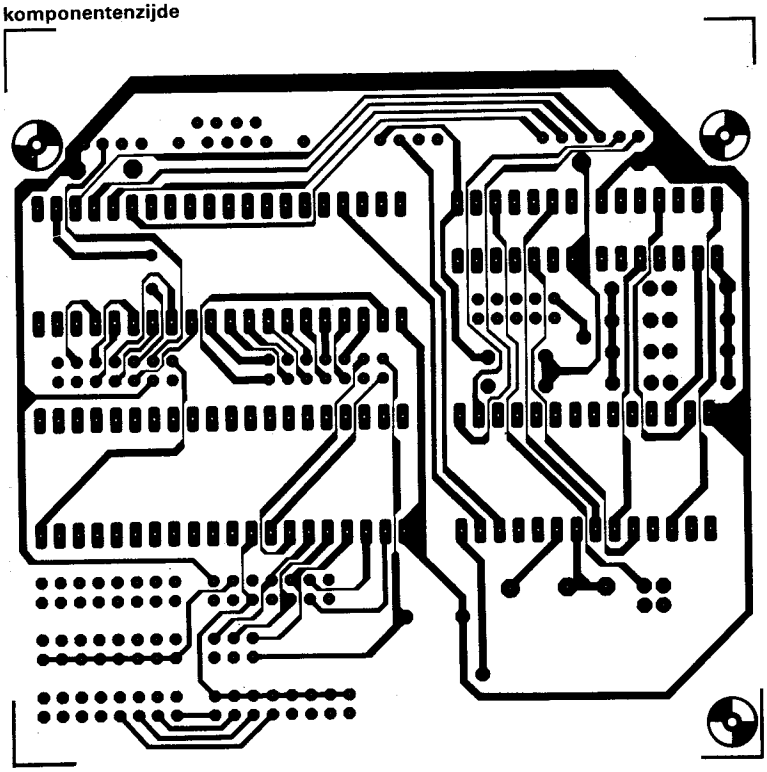
Pin No.	Signal Name	Pin No.	Signal Name
1	DATA STB	19	TWISTED PAIR GND (PIN 1)
2	DATA 1	20	TWISTED PAIR GND (PIN 2)
3	DATA 2	21	TWISTED PAIR GND (PIN 3)
4	DATA 3	22	TWISTED PAIR GND (PIN 4)
5	DATA 4	23	TWISTED PAIR GND (PIN 5)
6	DATA 5	24	TWISTED PAIR GND (PIN 6)
7	DATA 6	25	TWISTED PAIR GND (PIN 7)
8	DATA 7	26	TWISTED PAIR GND (PIN 8)
9	DATA 8	27	TWISTED PAIR GND (PIN 9)
10	ACK	28	TWISTED PAIR GND (PIN 10)
11	INPUT BUSY	29	TWISTED PAIR GND (PIN 11)
12	PE	30	TWISTED PAIR GND (PIN 31)
13	SELECT	31	INPUT PRIME
14	0 V	32	FAULT
15	NC	33	0 V
16	0 V	34	NC
17	CHASSIS GND	35	NC
18	+5 V DC	36	INPUT-BUSY



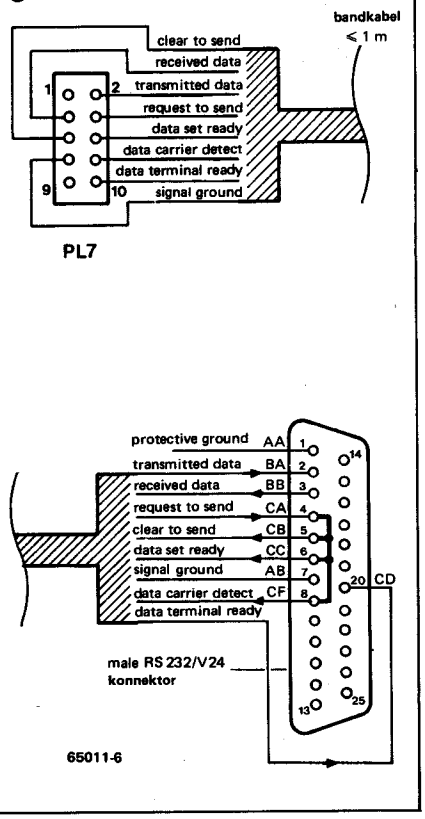
duidelijkheid geslopen. Het kader dat om dat adresbuslijnen is gedrukt boven IC12, heeft het negatiestreepje boven de bovenste A doen verdwijnen. Houdt men de print zoals die hier is afgedrukt, dan bevat de bovenste rij de geïnverteerde adresbuslijnen A8...A15. Op de tekening hier in deze Special hebben we dat nog kunnen korrigeren, maar voor de printopdruk was het al te laat. Met PL14 wordt de klokfrekwentie op 1 MHz ingesteld (brug 3-4). Het kan best zijn dat dit later nog eens gaat veranderen. Men kan voor de CPU, de VIA's en de ACIA de 2-MHz-CMOS-versies toepassen als men deze onderdelen toch nog moet aanschaffen. Maar houd er rekening mee dat die ook op 1 MHz moeten draaien, zolang de software nog niet hierop aangepast is.

Voor de IC's kan men het beste voetjes gebruiken. Neem wel voetjes van zeer goede kwaliteit, dan gaat ook alles nog goed bij hogere klokfrekwenties. IC5 en IC6 steekt men zó in het voetje dat de pennen 1 en 2 en ook 27 en 28 vrij blijven. Men kan natuurlijk ook twee 24-pens voetjes gebruiken in plaats van 28-pens,

4 komponentenzijde



6



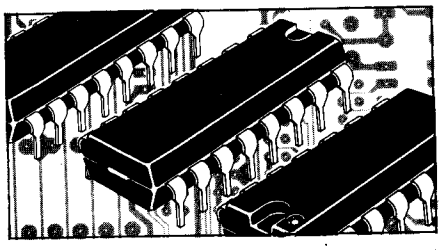
Tabel 2.

konnektor PL3			
verbinding			funktie
1-2	3-4	5-6	pariteitsbit:
X	X	0	geen (X = don't care)
0	0	1	odd
0	1	1	even
1	0	1	mark
1	1	1	space
7-8			mode:
0			normal
1			echo
9-10	11-12		transmitter-controls
0	0		transmitter interrupt disabled $\overline{RTS}$-level high, transmit. off
0	1		transmit. int. enabled $\overline{RTS}$-level low, transmit. on
1	0		transmit. int. disabled $\overline{RTS}$-level low, transmit on
1	1		transmit. int. disabled $\overline{RTS}$-level low, transmit break
13-14			$\overline{IRQ}$-interrupt
0			enabled
1			disabled
15-16			receiver + interrrupts:
0			disable
1			enable

konnektor PL4				
verbinding				funktie
1-2				aantal stopbits:
0 (= open)				1 stopbit
1 (= gesloten)				2 stopbits (1.5 bij 5 bits woordlengte)
3-4	5-6			woordlengte:
0	0			8 bits
0	1			7 bits
1	0			6 bits
1	1			5 bits
7-8				baudrate-generator:
0				extern
1				intern
9-10	11-12	13-14	15-16	baudrate:
0	0	0	1	50 baud
0	0	1	0	75 baud
0	0	1	1	109,92 baud
0	1	0	0	134,58 baud
0	1	0	1	150 baud
0	1	1	0	300 baud
0	1	1	1	600 baud
1	0	0	0	1200 baud
1	0	0	1	1800 baud
1	0	1	0	2400 baud
1	0	1	1	3600 baud
1	1	0	0	4800 baud
1	1	0	1	7200 baud
1	1	1	0	9600 baud
1	1	1	1	19200 baud
0	0	0	0	16x externe klok

konnektor PL8	
verbinding	funktie
1-2	low-speed modem
3-4	high speed VT-52-terminal

maar monteer ze dan zoveel mogelijk naar de rand van de print. Een en ander is op de printtekening duidelijk aangegeven door middel van een stippellijn.
Voordat men de kleine print op de grote monteert (hiervoor 3 afstandsbusjes gebruiken), moeten eerst de benodigde verbindingen tussen de beide printplaten worden aangebracht. Deze verbindingen liggen precies onder elkaar, zodat ze met korte stukken bandkabel kunnen worden gemaakt. In figuur 4 hebben we die aansluitpunten met een gestippeld kader aangegeven. Deze aanduiding staat echter **niet** op de print! Let vooral op de juiste positie van de twee printen ten opzichte van elkaar.



De VDU-kaart

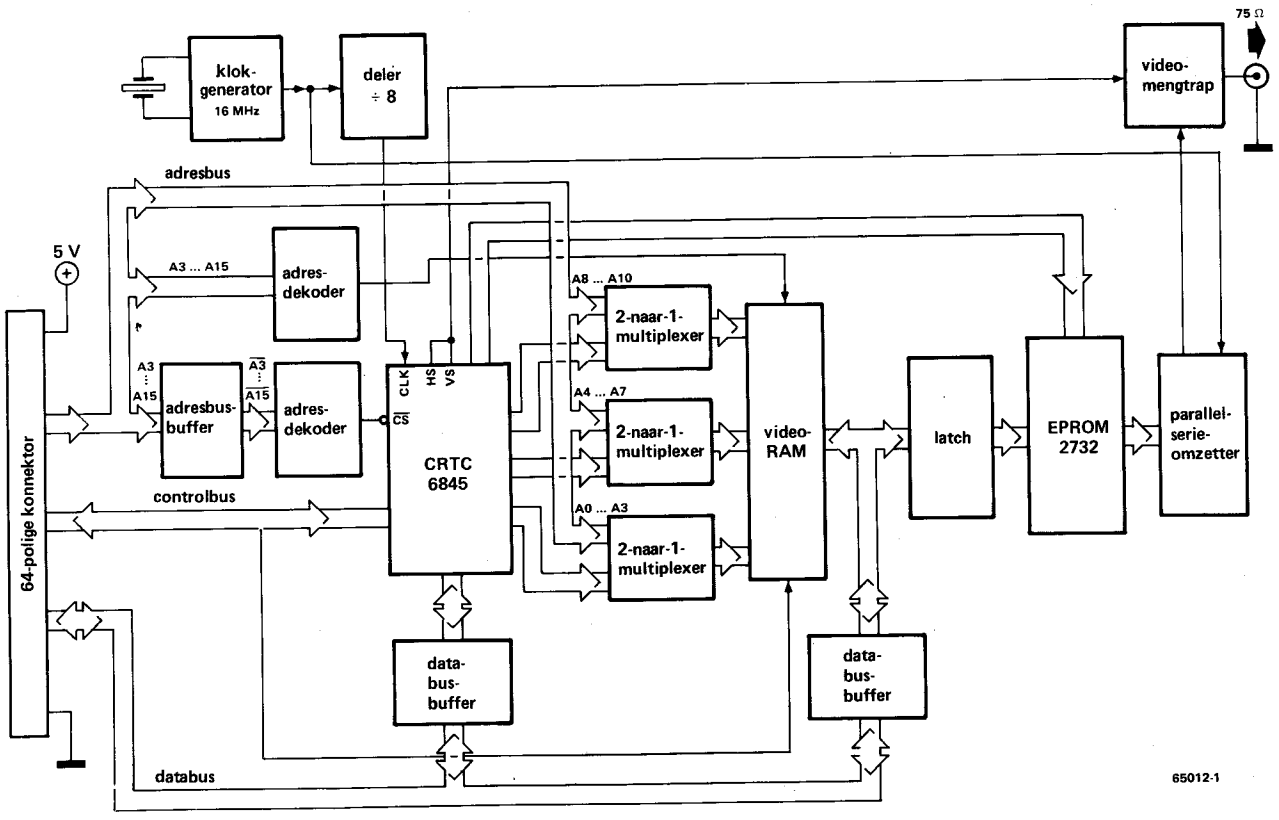
Octopus' optiek

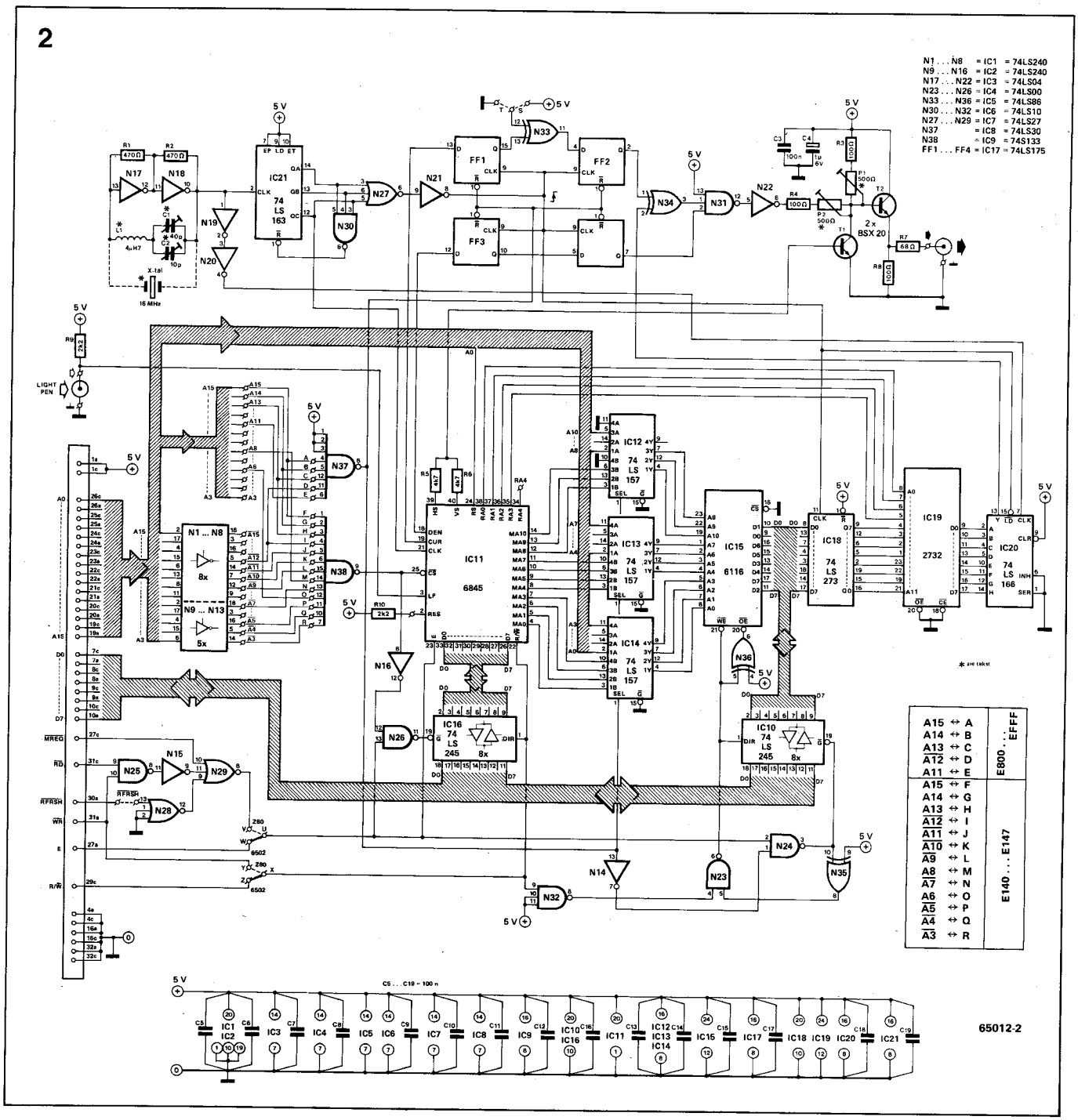
Ook van de VDU-kaart (VDU = Video Display Unit) bekijken we eerst het blokschema (figuur 1), omdat daardoor het bespreken van het eigenlijke schema aanmerkelijk eenvoudiger wordt.
Geheel links zien we weer de 64-polige konnektor voor de verbinding van de kaart met de Elektuurbus en de andere Octopus-kaarten. Daarnaast zit de adresbusbuffer die niet alleen de adreslijnen buffert, maar ook de adressignalen inverteert, zodat deze zowel geïnverteerd als niet-geïnverteerd ter beschikking staan. Dat doen de databus-buffers (onderaan in het blokschema) echter niet. De controlbus-lijnen zijn niet gebufferd, omdat dat nu eenmaal niet nodig is. Rechts van de adresbus-buffer zien we twee adresdekoders. De bovenste dekodeert de video-RAM-adressen, de onderste de CRTC. Op de adresdekodering komen we verderop nog terug.
Het IC van het type 6845 (rechts naast de adresdekoders) is de CRTC (= Cathode Ray Tube Controller). Deze regelt het "digitale verkeer" op de VDU-kaart en heeft de volgende taken:

* het opzoeken van het adres van het karakter dat op de monitor moet worden gezet.
* het omzetten van dat karakter in de bijbehorende puntenmatrix.
* het opwekken van de horizontale en vertikale beeldsynchronisatiepulsen op het juiste moment.
* het verzorgen van de licht-donkersturing van de beeldpunten op elke beeldlijn aan de video-ingang van de monitor.

Het IC is eigenlijk nog tot veel meer in staat, maar dat is voor het funktioneren van de VDU-kaart overbodig. Wie meer wil weten over dit IC en zijn registerorganisatie, kan hierover het een en ander vinden in de uitgave "Paperware 3".
Het IC ontvangt zijn werkfrekwentie van de klokgenerator via een 8-deler (zie rechtsboven in het blokschema). De drie rechts van de CRTC getekende 2-naar-1-multiplexers horen nog bij de adresdekodering van de video-RAM die weer rechts daarvan is getekend. In de video-RAM worden alle karakters opgeslagen die op het beeldscherm moeten worden geschreven. Voor 80 x 24 karakters zijn 1920 geheugenplaatsen nodig, zodat een 2-Kbyte-RAM voor de opslag daarvan voldoende is.
Bijna geheel rechts vinden we een EPROM dat als karaktergenerator werkt. Hier is de puntopbouw van alle beschikbare karakters vastgelegd, (hoofd- en kleine letters, getallen, leestekens en grafische tekens). De parallel-serie-omzetter (in het blokschema geheel rechts) zet de parallelle punt-informatie om in een serieel signaal en voert dat toe aan de videomengtrap (rechts bovenaan). Links van de karaktergenerator vinden we nog een latch. Deze ontvangt de kontinu door de video-RAM geproduceerde data en slaat die tijdelijk op. De latch wordt pas "geklokt" als de data van de RAM stabiel is, waarna deze informatie als adres voor de karaktergenerator wordt doorgegeven.

1





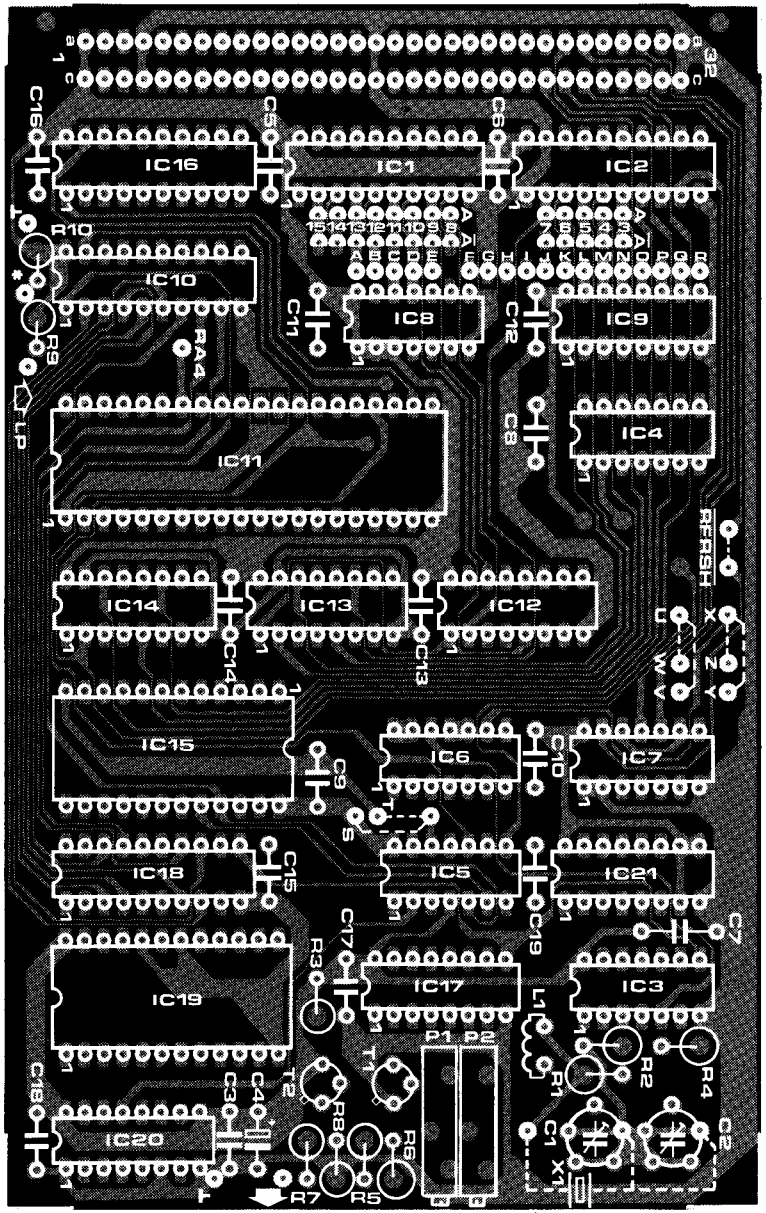
Figuur 1. Het blokschema toont kwa opzet een sterke gelijkenis met het echte schema, zodat men de verschillende delen gemakkelijk kan herkennen in figuur 2.

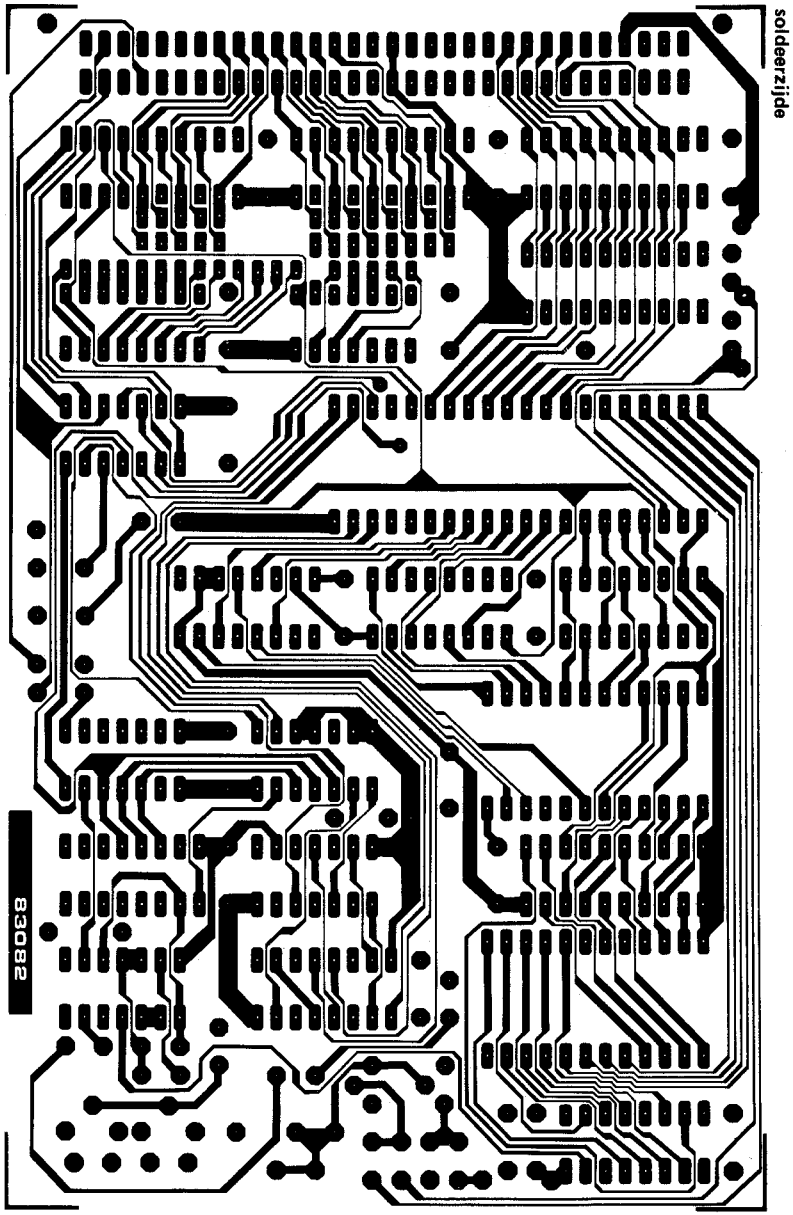
Figuur 2. Het schema van de VDU-kaart, waarbij alles draait rond de CRTC die uitvoerig is beschreven in Paperware 3. In deze Paperware-uitgave vindt men ook de nodige informatie over de block-graphics van de VDU-kaart.

Het schema in figuur 2 is weer op gelijke wijze als het blokschema opgebouwd, zodat de werking gemakkelijker in detail is te volgen. De adresbus-buffers liggen weer tussen de 64-polige konnektor en de CRTC IC11. IC10 en IC16 zijn de databus-buffers. IC16 is alleen nodig als men een lichtpen wil toepassen. Is dat niet het geval, dan kan dit IC dus vervallen en door 8 draadbrugjes worden vervangen. De adreslijnen A0...A10 zijn aangesloten op de A-ingangen van de 2-naar-1-multiplexers IC12...IC14. Door N1...N13 worden de adreslijnen A3...A15 geïnverteerd. De positie van de aansluitpunten A3...A15 en $\overline{A3}$...$\overline{A15}$ in het principeschema is overeenkomstig de opstelling op de printplaat getekend. Deze aansluitpunten worden met de ingangen A...R van de dekoders N37 en N38 verbonden. In het schema zijn de verbindingen nog eens rechts afgedrukt, en bovendien zijn ze ook in tabel 7 vermeld. N37 dekodeert de adressen voor de video-RAM IC15 en N38 doet hetzelfde voor de CRTC.

De videomengtrap bestaat uit N34, N31, N22 en de schakeling rond de transistoren T1 en T2. Hier wordt het Y-signaal van IC20 (de seriële punt-informatie) met de lijn- en beeldsynchronisatiepulsen (pen 39 en 40 van IC11) gemengd. De IC's 11, 12...14, 15 en 18...20 veroorzaken een tijdvertraging van enige honderden nanosekonden. Om alle signalen op het juiste tijdstip bij de videomengtrap te laten arriveren, vertragen de flipflops FF1...FF4 de signalen DEN en CUR van IC11 twee karaktertijden. Daarmee is alles weer "op tijd".

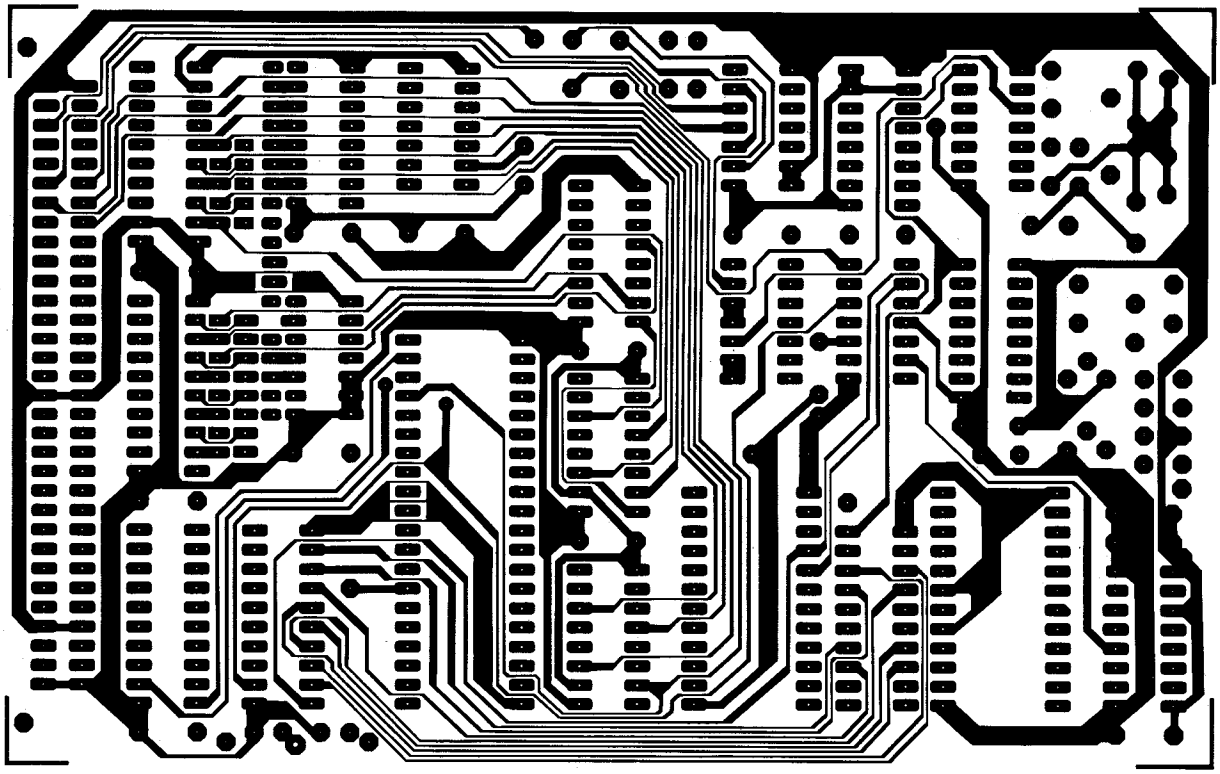
Nog iets over de draadbrugjes. Met de draadbrug bij N33 kan het beeld worden geïnverteerd. In positie T krijgt men het normale beeld, dus lichte karakters op


IC16
IC1
IC2
IC10
IC8
IC9
IC11
IC4
IC14
IC13
IC12
IC15
IC6
IC7
IC18
IC5
IC21
IC19
IC17
IC3
IC20
P1 P2
RA4
REFRSH


3
soldeerzijde
83082

3

komponentenzijde



Onderdelenlijst

Weerstanden:
R1,R2 = 470 Ω
R3,R4,R8 = 100 Ω
R5,R6 = 4k7
R7 = 68 Ω
R9,R10 = 2k2

Kondensatoren:
C1 = 40 p trimmer
C2 = 10 p trimmer
C3,C5...C19 = 100 n
C4 = 1 μ/6 V

Halfgeleiders:
T1,T2 = BSX 20 (of BC 547B)
IC1,IC2 = 74LS240
IC3 = 74LS04
IC4 = 74LS00
IC5 = 74LS86
IC6 = 74LS10
IC7 = 74LS27
IC8 = 74LS30
IC9 = 74LS133
C10,C16 = 74LS245
IC11 = 6545 of 6845
IC12,IC13,IC14 = 74LS157
IC15 = 6116
IC17 = 74LS175
IC18 = 74LS273
IC19 = 2732
IC20 = 74LS166
IC21 = 74LS163

Diversen:
X1 = Xtal 16 MHz (bij een display-konfiguratie van 80 × 24 karakters; bij gebruik van kristal vervallen C1, C2 en L1)
L1 = 4μ7
Konnektor volgens DIN 41612, 64-polig male, A- en C-rijen bezet

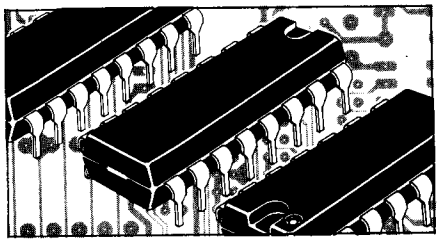
Figuur 3. De print voor de VDU-schakeling. Let extra goed op bij het leggen van de verbindingen bij de adresdekoder.

Tabel 1. Zo moeten de aansluitpunten van de adreslijnen worden verbonden met de ingangen van de poorten N37 en N38.

Tabel 1. Adresdekodering Octopus 65.

	aansluiting	adreslijn
poort N37	A	A15
	B	A14
	C	A13
	D	$\overline{A12}$
	E	A11
poort N38	F	A15
	G	A14
	H	A13
	I	$\overline{A12}$
	J	$\overline{A11}$
	K	$\overline{A10}$
	L	$\overline{A9}$
	M	$\overline{A8}$
	N	$\overline{A7}$
	O	$\overline{A6}$
	P	$\overline{A5}$
	Q	$\overline{A4}$
	R	$\overline{A3}$

donkere ondergrond, terwijl positie S donkere karakters op lichte ondergrond oplevert. Omdat de Octopus 65 met de 6502-CPU-kaart werkt, moeten de brugjes U-W en X-Z worden aangebracht. De brug RFRSH bij N28 vervalt. Als men de klokgenerator voorziet van een kwartskristal (het beeld is dan stabieler) vervallen L1, C1 en C2. Voor de transistoren T1 en T2 wordt in de onderdelenlijst het type BSX20 opgegeven, maar men kan ook het type BC547B toepassen.
De schakeling heeft een voedingsspanning van 5 V nodig, waarbij ca. 450 mA wordt opgenomen.
Figuur 3 laat de print zien voor de VDU-schakeling. Hier gelden dezelfde aanwijzingen voor de bouw als bij de CPU-kaart.
Met de instelpotmeters P1 en P2 kunnen de helderheid en het kontrast worden ingesteld (dit hangt ook af van de gebruikte monitor). De beide potmeters worden eerst in de middenstand gezet. Daarna zoekt men een instelling waarbij een duidelijk beeld ontstaat. Denk er aan dat de potmeters elkaar beïnvloeden.
Met de trimmers C1 en C2 kan de oscillatorfrekwentie worden afgeregeld (grof- en fijnregeling). Bij toepassing van een 16-MHz-kristal is die afregeling natuurlijk overbodig.

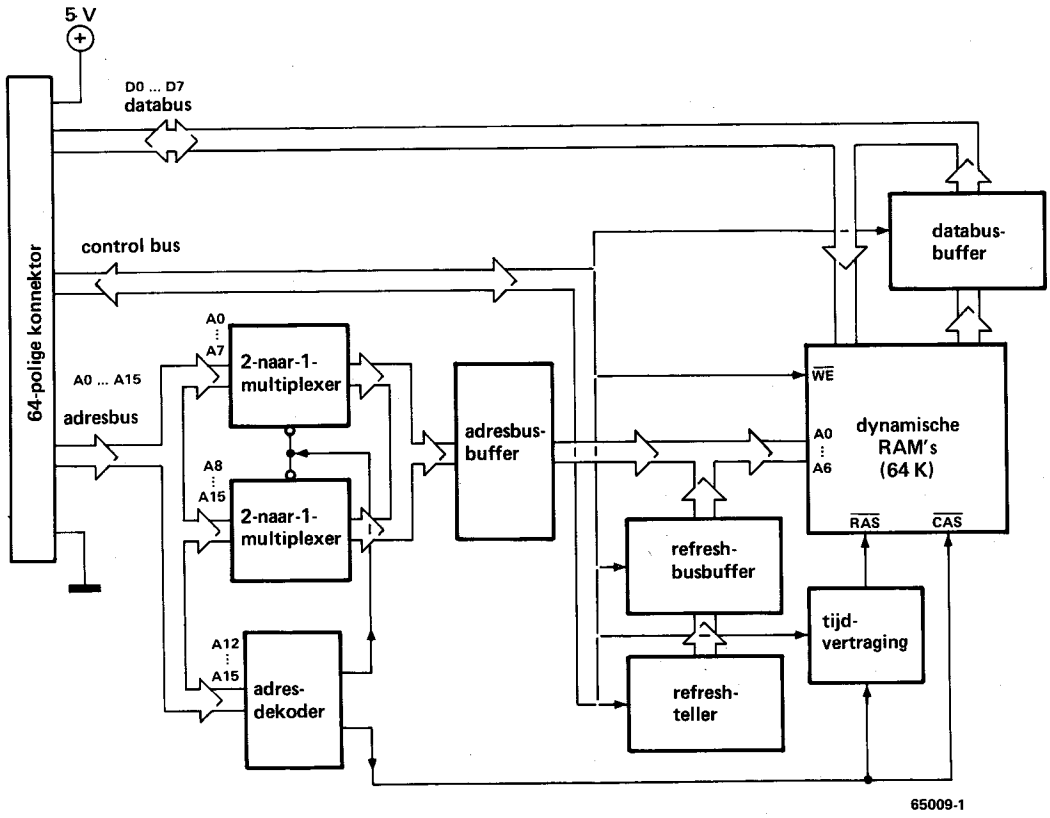


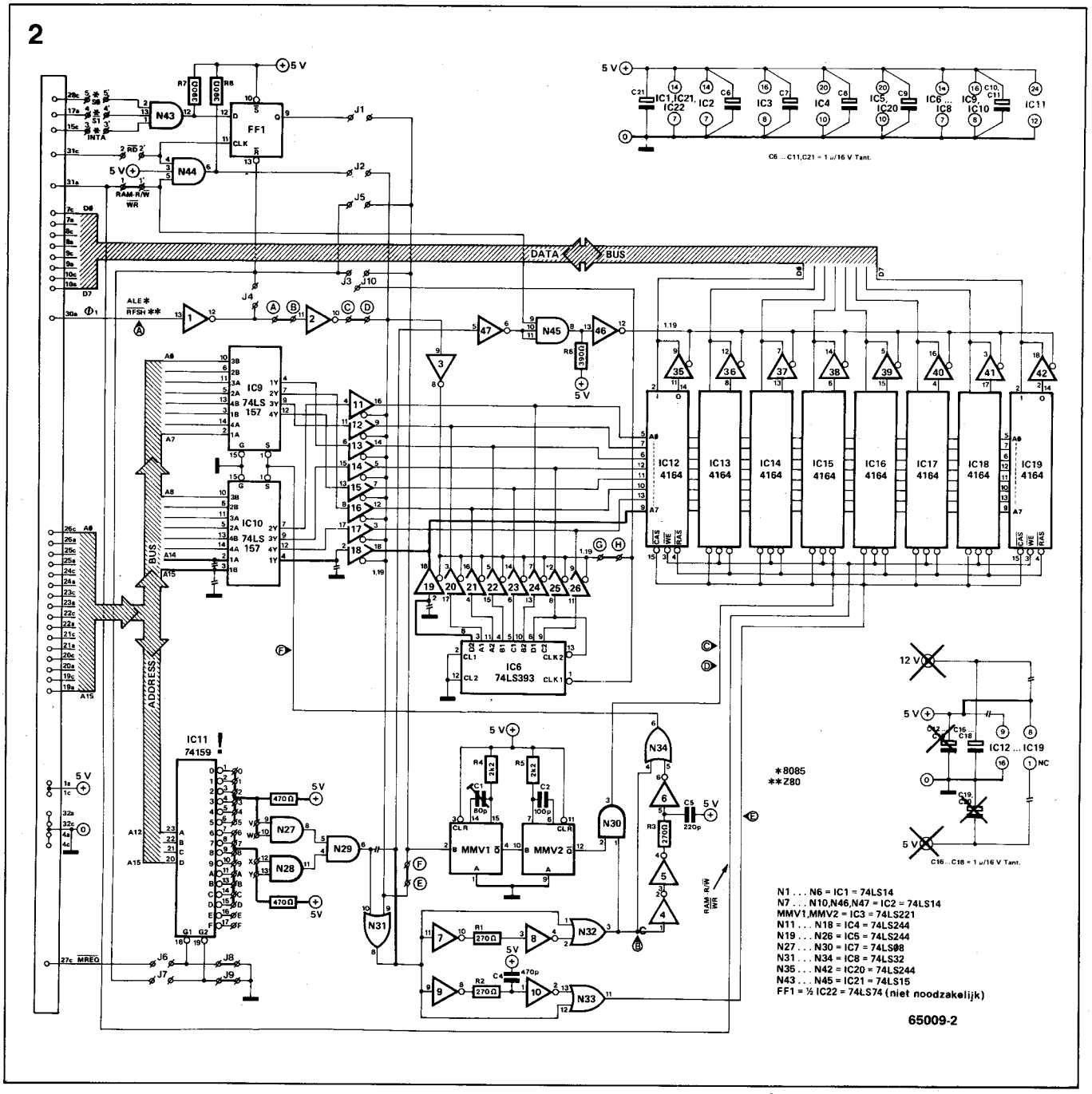
De DRAM-kaart

Octopus' vluchtige geheugen

Er is in de redaktie nogal wat gediskussieerd over de vraag of we de bestaande dynamische geheugenkaart zouden blijven gebruiken, of dat een nieuwe kaart moest worden ontworpen. Hoewel een nieuwe kaart er mooier en beter zou kunnen uitzien, hebben we toch besloten dat niet te doen, omdat vele lezers reeds over de bestaande DRAM-kaart beschikken en een groot aantal daarvan de uitbreiding van 16 K naar 64 K reeds zal hebben uitgevoerd. Bovendien biedt deze kaart een goedkope mogelijkheid voor het opbouwen van een 64-K-geheugenkaart. Men moet dan wel een beetje extra werk voor de aanpassingen voor lief nemen.
Wie nog een 16-Kbyte-geheugenkaart bezit, doet er verstandig aan deze voorlopig onveranderd te laten en de ombouw naar 64 K later, na de ingebruikname van de Octopus 65, te realiseren. Zie hiertoe ook het hoofdstuk "De Octopus 65 wordt operationeel". De Octopus aksepteert eigenlijk elke samenstelling van RAM-kaarten, mits ze maar op de juiste wijze worden geadresseerd (géén dubbele adressering) en bij de dynamische types rekening wordt gehouden met de "refresh". Men mag ook Elektuur-vreemde geheugenkaarten toepassen, mits deze maar aan de Elektuurbus zijn aangepast.
Het geheugenbereik van (hex)0000 tot DFFF kan worden gebruikt voor RAM-geheugen. Daarbij is het gemakkelijk wanneer het gedeelte van D000 tot DFFF met een schakelaar kan worden afgeschakeld, zoals we dat verderop zullen beschrijven. Dit is zelfs noodzakelijk als men de Octopus ook als ontwikkelingssysteem wil gaan gebruiken en EPROMs wil programmeren. De RAM moet vanaf 0000 als één blok in het adresbereik liggen, waarbij de grootte er in eerste instantie niet toe doet. We zijn van mening dat 32 Kbyte wel de minimale geheugenkapaciteit is om zinvol met de Octopus te kunnen werken. Beter is 48 K, maar 56 Kbyte is optimaal, waarbij we natuurlijk niet het voor speciale doeleinden gereserveerde 2-Kbyte-IC op de CPU-kaart en de 2-K-video-RAM op de VDU-kaart meerekenen. Zo werken alle tot nu toe (bij ons) gebouwde Octopussen met de 64-K-kaart die we nu nader zullen bespreken.
We beginnen weer met het blokschema (zie figuur 1). Links weer de 64-polige konnektor voor aansluiting van de DRAM-kaart op de Elektuurbus. De adreslijnen A0...A15 zijn in drie blokken onderverdeeld. De groepen A0...A7 en A8...A15 zijn elk op een 2-naar-1-multiplexer aangesloten die de in totaal 16 adreslijnen tot 8 reduceren. De multiplexers worden zodanig gestuurd door de control-logika van de kaart, dat steeds op het juiste moment het rij-adres of het kolom-adres via de adresbusbuffer aan de adres-ingangen van de RAM verschijnt. Een derde adresblok bevat nogmaals de lijnen A12...A15. Dit is op de adresdekoder aangesloten, waarmee de kaart wordt geaktiveerd. Deze dekoder geeft de mogelijkheid om met behulp van draadbrugjes willekeurige 4-Kbyte-blokken te selekteren. Via een tweede busbuffer worden, eveneens onder kontrole van de control-logika, de van de refresh-teller afkomstige adressen naar de adres-ingangen van de RAM's gevoerd.
Voor het schrijven van data naar de RAM's is geen buffertrap nodig, wel echter bij lezen uit het geheugen. Hiervoor is een databus-buffer aanwezig, die door een control-bus-signaal wordt geaktiveerd.
Ook het schema (zie figuur 2) is weer hetzelfde als het blokschema opgezet. De IC's 12...19 vormen het eigenlijke geheugen: 64 K x 8 bit. De tri-state-poorten N35...N42 vormen de databusbuffers voor de data-uitgangen. De 2-naar-1-multiplexers IC9 en IC10 reduceren de

1





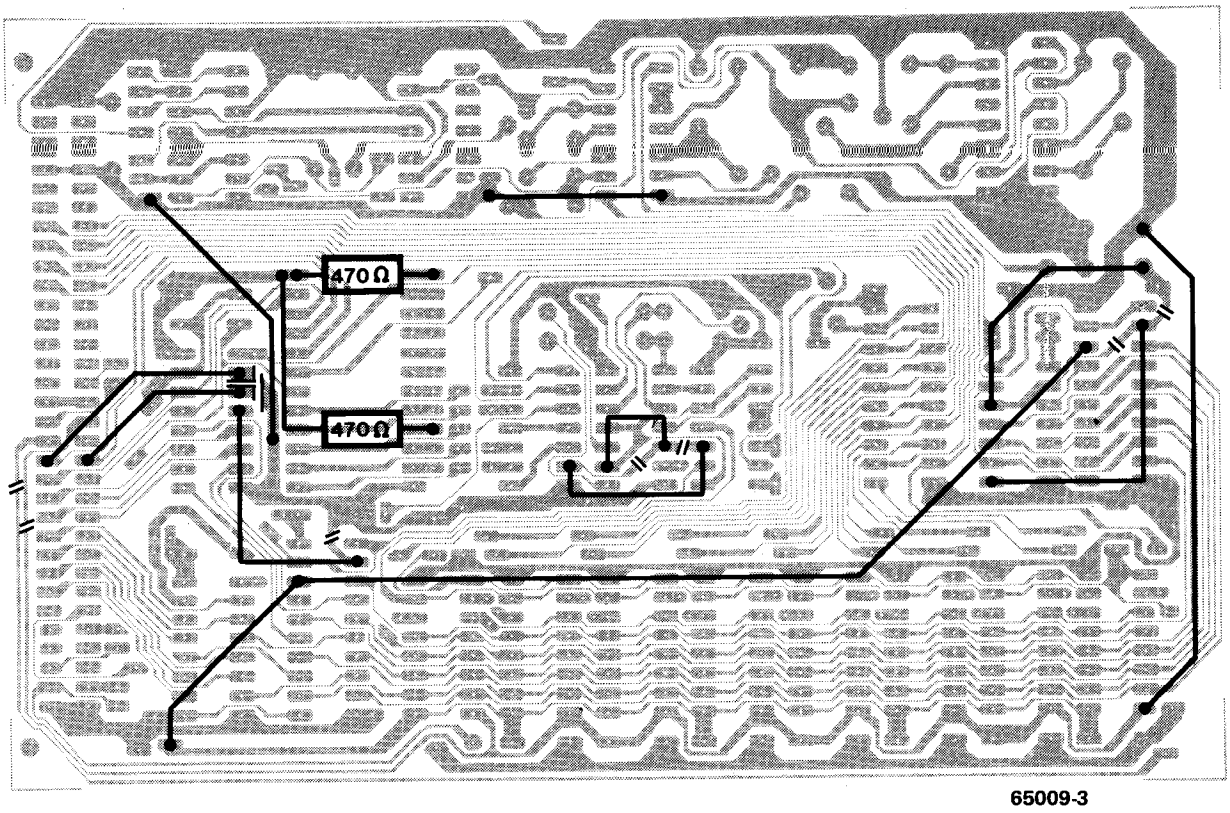
Figuur 1. Het blokschema kan worden onderverdeeld in drie grote blokken: adresmultiplexer en adresdekoder, refreshopwekking en tijdvertraging, en tenslotte het belangrijkste deel, het eigenlijke geheugen.

Figuur 2. Het schema van de 64-K-DRAM-kaart. Deze kaart is universeel van opzet en kan ook bij andere computersystemen worden toegepast.

16 adreslijnen tot 8 lijnen, die via de adresbusbuffers N11...N18 naar de adresingangen van de RAM's worden gevoerd. IC11 is de adresdekoder voor de 4-Kbyteblokken op deze kaart. De 8-bit-teller IC6 zorgt voor het regelmatig opfrissen van de RAM's. De uitgangen van IC6 zijn via de busbuffers N19...N26 (in IC5) verbonden met de adres-ingangen van de RAM's. MMV1 en MMV2 zorgen voor een tijdvertraging, zodat kort na de positieve klokpulsflank een neergaande puls op de RAS-ingangen van de RAM's verschijnt. Naast deze tijdvertraging moet er ook nog voor worden gezorgd dat de pulsflanken in de juiste volgorde aan de RAM-IC's en de multiplexers worden toegevoerd. De daarvoor benodigde timing-logika bestaat uit N4...N10, R1...R3 en C3 en C5. Een juiste timing is noodzakelijk voor het "samenspel" van de RAS- en CAS-ingangen van de RAM's. De WE-ingangen zijn direkt met de desbetreffende punten van de 64-polige konnektor verbonden. De hier verder niet genoemde poorten en flipflop FF1 dienen voor de eventuele aanpassing van de RAM-kaart aan andere computers. Daarom kan IC22 op de print rustig worden weggelaten als men van deze mogelijkheid toch geen gebruik maakt. Wie het schema goed bekijkt, ontdekt enkele onderbrekingen in de verbindingen, maar ook een aantal vetter getekende verbindingen. De schakeling was oorspronkelijk als 16-K-geheugen ontworpen. Dat is ook het geval met de print waarvan figuur 3 de koperzijde laat zien. Daarin zijn de veranderingen om van de 16-K-kaart een 64-K-kaart te maken duidelijk aangegeven. Om zeker te zijn dat niets wordt vergeten, noemen we hier alle noodzakelijke wijzigingen stuk voor stuk.

* Kopersporen onderbreken tussen:
- — aansluiting 2 van IC2 (N18) en massa
- — aansluiting 2 van IC5 (N19) en massa (let op, die massabaan moet later weer worden gebruikt!)
- — aansluiting 8 van IC12...IC19 en +12 V
- — aansluiting 1 van IC12...IC19 en —5 V
- — aansluiting 6 van IC7 (N29) en aansluiting 5 van IC2 (N47)
- — aansluiting 5 van IC2 en aansluiting 10 van IC8 (N31)
- — aansluiting 2 van IC10 en massa
- — aansluiting 3 van IC10 en massa
- — aansluiting 2 van IC10 en aansluiting 3

3



van IC10.
Test nu met een ohmmeter of de sporen werkelijk onderbroken zijn!

* Nieuwe doorverbindingen leggen tussen:
— aansluiting 8 van IC12...IC19 en de aansluitingen 1a/1c van de 64-polige konnektor (voedingsspanning +5 V)
— aansluiting 6 van IC7 (N29) en aansluiting 10 van IC8 (N31)
— aansluiting 8 van IC8 (N31) en aansluiting 5 van IC2 (N47)
— aansluiting 8 van IC6 en aansluiting 2 van IC5 (N19)
— aansluiting 4 van IC10 en aansluiting 2 van IC4 (N18)
— aansluiting 2 van IC10 en aansluiting 19c van de konnektor (adreslijn A14)
— aansluiting 3 van IC10 en aansluiting 19a van de konnektor (adreslijn A15)
— aansluiting 18 van IC4, aansluiting 18 van IC5 en aansluiting 9 van IC12...IC19 (adreslijn A7)
— aansluiting 9 en 10 van IC7 (V/W)
— aansluiting 12 en 13 van IC7 (X/Y)
— aansluiting 9 en 10 van IC7, via een weerstand van 470 Ohm naar +5 V
— aansluiting 12 en 13 van IC7, via een weerstand van 470 Ohm naar +5 V
— twee nieuwe massalijnen leggen (zoals aangegeven in figuur 3), uitgaande van de massalijn van de aansluitingen 4a en 4c van de 64-polige konnektor.

Als alle wijzigingen op de soldeerzijde van de print zijn aangebracht, beginnen we met de montage van de onderdelen. De tekening van figuuur 4 laat de oorspronkelijke 16-K-versie zien. Maar u laat eenvoudig alles weg wat niet in de onderdelenlijst is opgenomen. Let op: de draadbrug die naast IC9 ligt en gestippeld is aangegeven, mag beslist **niet** worden aangebracht!

Figuur 3. Er moeten enige modifikaties worden uitgevoerd om de oorspronkelijke 16-K-kaart om te bouwen tot een 64-K-kaart.

Figuur 4. Print-Layout en komponentenopstelling van de DRAM-kaart. Let bij de montage van de onderdelen goed op de aanwijzingen in de tekst.

Figuur 5. Zo ziet de geheugen-indeling er uit van de Octopus 65.

Onderdelenlijst

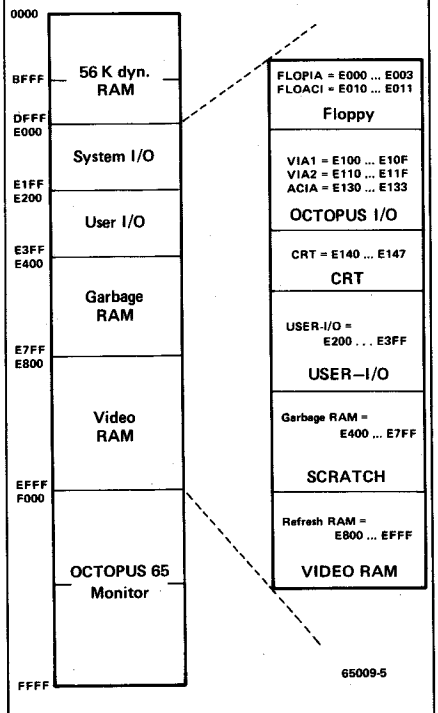
Weerstanden:
R1...R3 = 270 Ω
R4,R5 = 2k2
R6 = 390 Ω

Kondensatoren:
C1 = 80 p (trimmer)
C2 = 100 p
C4 = 470 p
C5 = 120 p
C6...C11, C16...C18, C21=1 μ/16 V tantaal

Halfgeleiders:
IC1,IC2 = 74LS14
IC3 = 74LS221
IC4,IC5,IC20 = 74LS244
IC6 = 74LS393
IC7 = 74LS08
IC8 = 74LS32
IC9,IC10 = 74LS157
IC11 = 74159
IC12...IC19 = 4164
IC21 = 74LS15

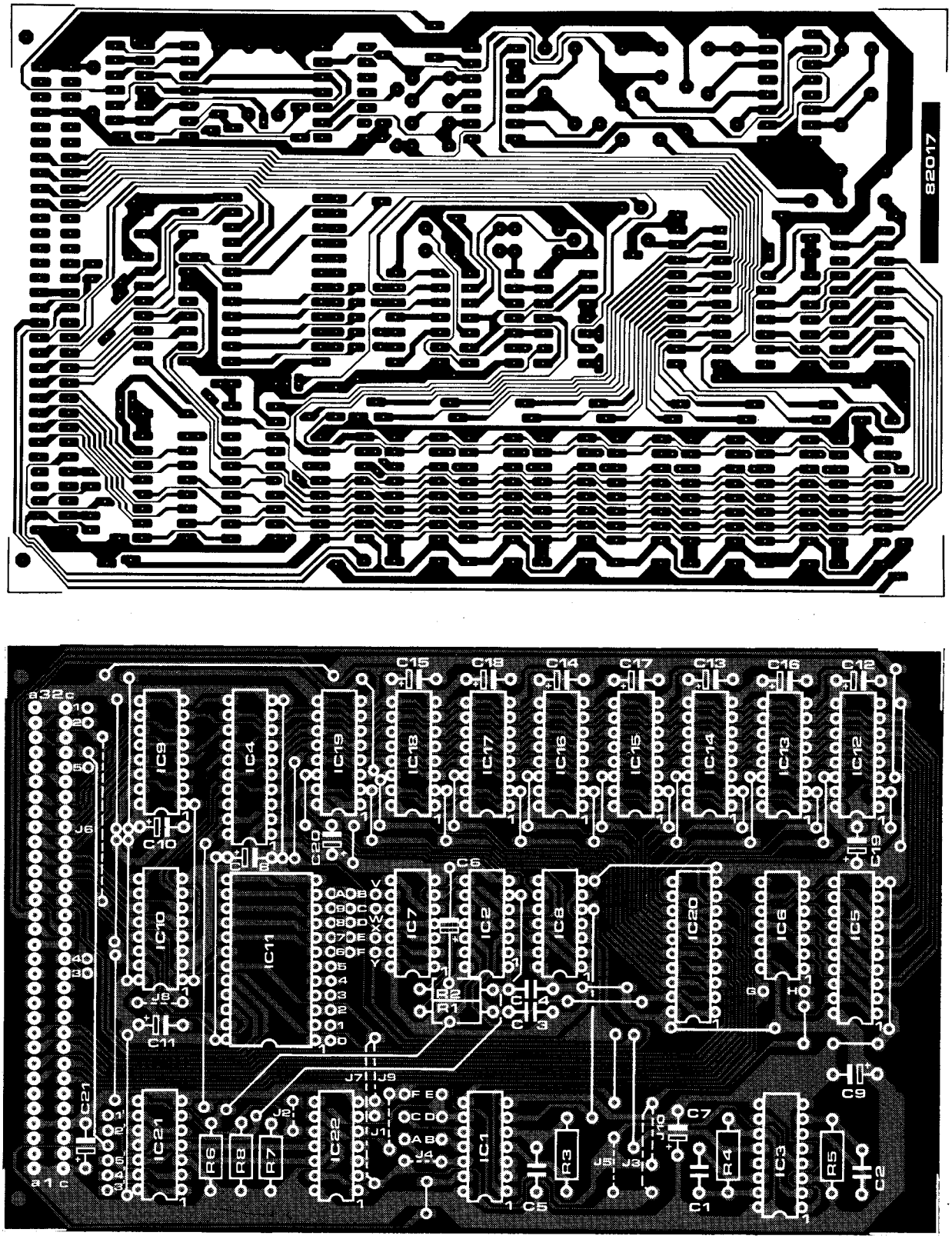
Diversen:
konnektor volgens DIN 41612, 64-polig, a- en c-rijen bezet
2 weerstanden van 470 Ω

5



Let verder op de volgende punten:
1. Monteer C7 vóórdat u trimmer C1 op de print zet, want deze laatste steekt over het gat voor de plus-aansluiting van C7 heen.
2. Voor de Octopus moeten de volgende draadbrugjes worden gemonteerd: 1-1'; A-B; C-D; E-F; G-H; J8 en J9. Verder alle doorverbindingen (behalve die bij IC9!) die niet nader zijn aangeduid, maar met

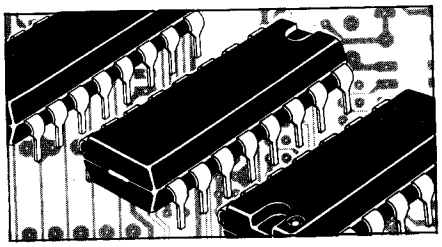
4



doorgetrokken lijnen zijn aangegeven.

3. De adressering van de geheugenblokken hangt alleen en uitsluitend af van de verbindingen Ø...F (uitgangen van IC11) en van de aansluitingen V/W en X/Y. Octopus reserveert de bovenste 8 K voor het monitorprogramma, de memory-mapped-I/O en de 2-Kbyte-RAM op de CPU-kaart. Er blijven dus 56 Kbyte over voor het werkgeheugen. Voor de adressering worden bij IC11 de uitgangen Ø...5 en 6...D met elkaar doorverbonden. Daardoor ontstaat een 24-K-blok (Ø...5) en een 32-K-blok (6...D). Legt men nu de 24 K aan X/Y en de 32 K aan V/W, dan staan ons 56 K ter beschikking.

Tenslotte nog een opmerking; wie van deze 56 K een blok van 4 K wil kunnen uitschakelen (bijv. voor het aansluiten van een EPROM-programmer) kan hiervoor blok D van de 56 K gebruiken, zodat nog 52 K overblijft. Men verbindt dan bij IC11 niet de uitgangen 6...D met elkaar maar 6...C; dat wordt dus een 28-K-blok. Met die 4 K van D kan men dan iets anders doen. Men kan het zelfs mooi maken door blok D met een schakelaartje bij of af te schakelen.



De FCU-kaart

Octopus' schijfgeheugen

De FCU-kaart, alsmede de opbouw en werking van disk drives, werd reeds uitvoerig behandeld in Elektuur. Dit voor degenen die hierover graag wat meer achtergrondinformatie willen hebben (zie het november- en decembernummer van 1982).

De FCU-kaart (FCU = Floppy Control Unit) is dus ook een oude bekende voor de trouwe Elektuurlezers, want wie de bovengenoemde artikelen heeft gelezen zal de hier gepresenteerde uitvoering meteen herkennen.

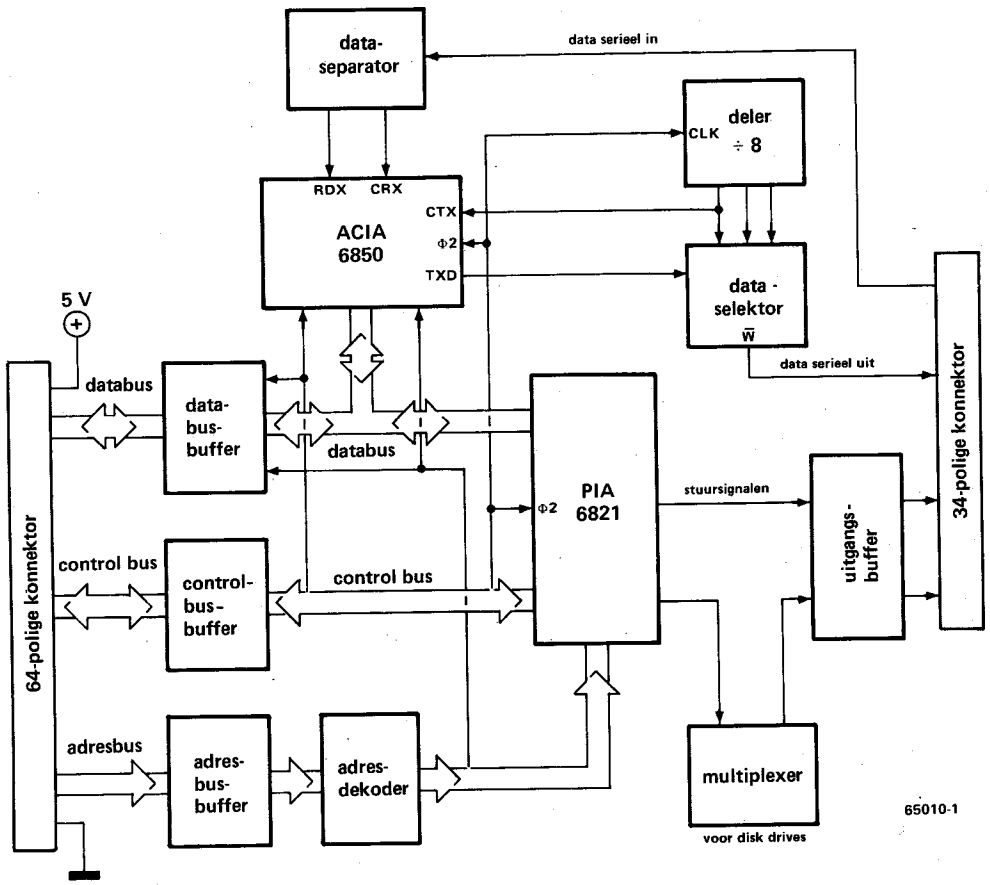
Natuurlijk was er een nieuwere, technisch betere oplossing mogelijk, maar dat zou alleen zin hebben als we dan tevens zouden overstappen op een ander opnameformaat, en daar willen we momenteel niet aan beginnen om diverse redenen:

1. We willen bereiken dat onze Elektuurlezers voor het realiseren van de Octopus 65 zoveel mogelijk bestaand (reeds gebouwd) materiaal kunnen toepassen; dat geldt niet alleen voor de hardware, maar vooral ook voor de software.
2. Het tot nu toe toegepaste opslagsysteem heeft bewezen enorm betrouwbaar te zijn. Dat heeft dan wel als konsekwentie dat er wat minder data op de diskette kan worden geschreven dan bij andere systemen, maar anderzijds zijn de prijzen van diskettes zo sterk gedaald dat we de voor- en nadelen van het omschakelen naar een ander opnameformaat eerst nog eens nauwkeurig willen onderzoeken.
3. Als u hierbij bedenkt dat een double-sided disk drive met 80 tracks per zijde thans minder kost dan een enkelzijdige 40-track-drive in nov./dec. 1982 (toen de kaart gepubliceerd werd), dan zal deze beslissing wel duidelijk zijn. Bovendien worden alle aanpassingen in deze Computer Special nauwkeurig beschreven.
4. Zouden we in de toekomst alsnog besluiten op een ander systeem over te gaan, dan willen we trachten dat zodanig te doen dat alle bestaande software op eenvoudige wijze naar het nieuwe systeem kan worden overgebracht.

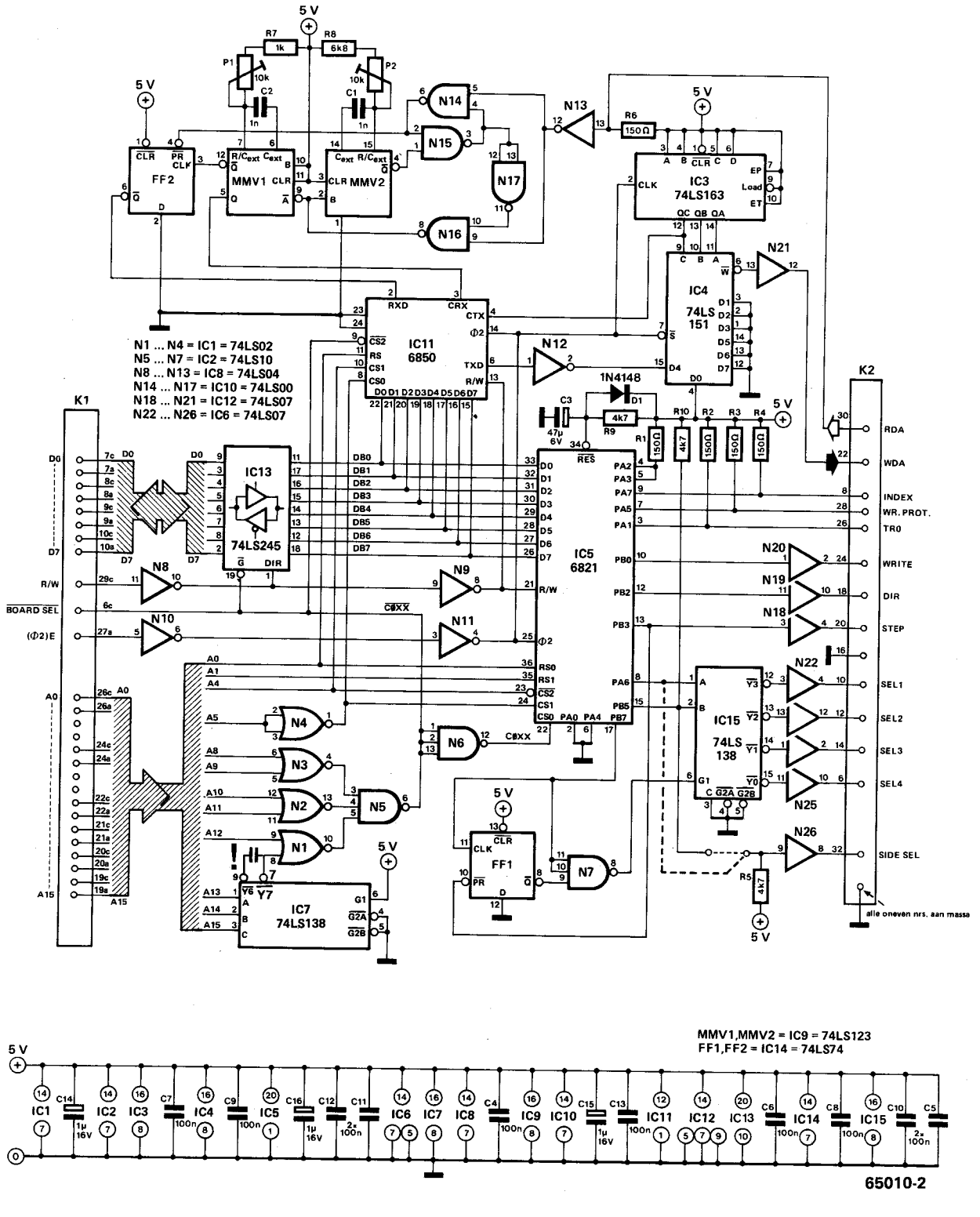
Het zal u niet verbazen dat we ook hier beginnen met de bespreking van het blokschema van de FCU-kaart (zie figuur 1). Links weer de 64-polige konnektor die de aansluiting op de Elektuurbus verzorgt. Op de FCU-kaart zijn alle adres-, data- en control-lijnen gebufferd (links in het blokschema). Naast de adresbuffer de adresdekoder die de PIA (= Peripheral Interface Adaptor), de ACIA (= Asynchronous Interface Adaptor) en de databusbuffer aanstuurt. De control-bus bepaalt met het R/W– signaal in welke richting de databusbuffer wordt geschakeld.

De ACIA maakt overdrachtsnelheden van 125 000 baud mogelijk. De werkklok wordt afgeleid door de systeemklok Φ2 met behulp van de 8-deler. Het aldus opgewekte signaal gaat naar de CTX-ingang van de ACIA. Dat geldt echter alleen voor de data-overdracht van de computer naar de drive. De ACIA leest de computerdata parallel in, waarna ze in seriële vorm ver-

1



2



Figuur 1. Het blokschema van de FCU-kaart laat goed zien hoe de data naar en van de floppy drive worden verwerkt.

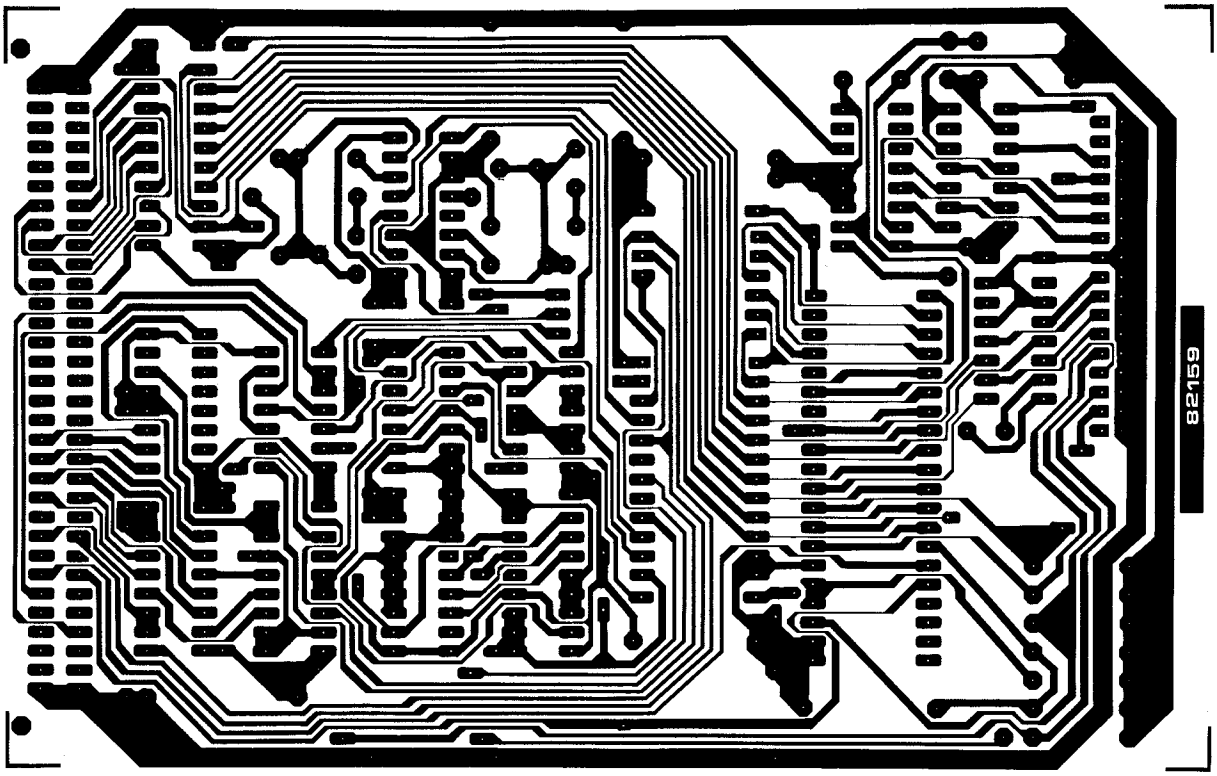
Figuur 2. Het schema van de floppy-interface. Doordat de data-uitwisseling tussen computer en drives door software gestuurd wordt, is geen speciale floppy-disk-controller nodig. In de schakeling worden alleen gewone onderdelen gebruikt.

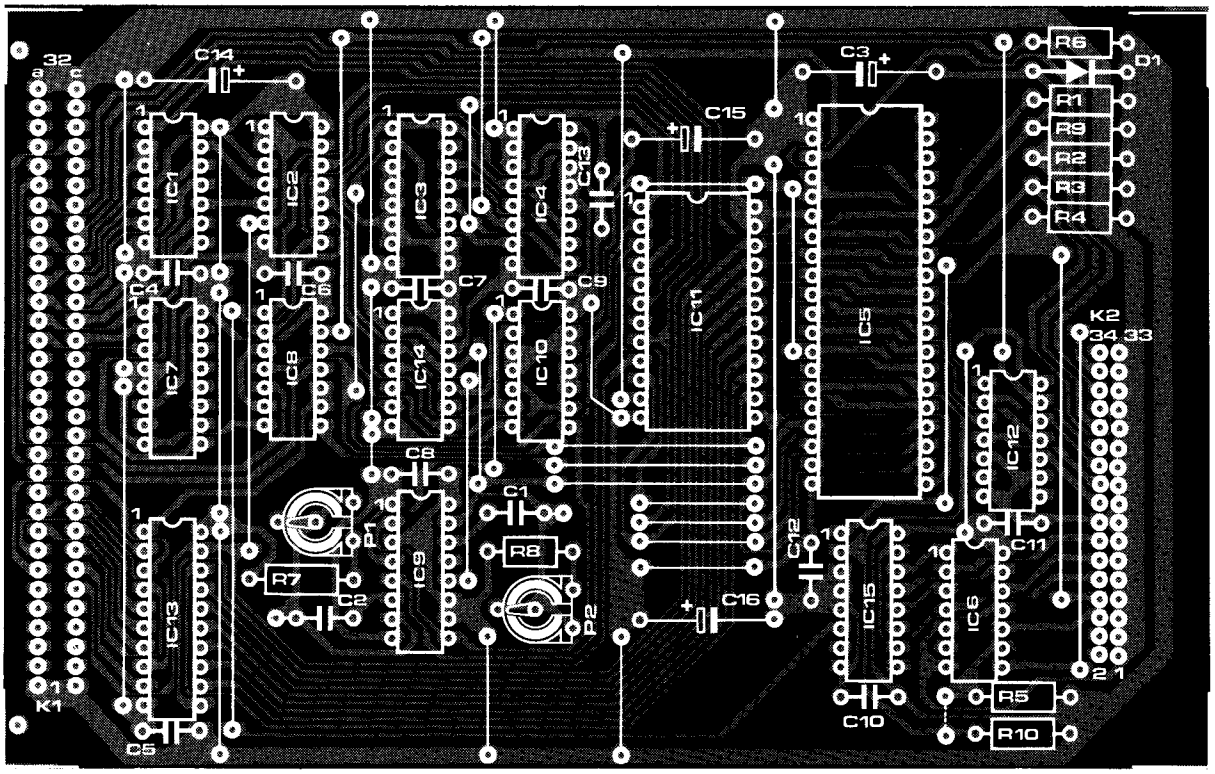
schijnen aan de TXD-uitgang in de volgorde: startbit, 8 databits, een pariteitsbit (even) en een stopbit. De dataselektor maakt daarvan een frekwentiegemoduleerd signaal. Via de 34-polige steker (rechts in het blokschema) wordt de FCU met de disk drive verbonden.

Als de Octopus data leest vanaf de disks, dan is de werkrichting omgekeerd. Via de 34-polige steker komen de seriële en frekwentiegemoduleerde data bij de dataseparator (boven), waar de data uit het FM-signaal worden gehaald. Het seriële signaal verschijnt aan de RXD-ingang van de ACIA, die in het ritme van het kloksignaal van de CRX-aansluiting de data inleest. De ACIA maakt nu van het seriële signaal weer een parallel-signaal dat via de databuffer naar de Elektuurbus gaat.

Voor de besturing van de floppy drive moeten nog verschillende signalen ter beschikking staan. Dit zijn: INDEX, WR.PROT., TR0, WRITE, DIR en STEP. Een multiplexer maakt verder van twee PIA-signalen vier, zodat de FCU-kaart vier enkelzijdige of twee dubbelzijdige drives

3





kan sturen.
Het schema (zie figuur 2) is weer op gelijke wijze opgezet als het blokschema. De databuffers in IC13 worden door het R/W-signaal als in- of uitgang geschakeld. IC7 verzorgt samen met de poorten N1...N6 de adresdekodering en de buffering. In eerste instantie verzorgt de ACIA (IC11) de data-overdracht. De computer schrijft het over te brengen byte via de databus in het zendregister van de ACIA, waarna dit parallel ingelezen karakter in seriële vorm aan de TXD-uitgang verschijnt. Dit wordt nu via N12, de dataselektor IC4 en N21 naar de WDA-aansluiting van de disk drive gevoerd.
Vanuit de disk drive wordt het signaal direkt naar de dataseparator gevoerd, die bestaat uit N13...N17, MMV1, MMV2 en FF2. De ACIA ontvangt het signaal via de

Figuur 3. De komponentenopstelling en de koperzijde van de FCU-kaart. Na het opbouwen van de andere kaarten zal dit ook wel prima lukken.

Onderdelenlijst

Weerstanden:
R1...R4,R6 = 150 Ω
R5,R9,R10 = 4k7
R7 = 1 k
R8 = 6k8
P1,P2 = 10 k instelpotmeter

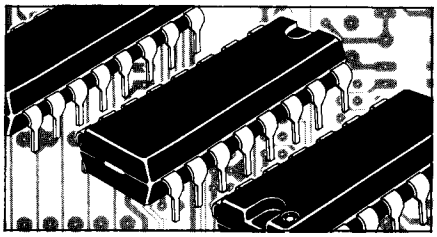
Kondensatoren:
C1,C2 = n MKT
C3 = 47 μ/6,3 V
C4...C13 = 100 n
C14,C15,C16 = 1 μ/16 V

Halfgeleiders:
D1 = 1N4148
IC1 = 74LS02
IC2 = 74LS10
IC3 = 74LS163
IC4 = 74LS151
IC5 = 6821
IC6,IC12 = 74LS07
IC7, IC15 = 74LS138
IC8 = 74LS04
IC9 = 74LS123
IC10 = 74LS00
IC11 = 6850
IC13 = 74LS245
IC14 = 74LS74

Diversen:
1 64-polige konnektor volgens DIN 41612, male
1 34-polige bandkabelkonnektor

aansluiting RXD (de klok hiervoor ligt aan CRX). De computer kan nu het signaal parallel uit het ontvangstregister lezen. Via pen CSØ wordt de PIA (IC5) geselekteerd om de reeds genoemde stuursignalen te leveren. IC15 multiplext de twee stuurlijnen PA5 en PA6, zodat via de 34-polige kabel vier enkelzijdige drives kunnen worden bestuurd. Indien men één of twee dubbelzijdige drives wil gebruiken, is een kleine aanpassing nodig. Dan moet vanaf N26, pen 9, een verbinding worden gelegd naar IC15, pen 1. Deze verbinding is in het principeschema als stippellijn aangegeven. In het software-deel van deze Special (zie onder "DOS-kommando's") hebben we de werking hiervan nader verklaard. Daar vindt u ook de noodzakelijke aanwijzingen voor het werken met dubbelzijdige drives.
Onafhankelijk van het aantal en soort disk drives dient nog een wijziging van de adresdekodering te worden aangebracht. Niet pen 9 van IC7, maar **pen 10** van IC7 **moet** met pen 8 van IC1 worden verbonden. Om dit te bereiken dient men pen 9 van IC7 zo om te buigen dat deze bij het insteken van IC7 in de voet vrij blijft en dus géén kontakt met de voet kan maken. Gebruikt men geen voet, dan kan pen 9 worden afgeknipt. Voor de verbinding van pen 10 van IC7 naar pen 8 van IC1 kan men het beste een stukje dunne kabel of gelakt koperdraad gebruiken, dat men direkt aan de soldeerzijde van de print op de desbetreffende punten soldeert.
De montage van de onderdelen op de kaart is vrij gemakkelijk. Aan de hand van de komponentenopdruk van figuur 3 worden eerst alle brugjes aangebracht, dan de weerstanden, kondensatoren, de diode D1 en de beide konnektoren. Als men IC-voetjes wil gebruiken, moeten dit voetjes van goede kwaliteit zijn. Dit geldt speciaal voor IC5 en IC11.
Nadat de voedingsspanning is ingeschakeld zal de floppy-interface gewoonlijk direkt werken als de twee instelpotjes in de middenstand staan. Is toch een afregeling nodig, dan dient deze als volgt te geschieden:
1. De steker van konnektor K2 verwijderen.
2. Daarna wordt de WDA-uitgang aan de koperzijde van de print door middel van een stukje draad doorverbonden met de RDA-ingang.
3. Uitgang $\overline{Q}$ van MMV2 wordt op een skoop aangesloten en daarna kan men P2 zo instellen dat de duur van de pulsen precies 5,5 μs is.
4. Als laatste wordt P1 zo verdraaid dat uitgang Q van MMV1 pulsen met een duur van 1 μs geeft. Deze laatste instelling is echter niet kritisch.

Tot slot nog het volgende: met deze FCU-kaart blijft de motor van de drive kontinu draaien, ook als de floppy niet door de computer wordt aangesproken. Dit kontinu draaien heeft het voordeel dat de toegangstijd zeer kort is omdat men niet behoeft te wachten tot de motor op toeren is. Maar aan de andere kant geeft het kontinu draaien van motor en floppy behoorlijk wat slijtage aan de floppy en aan de koppen. In het artikel "motorsturing voor disk drives" in Elektuur april 1984 werd een schakeling gepubliceerd die men nog kan toevoegen aan de bestaande kaart en die de motor automatisch uitschakelt.



De opbouw van de Octopus 65

Tot nu toe hebben we de aandacht gericht op de vier belangrijkste bouwstenen van de Octopus 65. Maar, al was de bouw daarvan een heel karwei, we zijn nog lang niet klaar. De vier kaarten moeten we met elkaar doorverbinden en voorzien van zaken zoals een voeding, een keyboard, een monitor, disk drive(s), enz.
Voor de verbindingen is een busprint nodig en we hebben daarvoor gekozen de in december 1983 in Elektuur gepubliceerde Omnibus (EPS 83102). Deze onderscheidt zich vooral van haar voorgangers door haar kapaciteit: zeven 64-polige "vrouwelijke" konnektors.
Het voor de Omnibus gekozen bussysteem werd reeds jaren geleden door Elektuur vastgelegd. Een uitgebreide definitie van de voor de 6502 en andere microprocessoren vastgelegde signalen kunt u vinden in het hiervoor genoemde nummer van Elektuur.
Zoals reeds eerder gezegd, is de Octopus 65 ontworpen om te werken met één tot vier disk drives. Dat kunnen dan ook één of twee dubbelzijdige drives zijn, afhankelijk van uw wensen en mogelijkheden. Het is mogelijk reeds in uw bezit zijnde drives in te bouwen, maar als u nog niet over drives beschikt raden we u aan te kiezen voor één of twee dubbelzijdige drives met 80 tracks per kant. We hebben proeven gedaan met de TEAC-drive type FD55F en de BASF type 6138. Met onze FCU-kaart en het OHIO-DOS-systeem kunnen beide drives worden omgeschakeld op 40 of 80 tracks. Het BASF-type is wat nieuwer en geruislozer. Het omschakelen van de TEAC kan zowel via hardware als software gebeuren, bij de BASF echter alleen via software. Deze softwaremethode is echter nog niet helemaal klaar. Met de TEAC zal het geen problemen geven; met de verderop gepubliceerde informatie kunt u de systeemsoftware van 40-track-diskettes naar 80-tracks overzetten. Overigens kan men elke bestaande disk drive toepassen, mits deze een "Shugart-bus" bezit. Helaas kan men de Apple- en Commodore-drives niet toepassen (althans niet de originele uitvoeringen met variabele toerentallen).
Er zijn natuurlijk verschillende mogelijkheden om voor de Octopus 65 een geschikte voeding te bouwen. Als men de voeding echter in de kast van de computer wil inbouwen (en dat is verreweg te verkiezen boven een aparte voeding) dan hebben wij een duidelijke voorkeur voor het ontwerp dat onder nummer 40 werd gepubliceerd in de Halfgeleidergids van 1984. We hebben daar drie redenen voor:
1. Door gebruik te maken van een speciale, maar goed verkrijgbare ringkerntrafo is een zeer kompakte opbouw mogelijk.
2. Het schema is gebaseerd op beproefde elektronische "recepten" en bekende, goed verkrijgbare regel-IC's. De prijs/kwaliteitsverhouding is daardoor uitstekend.
3. De hele voeding bestaat uit één trafo en één print en is in staat alle benodigde spanningen te leveren, inklusief de voeding voor twee disk drives: 5 V/3 A, 12 V/2 A (disk drives), 12 V/250 mA (aparte interface-spanning) en —12 V/250 mA.

De toegepaste trafo is van het fabrikaat

ILP (type 4T344), die over drie gescheiden sekundaire wikkelingen beschikt. Wil men rekening houden met toekomstige uitbreidingen, dan kan men het type ILP 6T345 toepassen, die het dubbele vermogen kan leveren. Wel moet men dan aan elke aanwezige TIP 142 een tweede exemplaar met emitterweerstand parallel schakelen om het vereiste vermogen te kunnen leveren. Voor verdere gegevens voor de nabouw, printplaat, montage, enz. verwijzen we naar genoemd artikel. Besluit men om drie aparte trafo's te gebruiken, dan heeft men minimaal nodig: een trafo van 9 V/5 A, een van 15 V/3,2 A en eentje van 15 V/0,4 A.

Het spreekt vanzelf dat we voor de kast, waarin we het geheel gaan onderbrengen, een stevig exemplaar moeten kiezen. Niet alleen de omnibus met kaarten, maar ook het zwaardere spul, zoals de disk drives en de voeding moeten er in kunnen worden ondergebracht. In principe kan men de Octopus 65 in elke 19"-kast onderbrengen. In het algemeen zal men dan nogal wat aanpassingswerk moeten verrichten om de kast "pasklaar" te maken. Zoekt u echter een gemakkelijker oplossing dan kunnen we u de door de Duitse firma Feltron/Zeissler gefabriceerde kast aanbevelen, die reeds geschikt gemaakt is voor de montage van de omnibusprint en verder voorzien is van alle gaten voor de montage van de overige delen. De foto's in deze Special geven daar een goed beeld van.

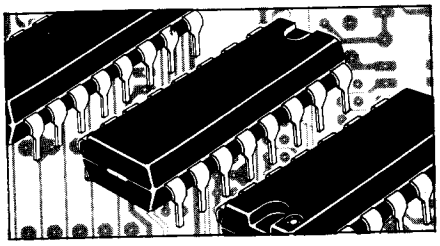
Hoewel randapparatuur eigenlijk niet behoort tot het bouwprojekt Octopus 65, hangt het sukses van het geheel natuurlijk ook af van de benodigde randapparatuur. Vandaar dat wij het noodzakelijk achten om onze zienswijze en ervaring hierover te geven, zodat u er uw voordeel mee kunt doen en de Octopus 65 tot volle tevredenheid van uzelf én van ons kunt gebruiken. We beginnen met het **keyboard**. Eigenlijk vinden we dat het keyboard een onverbrekelijk deel vormt van het Octopus 65 projekt. Daarom adviseren we de toepassing van het in Elektuur mei 1983 gepresenteerde ASCII-keyboard, dat bij de ontwikkeling van de Octopus 65 tot volle tevredenheid is gebruikt. De binaire informatie kan zowel parallel als serieel worden afgegeven en de printplaat biedt plaats voor een extra EPROM als men behoefte heeft aan omkoderingen. Maar men is natuurlijk niet gebonden aan dit keyboard. In principe kan men elk goed keyboard gebruiken dat ASCII-signalen produceert, hetzij parallel of serieel. Het verdient aanbeveling een type te gebruiken dat voorzien is van "auto-repeat", wat het werken met de "full-screen-BASIC-editor" en met de wordprocessor in belangrijke mate vereenvoudigt. Bovendien is het prettig als men het keyboard door middel van een EPROM kan omkoderen, omdat men dan de verschillende funktietoetsen (indien aanwezig) kan aanpassen aan de eigen behoefte. We willen nog even vermelden dat de reeds eerder genoemde firma Feltron/Zeissler een uitstekend ASCII-keyboard levert dat helaas niet goedkoop is, maar wel aan de strengste eisen voldoet.

Zeer belangrijk is natuurlijk ook de toegepaste **monitor**. Ook hier een paar "eisen". De bandbreedte moet minstens 16 MHz zijn. Denkt men in de toekomst veel met kleuren te werken (in voorbereiding is een grafische kleurenkaart), dan is de aanschaf van een kleurenmonitor gerechtvaardigd. Let hierbij op de aanwezige ingangen van de monitor; behalve de voor kleur noodzakelijke RGB-ingangen moet ook een "normale" ingang ter beschikking staan om tevens monochroom te kunnen werken. De grafische kaart zal ook de "normale" funkties van de VDU-kaart kunnen overnemen. De software hiervoor is echter zodanig opgezet, dat parallelwerken mogelijk is, dus gelijktijdige aansluiting van de VDU-kaart èn de grafische kaart, waarbij het programma zichtbaar is op de monochrome monitor en de grafieken in kleur op de kleurenmonitor. Voor de monochrome monitor hebt u de keuze uit groen en amber. Dat is een persoonlijke smaak; de één zweert bij groen, de ander wil niets anders dan amber. Ons advies: ga hier en daar eens kijken en maak voor uzelf uit wat het prettigst voor uw ogen is.

Dan iets over **printers.** Vroeg of laat kan voor u het moment komen dat u wel eens wat op papier wilt zetten. Dat kan al gebeuren als u een wat langer programma wilt bestuderen. Een listing op papier is dan noodzakelijk, want met de monitor alleen raakt u al gauw het overzicht kwijt. Zelf overschrijven, zo van de buis, is natuurlijk monnikenwerk, zodat u aangewezen bent op een printer. Nu zijn er printers en printers. U dient ook hier weer voor uzelf na te gaan waarvoor u die printer wilt gebruiken. Blijft het alleen bij het maken van listings of een boeken- of platenlijst voor eigen gebruik, dan is de kwaliteit niet het allervoornaamste en kunt u met een relatief goedkope printer uit de voeten. Maar voor kwalitatief betere afdrukken dient men toch dieper in de geldbuidel te tasten. Men dient dan een matrixprinter van hoge kwaliteit of een daisy-wheel-printer aan te schaffen. Hoewel de schriftkwaliteit van die laatste typen uitstekend is, staat daar een grotere investering tegenover. Voor het produceren van brieven of rapporten via het tekstverwerkingsprogramma bent u toch wel aan deze klasse gebonden.

Plotters zijn weer aan andere criteria onderhevig. Hierbij worden de letters, tekens, lijnen enz. door een gestuurde schrijfstift op papier gezet. Zij zijn ideaal voor het maken van grafieken en diagrammen, waartoe u dan diverse kleuren ter beschikking staan. Maar zij zijn doorgaans te traag om te gebruiken voor listings e.d. Een kleine X-Y-plotter werd beschreven in Elektuur van april 1985.

Dan nog iets over **modems.** Men kan tegenwoordig zeer veel informatie opvragen via de telefoon. Men kan zo'n modem huren (bij de PTT), kopen of zelf bouwen. Denk er echter wel aan dat voor zo'n telefoonmodem een PTT-goedkeuring moet worden verleend.



De Octopus 65 wordt operationeel

Als al het soldeer- en montagewerk gebeurd is, bent u natuurlijk hoogst benieuwd om te weten of alles nu naar behoren zal funktioneren. Het is dan erg verleidelijk om zonder meer de stekker in het stopkontakt te steken en de netschakelaar te bedienen.

Maar het is verstandiger om met wat meer overleg te werk te gaan. Bovendien moeten er nog enige afregelingen plaats vinden. Als alles inderdaad zonder fouten is gemonteerd, is er geen probleem. Maar als onverhoopt toch iets mis is, kunnen ongewenste effekten optreden en kan grote schade ontstaan. Vandaar dat we systematisch te werk gaan.

Octopus wordt stap voor stap in gebruik genomen. Als er dan fouten zijn weten we meteen waar we die moeten zoeken. Allereerst gaan we alle kaarten, die we natuurlijk al even in de buskaart hebben gestoken om te zien hoe dat er uit ziet, uit de bus-kaart halen. We schakelen nu in en testen eerst de netvoeding. Meet of alle gelijkspanningen op de buskaart aanwezig zijn. Is dat niet het geval, dan moet de fout eerst gelokaliseerd en verholpen worden. Daarna steken we de CPU-kaart op haar plaats en testen weer alle voedingsspanningen. Dan doen we hetzelfde met de VDU-kaart nadat de CPU-kaart weer is verwijderd. Daarna herhalen we datzelfde met de RAM-kaart als de VDU-kaart er weer uit is. Steeds testen of de voedingsspanningen nog in orde zijn. Nu schakelen we de netvoeding weer uit en steken de drie kaarten samen op de buskaart. Nu moet er al iets op de monitor verschijnen, hoewel waarschijnlijk wat vervormd. Dan moet de VDU-kaart worden afgeregeld.

Geheel links boven op het monitorscherm staat nu in duidelijke karakters:

— OCTOPUS 65 —

Het is ook mogelijk, dat er willekeurige

tekens kriskras over het scherm staan. Dat ligt dan waarschijnlijk aan de RAM-kaart. Wie nog een 16-Kbyte-RAM-kaart ter beschikking heeft of van iemand kan lenen, doet er goed aan die voor de afregeling te gebruiken in plaats van de 64-Kbyte-kaart. Maar ook met die tekens (dat kunnen ook grafische tekens zijn) kan men de VDU-kaart afregelen en zorgen dat die tekens helder en stabiel op het scherm staan.

Dan komt de RAM-kaart aan de beurt. De trimmer C1 hebben we in de middenstand gezet. Meestal is dat al goed, want die instelling is helemaal niet kritisch. De trimmer wordt telkens een stukje verdraaid, waarna steeds gereset wordt, totdat

— OCTOPUS 65 —

op het scherm verschijnt. Daarna voeren we in:

4000 = F000, FFFF cr

cr betekent steeds dat de Return-toets (carriage return, ook vaak aangeduid met Enter-toets) moet worden ingedrukt. Hierdoor wordt het gehele monitorprogramma in het RAM-geheugen vanaf $4000 gekopieerd. Hierna geven we in:

.H cr

Nu schrijft Octopus op het scherm Hexdump:

en wacht op de begin- en eindadressen voor het geheugengedeelte waarvan hij een hexdump moet laten zien. Dus voeren we in:

4000, 4100 cr

Op het scherm verschijnt nu de inhoud van het werkgeheugen: op het linker deel hexadecimaal, op het rechter deel van het scherm in de ASCII-kode als het desbetreffende byte overeenkomt met een ASCII-teken. Dat rechter deel van het scherm moet er uit zien zoals in figuur 1 is weergegeven. Daarna volgt weer een kontrole, waartoe we invoeren:

.H cr

en daarna

4AB0, 4BB0

De ASCII-kode in het rechter deel moet nu hetzelfde te zien geven als in figuur 2 is afgedrukt. Is dit niet het geval, dan moeten we de trimmer op de RAM-kaart nog iets naregelen en alles opnieuw proberen. Lukt het niet, dan moet men de moed niet opgeven; waarschijnlijk is een soldeerbrugje, dat bij de adressering van een der kaarten hoort, verkeerd geplaatst, of er is zo'n kortsluitbrugje op de CPU-kaart verkeerd gezet. Misschien is een van de wijzigingen op de RAM-kaart misgegaan?

In zo'n geval is het prettig als men iemand kent, die reeds een goed funktionerende Octopus 65 heeft. Men kan dan door het systematisch omwisselen van de kaarten eenvoudig te weten komen waar men de fout moet gaan zoeken. Men kan ook een van de in deze Special genoemde clubs benaderen, waar men u graag zal willen helpen.

Er is overigens een simpele methode om uit te zoeken of de fout in de CPU- of VDU-kaart zit, of dat de RAM-kaart de schuldige is. In figuur 3 zien we een kleine hulpschakeling. U gaat die even bouwen. De schakeling bestaat uit een enkele statische RAM van het type 6116 en een kondensator over de voedingsspanning. Daarnaast heeft men een 24-polige IC-voet nodig, een 64-polige steker, een stukje gaatjesprint en wat draad. De schakeling is zo uitgekiend dat hierop alle beslist noodzakelijke geheugenadressen staan, vandaar de wat merkwaardige adressering. Deze schakeling steekt men in de buskaart in plaats van de RAM-kaart en schakelt de Octopus 65 dan weer in. Nu moet weer links bovenaan op het scherm:

— OCTOPUS 65 —

verschijnen. Als dat niet het geval is, moet de fout in de CPU- of VDU-kaart worden gezocht. Hierbij nemen we even aan dat u die hulpschakeling in elk geval zonder fouten hebt gebouwd. Komt de tekst in de juiste vorm op het scherm, dan zit de fout in de RAM-kaart.

Van de bij ons gebouwde Octopussen, ca. een dozijn, werkte 60% onmiddellijk. De meest voorkomende fout bij de overige kaarten was de foutieve plaatsing van soldeerbrugjes. Maar wij hadden dan ook niet zo'n duidelijke en overzichtelijke handleiding ter beschikking als deze Computer Special! Voor de adressering hadden we slechts enkele aantekeningen tot onze beschikking. Verdere fouten waren: verkeerde modifikaties op de RAM-kaarten en foutief gemonteerde onderdelen. Dat kan ons ook gebeuren, meestal als iets snel klaar moet!

Als alles nu goed werkt, komt de FCU-kaart aan de beurt. Eerst doen we hetzelfde als met de drie eerste kaarten. Die worden verwijderd, we steken alleen de FCU-kaart op zijn plaats en kontroleren weer de voedingsspanningen. Daarna komt de disk drive aan de beurt: we kontroleren of alle voedingsspanningen aanwezig zijn. Als dat in orde is, worden de andere kaarten bij uitgeschakelde voedingsspanning in de bus gestoken, waarna men weer inschakelt en opnieuw de spanningen kontroleert. Daarna gaan we de zojuist beschreven eenvoudige geheugentest nog eens doen. Is alles in orde, dan worden pas de FCU-kaart en de disk drive(s) via een 34-polige bandkabel en de juiste konnektoren met elkaar verbonden. Zo'n floppy-drive-kabel is kompleet te koop, maar kan men ook eenvoudig zelf maken.

Eindelijk is het dan zover: we gaan "booten" (spreek uit: boeten). Eerst zetten we de twee instelpotmeters op de FCU-kaart in de middenstand. Dan gaan we naar het hoofdstuk "menu" in het software-deel van deze Special. Als alles gaat zoals daar is beschreven, kan men de rest van dit hoofdstuk verder vergeten. Maar als het niet lukt volgen hier nog enkele aanwijzingen.

In 20% van alle gevallen blijkt toch nog iets mis te zijn. Probeer eerst de instelling van de twee instelpotmeters op de FCU-kaart iets te wijzigen. Daarbij draait u beide potmeters tegelijk een stukje naar links en probeert alles opnieuw. Dat herhaalt u enkele malen totdat òf het menu op het scherm verschijnt òf de potmeters op ca. ¼ van de aanslag staan. In dat laatste geval draait u ze weer terug in de middenstand en probeert het nu in de andere richting. Als het menu eenmaal op het scherm verschijnt weet u dat u in elk geval in de goede richting zit. Bij het maken van de "back-up"-schijf (kopie van de originele schijf) van de systeem-

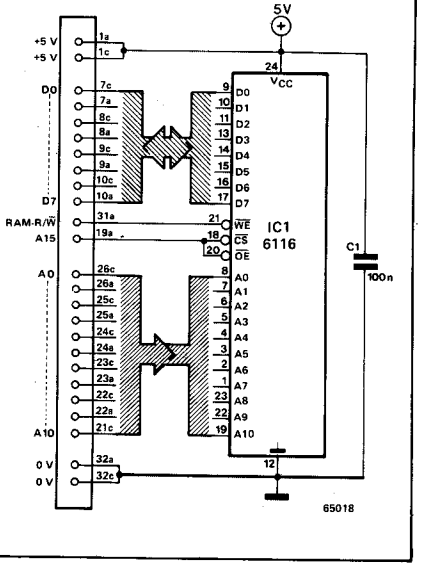
1

```
4000: H.H.H.c#.......h
4010: .h.h`S65D ö.....
4020: ..........1..L.
4030: ...... ..L.. ..L
4040: .. .. 1.L.. ..L.
4050: . ..L.. ..L.. ..
4060: L.. ..L.. ..L..
4070: 1.L..... d. ..
4080: L...c#. ..L.. ..
4090: L..8..........
40A0: ..)....`8.....
40B0: .......)....`8..
40C0: ..........)...
40D0: .`....m.....i
40E0: .)....`....m....
40F0: ....i.)....`....
4100: m.....i.)....
4110:
```

2

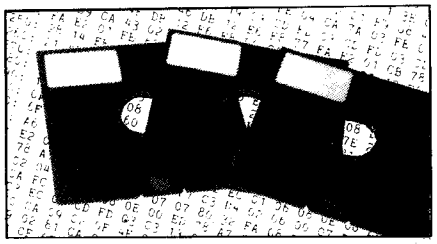
```
4AB0: .....-OCTOPUS 65
4AC0: - .....A  X  Y
4AD0: SP        PC   N
4AE0: V BDIZC... AT .L
4AF0: .P?.FULL.NR.?.WH
4B00: AT?.READY.HEXDUM
4B10: P: ....ASCII . .
4B20: ..^D,^L,^P,^S ?.
4B30: ..DISASM: .+1=..
4B40: 00..:0..A0..G0..
4B50: .`.@0..i.).`..h.
4B60: .h....h.........
4B70: ..  ..Lk..T.¥.r .
4B80: .  ..............
4B90: Lf....8.........
4BA0: ...L...W........
4BB0: ........L...N..
4BC0:
```

3



diskette "SYSTEM LOYS" heeft men gauw in de gaten of en hoe men verder moet: het gaat goed of het kopieerprogramma vertelt u in welke track er iets is misgegaan. Dus we draaien de beide potmeters nog iets verder in dezelfde richting en we proberen het opnieuw. Hierbij geldt steeds: eerst op de reset-knop drukken en dan opnieuw "booten". Deze simpele methode lost minstens 7/10 van die 10.. ..20%-probleemgevallen op.
Wil de zaak nog steeds niet naar behoren funktioneren, dan wordt de FCU-kaart afgeregeld zoals dat in de beschrijving van deze kaart is opgenomen. Bent u nog niet uit de zorgen, dan vindt u in het hoofdstuk "Waar vinden we wat in de DOS" nog aanwijzingen omtrent mogelijke fouten in de drive zelf (bijv. als bij gebruikte drives de kopafstelling niet meer in orde is).
Nog een opmerking tot slot: u hebt misschien bij het lezen van dit hoofdstuk de indruk gekregen dat de afregeling van de Octopus 65 moeilijk is. Dan vergist u zich beslist. Als men alle hiervoor beschreven aanwijzingen bij de bouw zorgvuldig heeft opgevolgd en geen montagefouten heeft gemaakt, dan heeft men zelfs bij uitgebreid kontroleren niet meer dan twee uurtjes nodig voor het opstarten van het systeem. De reden dat we in dit hoofdstuk toch zo uitvoerig op mogelijke fouten ingaan, is gelegen in ons streven u zo goed mogelijk te willen helpen bij het opzetten van de Octopus 65. Degene die bestaande units in de Octopus opnieuw wil toepassen, kan dan ook het beste die kaarten eerst aan een deugdelijke kontrole onderwerpen en moet vooral nagaan of alle aangegeven wijzigingen wel op de juiste wijze zijn doorgevoerd. Let hierbij vooral op de adressering.
Kunt u er echt helemaal niet meer uitkomen, schrijf ons dan gerust. We zullen dan trachten u uit de brand te helpen. Brieven met de strekking "mijn Octopus werkte na de afregeling meteen perfekt" zijn natuurlijk eveneens welkom.

SOFTWARE



Diskette-software

Een overzicht

Elke DOS-computer heeft ten opzichte van computers zonder disk drive(s) een groot principieel voordeel: hij kan zeer flexibel werken omdat het eigenlijke operating system waarmee de gebruiker werkt niet vast in de computer is opgeslagen (in ROM's of EPROM's), maar op disk. Dat biedt de mogelijkheid praktisch elk willekeurig operating system in de computer te laden, mits men ze op disk ter beschikking heeft.

Ter verduidelijking: hier worden niet de gebruikerprogramma's bedoeld, zoals een bankprogramma of een programma voor statistische, wiskundige en technische berekeningen. Natuurlijk kunnen ook deze programma's, nadat men ze in de computer heeft geladen en op diskette heeft gezet, van de diskette in de computer geladen worden als men ze nodig heeft. De computer laadt ze echter in een bepaald deel van het RAM-geheugen, het werkgeheugen (workspace).

Het operating system, dus het programma dat van de computer pas een produktief apparaat maakt voor de gebruiker, wordt daarentegen in een ander, daartoe gereserveerd deel van het geheugen geladen. Zo'n operating system bestaat grofweg uit twee delen: een basisgedeelte, waartoe naast het eigenlijke disk operating system (DOS) ook nog I/O-routines behoren, en de "transient language processor". Deze transient language processor kan bijvoorbeeld een BASIC-interpreter zijn, maar ook een assembler of een tekstverwerkingsprogramma. Zo kan een DOS-computer de ene keer een Pascal-computer zijn, de andere keer een Forth-computer, enzovoorts.

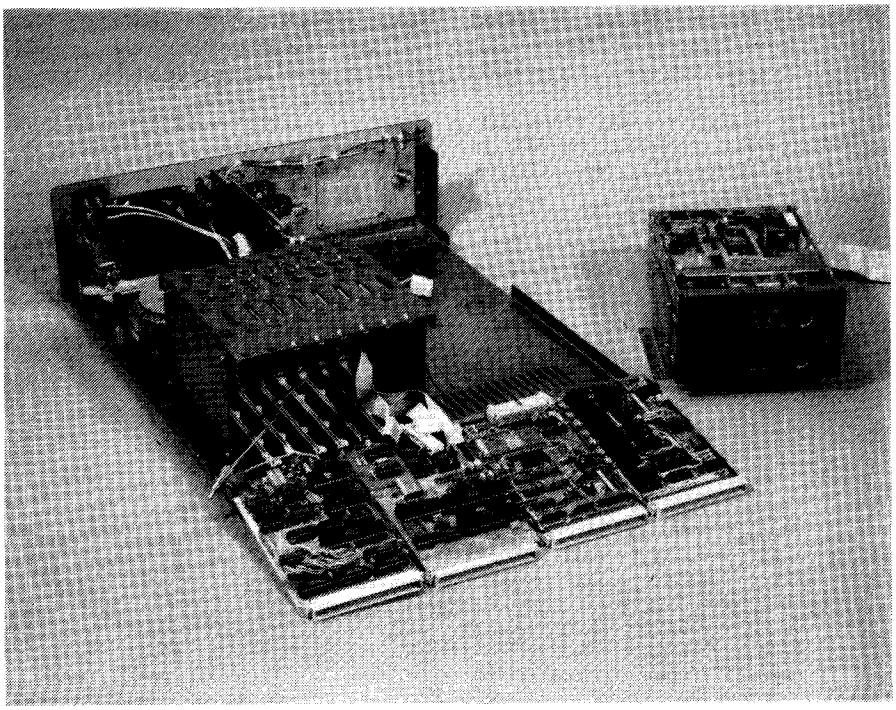
Het hele systeem is zo opgezet, dat de gebruiker niets aan het geladen operating system kan wijzigen en dus geen fouten kan maken, tenminste niet onder normale omstandigheden. Natuurlijk zijn er wel mogelijkheden om wijzigingen aan te brengen in uitzonderlijke situaties. Bij veel moderne computers heeft de gebruiker echter geen toegang tot het operating system, maar kan alleen de fabrikant dat. Dit is gelukkig niet het geval bij onze Octopus 65; daar zijn geen geheimen, zoals u zult merken in het verdere verloop van deze special.

Maar terug naar de flexibiliteit van DOS-computers in het algemeen en van de Octopus 65 in het bijzonder. Bij de Octopus 65 staan vijf operating systems ter beschikking:

* De BASIC-interpreter (Microsoft-BASIC)
* De Micro-Ware-assembler met editor en symbolische disassembler
* De door Elektuur ontwikkelde supermonitor met tracer en disassembler
* De oorspronkelijke OHIO-assembler.
* Het tekstverwerkingsprogramma "wordprocessor".

Maar dat is nog niet alles. Bij de aanschaf van het OHIO-DOS-pakket ontvangt men nog een aantal praktische utility-programma's, gedeeltelijk van en met het OHIO-DOS en gedeeltelijk door ons toegevoegd. Bij andere computers, zoals een Apple II, Acorn, Commodore 64, ZX-Spectrum, enz., moet u voor elk utility-programma tussen de *f* 50,— en *f* 100,— betalen, maar bij de Octopus niets! Het hierna volgende overzicht geeft de belangrijkste van deze utility-programma's (in alfabetische volgorde):

* "ATNENB". In- en uitschakelen van de arctangensfunktie. Is de arctangensfunktie uitgeschakeld, dan zijn de uitgebreide PRINT-funkties, zoals PRINT USING, beschikbaar.
* "BEXEC*". Een utility-programma dat steeds na het "booten" wordt opgeroepen (BASIC execution system). Het vormt de schakel tussen mens en computer.
* "BUFFER". Een utility-programma waarmee een buffer van willekeurige grootte voor een BASIC-programma kan worden gemaakt. Het is tevens mogelijk een reeds gedefinieerde buffer te vergroten, te verkleinen of te wissen.
* "CHANGE". Een utility-programma waarmee men het gehele werkgeheugen voor een bijzondere toepassing kan konfigureren, zoals:
  — Instellen van de hoogst mogelijke "pagina" (page) voor BASIC.
  — Instellen van de beide page-grenzen van een BASIC-programma.
  — Instellen van de BASIC-invoerbuffer (aantal karakters)
* "COMPAR". Een utility-programma waarmee men de inhoud van twee diskettes kan vergelijken.
* "CREATE". Een utility-programma dat



Fotoserie, figuur 1. Alle printen zijn opgebouwd en liggen hier klaar voor de montage in de kast. De twee drives zijn reeds gemonteerd tussen twee hoekstukken die straks in de kast worden vastgeschroefd. Het netvoedingsgedeelte is al in de kast gebouwd. De volgende foto's tonen nog enkele details voordat we overgaan tot de samenbouw van het geheel.

toegang geeft tot de directory.
* "DATRAN". Een utility-programma waarmee men files van de ene diskette naar de andere kan kopiëren.
* "GARBAG". Een utility-programma waarmee bytes of karaktergroepen in het geheugen of op een diskette kunnen worden opgezocht.
* "MERGE". Een utility-programma waarmee men twee BASIC-programma's aan elkaar kan koppelen.
* "MODEM". Een utility-programma dat zorgt voor een juiste koppeling tussen de Octopus en een modem.
* "NUMCON". Een utility-programma om getallen om te rekenen (dec/hex en hex/dec).
* "RANLST". Een utility-programma waarmee men de inhoud van random-files direkt naar de monitor of de printer kan sturen.
* "REPACK". Een utility-programma waarmee men de REM-statements en/of de spaties uit een BASIC-programma kan verwijderen.
* "RSEQ". Een utility-programma om de regels van een BASIC-programma te hernummeren (bijv. in stappen van tien).
* "SECDIR". Een utility-programma dat de sektorstruktuur van alle tracks aangeeft.
* "SEQLST". Een utility-programma dat de inhoud van een sekwentiële file toont.
* "TERMIN". Een utility-programma dat de Octopus omtovert in een VT52-terminal. Zo kan bijvoorbeeld een FORCE-2-computer, die met de 16-bit-processor MC 68000 werkt, direkt op de Octopus worden aangesloten.
* "TRACE". Een utility-programma waarmee men een BASIC-programma stap voor stap kan doorlopen. Heel praktisch bij het testen van zelfgeschreven of aangepaste BASIC-programma's.
* "ZERO". Een utility-programma dat sekwentiële of random-files initialiseert.

Er zijn nog veel meer utility-programma's, bijvoorbeeld GSOSRT, maar door plaatsgebrek kunnen we niet eens de hier genoemde uitvoerig bespreken. Maar u kunt alles in het OHIO-DOS-manual (65D Tutorial and Reference Manual, Ohio Scientific, August 1981) nalezen, of de gebruiksaanwijzing bekijken, of het programma laten "listen".
Tenslotte staan ons nog minstens twee andere programmeertalen ter beschikking: fig.FORTH kan, aangepast aan de Octopus 65, worden betrokken van de 6502-kenners-club (zie adres achterin) en een lid van de FORTH-interesse-groep heeft ons een in fig.FORTH ingepast tiny-PASCAL-programma gestuurd om op de Octopus 65 te testen (waar we nu druk mee bezig zijn). U ziet, er zijn genoeg mogelijkheden.

Menu

En de eerste stappen.

Het grote ogenblik is aangebroken. De hardware is klaar, de tests zonder de floppy-controller zijn tot volle tevredenheid afgesloten, nu moet er ge"boot" worden. Voor de nieuwelingen in het gezelschap van computeraars met disketsysteem vertellen we even dat het laden van het disk-operating-system van de diskette "booten" wordt genoemd. Maar laten we eerst nog even goed kijken: hebt u alle handelingen en kontroles, zoals in het voorgaande beschreven, uitgevoerd en klopt alles voor 100%? Want verder gaan heeft alleen zin als het hardwaregedeelte volledig in orde is!
Goed. Dan steken we de aangepaste systeem-diskette ("SYSTEM LOYS") in drive A en sluiten de klep. Op het beeldscherm staat nog steeds:

— OCTOPUS 65 —

Maar dat gaat snel veranderen. Men tikt .b of .B in. De CR-toets hoeft men niet eens te bedienen. Daar krijgt men niet eens de tijd voor, want drive A komt direkt zichtbaar en hoorbaar in aktie. Meteen daarna verschijnt op het beeldscherm het "menu". Dat behoort er uit te zien zoals in figuur 1. Klopt dat? Gefeliciteerd, dat betekent dat alle hardware goed funktioneert. Uw Octopus 65 is klaar voor het grote werk. Er moet weliswaar nog een kleine schakeling worden gebouwd, maar dat komt straks nog wel. Nu eerst aan de slag met de software. De meer ervaren OHIO-DOS-gebruikers hebben al gezien dat we BEXEC* uitgebreid hebben. Lees daarom dit hoofdstuk nauwkeurig door, want het is niet alleen voor de nieuwelingen bedoeld.
Wat is er nu eigenlijk gebeurd? De computer heeft tegelijkertijd vier dingen gedaan: hij heeft het disk-besturingssysteem (DOS), de BASIC-interpreter en het utility-programma BEXEC* geladen en is gelijk begonnen met het "runnen" van BEXEC*. De utility BEXEC* is een BASIC-programma dat de gebruiker 11 standaard-mogelijkheden biedt. Daartoe kiest men een getal tussen 1 en 11 (de mogelijkheden zijn duidelijk in het menu aangegeven) en drukt op de CR-toets, (CR = carriage return, afgeleid van de wagenterugloop en het beginnen op een nieuwe regel bij een schrijfmachine). Kiest u nu eens 1 en druk CR. Direkt verschijnt dan een nieuw menu met nieuwe keuzemogelijkheden. De Octopus 65 vraagt u dan: van welke disk drive moet de "directory" worden getoond? Even tussendoor: een directory bevat de inhoud van de diskette, maar kan nog meer informatie bevatten. Daarom noemen we het geen inhoud en houden we ons aan het Engelse woord directory. Nu zijn er vier mogelijkheden, want er kunnen maximaal vier drives op de FCU-kaart worden aangesloten. Kiest men drive A dan kan men A intikken en daarna CR, maar in dit geval is CR reeds voldoende. Voor de andere drives **moet** men eerst B, C of D kiezen en daarna nog op CR drukken. Goed. Nu dus alleen CR indrukken. Meteen komt Octopus 65 met de volgende vraag: moet de directory op de printer worden afgedrukt? Indien dat niet het geval is kan men N intikken, maar ook hier is alleen CR voldoende. Als u de vraag bevestigend wilt beantwoorden, tikt u Y in, maar dan moet er natuurlijk wel een printer zijn aangesloten en ingeschakeld. Voorlopig dus alleen CR indrukken. En meteen flitsen de regels van de directory over het scherm.
Zij die het OHIO-DOS-systeem uit ervaring kennen, zullen wel verbaasd kijken, want ook hier is het een en ander veranderd. Maar we zullen alles exakt toelichten.
Eerst gaan we iets doen dat heel erg belangrijk is. We gaan van de systeemdiskette "SYSTEM LOYS" en van de op de

```
--UTIL 5--                                  --- Aug. 9, 1984 ---

          1 > Directory
          2 > Create a new file
          3 > Change a file name
          4 > Delete file from diskette
          5 > Create blank data diskette
          6 > Create data diskette with files
          7 > Create buffer space for data files
          8 > Single or dual disk drive copier
          9 > Enter OS-65D system
         10 > Load assembler
         11 > Load word processor

Type the number of your selection
and depress RETURN ?
```

Octopus 65 aangepaste "Tutorial Disk 2" zogenaamde "back-ups" maken. Dat zijn kopieën van de originele schijven. We doen dat als veiligheidsmaatregel zodat we altijd op dat exemplaar kunnen teruggrijpen, mocht er ooit iets mis gaan met de diskette waarmee we werken. Uit ervaring weten we dat foutjes en ongelukjes nog steeds in een klein hoekje zitten. Men doet gauw een ondoordachte handeling en Octopus 65 is nu eenmaal razend snel. Wat nu volgt kunnen degenen die dit al vaker met OHIO-DOS hebben gedaan dus even overslaan. We gaan nu zo'n back-up maken en die moet straks dan als een archief-exemplaar worden bewaard. Men legt deze op een plaats waar hij beschermd is tegen magnetische velden (praktisch alle elektronische apparaten maken zich daaraan schuldig) en ook tegen grote hitte. Maar ook "onbevoegden" mogen niet de kans krijgen onbezonnen daden met uw computer te verrichten in een onbewaakt ogenblik en uw naarstig gekreëerde programma's ongewild onbruikbaar te maken.

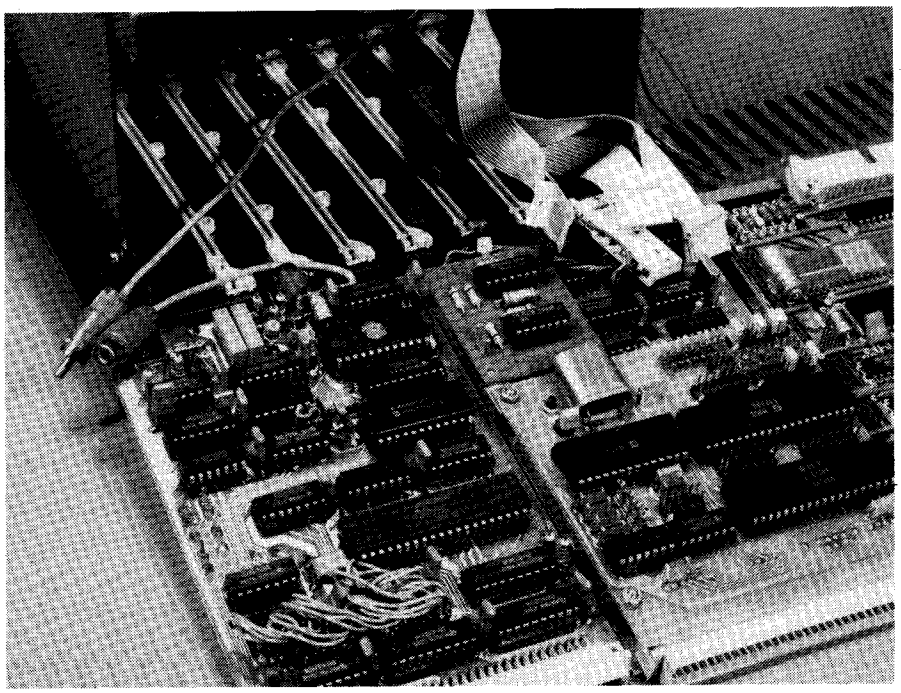
Eerst weer CR drukken en direkt verschijnt weer het hoofdmenu (dus het menu dat direkt na het "booten" op het beeldscherm verschijnt). Heeft men ongelukkigerwijze tussendoor al een fout gemaakt, dan de diskette uit de drive nemen, de netspanning uit- en daarna weer inschakelen en het systeem-programma opnieuw laden.

We tikken nu 8 in en daarna CR; prompt komt de kopieer-utility in aktie en vraagt: van welke drive moet worden gekopieerd? In principe kan men met A, B, C of D antwoorden, maar omdat de systeemdiskette in drive A werd gestoken tikken we eenvoudig A in en daarna weer CR. De volgende vraag van de Octopus 65 luidt nu: op welke drive moet de kopie worden gemaakt? Beschikt men over slechts één drive, dan is het antwoord eenvoudigweg A. Heeft men er meer, dan kiest men bijvoorbeeld B. Dan komt als volgende vraag: hoeveel tracks moeten worden gekopieerd? Ons antwoord: alle 40. Opgelet: omdat de nummering van de tracks bij 0 begint moet men 39 intikken! Daarna komt de laatste vraag van de Octopus 65; alles klaar voor het kopiëren? Ja natuurlijk, dus tikken we Y in. Wie over tenminste twee drives beschikt heeft het verder gemakkelijk, de rest verzorgt de computer. Heeft men er slechts één, dan moet men tussendoor nog de diskette verwisselen. Maar dat vraagt Octopus 65 u wel via het beeldscherm. Er verschijnt dan het Engelse woord "Insert" en bovendien wacht de computer op CR, zodat men zich nauwelijks kan vergissen. Alleen mag men de floppies natuurlijk niet verwisselen, de "master" is de diskette die gekopieerd wordt terwijl met "blank" de diskette bedoeld wordt waarop gekopieerd gaat worden. Als de kopie klaar is vraagt de Octopus 65 of er nog een andere diskette gekopieerd moet worden ("Do you want...?"). Ja, dat willen we inderdaad, want op dezelfde wijze maken we ook een back-up van de aangepaste Tutorial Disk 2. Dus Y intikken en direkt volgt de opdracht de "master" in de drive te steken. Daarna gaat alles zoals juist werd beschreven.

Heeft men twee drives, dan kan men natuurlijk ook van B naar B kopiëren, maar dat heeft eigenlijk weinig zin. Het gaat veel eenvoudiger en sneller om van A naar B te kopiëren. Maar let op: de Octopus 65 begint direkt nadat u Y hebt ingetikt, dus vóór die tijd moet de goede disk reeds op de goede plaats aanwezig zijn. Als men klaar is met het kopiëren kan men de bijgeleverde etiketten op de disks plakken en noteren welk programma zich op de disk bevindt. Schrijf in geen geval met een balpen direkt op de diskette, want dat kan fatale gevolgen hebben! Dus of eerst op het etiket schrijven en dan pas opplakken, of na het opplakken heel voorzichtig de tekst met een zachte viltstift aanbrengen.

Het kopiëren moet nu verder geen probleem meer zijn. Tegelijkertijd heeft men geleerd hoe men met de computer moet communiceren, want dat is het vraag-en-antwoord-spel toch eigenlijk. Bij de volgende voorbeelden kunnen we dat door plaatsgebrek niet meer zo uitvoerig beschrijven maar de door de computer gestelde vragen spreken meestal voor zich. In het geval u er toch moeite mee heeft kan een woordenboek (er zijn speciaal voor computergebruik samengestelde woordenboeken) een goede hulp zijn. Ook de reeds eerder genoemde mogelijkheid om steun te zoeken bij een computerclub kan hier weer uitkomst bieden. Het praten over uw problemen met anderen kan een zeer leerzame ondersteuning betekenen.

Maar we gaan terug naar het hoofdmenu: met de keuzen 9, 10 en 11 kan men "dieper" in het systeem duiken. Voor BASIC tikt men 9 in (of simpelweg CR), wil men met de assembler werken, dan tikt men 10 in en voor de wordprocessor kiest men 11. In het volgende hoofdstuk gaan we ons met keuze 9 bezig houden. De opties 2...7 roepen enkele nuttige utilities op. Vele daarvan komen nog aan de orde bij de beschrijving van de DOS in een der volgende hoofdstukken. Maar dat hoeft u nog niet te weerhouden het alvast eens te proberen. Werkt u voor het eerst met de OHIO-DOS, dan raden we u aan nog een kopie van de systeemdiskette te maken, waarna deze plus een lege (nog niet geinitialiseerde) diskette naast de computer worden gelegd en alle andere diskettes in veiligheid worden gebracht. En dan kunt u starten met de opties 2...7, waarbij het nodige vraag-en-antwoordspel u de gewenste ervaring zal geven in de kommunikatie met de Octopus 65. Houd in ieder geval het 65D Tutorial and Reference Manual (Ohio Scientific, August 1981) en het Basic Reference Manual (Ohio Scientific, 1981) bij de hand, want die geven zeer nuttige informatie.



Fotoserie, figuur 2. Een interessant detail op deze foto is het printje dat naast het kristal van de ACIA op de CPU-kaart is bevestigd. Dit printje bevat de tracer-schakeling, een onmisbaar stukje elektronica voor iedereen die de Octopus als ontwikkelingssysteem wil gebruiken. Verder ziet men op de foto duidelijk de kortsluitstekertjes op de CPU-kaart.

BASIC-kommando's

We zullen maar meteen eventuele misverstanden uit de weg ruimen door te vertellen dat dit hoofdstuk beslist geen spoedkursus BASIC bevat; daar is andere gespecialiseerde literatuur veel geschikter voor. Hier gaat het alleen om bepaalde details die met de Octopus 65 te maken hebben.

De eerste bijzonderheid werd reeds gepubliceerd in Elektuur december 1984, namelijk "enkeltoets-BASIC-kommando's". Dit "steno-BASIC" is bij de Octopus 65 vast ingebouwd. Door gebrek aan plaatsruimte verwijzen we hiervoor naar de bewuste Elektuur. Wie dat nummer niet (meer) heeft kan het alsnog bestellen. Kent u het artikel helemaal niet, dan toch even een kleine toelichting. Het "steno-BASIC" vergemakkelijkt het intikken van BASIC-kommando's. Zo hoeft men niet meer volledig "PRINT" in te tikken maar alleen de "Escape"-toets en de toets p in te drukken. Meteen verschijnt "PRINT" op de monitor. Natuurlijk vereist dat enige oefening, maar de benodigde tijd voor het ingeven van BASIC-opdrachten wordt er belangrijk door verkort. Op het voorin aanwezige overzicht van BASIC-kommando's kunt u de steno-BASIC-kommando's eveneens vinden.

De tweede bijzonderheid is ervaren OHIO-DOS-gebruikers reeds bekend: de direct-mode (in het Engelse DOS-manual: The BASIC Immediate Mode). Dit betekent het volgende: BASIC-opdrachten die zonder de gebruikelijke regelnummers worden ingetikt, worden na indrukken van de CR-toets onmiddellijk uitgevoerd. Een uitzondering vormen de opdrachten DEF, INPUT, READ, en DATA. Opdrachten als GOTO kunnen alleen betrekking hebben op een regelnummer van een programma in het werkgeheugen. De FOR-NEXT-lus kan in de direct-mode worden gebruikt maar moet daartoe wel in één enkele regel worden geplaatst. Dat is ook logisch, want als de lus zonder regelnummer in verschillende regels wordt opgenomen, ontbreekt de juiste sturing en gaat het mis. Belangrijk en ook zeer nuttig is de mogelijkheid om in de direct-mode ook met de opdrachten PEEK en POKE te kunnen werken. Maar voor we u verder op de computer "loslaten", gaan we eerst kijken naar de full-screen-editor en naar de diskette-kommando's.

De Editor is een nieuw programma dat tot nog toe niet werd gepubliceerd en dat dient voor het eenvoudig bewerken (wijzigingen, uitbreidingen, enz.) van in het werkgeheugen aanwezige BASIC-programma's. Daarmee kan men het op het beeldscherm aanwezige deel van een BASIC-programma erg simpel "editen", dus korrigeren, wijzigen of uitbreiden. We zullen dat straks in een praktisch voorbeeld laten zien.

Als laatste punt, vóór we met de praktijk beginnen, nemen we de voor het programmeren in BASIC belangrijke disk-opdrachten even door. Zij vormen een praktische toepassing van de direct-mode en eigenlijk een voorproefje van het hoofdstuk "DOS-kommando's". Dat hoofdstuk hoeft men niet door te lezen als men de Octopus 65 (voorlopig) alleen in BASIC programmeren wil en daarom hebben we die opdrachten hier opgenomen. De belangrijkste disk-opdrachten zijn PUT en LOAD. Met de PUT-opdracht wordt een programma op de diskette geschreven en met de LOAD-opdracht wordt een programma van de diskette in het werkgeheugen geladen. Voor de gelukkige bezitters van twee drives (of misschien nog meer) is ook de SELECT-opdracht belangrijk, waarmee een van de drives kan worden gekozen. Dan is er nog... nee, dat vertellen we wel bij de praktische voorbeelden.

Daar gaan we. Schakel de computer in, met .b "booten" en 9 CR intikken. Alleen CR kan ook, dat hebben we als extra komfort in het programma opgenomen. De BASIC-interpreter meldt zich met: "Your system is now open for modification" en de "prompt" OK. Deze prompt OK verschijnt steeds op het scherm als de computer op een nieuwe opdracht van de gebruiker wacht. We willen de Octopus 65 nu eens in de direct-mode als rekenmachine gebruiken en hem de wortel uit 4 laten trekken. Daarvoor tikken we in:

?SQR (4)

Hierbij even twee opmerkingen:

1. Als er nu niets gebeurt, dan hebt u vergeten de CR toets te drukken (niet meer vergeten!).
2. Het ? is bij Microsoft-BASIC een afkorting voor de opdracht "PRINT".

De uitkomst is natuurlijk 2, daarvoor heeft men de Octopus 65 niet nodig en ook geen rekenmachine. Daarom nu iets dat wat ingewikkelder is. We laten Octopus 65 de vergelijking:

$y = x^4 + 4x^3 - 7x^2 + 7$

uitrekenen. Daartoe moeten we nog wel een waarde voor x ingeven; we nemen als voorbeeld 7. We tikken nu in de direct-mode de volgende regel in:

X=7: ? X↑4 + 4*X↑3 — 7*X↑2 + 7

Eigenlijk zou nu de uitkomst 3437 op het scherm moeten verschijnen, maar wat zien we... 3437,00001? Maar een BASIC-computer maakt, in tegenstelling tot een echte rekenmachine, afrondingsfouten en die komen tot uiting in het antwoord. Het is beter om de volgende opdracht te geven:

X=7: ? INT(X↑4 + 4*X↑3 — 7*X↑2 + 7)

De afrondingsfout wordt dan weggelaten. Genoeg gerekend voorlopig. We gaan nu een BASIC-programma van de disk laden en als voorbeeld nemen we het programma "NUMCON", dat verderop in deze Special nog wordt beschreven. Dus we tikken in:

DISK!"LOAD NUMCON"

of, afgekort:

DISK!"LO NUMCON

Met DISK!" vertellen we de BASIC-interpreter dat hij het disk-operating-system moet inschakelen. Alleen de twee eerste letters zijn daarvoor voldoende, de andere dienen slechts ter verduidelijking voor de gebruiker en kunnen eventueel worden weggelaten. De tussenruimte (space) tussen LOAD (of LO) en de naam van het programma mag men niet weglaten! De computer leest ná LO niets meer van de rest van dat woord, dus de tussenruimte moet aangeven dat een nieuw woord begint. De laatste ", aan het eind van de opdracht, kan eveneens vervallen, mits men niet direkt daarna een volgende opdracht op dezelfde regel invoert. Na indrukken van CR wordt de drive even aktief en dan volgt de terugmelding "OK". Het programma "NUMCON" is nu geladen en kan met RUN worden gestart, waarna het bijbehorende menu op het beeldscherm verschijnt. Bij de OHIO-DOS kan men de opdracht RUN ook als opdracht voor het laden gebruiken. Het NUMCON-menu staat nog op het scherm, maar we gaan even terug naar BASIC door simpelweg CR in te toetsen. Direkt verschijnt de prompt OK. Nu gaan we eens intikken:

RUN"NUMCON

Let op! RUN mag hier niet worden afgekort tot RU, want dat is geen DOS-opdracht! Weer wordt drive A even aktief, maar in plaats van het OK verschijnt nu direkt het menu van NUMCON. De computer heeft dus het programma geladen en direkt gestart. Heeft men twee of meer disk drives, dan wordt er steeds één gekozen. Na het "booten" is dat automatisch A. Wil men op een andere drive omschakelen, dan geeft men de volgende opdracht:

DISK!"SELECT B"

of

DISK!"SE B

Men kan natuurlijk ook C of D kiezen als men over deze drives beschikt. In de direct-mode kan men hier of bij andere toepassingen meerdere opdrachten in één regel opnemen, mits ze worden gescheiden door : . In het volgende voorbeeld willen we de PUT-opdracht gaan gebruiken en we nemen even aan dat twee drives ter beschikking staan. We willen een programma met de naam PROBE op de disk laden en tikken in:

DISK!"PUT PROBE"

of afgekort:

DISK!"PU PROBE

Maar voor men een programma op de diskette kan zetten moet het eerst in het werkgeheugen staan. We hebben echter nog geen programma ingetypt. Daarom laden we een programma van een diskette en zetten het op een andere diskette. We laten de computer dat in één keer doen. Maar let op: indien men een nieu-

we, nog ongebruikte diskette wil gebruiken, moet deze eerst met utility 5 van het hoofdprogramma worden geïnitialiseerd. De gebruiksaanwijzingen daartoe zijn heel duidelijk. Daarna kunnen we onze meervoudige opdracht op de computer loslaten. Om dat te realiseren wordt de diskette "SYSTEM LOYS" in drive A geladen, terwijl in drive B de nieuwe, geïnitialiseerde diskette wordt gestoken. We gaan nu het programma BEXEC* van drive A overbrengen naar drive B onder de titel PROBE::

```
DISK!"SE A":DISK!"LO BEXEC*":
                DISK!"SE B":DISK!"PU PROBE
```

Bij het laatste kommando in de regel kan men " weglaten, bij de andere opdrachten niet!

Nu gaan we de editor bespreken. Om dat te proberen kan men het beste een willekeurig programma van de diskette in het werkgeheugen laden; we nemen als voorbeeld het reeds genoemde programma NUMCON. De editor kent de volgende opdrachten (de ↑ betekent weer dat de desbetreffende toets tegelijk met de control-toets moet worden ingedrukt):

↑K — een regel naar boven
↑M — een regel naar beneden
↑P — cursor naar rechts
↑H — cursor naar links
↑O — regel tussenvoegen
↑D — regel zowel op monitor als in geheugen wissen
↑R — cursor naar eind van de regel (rear)
↑F — cursor naar begin van de regel (front)
↑I — cursor naar volgende tabulator
↑L — beeldscherm wissen / cursor naar home-positie
↑@ — cursor naar home-positie brengen
↑Delete (DEL-toets) — wis het karakter onder de cursor
CR (carriage return) — cursor naar begin volgende regel.

Invoegen van karakters — cursor naar de gewenste positie brengen en het (de) karakter(s) intikken. Het op die plaats reeds aanwezige karakter wordt met alle volgende naar rechts opgeschoven. Dit gaat alleen zolang er in die regel voldoende plaats is.

Wie het keyboard uit Elektuur van mei 1983 bezit, hoeft de cursor-besturingstekens niet uit het hoofd te leren. Voor de cursorbesturing zijn op dit keyboard speciale toetsen aangebracht. De toetsen HOME en CLEAR kan men zonder meer gebruiken. Ook de auto-repeat-funktie kan bij alle opdrachten van de BASIC-editor worden toegepast.

Nu tikken we in:

```
LIST 1-15
```

Op het beeldscherm verschijnen nu de regels 1...15 van het programma. In elke regel blijft de eerste, meest linkse plaats vrij. De cursor kan nu in die voorste positie (en alleen in díe positie!) met ↑K en ↑M naar boven of beneden worden bewogen. Ook de kommando's ↑O en ↑D funktioneren slechts vanuit die positie. Met ↑D wordt de regel achter de cursor gewist en met ↑O kan men regels vóór de desbetreffende regel invoegen. Deze vier kommando's moet men nu eens proberen op het programmadeel dat op het beeldscherm staat. Hiertoe moet men de cursor eerst met ↑K in dit gedeelte brengen.

Wil men een bepaalde regel gaan wijzigen, dan brengt men de cursor eerst op de positie aan het begin van die regel. Daarna brengt men de computer in de editor-mode door een keer ↑P in te drukken. De cursor springt razendsnel door de regel heen en komt weer op de beginpositie terug. Dit gaat zo snel dat men het nog maar juist met het oog kan volgen. In deze korte tijd wordt de regel vanuit het beeldschermgeheugen in de BASIC-ingangsbuffer opgeslagen (via een "block move") en daarna opnieuw op de oude plaats op het beeldscherm geplaatst. Nu kan men met ↑P de cursor in die regel brengen, waarna met de opdrachten ↑P, ↑H, ↑R, ↑F, ↑I de cursor kan worden gemanipuleerd, terwijl men met Delete kan wissen en nieuwe karakters op de normale manier kan invoegen. Bij keyboards met auto-repeat kan men door het vasthouden van de DEL-toets niet alleen het karakter onder de cursor wissen maar ook alle karakters die daarna volgen. Zolang men de toets ingedrukt houdt, wist de cursor door tot maximaal het einde van de desbetreffende regel. Houdt men de DEL-toets nog steeds vast, dan gaat de cursor terug, dus van rechts naar links, waarbij ook het resterende deel van de regel wordt gewist. De cursor komt dan tot stilstand op de beginpositie van de regel. Dat alles kan natuurlijk eenvoudiger met ↑D, maar zo leert men wel alle mogelijkheden kennen.

Wil men de editing-mode verlaten, dan bedient men weer CR. Met CR, ↑J en ↑K kan men de cursor naar de volgende regel brengen waar men iets in wil wijzigen. In totaal kan men 21 regels op het scherm editen, omdat na een LIST-opdracht steeds twee lege regels en een regel met de prompt OK op het scherm verschijnen. De editor funktioneert ook in de directmode. Als men bijvoorbeeld in plaats van

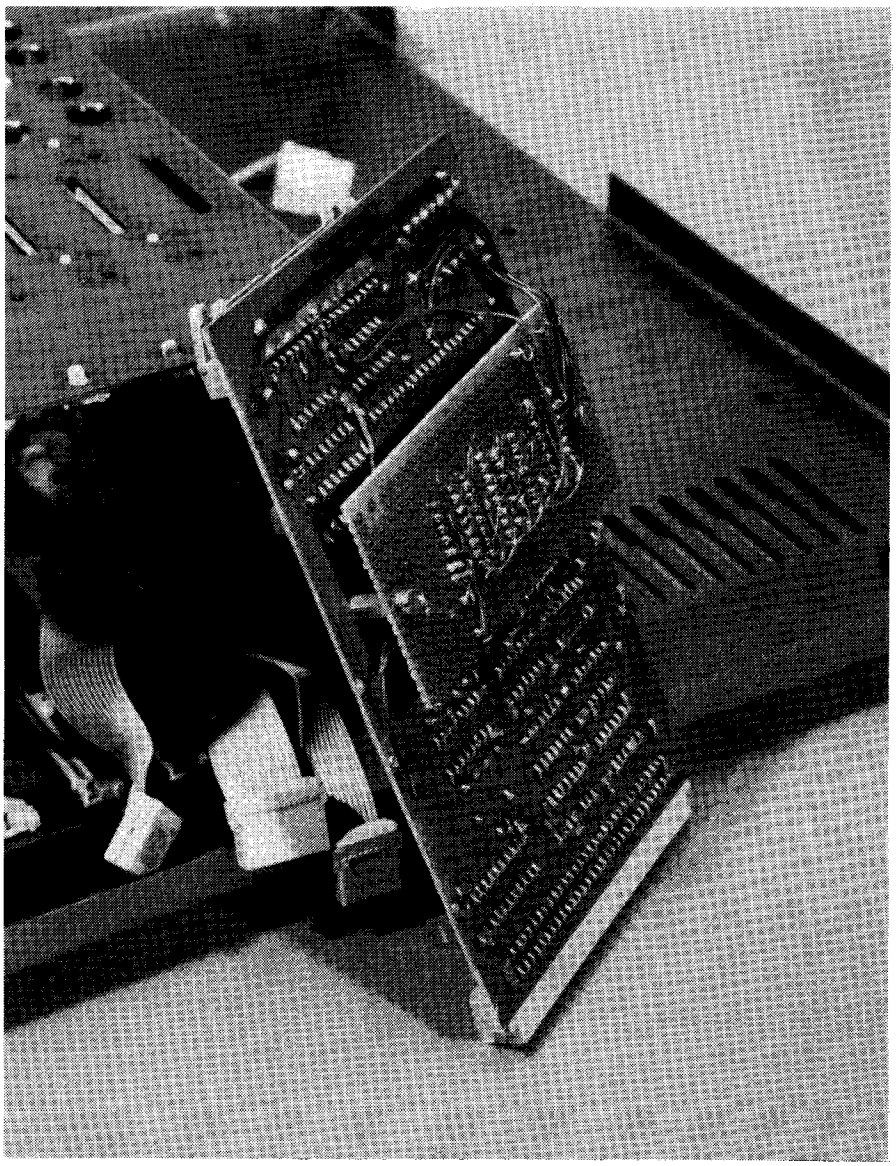
```
DISK!"SE B"
```

zou intikken

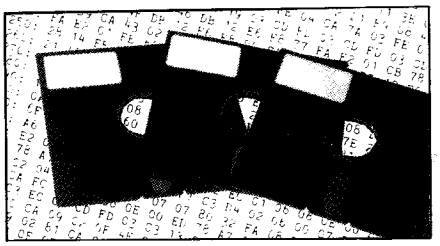
```
DISK!"SI B",
```

dan behoeft men niet de gehele opdracht opnieuw in te tikken, maar brengt de cursor op de I, drukt de DEL-toets in en tikt dan E.

Het beste kan men eens uitgebreid proefdraaien met de editor op het NUMCON-programma. Men mag het programma hierbij gerust "om zeep helpen", want het NUMCON-programma zelf staat toch veilig op diskette. Houd het BASIC Reference Manual bij de hand, want daar hebt u veel steun aan.



Fotoserie, figuur 3. De achterzijde van de CPU-kaart, waarop de motorschakeling is toegevoegd met behulp van twee schroefjes met afstandbusjes. Deze motorschakeling is gepubliceerd in Elektuur april 1984.



De wordprocessor

Eerst even een inleidende opmerking: We hebben tot nu toe de bediening van de Octopus 65 tamelijk uitvoerig behandeld, om vooral degenen die pas beginnen zo goed mogelijk wegwijs te maken. Wie de tot nu toe gegeven uitleg serieus en nauwkeurig heeft gevolgd en zijn systeem met succes "tot leven" heeft gebracht, heeft dan nog wel geen al te grote ervaring, maar een beginneling is hij ook niet meer! Van nu af gaan we de bediening van het systeem wat minder uitvoerig beschrijven om in deze Special voldoende ruimte over te houden voor het presenteren van de zeer interessante software bij de Octopus 65. We gaan alles wat minder uitgebreid vertellen en we vertrouwen er op dat dit geen bezwaar zal zijn, omdat proberen toch nog meer ervaring geeft dan het bestuderen van een hoop papier. Niet dat er informatie zal ontbreken, integendeel, maar we geven het tamelijk gecomprimeerd en laten de praktijk de beste leermeester zijn.

"Wordprocessor" is eigenlijk alleen maar een werknaam voor het tekstverwerkingsprogramma waaraan we nog steeds werken. Gelukkig kan men er al heel goed mee uit de voeten, dus het basiskoncept behoort tot het Octopus 65-pakket. Het is een full-screen-tekst-editor met automatische scroll-up/scroll-down. Naar onze mening ontbreken er nog wat "extra's", zoals tekst-formatter, blokverschuiving, blokduplicering, uitvullen, enz., zoals dat bij kommerciële tekstverwerkingsprogramma's reeds lang gebruikelijk is. Als men echter bedenkt dat wij de basisversie gratis weggeven, terwijl we nog aan de uitgebreide versie werken, dan zal wel niemand bezwaren maken.

Nu aan de slag. Eerst start men het systeem zoals gewoonlijk met .B, daarna kiest men nummer 11. Het editor-systeem meldt zich en de cursor staat te knipperen. Drukt men nu een willekeurige toets in, dan verdwijnt de aankondiging weer en de cursor staat in de startpositie, dus geheel links op de eerste regel. U tikt nu bij wijze van proef een lange tekst in, die drie tot vier beeldschermoppervlakken beslaat. Hierbij hoeft men niet alle regels te vullen met 80 karakters per regel. Maak gerust veel fouten in die tekst zodat u die daarna kunt gaan korrigeren. Maar eerst gaan we de cursor-besturing bekijken. Er zijn daarvoor 6 opdrachten. Het teken "↑" betekent weer dat met de desbetreffende letter tegelijk de control-toets moet worden ingedrukt.

↑W — stuurt de cursor in de regel naar rechts, aan het eind van de regel wordt automatisch naar de volgende regel gesprongen.

↑Q — stuurt de cursor in de regel naar links, aan het begin van de regel wordt automatisch naar de vorige regel teruggesprongen.

↑E — cursor een regel naar beneden. Aangekomen aan de onderrand van het scherm volgt automatische "scroll-up" naar de daaronder liggende regel in het werkgeheugen.

↑R — cursor een regel naar boven. Bij de bovenrand volgt "scroll-down" naar de erboven liggende regel in het werkgeheugen.

Hebt u een keyboard met autorepeat, dan kan de cursor heel gemakkelijk door de tekst worden gestuurd. Probeer dit meteen even. Maar om snel door de tekst te fietsen hebben we nog twee andere opdrachten ter beschikking:

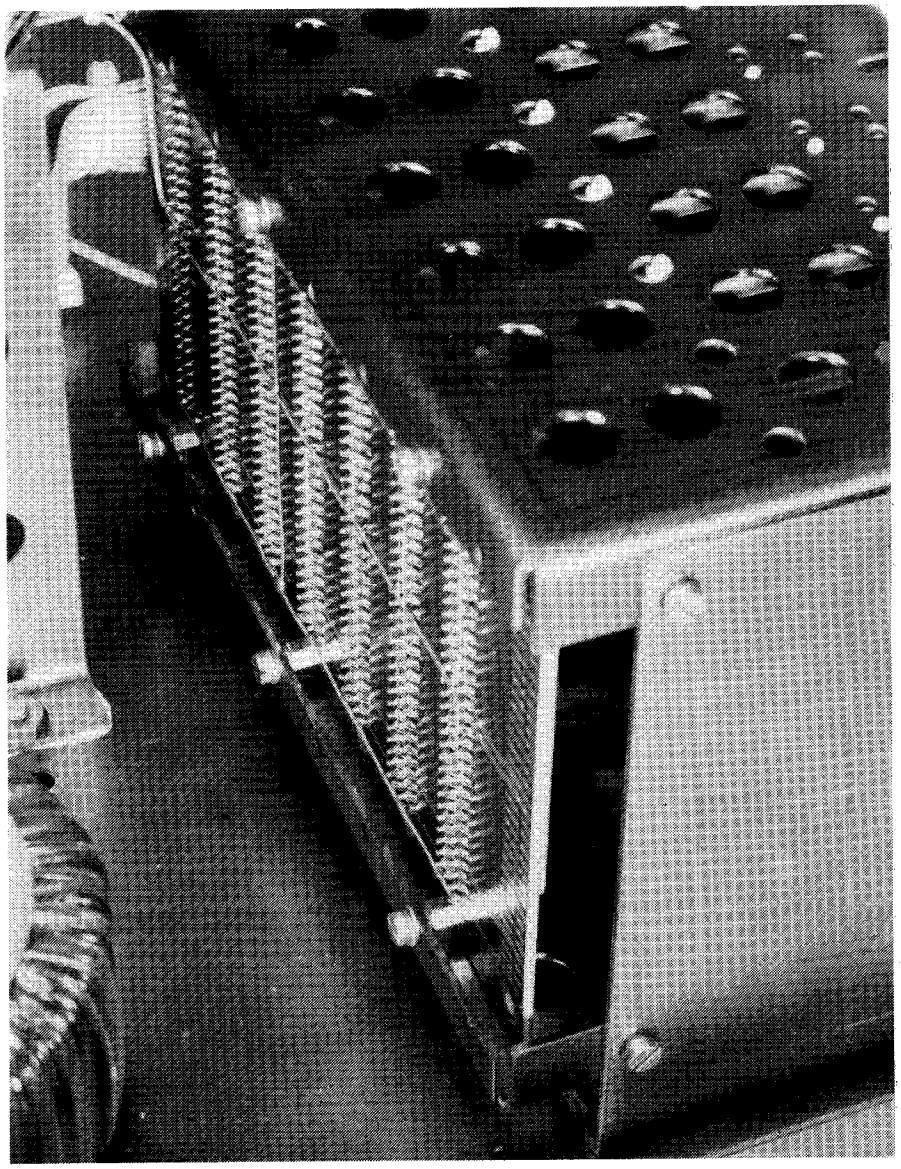
↑T — terugsprong naar de eerste beeldschermpagina (begin van de tekst), de cursor staat nu links beneden op het scherm.

↑S — schuift de listing van het programma een volle pagina op, waarbij de regel waarin de cursor staat nu de bovenste regel op het scherm wordt. De cursor staat nu linksonder.

Het laatste cursor-stuurkommando dat we noemen is de CR-toets. Die wordt ook in de tekst toegepast, eveneens met autorepeat.

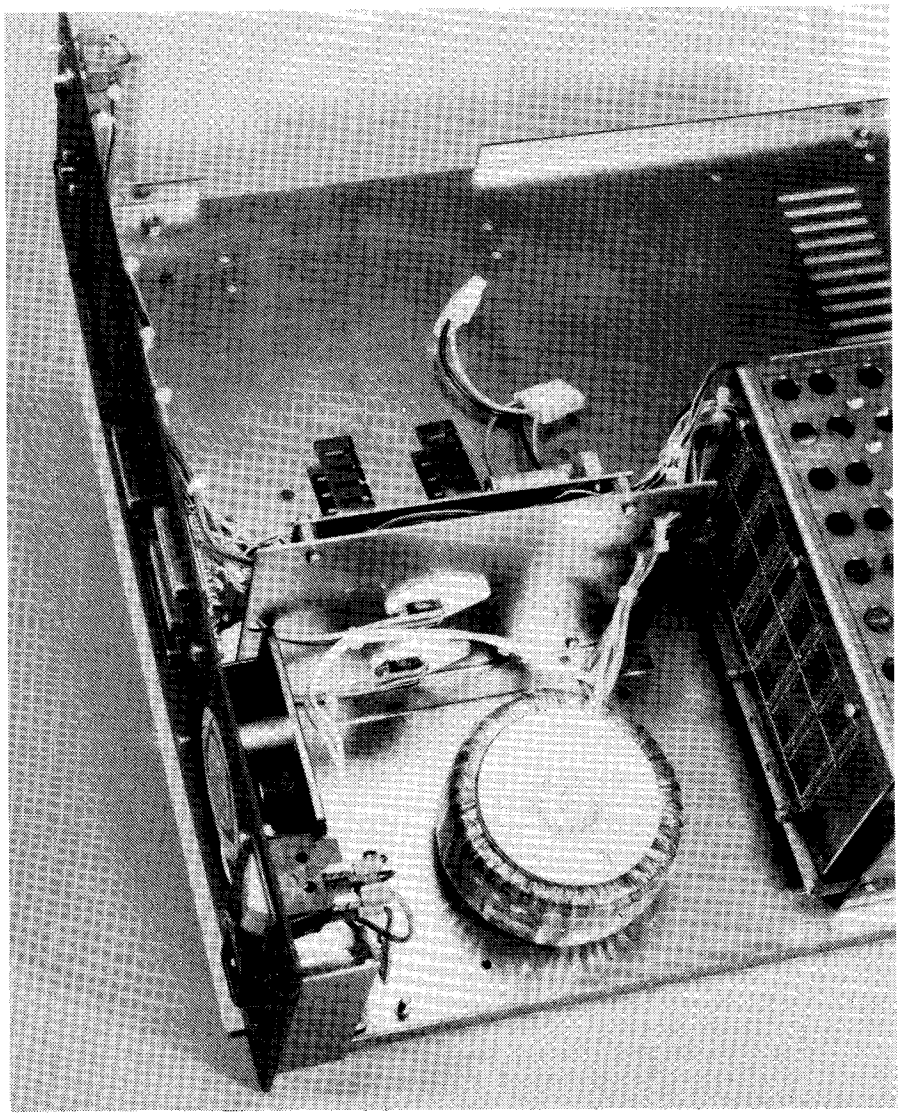
We raden u aan ook met deze drie opdrachten goed te oefenen.

Dan komt de tekst-editing. Foutieve karakters kan men korrigeren door de cursor op die positie te plaatsen en het gewenste karakter in te tikken. Moet er een karakter worden verwijderd zonder door een ander te worden vervangen, dan moet de DEL-toets worden ingedrukt. Het karakter



Fotoserie, figuur 4. Op deze foto ziet men twee interessante details. Ten eerste zijn de koperbanen voor de voedingsspanningen "versterkt" met blanke koperdraad (alleen 5 V en massa) en ten tweede hebben we de busprint een stukje van de metalen delen van de behuizing verwijderd met behulp van enkele afstandsbusjes.

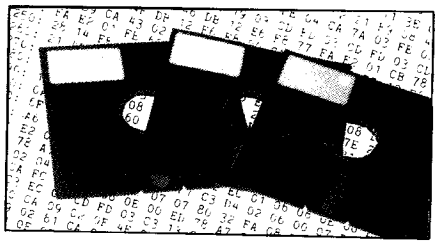
onder de cursor wordt zo gewist en de volgende karakters schuiven automatisch op. Met autorepeat gaat het wissen van daaropvolgende karakters op dezelfde wijze, zo lang men die DEL-toets ingedrukt houdt. Aan het einde van de regel wordt gestopt. Met ↑I schakelt men in de insert-mode (invoegstand). Er kunnen na het plaatsen van de cursor op de gewenste plaats meerdere karakters worden ingevoegd, totdat men een andere stuurtoets indrukt of totdat de regel vol is. Indien nodig kan men met de DEL-toets aan het einde van de regel meer plaatsruimte kreëren. Met ↑L kan vanaf de cursorpositie een regel worden ingevoegd. Drukt men meerdere malen of houdt men de toets vast (bij autorepeat), dan kunnen dus meerdere regels worden ingevoegd. Met ↑D kan men de regel wissen waarin zich de cursor bevindt. Oppassen want de autorepeat werkt hier ook!
De tot nu toe besproken opdrachten hebben alle met de eigenlijke tekstverwerking te maken. Maar er zijn nog een aantal kommando's die andere taken vervullen. Zij worden alle opgeroepen door het indrukken van de ESC-toets (escape-toets). Nadat men deze toets heeft ingedrukt gaat de cursor knipperen, waarna men een van de volgende tekens kan gebruiken.
esc k is een van de genoemde uitzonderingen; deze beïnvloedt de tekst wel heel radikaal. Het is de opdracht om het werkgeheugen te wissen! Maar voordat dit gebeurt, vraagt de wordprocessor tweemaal of inderdaad gewist moet worden, wat men dan steeds met "Y" (yes) moet beantwoorden. Men kan de tekst beter eerst op een diskette opslaan.
esc s dient om een disk drive te kiezen; het systeem vraagt u een van de drives A...D te kiezen. Na die keuze wordt automatisch de inhoud van de in de desbetreffende drive aanwezige diskette op het beeldscherm zichtbaar. Men kan in de tekst terugspringen door simpelweg een willekeurige toets in te drukken. Als een fout wordt gemaakt, bijvoorbeeld het kiezen van een niet aanwezige drive, dan meldt de "CCP" dit met A*, B*, C* of D*. Terugspringen naar de wordprocessor wordt direkt beschreven.
esc p "save" de tekst op de diskette. Het systeem vraagt welke titel of naam (file name) er aan moet worden gegeven. Staat deze al in de directory, dan probeert de Octopus 65 de tekst in de aanwezige file op te slaan. Is de file te kort, dan volgt een foutmelding. Dat gebeurt eveneens als op de diskette of in de directory geen plaats meer is. Deze foutmeldingen komen van de "CCP" (zie het hoofdstuk "DOS-opdrachten"). Men kan van de "CCP" in de tekst en het tekstsyteem terugspringen door
GO ØBBC
in te tikken (de tekst-editor is dan weer startklaar)
esc l laadt teksten van de diskette. Men geeft een "file name", waarna de bijbehorende tekst in het werkgeheugen wordt geladen. Men kan ook meerdere titels opgeven, steeds gescheiden door een "space" (spatiebalk). De bijbehorende teksten komen dan in de ingegeven volgorde in het werkgeheugen. Op het beeldscherm ziet men het begin van de eerste tekst.
esc e wist een naam in de directory. Hiermee wist men eigenlijk de volledige tekst die onder die naam staat opgeslagen, want die wordt dan bij een volgende opslagprocedure door een andere tekst overschreven. Het systeem vraagt u de naam van de betreffende tekst te geven, waarna het wissen van die naam volgt.
esc d geeft de directory van de diskette in de gekozen drive. Men kan terugspringen in het tekstsysteem door een willekeurige toets te drukken. Bij fouten (als er bijvoorbeeld geen diskette in de gekozen drive aanwezig is) kan men weer terugspringen zoals boven beschreven.
esc + en esc — . Als voor de eerste maal tekst op de diskette wordt geladen (als er dus nog geen filename in de directory aanwezig is) wordt automatisch een track méér dan nodig voor de desbetreffende tekst gereserveerd, zodat er nog ruimte is voor latere uitbreidingen. Wil men dat niet, dan tikt men esc — in. Verandert men van gedachte, dan kan men weer esc + geven. Bij een gewijzigde tekst die langer is geworden tikt men esc — in, omdat het systeem geen onderscheid maakt tussen oude en nieuwe teksten. Doet men dit niet, dan geeft de "CCP" een foutmelding. Men springt terug zoals boven aangegeven (GO ØBBC) waarna men alsnog het verzuimde teken esc — geeft. Maar als de tekst meer dan één track langer geworden is, is esc — ontoereikend; dan moet men de tekst onder een andere naam opslaan.
esc h stuurt de tekst in het geheugen naar de Centronics-uitgang (printer-parallel-uitgang). De h betekent hier "hard copy". De tekst kan ook serieel worden uitgegeven via de "CCP"; zie hiertoe de DOS-opdrachten. De overdracht geschiedt dan òf direkt via de CCP òf na "initialisering" van de seriële printer-uitgang en terugspringen via esc h.
esc * schakelt om naar de CCP. Zie hiervoor het volgende hoofdstuk.
Zo, nu kunt u aan de slag met de tekstverwerking. Nog een opmerking tot slot: als u daartoe in de gelegenheid bent moet u de werksnelheid van onze wordprocessor eens vergelijken met een andere tekstverwerker; u zult versteld staan over het verschil. En, zoals gezegd, dit is "pas" de basisversie; de echte uitvoering komt nog!



Fotoserie, figuur 5. Een lucht-opname van het voedingsgedeelte: een flinke netschakelaar, een netsnoer-aansluiting met ingebouwd ontstoringsfilter, een dikke ringkerntrafo en een voedingsprint waarbij de twee vermogenstransistoren op een koelplaat zijn bevestigd. Aan de achterzijde van de kast is bovendien een ventilator ingebouwd om de temperatuur in de kast zo laag mogelijk te houden (nog niet per se nodig, maar wel later als nog meer uitbreidingen worden toegevoegd).

Aanpassing van de wordprocessor aan kleinere geheugens

Als uw Octopus 65 niet over de volle 56 Kbyte RAM beschikt, of als u de wordprocessor op een andere OHIO-DOS-computer wilt gebruiken, moeten drie adressen worden aangepast. Daartoe moet men eigenlijk eerst de monitor leren kennen. En wil men die aanpassing blijvend op de diskette zetten, dan moet men ook de DOS-opdrachten goed kennen. Men moet daartoe track 28 met een CALL-kommando in het geheugen opslaan vanaf $Ø2ØØ.
Daarna wijzigt men met de monitor de inhoud van drie adressen. Voor 32 Kbyte wordt dat:
$Ø698 van AØ naar 7C
$Ø69A van DE naar 7E
$Ø69C van DF naar 7F
Bij 48 Kbyte of 52 Kbyte behoeft men slechts twee adressen te veranderen.
Voor 48 Kbyte:
$Ø69A van DE naar BE
$Ø69C van DF naar BF
en voor 52 Kbyte:
$Ø69A van DE naar CE
$Ø69C van DF naar CF
Hierna kan men met het SAVE-kommando de wijziging op de diskette zetten. Aan het eind van het hoofdstuk DOS-opdrachten worden gelijksoortige wijzigingen uitvoerig beschreven.



DOS-kommando's

Het "disk operating system" (diskettebesturingssysteem, afgekort DOS) is een zelfstandig besturingssysteem met eigen kommando's. In het hoofdstuk "Waar vinden we wat in de DOS" zullen we een en ander exakter uiteenzetten, maar hier volgen eerst alle kommando's van de DOS. Het DOS-gedeelte dat de kommando's in de computer laadt, is de reeds eerder genoemde "console command processor" (CCP), welke eveneens in het bovengenoemde hoofdstuk nader wordt beschreven. De kommando's worden zoals gewoonlijk via het keyboard ingetikt. Men kan drie groepen onderscheiden:

1. Kontrole-opdrachten, waarmee men naar de DOS kan springen en ook weer terug.
2. Algemene DOS-kommando's.
3. De eigenlijke diskette-kommando's.

Kontrole-kommando's

Men kan in de DOS springen door:
EXIT — vanuit BASIC
.* — vanuit assembler
.OS65D — vanuit de "super-monitor" in de EPROM
esc * — vanuit de "wordprocessor"
EX — vanuit de "extended monitor" van OHIO
Eigenlijk horen deze opdrachten niet tot de DOS-kommando's, maar tot die delen van de totale software van waaruit men naar de DOS springt, dus BASIC, assembler, enz. De DOS meldt zich dan met de prompt A*, B*, C* of D*, afhankelijk van de gekozen disk drive (de letter duidt dus steeds de gekozen drive aan).
Men kan DOS weer verlaten (althans bij SYSTEM LOYS) via
BA — naar BASIC (of RE BA)
AS — naar de assembler (of RE AS)
WP — naar de "wordprocessor" (of RE WP)
RE M — naar de monitor in de EPROM
EM — naar de "extended monitor" van OHIO
Op de diskette "SYSTEM LOYS" is nog een uitbreiding toegevoegd. Octopus 65 weet steeds welke transient-processor op een bepaald moment geladen is. Daarom wordt automatisch, indien nodig, een nieuwe transient processor geladen of naar de aktueel geladen transient-processor teruggesprongen. Het RE(TURN) kommando is dus eigenlijk onnodig, maar is gehandhaafd ten behoeve van de kompatibiliteit met de OHIO-DOS.

Algemene DOS-kommando's

Het woord algemeen kan misschien tot misverstanden aanleiding geven, maar we bedoelen hier die DOS-opdrachten die niet met "in- of uitspringen" te maken hebben, en ook niet direkt betrekking hebben op de disk drives of diskettes. Het gaat om de volgende opdrachten:
IO, GO, ME en "underline".
De IO-opdracht dient om de invoer/uitvoer-flags te setten en resetten. In het hoofdstuk "Waar vinden we wat in de DOS" wordt dit nader uiteengezet, maar hier alvast een paar voorbeelden:
IO, Ø9 — selekteer de aan de Centronics-uitgang aangesloten printer (gelijktijdige verzending naar de printer en het beeldscherm).
IO Ø4 — selekteer het seriële keyboard. De 6551-ACIA moet echter eerst geprogrammeerd worden (kortsluitstekers op de CPU-kaart).
IO Ø4,Ø5 — zenden en ontvangen via de ACIA. Men kan nu een modem aansluiten. Gelijktijdige weergave op het beeldscherm.
De andere opdrachten noemen we alleen, want men heeft ze zelden nodig.
ME NNNN, MMMM — lees dat uit het geheugen, te beginnen bij adres $NNNN. Sla data in het geheugen op vanaf adres $MMMM. Hier betekent NNNN de input-pointer en MMMM de output-pointer.
GO NNNN — Start de uitvoering van een programma op adres $NNNN. Eindigt het programma met een RTS-opdracht, dan wordt teruggesprongen naar de DOS. Let wel, het gaat hier om machinekode, dus op het aangegeven adres moet een zinvolle opcode staan.
"Underline" — Met deze toets kan het laatst ingegeven karakter weer gewist worden. Hierbij gaat de cursor een plaats of (bij meermaals indrukken) meerdere plaatsen naar links. Met deze funktie kan men foutief ingetoetste kommando's weer korrigeren (wissen en goed invoeren).

"Afspraken"

Voor we bij de eigenlijke diskettekommando's komen, moeten nog enkele aanwijzingen worden gegeven en enkele afspraken worden gemaakt.
— DOS-opdrachten mogen niet langer zijn dan 17 karakters;
— parameters moeten van het kommando gescheiden worden door een spatie.
Maatgevend zijn alleen de twee eerste letters van een woord, soms slechts één. Men kan dus in plaats van RE M ook ingeven RETURN MONITOR of in plaats van BA BASIC en in plaats van AS ASSEMBLER. Maar bij het omschakelen naar "extended monitor" of de "wordprocessor" moet men oppassen: hier moét EM resp. WP ingetikt worden, omdat deze afkortingen niet identiek zijn aan de eerste twee letters van de bijbehorende woorden.
Programmanamen (file names) moeten steeds met een alfa-karakter beginnen en mogen maximaal 6 karakters lang zijn.
Een goed voorbeeld is:
TEXT1
X#/U1+
loonaf
text1
Foutief is:
1.Text
#blurp
loonaftrek
Een bijzonderheid moet nog worden vermeld bij de wordprocessor. Daar worden bij het laden van programmanamen kleine letters automatisch in hoofdletters omgezet. Wil men in zijn programmanamen beslist kleine letters toepassen, dan is ook daarvoor een oplossing. Men moet dan in DOS springen en zijn programma's met de hierna beschreven opdrachten invoeren en opslaan.
Bij de nu volgende diskette-kommando's moeten alle plaatsbepalende gegevens worden ingevoerd, bijv. TT voor het tracknummer. Waar TT staat moet men als tracknummer niet 3 maar Ø3 zetten. Dat geldt ook voor adressen: voor NNNN moet men Ø2ØØ ingeven en niet 2ØØ. We zullen dezelfde letters aanhouden als in het OHIO-DOS-manual, zodat men niet in de war raakt:

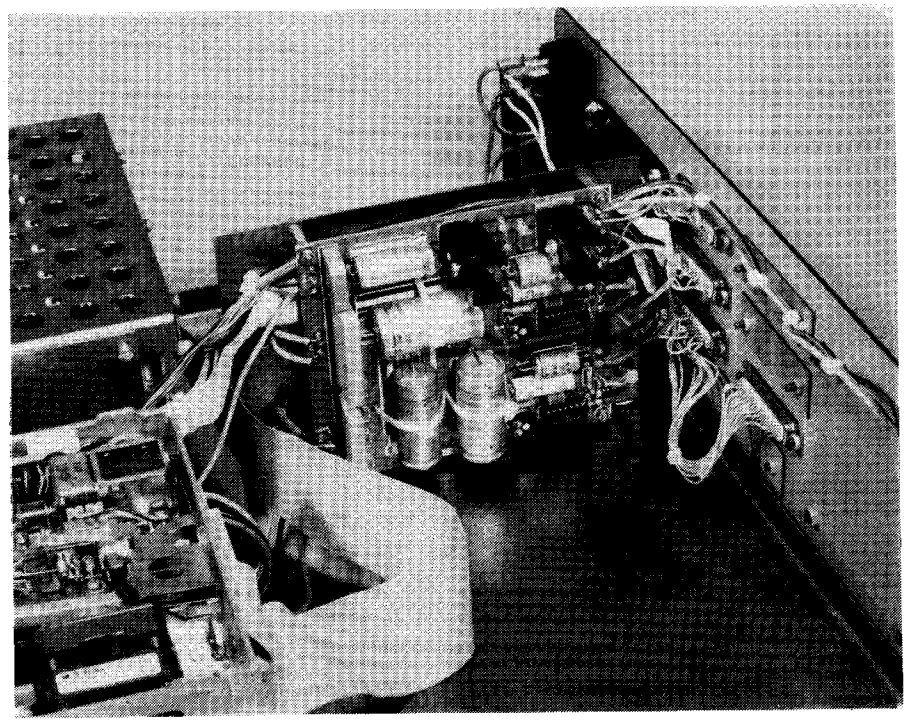
TT — Track-nummer. Het track-nummer wordt altijd decimaal ingegeven.
S — Sektor-nummer. Wordt eveneens decimaal ingegeven.
NNNN of MMMM — Adressen, hexadecimaal.
X — Drive-nummer. Hier moet A, B, C of D worden ingetypt. In de gebruiksaanwijzingen van drives kan men andere aanduidingen vinden, bijv. 0, 1, 2, 3,... Voor de Octopus 65 betekent dat: 0=A, 1=B, 2=C, enz. Soms ziet men ook 1, 2, 3, 4,... Dat betekent dan voor de Octopus 65: 1=A, 2=B, 3=C, enz. Drive A is bij de OHIO-DOS steeds de drive waarop het systeem wordt gestart (geboot).
P — Aantal "pages". Een page is 256 bytes (¼ Kbyte), dus precies het aantal bytes dat door twee hexadecimale getallen (00.. ..FF) volledig geadresseerd kan worden.
Nog even een opmerking. Foutief gegeven opdrachten veroorzaken de foutmelding:
ERR#7
We komen nog terug op de foutmeldingen, want tenslotte maakt iedereen wel eens een typefoutje.
Nu komen de eigenlijke diskette-opdrachten aan de beurt.

Disk-kommando's

Hier volgen alle disketteopdrachten; eerst geven we de (Engelse) aanduiding en daarna tussen haakjes de mogelijke afkorting:
LOAD (LO), PUT (PU), CALL (CA), SAVE (SA), SELECT (SE), DIRECTORY (DI), HOME (HO), ERASE (ER), INIT (IN), XQT (XQ) en EXAMINE (EX).
Deze opdrachten kunnen op meerdere manieren worden gebruikt; we zullen dat straks verduidelijken. Daarbij gebruiken we natuurlijk meteen de afkortingen, zodat u die snel onder de knie krijgt (dat bespaart later een hoop moeite). Ondertussen geven we nog enkele praktische aanwijzingen.
IN — initialiseert de gehele diskette (maakt ze klaar voor opname van data). Maar wees voorzichtig, want daarbij wordt alles gewist dat er reeds opstaat. Een nieuwe, nog ongebruikte diskette bevat nog geen magnetische informatie en moet daarom eerst worden "geformatteerd". Het resultaat van het IN-kommando is, dat ringen met magnetische sporen op de diskette worden geschreven, die daarna data kunnen bevatten. De nummering van de tracks op de diskette loopt van buiten naar binnen en van 00 tot 39 (of van 00 tot 79 bij 80-track-drives). Elke track begint met een "track-header", waarin de gegevens van de betreffende track kunnen worden opgeslagen; deze header wordt door de IN-routine op elke track gezet. Hierop komen we later in dit hoofdstuk terug.
IN TT — initialiseert track TT (en alleen deze).
SE X — selekteert drive X. De computer "onthoudt" steeds de laatst gekozen disk drive. Wordt geen andere SE-opdracht gegeven, dan wordt bij volgende diskette-bewerkingen steeds de laatst geselekteerde drive ingeschakeld.
DI — roept de directory op van de geselekteerde diskette en toont deze op het beeldscherm. Is d.m.v. een I/O-opdracht ook de printer geaktiveerd, dan wordt de directory tevens naar de printer gestuurd.
DI TT — geeft op het beeldscherm (en eventueel ook op de printer) aan met hoeveel pages welke sektoren van track TT gevuld zijn. Verderop verklaren we nog precies wat een sektor is.
HO — de kop van de gekozen drive wordt op zijn startpositie op track 00 gezet. Wie een drive heeft waarvan kan worden omgeschakeld tussen 40 en 80 tracks, moet steeds na het omschakelen een HO-kommando geven.
SA TT,S=NNNN/P — zend P pages vanaf adres NNNN van het werkgeheugen naar de diskette en sla die op in sektor S van track TT.
Dit kommando zoekt dus track TT op en daarin sektor S. Dan worden P pages op de diskette opgeslagen en sektor S genoemd. Is in deze sektor reeds data opgeslagen, dan wordt tevens gewist en de nieuwe data opgeslagen zonder dat met een foutmelding hiervoor wordt gewaarschuwd. Als de beschikbare sektor korter is dan P pages, wordt dit eerst gemeld. De voor deze sektor beschikbare plaats wordt echter overschreven, de volgende sektor is beveiligd tegen overschrijven. Waarschuwing: met het SAVE-kommando moet men dus zorgvuldig omspringen. Beginners raden we daarom aan bij voorkeur het PUT-kommando toe te passen (zie verder). Overigens is dit punt in het OHIO-DOS-manual niet geheel korrekt beschreven. Let daar goed op!
CA NNNN=TT,S — laad de inhoud van track TT van de diskette, sektor S, in het werkgeheugen vanaf adres NNNN. Daarbij worden track-header, sektor-header en trailer niet in het werkgeheugen opgeslagen. Wat precies track-header, sektor-header en trailer zijn leggen we u direkt nog uit.
EX NNNN=TT — laad de inhoud van track TT met inbegrip van track-header, sektor-header en trailer in het werkgeheugen vanaf adres NNNN.
EXAM (examinate, test) is een speciale opdracht die men onder normale omstandigheden niet nodig heeft. Hiermee is het mogelijk fouten in de DOS op te sporen, waarbij de totale inhoud van een track met inbegrip van header en trailer, die de DOS anders slechts voor interne verwerking nodig heeft, in het werkgeheugen wordt geplaatst. Daarbij wordt de gehele inhoud zonder uitzondering overgenomen, ongeacht welke nonsens zich in die track bevindt en waarvoor de DOS in normale gevallen een foutmelding zou geven of het programma onderbreken. Het is dus een zeer nuttige opdracht, want men kan er niet alleen fouten mee opsporen maar in geval van nood niet meer toegankelijke diskette-inhouden "redden" door die via het werkgeheugen op de printer te "listen" als men geen back-up van die inhoud heeft. Daarbij kunnen vaak een of meer extra karakters voor de track-header of de sektor-header opduiken. De oorzaak ligt in een onjuiste synchronisatie. De headers (en natuurlijk ook de data) zijn echter korrekt. Meestal zijn deze "extra" tekens FF.

Enkele belangrijke toelichtingen

Het is tijd nu enkele belangrijke toelichtingen te geven, ofschoon een paar waardevolle kommando's nog niet zijn behandeld.
De track-header bestaat evenals alle andere informatie op de diskette uit magnetische "nullen" en "enen". Zo'n header bevat (behalve op track 00) de volgende bytes:
43 (hexadecimaal)
57 (hexadecimaal)
het track-nummer (in BCD-kode)
58 (hexadecimaal)
Het systeemspoor track 00 wordt geladen



Fotoserie, figuur 6. Een blik op de voedingsprint. Intussen zijn ook de loopwerken ingebouwd. Rechts van de voedingsprint herkent men de verschillende konnektoren die zorgen voor de konnekties met de buitenwereld (keyboard, printer, etc.).

door de "bootstraploader" (deze is bij het monitorprogramma SAMMON in de EPROM op de CPU-kaart opgeslagen). De track-header hiervan bevat de volgende (hexadecimale) bytes:
— laad-adres, meest signifikante byte
— laad-adres, minst signifikante byte
— aantal pages op track 00
Bij de Octopus 65-systeemdiskette (de vroegere Tutorial disk 5) is het oer-laad-adres 2200 (zie ook "Waar vinden we wat in de DOS").
Het aantal sektoren op een track is bij de OHIO-DOS (in tegenstelling tot bijv. CP/M) niet vastgelegd, zodat men zeer ekonomisch met de geheugenkapaciteit om kan springen. Elke sektor begint met een sektor-header die uit 3 bytes bestaat en eindigt met een trailer die 2 bytes in beslag neemt.
Sektor-header:
76 (hexadecimaal)
sektor-nummer (binair)
sektor-lengte in pages (binair)
Trailer:
47 (hexadecimaal)
53 (hexadecimaal)
Tussen header en trailer wordt een geheel aantal pages opgeslagen en dat vormt dan een sektor. Bij 5 ¼"-drives zijn dat maximaal 8 pages (= 2 Kbyte) indien slechts één sektor op de track wordt opgeslagen. Bij meer dan een sektor per track zijn het maximaal 7 pages omdat anders de extra bytes van de bijkomende headers en trailers de track te lang zouden maken en het eind van de track het begin zou overschrijven.

File-opdrachten

De tot nu toe behandelde DOS-kommando's hadden steeds betrekking op enkele tracks of op de gehele drive. Bij de meeste toepassingen wil men echter een bepaald programma of een bepaalde tekst op de diskette opslaan of wissen, dus een groep data, een file. Daarvoor gebruikt men de DOS-kommando's PUT, LOAD, XQT en ERASE. Elke file krijgt een speciale file-header die uit vijf bytes bestaat. Deze header bevat het begin- en eindadres waartussen de file in het werkgeheugen moet worden gezet, evenals het aantal benodigde tracks (zie verder "Waar vinden we wat in de DOS"). Nu kan men wel opmerken dat de data die met SAVE en CALL worden verwerkt, dus in één track, ook groepen data zijn en dus een file vormen. Dat is juist. Er moet toch een duidelijk verschil zijn. In het DOS-manual van Ohio-Scientific worden daarom de door de opdrachten PUT en LOAD verwerkte files als "source files" aangeduid (de ERASE-opdracht hebben wij aan de OHIO-DOS toegevoegd). Dat is in het algemeen juist, maar eigenlijk niet maatgevend voor het verschil tussen (specifiek) CALL en SAVE enerzijds en LOAD en PUT anderzijds.
Iets verderop in deze Special (bij de beschrijving van het werken met de assembler) wordt het verschil tussen "object code" en "source code" duidelijk gemaakt, we behandelen hier alleen de diskette-kommando's. Bij het gebruik van CALL en SAVE geeft de gebruiker aan, naar of vanaf welk startadres deze kommando's moeten werken. De lengte van de te transporteren file wordt door de lengte van de sektor bepaald of wordt door de gebruiker door het aantal pages vastgelegd. Het maakt geen verschil of het gaat om object-code (machinekode) of om source-code. Het maakt ook geen verschil of het kommando zinvol is of (om een sterk voorbeeld te geven) dat met een CALL-kommando wordt geprobeerd de monitor-EPROM om te programmeren. Men kan het dus ook zo zeggen: CALL en SAVE zijn "zuivere" DOS-opdrachten.
Bij de toepassing van de kommando's LOAD en PUT is, in tegenstelling tot het bovenstaande, steeds een "transient processor" aktief, of dat nu de BASIC-interpreter is, de assembler, de wordprocessor of een andere. Die bepaalt uiteindelijk het begin- en eindadres van de file in het werkgeheugen. Het begin-adres is meestal het laagste adres van het werkgeheugen. Een uitzondering hierop zijn buffers, die van het "onderste" deel van het werkgeheugen "afgetakt" worden. Maar dat verandert niets aan het principe; zodra de gebruiker zo'n buffer met een kommando definieert, is er een nieuw en eenduidig start-adres. Het eind-adres wordt bepaald door het einde van het BASIC- of assembler-programma of het einde van de tekst. Die adressen worden door de transient processor berekend en aan de DOS doorgegeven.
Het ligt dus voor de hand zo'n file, die uiteraard een source-code-file is, een bepaalde naam te geven, de file-naam. Daarmee gaan we dan werken in plaats van met track-nummers. In het OHIO-DOS-manual is de volgorde juist andersom; en dat is eigenlijk een der weinige punten van kritiek die we op dit overigens zeer goede en jammer genoeg alleen in de Engelse taal verkrijgbare manual hebben. Maar we keren nu weer terug naar de voor Octopus 65 geldige DOS-opdrachten:
PU file-naam — laad de op dat moment in het werkgeheugen aanwezige gegevens onder de naam "file-naam" op de diskette.
LO file-naam — laad de gegevens met de naam "file-naam" in het werkgeheugen.
XQ file-naam — laad een machinetaal-programma onder de naam "file-naam" in het werkgeheugen en start de uitvoering ervan. Het machinetaal-programma wordt bij de versie V3.2 op adres $3279 en bij V3.3 op $3A79 geladen. De sprong leidt bij V3.2 naar adres $327E en bij V3.3 naar $3A7E.
ER file-naam — wis de file-naam in de directory. De voor deze file gereserveerde tracks kunnen opnieuw worden gebruikt. ER(ASE) is echter alleen aanwezig bij de Micro-Ware-assembler en bij de wordprocessor, omdat deze met de DOS-versie 3.2 werken. Bij de DOS-versie 3.3 is "Erase" door plaatsgebrek nog niet aanwezig, maar we werken nog aan een oplossing hiervoor. Bij BASIC moet men thans nog met de utility "delete a file from diskette" van het hoofdmenu werken.
Het nieuwe, door ons ontwikkelde PUT-kommando onderzoekt eerst of zich reeds een file met de naam "file-naam" in de directory bevindt. Als dit het geval is, wordt daarna getest of het daarvoor gereserveerde aantal tracks voldoende ruimte biedt. Als dit inderdaad het geval is, dan wordt de file op deze tracks opgeslagen waarbij de oude inhoud wordt gewist. Pas dus op bij het geven van namen! Als men niet zeker is kan men met de DI-opdracht snel en zonder de inhoud in het werkgeheugen te beïnvloeden kontroleren of die naam reeds is toegepast. Zo ja, dan maar een andere naam bedenken. Indien de naam reeds bestaat maar het aantal beschikbare tracks onvoldoende is, komt de foutmelding ERR D. Ook hier geldt: geef een nieuwe naam, de gegevens in het werkgeheugen worden er niet door beïnvloed.
Tot nu toe blijkt nog geen verschil met het oude PUT-kommando. Bij het oude kommando moest de file-naam echter reeds in de inhoud aanwezig zijn, vóór de file met de opdracht PU "file-naam" kon worden opgeslagen. Daarvoor diende het utility-programma "Create a new file" (keuze 2 van het hoofdmenu). Een moeizame en vaak problematische methode, en met het nieuwe PUT-kommando overbodig. Indien de file-naam nog niet in de inhoud aanwezig is, zorgt de nieuwe PUT-opdracht voor de benodigde handelingen. Hij stelt vast hoeveel tracks nodig zijn, kijkt in de inhoud of nog voldoende opeenvolgende tracks vrij zijn, neemt de file-naam en het aantal daarbij behorende tracks op in de directory, slaat de file op in deze tracks en meldt dat alles klaar is met de bekende "prompt". Maar hiervoor moet aan twee voorwaarden zijn voldaan, m.a.w. twee foutmeldingen zijn mogelijk:
ERR E — er zijn onvoldoende opeenvolgende tracks beschikbaar op de diskette.
of
ERR F — er is onvoldoende ruimte in de directory.
In beide gevallen brengt een lege, geïnitialiseerde reserve-disk uitkomst. In veel gevallen kan de ER-opdracht helpen indien overbodig geworden files op de diskette staan. Lees hiertoe zo dadelijk het stukje "De directory". We gaan nu over naar een andere bijzonderheid van de OHIO-DOS: de kommando's:
— PU TT
— LO TT
— XQ TT
Hun werking is in feite hetzelfde als die van de reeds behandelde kommando's met dit verschil, dat hier het laagste tracknummer van de file als file-naam funktioneert. Het PUT-kommando moet echter met grote voorzichtigheid worden gebruikt. De file wordt gewoon vanaf track TT opgeslagen, ongeacht het feit dat daarbij een andere file kan worden overschreven. Daarom alleen geschikt voor meer ervaren programmeurs!

De directory

Bij de OHIO-DOS zijn voor de directory de beide eerste sektoren van track 12 gereserveerd, in principe zelfs de hele track. Maar we vonden dat zonde van de ongebruikte ruimte en hebben daarom op de systeemdiskette "SYSTEM LOYS" in de volgende sektoren enkele van onze uitbreidingen geplaatst. Elke file-naam beslaat in de directory 8 bytes: de eerste 6 bytes bevatten de file-naam in ASCII-kode, het zevende byte bevat het eerste tracknummer van de file en het achtste byte het laatste track-nummer (track-

nummers decimaal).
Twee pages gedeeld door acht: dat zijn precies 64 file-namen. In verschillende Octopus 65-meldingen worden ze als "entries" aangeduid. Nu even opgelet! De nieuwe PUT-opdracht veroorlooft slechts 32 entries in de directory, wat voor 40-track-drives meer dan voldoende is omdat track 00 en track 12 reeds door het systeem zijn gereserveerd. Verderop beschrijven we nog de aanpassing van de OHIO-DOS aan 80-track-drives. Daar zou men aan 32 namen te kort kunnen hebben. Daarom werken we aan een uitbreiding van de directory waarbij tenminste de oorspronkelijke 64 entries mogelijk zijn.
Opdat er geen misverstand ontstaat: natuurlijk kan men over de volle kapaciteit van de directory beschikken als men de utility "create a new file" gebruikt. Hierbij de naam eerst in de directory plaatsen voor men de file op de disk schrijft. Dus zoals men het bij het oude PUT-kommando gewend was te doen.

Foutmeldingen

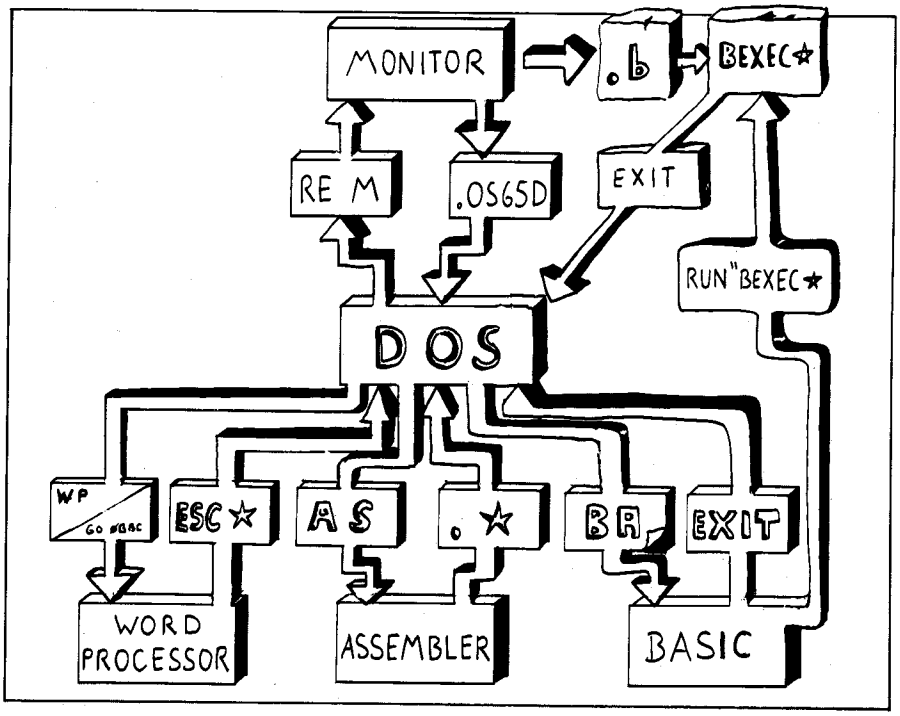
Ieder mens maakt fouten en omdat computers door mensen worden gebouwd maken die ook fouten. Om de gebruiker hierbij te ondersteunen bestaan de volgende foutmeldingen bij de Octopus:

1 — sektor kan niet gelezen worden (pariteitsfout)
2 — sektor kan niet beschreven worden (terugleesfout)
3 — track 00 is tegen beschrijven beveiligd
4 — diskette beveiligd tegen beschrijven
5 — zoekfout (track-header past niet bij de track)
6 — disk drive niet klaar
7 — syntax-fout in kommando-regel
8 — foutief track-nummer
9 — track-adres kan tijdens een omwenteling van de diskette niet worden gevonden
A — sektor vóór de beginsektor kan niet worden gevonden
B — foutieve waarde sektor-lengte
C — de file-naam kan in de directory niet gevonden worden
D — schrijf- of leespoging na het einde van de opgegeven file
E — op deze diskette is een voldoende aantal aaneengesloten tracks niet meer ter beschikking
F — geen plaats meer in de directory

De foutmeldingen 1 ... D zijn onveranderd ten opzichte van de OHIO-DOS. De foutmeldingen E en F hebben betrekking op de door ons aangebrachte wijzigingen in de OHIO-DOS.

Aanpassing van de DOS naar 80 tracks

Hier komen we bij een tot nog toe niet genoemde utility: de Trk0-R/W-utility (de track 0-read/write-utility). Eerst starten we het systeem (booten), dan met CR naar BASIC en van daar uit met EXIT naar de DOS. De computer meldt zich nu met:
A*
Nu geven we in:
CA 0200=36,1
Daardoor wordt track 36 in het TPA-bereik gelezen. Terugmelding weer A*, waarna we intikken:
GO 0200
Het programma in de TPA wordt hierdoor gestart en meldt zich in de vorm van een klein menu. We kiezen nummer 2 door 2 CR in te tikken. Nu verschijnt het Trk0-R/W-menu. Dat kan men eerst eens rustig bestuderen, maar men kan ook direkt intikken:
R6200 CR
Na de terugmelding A* geeft men RE M (sprong naar de monitor). Met het monitorprogramma (zie twee hoofdstukken verder) kan men de inhoud wijzigen van de adressen:
66CA van 40 in 80
6769 van 39 in 79
6779 van 39 in 79
Dan geeft men de opdracht .OS65D (sprong van monitor naar DOS). Nu starten we het programma met
GO 0200
daarna met optie 2 de Trk0-R/W-utility kiezen en daarna intikken
W6200/2200,8
Daarmee is track 00 gewijzigd. Men kiest nu bij de terugmelding van de Trk0-R/W-utility de keuze E (EXIT) en komt zo weer in de DOS. Hierna moet track 01 worden gewijzigd:
CA 6A00=01,1
Daarna springen we weer naar de monitor (RE M) en wijzigen de adressen
6DA7 van 39 in 79
6DB7 van 39 in 79
Daarna met .OS65D terug in de DOS en dan intikken:
SA 01,1=6A00/8
Daarmee is de DOS aangepast, met uitzondering van de nieuwe opdrachten PU, DI en ER, die nu aan de beurt komen:
CA 6400=06,4
Dan weer RE M en de adressen wijzigen:
6496 van 27 in 4F
64CE van 28 in 50
64FD van 28 in 50
Daarna weer met .OS65D terug in de DOS en intikken:
SA 06,4=6400/4
Tenslotte moet de DOS-versie V3.2 op de nieuwe systeemdiskette eveneens gewijzigd worden, omdat de wordprocessor en de Micro-Ware-assembler die versie gebruiken. Zij bevindt zich op de tracks 10 en 11. We tikken in:
CA 6200=10,1
daarna, na de melding A*, direkt ingeven
CA 6A00=11,1
Daardoor wordt de inhoud van de beide tracks in het werkgeheugen aaneengekoppeld. Nu gaan we weer met RE M in de monitor en dan wijzigen we
66CA van 40 in 80
6769 van 39 in 79
6779 van 39 in 79
6DA7 van 39 in 79
6DB7 van 39 in 79
Weer met .OS65D in de DOS terug en intikken:
SA 10,1=6200/8
terugmelding A* afwachten en daarna intikken:
SA 11,1 6A00/8
De DOS is nu geheel aangepast, maar het BASIC-programma BEXEC* heeft in regel 30 nog een 39 staan die in een 79 moet worden veranderd. Daarom gaan we het systeem opnieuw booten, zodat de BASIC-interpreter weer in de TPA geladen wordt, dan met LIST 30 de regel in BEXEC* oproepen en met de BASIC-editor (beschreven in het hoofdstuk BASIC-kommando's) aan het einde van de regel de "(39)" in "(79)" wijzigen, door eenvoudig de 3 te wissen en een 7 in te tikken. Met
DISK!"PU BEXEC*
wordt nu de veranderde versie van BEXEC* op de diskette geschreven.
Er blijft nog een laatste probleem over; deze nieuwe versie moet op een 80-track-drive worden gezet. We hebben hiervoor met een TEAC FD55F gewerkt (waarbij men tussen 40 en 80 tracks kan omschakelen), samen met de utility "diskette

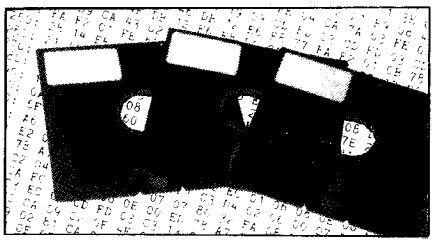


Figuur 1. Dit "steenblok"-schema werd door een onzer kollega's getekend om hem als praktische hulp te dienen. Het stamt uit de periode vóór de samenstelling van deze Computer-Special en we vinden een vereeuwiging voor het nageslacht alleszins de moeite waard.

copier". Daarbij hebben we gewoon na elke keer insteken van de "blank disket-te" op 80 tracks geschakeld en na elke keer insteken van de "master" op 40 tracks. Maar daartoe moet het RAM-bereik volledig ter beschikking staan. Er zijn ook nog andere manieren te bedenken; men koopt een 80-track-drive en leent een 40-track-drive die voor het kopiëren gebruikt wordt als drive A.
Een andere aanpassing is echter nog niet ingevoerd: de aanpassing van de utility "diskette copier". Het programma funktioneert verder als gewoonlijk, alleen is het maximale aantal te kopiëren tracks nog niet verhoogd. Dat zullen we in de toekomst ook nog veranderen.

Het werken met dubbelzijdige drives

Allereerst is voor het werken met dubbelzijdige drives een kleine, reeds voorbereide aanpassing van de hardware nodig op de FCU-kaart. Daartoe legt men aan de koperzijde van de kaart een brug tussen pen 9 van IC6 en pen 1 van IC15. Daardoor wordt de "bovenzijde" van de eerste drive drive A en de "onderkant" drive B. Heeft men nog een tweede drive, dan wordt de "bovenkant" C en de "onderkant" D.
Wat de software betreft doen we het volgende: de utility "Initialize blank data diskette" werkt uitsluitend op drive A: daarmee kan dus de "achterzijde" niet worden geïnitialiseerd. Men moet nu de achterzijde met SE B van de DOS kiezen (antwoord: prompt B*) en daarna met het IN-kommando initialiseren. Alleen ontbreekt nu nog de voorbereide lege directory, die normaal door de juistgenoemde utility op de diskette wordt aangebracht. Daartoe kiest men weer drive A en tikt dan in:
CA 4000=26,2
en na de prompt A*
CA 4100=26,3
Daardoor wordt dezelfde "lege" directory, die ook door de utility wordt gebruikt, in het geheugen geladen. Nu weer SE B intikken en na de prompt B*
SA 12,1=4000,1
en na prompt B* nog:
SA 12,2=4100/1
Nu heeft ook de achterzijde van de diskette haar directory en kan ze gewoon met het DISK!"PUT-kommando worden ingeschreven. De DISKOPEN- en DISKCLOSE-kommando's werken nu nog niet, omdat de GET- en PUT-overlays nog niet op track 12, sektor 3 en sektor 4 staan. Om een directory met alle mogelijkheden van de DOS te verkrijgen, moeten de volgende sektoren van de diskette "SYSTEM LOYS" naar de geïnitialiseerde achterkant van de (nieuwe) diskette worden overgebracht:
Diskette "SYSTEM LOYS" — "geïnitialiseerde achterkant van de diskette"
CA 4000=26,2 ..... SA 12,1=4000/1
CA 4100=26,3 ..... SA 12,2=4100/1
CA 4200=26,4 ..... SA 12,3=4200/1
CA 4300=26,5 ..... SA 12,4=4300/1
Zelfs voor een beginneling in deze materie zal het niet moeilijk zijn om voor deze sektor-transfer een kleine utility in BASIC te schrijven.



Sekwentiële en random files

Een van de sterke punten van de OHIO-DOS is de snelle verwerkingstijd van sekwentiële en random-files. Voor degenen die tot dusver nog niet met dit soort files hebben gewerkt, geven we eerst een korte inleiding.
Vaak staat de computergebruiker voor het probleem dat veel data in de computer moeten worden geladen en dat de computer, nadat alle bewerkingen zijn uitgevoerd, weer veel data moet versturen. Stel u eens voor dat een firma de adressen, klantnummers en telefoonnummers van circa 1000 klanten in de computer wil opslaan. Bovendien moeten ook de omzetten en rekeningnummers in de computer worden gezet. Dit alledaagse probleem kan op normale wijze nauwelijks worden opgelost omdat de ruimte van het werkgeheugen daartoe te klein is. Het ligt voor de hand die gegevens op de diskette op te slaan en ze door een daartoe gereserveerde RAM-buffer door de Octopus 65 te "sluizen", dus in delen te laten verwerken. Hoe worden nu die data door de RAM-buffer gesluist?
Denk eens aan de INPUT- en PRINT-statements in BASIC. Als de Octopus 65 in een BASIC-regel het statement INPUT A$ ontmoet, dan wacht hij op de ingave van een string; als Octopus in een BASIC-regel INPUT A ontmoet, dan wacht hij op de ingave van een decimaal getal. De string en het decimale getal bestaan elk uit een serie ASCII-tekens. Het verschil ligt hierin dat de string uit ASCII-tekens met de waarden $20...$7F (decimaal 32...127) bestaat en het decimale getal slechts uit ASCII-tekens met de waarden $30...$39 (decimaal 48...57).
Het INPUT-statement heeft echter nog een andere betekenis! Indien achter het INPUT-statement geen device-nummer volgt dan betekent dat voor de Octopus 65 dat de input moet worden gelezen uit device nummer 1. Device nummer 1 is het keyboard. INPUT A$ is dus gelijk aan INPUT#1,A$. Bij de beschrijving van de DOS is gezegd, dat de Octopus 65 zich kan richten op maximaal acht input-devices en acht output-devices. Twee van deze input-devices zijn RAM-buffers:

INPUT#6,... lees ASCII-tekens uit de RAM-buffer met device-nummer 6,
INPUT#7,... lees ASCII-tekens uit de RAM-buffer met device-nummer 7.

Komt de Octopus 65 in een BASIC-programma een van deze statements tegen, dan leest hij niet de data van het keyboard maar uit de RAM-buffer. Men kan zich dat zo voorstellen: Een pointer wijst een ASCII-teken in de RAM-buffer aan. Telkens als het INPUT-statement een karakter uit de RAM-buffer leest, wordt de pointer met één verhoogd en wijst dan het volgende karakter in de RAM-buffer aan. Een CR-teken in de RAM-buffer beëindigt het INPUT#6- of INPUT#7-statement op gelijke wijze als een CR-teken van het keyboard. De data worden vanuit de diskette in de RAM-buffer geladen vóór de leesoperatie geschiedt. Het laden vanaf de diskette gebeurt steeds in blokken van 2 Kbyte.
Het PRINT-statement is juist tegensteld aan het INPUT-statement. Als de Octopus 65 bijv. het statement PRINT A$ of PRINT A tegenkomt, dan wordt in het eerste geval de string A$ en in het tweede geval het decimale getal A naar output-device nummer 1 gezonden. Dit is bij de Octopus 65 steeds het beeldscherm. Net zoals ASCII-tekens met het PRINT- of PRINT#1-statement naar het beeldscherm kunnen worden gestuurd, kunnen ze ook naar de RAM-buffer worden gezonden. De statements hiervoor zijn:

PRINT#6,... schrijf ASCII-tekens in de RAM-buffer met device-nummer 6,
PRINT#7,... schrijf ASCII-tekens in de RAM-buffer met device-nummer 7.

De schrijf-operatie in de RAM-buffer kunt u zich als volgt voorstellen: steeds als het PRINT-statement een karakter in de buffer schrijft, wordt een pointer met één verhoogd. De pointer wijst dus naar de volgende geheugencel waarin het "te printen" ASCII-teken wordt geschreven. Het schrijven van de RAM-buffer naar de diskette gebeurt eveneens in blokken van 2 Kbyte groot.
Figuur 1a laat zien hoe bij V3.3 een BASIC-programma in het geheugen wordt opgeslagen. Het BASIC-programma begint bij adres $3A79 met het startadres (SAL, SAH), het eindadres (EAL, EAH) en het aantal benodigde tracks (#Tracks). Deze vijf bytes worden automatisch door de BASIC-interpreter ingevuld. Figuur 1b toont hoe bij gebruik van een RAM-buffer het BASIC-programma 2Kbyte naar beneden wordt geschoven. De RAM-buffer ligt dus vóór het BASIC-programma in het geheugen. Voordat de buffer kan worden gebruikt, moet deze eerst met BEXEC* (op de systeemdiskette LOYS) worden ge-

kreëerd met "7 buffer creator".
Er zijn twee soorten files: sekwentiële en random files. Een sekwentiële file is een file waarin data achter elkaar wordt geschreven. Dat gaat zo door vanaf het begin tot het eind van de file. Het speelt daarbij geen rol of de file uit slechts tien getallen en tien strings bestaat, of dat de file uit duizenden data bestaat die op meerdere aaneensluitende tracks op de diskette zijn opgeslagen. Een sekwentiële file moet steeds als één samenhangend extern data-geheugen worden beschouwd. Steeds als door een PRINT#6- of een PRINT#7-statement de 2-Kbyte grote RAM-buffer is volgeschreven, wordt deze in een file op de diskette geschreven, waardoor de RAM-buffer weer vrij is voor nieuwe data.
Met de statements INPUT#6 of INPUT#7 kan een sekwentiële file vanaf de diskette in de Octopus 65 worden gelezen. De INPUT-statements lezen de data vanuit de RAM-buffer op gelijke wijze als van het keyboard. Is de sekwentiële file meerdere tracks groot, dan wordt automatisch een nieuwe track geladen wanneer de hele RAM-buffer-inhoud (2 Kbyte) is uitgelezen.
Bij vele toepassingen worden sekwentiële files onhandelbaar omdat ze niet in kleinere delen (records) kunnen worden onderverdeeld. Moet bijv. de 210^{de} ingave van een sekwentiële file gelezen worden, dan moeten met een INPUT#-statement alle 209 voorgaande inschrijvingen gelezen worden voor de 210^{de} gelezen kan worden. Dat duurt erg lang! Random files zijn in zo'n geval een ideale oplossing. Deze onderscheiden zich van sekwentiële files doordat ze in records zijn onderverdeeld. Elke record krijgt een nummer. Een random-file kan wel uit honderden records bestaan. Elke record kan snel via het record-nummer worden gevonden en opgeroepen. Het is dus onnodig de gehele file te lezen, maar slechts die track in de RAM-buffer waarop zich het gewenste record-nummer bevindt. Het record-nummer staat niet op de diskette. Dat is ook niet nodig omdat de records een vaste lengte hebben (gewoonlijk 128 bytes). Op één track kunnen dus 16 records worden ondergebracht (16 x 128 bytes = 2048 bytes). Bij random files worden net als bij sekwentiële files de data in blokken van 2 Kbytes in de RAM-buffer geladen. De buffer-pointer wordt met 128 verhoogd om de volgende record in de random file te tonen. Als een random file meerdere tracks lang is en vanaf één van deze tracks een bepaalde record moet worden gelezen, dan berekent de DOS automatisch het juiste tracknummer en zet de buffer-pointer op het begin van de gewenste record.

Werken met sekwentiële files

Voordat BASIC met een sekwentiële file kan werken moet deze file eerst geopend worden. Het statement voor het openen van een file is DISKOPEN. Is een file geopend, dan kan het BASIC-programma met behulp van het PRINT#-statement data in de sekwentiële file schrijven. Met het INPUT#-statement kunnen data uit de sekwentiële file worden gelezen. Na een schrijf- of leesoperatie naar of uit de file moet de verbinding tussen BASIC en de RAM-buffer weer worden verbroken. Het statement hiervoor is DISKCLOSE.

Schrijven in een sekwentiële file

We laten nu zien hoe data in een sekwentiële file kunnen worden geschreven. Neem eens aan dat de naam van die file "SDATA" is. Het schrijven in de sekwentiële file verloopt nu in drie fasen:
1) openen van de sekwentiële file
2) schrijven in de sekwentiële file
3) sluiten van de sekwentiële file
Bij de Octopus 65 gaat dat als volgt:

```
DISKOPEN,6,"SDATA"   (file "SDATA"
                      openen)
PRINT#6, D1          (data in de file
PRINT#6, D2          "SDATA"
PRINT#6, D3          schrijven)
...
...
PRINT#6, Dn
DISKCLOSE,6          (file "SDATA"
                      sluiten)
```

De print#-statements schrijven de data D1...Dn in de file "SDATA". Het is ook mogelijk de data D1...Dn met een FOR/NEXT-lus in de sekwentiële file te schrijven.
De data kunnen zowel decimale getallen als strings zijn (D1$...Dn$).

Lezen uit een sekwentiële file

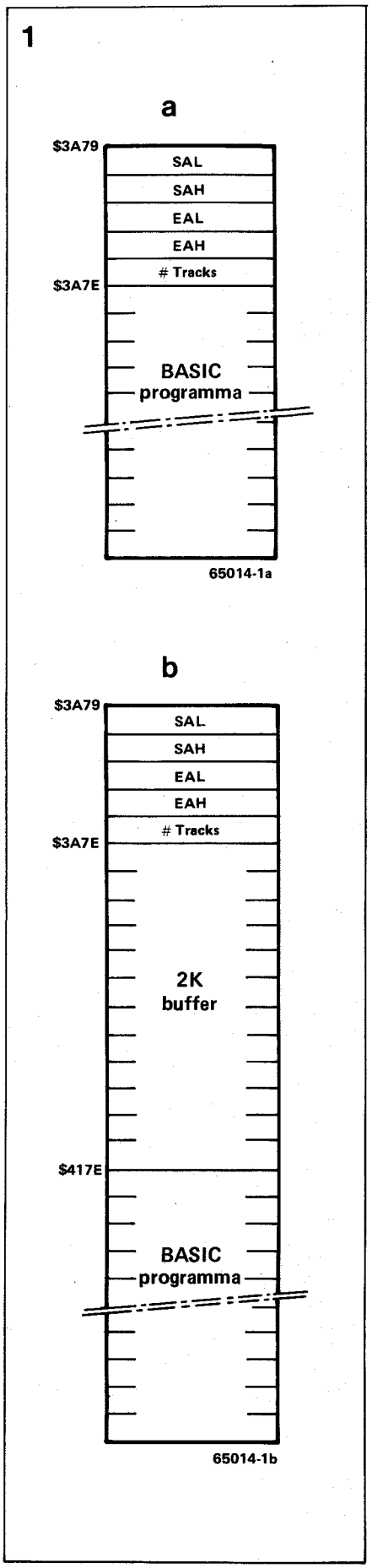
We geven nu aan hoe in het algemeen data uit een sekwentiële file kunnen worden gelezen. We nemen weer even aan dat de naam van de sekwentiële file "SDATA" is. Het lezen uit een sekwentiële file verloopt eveneens in drie fasen:
1) openen van de sekwentiële file
2) lezen uit de sekwentiële file
3) sluiten van de sekwentiële file
Bij de Octopus 65 gaat dat als volgt:

```
DISKOPEN,6,"SDATA"   (file "SDATA"
                      openen)
INPUT#6, D1          (data
INPUT#6, D2          uit
INPUT#6, D3          de
...                  file
...                  "SDATA"
INPUT#6,Dn           lezen)
DISKCLOSE,6          (file "SDATA"
                      sluiten)
```

De INPUT#-statements lezen de data D1...Dn uit de sekwentiële file "SDATA". Er kan ook hier weer een FOR/NEXT-lus worden gebruikt om uit de file te lezen. De data D1...Dn kunnen ook strings zijn (D1$...Dn$).

Werken met random files

Voordat BASIC met een random file kan werken, moet de file worden geopend. Het statement voor het openen van een random file is DISKOPEN. Als een random file geopend is, dan moet aan de BASIC worden verteld in welke record de data geschreven dienen te worden of uit welke record de data gelezen moeten worden. Het statement voor het opzoeken van het gewenste record-nummer is DISKGET,R waarbij R het record-nummer is (0...n). Is het DISKGET,R-statement uitgevoerd, dan kunnen data in die record van de random file met behulp van het PRINT#-statement worden geschreven. Met het INPUT#-statement kunnen data uit die record van de random file worden gelezen.



Figuur 1. De RAM-buffer voor sekwentiële en random files komt voor het BASIC-programma te liggen. De RAM-buffer moet met hulpprogramma 7 in BEXEC* op de diskette "SYSTEM LOYS" worden gekreëerd.

Schrijven in een random file

We laten nu zien hoe data in een random file kunnen worden geschreven. Laten we aannemen dat de naam van de random file "RDATA" is. Het schrijven in de random file gaat in vijf fasen:

1) openen van de random file
2) laden van de gewenste record in de RAM-buffer
3) schrijven in de record van de random file
4) de zojuist beschreven record opslaan op de diskette
5) sluiten van de random file

Bij de Octopus 65 gaat dat als volgt:

DISKOPEN,6,"RDATA"	(file "RDATA" openen)
DISKGET,R	(laden en pointer op record met nummer R zetten)
PRINT#6, D1	(data
PRINT#6, D2	in de
PRINT#6, D3	record schrijven)
...	
...	
PRINT#6, Dn	
DISKPUT	(alleen bij V3.2 nodig: record op de diskette zetten)
DISKCLOSE,6	(file "RDATA" sluiten)

De PRINT#-statements schrijven de data D1...Dn in de record met nummer R van de random file "RDATA". Het is echter ook mogelijk de data D1...Dn met een FOR/NEXT-lus in de record te schrijven. De data D1...Dn kunnen decimale getallen of strings zijn (D1$...Dn$).

Let op: bij toepassing van de versie V3.3 kan het DISKPUT-statement vervallen indien direkt voor de PRINT#-statements een DISKGET-statement staat, omdat in deze versie het DISKPUT-statement in het DISKGET-statement is ingebouwd. Het schrijven in de record van de random file gebeurt dan in vier fasen en het opzoeken van een record gaat tweemaal zo snel als bij versie V3.2!

Lezen uit een random file

We laten hier zien hoe data uit een random file gelezen kunnen worden. De naam van de random file is weer "RDATA". Het lezen uit de random file gaat in vier fasen:

1) openen van de random file
2) laden van de gewenste record in de RAM-buffer.
3) lezen uit de record van de random file
4) sluiten van de random file

Bij de Octopus 65 gaat dit als volgt:

DISKOPEN,6,"RDATA"	(file "RDATA" openen)
DISKGET,R	(laden en pointer op record met nummer R zetten)
INPUT#6, D1	(data uit de
INPUT#6, D2	record
INPUT#6, D3	lezen)
...	
...	
INPUT#6, Dn	
DISKCLOSE,6	(file "RDATA" sluiten)

De INPUT#-statements lezen de data D1...Dn uit de record met nummer R van de random file "RDATA". Het is ook mogelijk de data D1...Dn met een FOR/NEXT-lus uit de record te lezen. De data D1...Dn kunnen decimale getallen zijn of strings (D1$...Dn$).

Sekwentiële en random files in de praktijk

We gaan nu zowel met een sekwentiële als met een random file praktisch werken. Voor sekwentiële files kunnen de RAM-buffers #6 en #7 worden toegepast. Bij random files mag alleen de RAM-buffer #6 worden gebruikt. Men kan met de buffer creator op de diskette "System LOYS" twee RAM-buffers kreëren (4 Kbyte) en dan in een BASIC-programma zowel een sekwentiële file als een random file gelijktijdig openen. Er kan dan tegelijk met twee files worden gewerkt.

Voor u de programma's in de Octopus 65 kunt laden, moeten nog enkele voorbereidingen worden getroffen:

1) Boot de diskette "System LOYS".
2) Kies hulpprogramma 5 van BEXEC*.
3) Neem de diskette "System LOYS" uit disk drive "A".
4) Schuif een lege diskette in drive "A" en druk op CR.
5) Kreëer de file-naam "TRKØ" met het hulpprogramma 2 van BEXEC*. Reserveer één track, voor deze file maar initialiseer de file TRKØ nooit!
6) Kreëer de filenaam "SDATA" met het hulpprogramma 2 van BEXEC*. Reserveer voor deze file twee tracks op de diskette Initialiseer deze file.
7) Kreëer de filenaam "RDATA" met hulpprogramma 2 van BEXEC*. Reserveer vijf tracks voor deze file op de diskette. Initialiseer deze file.
8) Roep het hulpprogramma 7 van BEXEC* op (buffer creator). Kreëer een RAM-buffer. Tik dan niet het demo-programma uit figuur 2 in uw Octopus 65. Dit demo-programma gaat u vertrouwd maken met een sekwentiële file.
9) Na het intikken het programma saven op de diskette (DISK!"PUT SEQ").
10) Start nu weer het programma BEXEC* op de diskette "System LOYS" en roep het hulpprogramma 7 op (buffer creator). Kreëer een RAM-buffer. Tik nu het demo-programma uit figuur 3 in. Dit demo-programma zal u vertrouwd maken met een random file.
11) Na het intikken het programma saven op de diskette (DISK!"PUT RAN")

U hebt nu in de directory de file-naam "SDATA" voor een sekwentiële file opgeslagen en voor deze file twee tracks gereserveerd. Omdat bij een 5¼"-diskette 2 Kbytes op een track kunnen worden ondergebracht, kan de file "SDATA" maximaal 4 Kbytes aan data bevatten. Zou men teveel willen laden, dan geeft de DOS een foutmelding. Daarnaast hebt u in de inhoud de file-naam "RDATA" opgenomen en daarvoor vijf tracks gereserveerd. Omdat op een 5¼"-diskette 16 records van elk 128 bytes op een track ondergebracht kunnen worden, kan de file "RDATA" 5 × 16 = 80 records bevatten (recordnummers Ø...79). Probeert u nog meer records in te geven, dan geeft de DOS weer een foutmelding.

Nadat de sekwentiële file en de random file waren gedefinieerd, hebt u de beide demo-programma's SEQ en RAN met behulp van de "buffer creator" op de diskette "System LOYS" op uw diskette gesaved.

2

```
10 REM ::: sekw :::
20 :
30 :
40 REM open file en schrijf
50 :
60 DISKOPEN,6,"SDATA"
70 FOR X=1 TO 200
80 PRINT#6,"RECORD ": PRINT#6,X
90 NEXT
100 DISKCLOSE,6
110 END
120 :::
130 :::
140 REM open file en lees
150 :
160 DISKOPEN,6,"SDATA"
170 FOR N=1 TO 200
180 INPUT#6,X$
190 INPUT#6,X
200 PRINTX$;X
210 NEXT
220 DISKCLOSE,6
230 END
```

Figuur 2. Demonstratieprogramma voor een sekwentiële file.

Schrijven in de sekwentiële file: RUN

Het programma in figuur 2 schrijft data in de sekwentiële file "SDATA". In regel 6Ø wordt de file "SDATA" geopend en en delen we de BASIC mee dat RAM-buffer #6 wordt gebruikt. Omdat het om een sekwentiële file gaat, zou men ook de RAM-buffer #7 kunnen gebruiken (DISKOPEN,7,"SDATA"). Aansluitend schrijven we in de sekwentiële file:

Ingave 1
Ingave 2
Ingave 3
...
...
Ingave 199
Ingave 2ØØ

Deze ingaven worden met een FOR/NEXT-lus (regels 7Ø...9Ø) in de file geschreven. In regel 1ØØ volgt het sluiten van de sekwentiële file "SDATA". In regel 11Ø wordt het programma beëindigd. Start nu het programma met RUN en let op de aktiviteit van de drive. U kunt duidelijk horen hoe de Octopus 65, nadat de eerste track van de file "SDATA" vol geschreven is, automatisch de volgende track kiest en door gaat met het schrijven in de file.

Lezen van de sekwentiële file: RUN 16Ø

Met het tweede deel van het programma van figuur 2 kunt u de inhoud van de file

3

```
10 REM ::: random file schrijven :::
20 :
30 :
40 :
50 DISKOPEN,6,"RDATA" :::: REM open random file voor schrijven
60 FOR R=0 TO 79 ::::::::: REM op 5 tracks passen 5*16=80 records
70 DISKGET,R ::::::::::::: REM zet pointer op record
80 PRINT#6,"Dit is record nummer "
90 PRINT#6,R
100 DISKPUT
110 NEXT
120 DISKCLOSE,6 :::::::::: REM sluit random file
130 END
140 :
150 REM=============================================
160 REM=============================================
170 :
180 REM ::: random file lezen
190 :
200 DISKOPEN,6,"RDATA" ::: REM open random file voor lezen
210 FOR R=0 TO 79
220 DISKGET,R ::::::::::::: REM zet pointer op record
230 INPUT#6,X$
240 INPUT#6,X
250 PRINTX$,X
260 NEXT
270 DISKCLOSE,6 :::::::::: REM sluit random file
280 END
290 :
300 REM=============================================
310 REM=============================================
320 :
330 REM ::: lees een willekeurige record van de diskette
340 :
350 :
360 DISKOPEN,6,"RDATA" ::: REM open random file voor lezen
370 :
380 INPUT"Welke record wilt u lezen (0...15)";R
390 PRINT: PRINT
400 IF R<0 OR R>79 THEN PRINT: PRINT: GOTO 380
410 DISKOPEN,6,"RDATA" ::: REM open random file voor lezen
420 DISKGET,R ::::::::::::: REM zet pointer op record
430 INPUT#6,X$
440 INPUT#6,X
450 PRINT"De inhoud van deze record is:"
460 PRINTX$,X: PRINT: PRINT
470 GOTO 380
480 REM=============================================
```

Figuur 3. Demonstratieprogramma voor een random file.

"SDATA" lezen. In regel 160 delen we BASIC mee dat de file "SDATA" moet worden geopend. Met een FOR/NEXT-lus in de regels 170...210 wordt de file gelezen en aansluitend op het beeldscherm weergegeven. Het lezen geschiedt met de INPUT#-statements en het weergeven op het beeldscherm wordt verzorgd door PRINT X$,X. Let bij het lezen van de file "SDATA" eens op het funktioneren van de drive! Men kan precies horen hoe de DOS automatisch onschakelt op de volgende track van de sekwentiële file zodra alle data van de eerste track zijn gelezen.

Schrijven in de random file: RUN

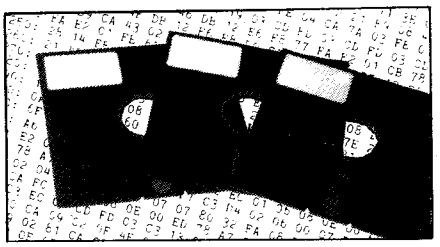
Start het programma RAN met de opdracht RUN "RAN". In regel 50 van figuur 3 vertellen we BASIC dat de file "RDATA" moet worden geopend. Omdat de random file "RDATA" vijf tracks groot is, kan ze 80 records opnemen. In een FOR/NEXT-lus schrijven we in elke record: "dit is record nummer X", waarbij X het nummer van de record is. In regel 70 kiezen we het gewenste recordnummer (DISKGET, R). In regel 80 en regel 90 wordt het aktuele record-nummer in de record geschreven. Het DISKPUT-statement stuurt de inhoud van de RAM-buffer naar de diskette. Het sluiten van de file volgt in regel 120.
Let op: bij de diskette "SYSTEM LOYS" kan het DISKPUT-statement worden weggelaten omdat direkt vóór de PRINT#-statements een DISKGET,R-statement staat. De verwerkingstijd per record kan worden gehalveerd door regel 100 in figuur 3 weg te laten!

Lezen van een random file: RUN 200

Nadat er data in de records van de random file "RDATA" zijn geschreven, kan deze file weer gelezen worden. In regel 200 van figuur 3 openen we de random file "RDATA". In de FOR/NEXT-lus (regels 210...260) lezen we de inhoud van alle 80 records van de random file en tonen die op het beeldscherm. In regel 270 wordt de random file weer gesloten. Let ook nu weer tijdens het lezen van de records op de aktiviteiten van de drive. Men kan duidelijk horen dat na het lezen van 16 records automatisch een nieuwe track in de RAM-buffer wordt geladen!

Lezen van een enkele record van een random file: RUN 360

Men kan ook slechts een enkele record van de file "RDATA" lezen. Het programma staat in de regels 360...470 van figuur 3.
Opmerking: in dit voorbeeld wordt de random file niet met een DISKCLOSE-statement gesloten. Bij het lezen van een random file kan het sluiten van de file vervallen. In de OSI-handboeken wordt nergens op deze eigenschap gewezen. Blijft een random file na het lezen geopend, dan zal de verwerkingstijd bij het zoeken van de records veel korter zijn!



Het monitorprogramma

De Octopus 65 is met een zeer krachtig monitorprogramma uitgerust. Het bijzondere van dit monitorprogramma is, dat het de gebruiker ook ten dienste staat als een transient-programma is geladen! Het monitorprogramma kan dus voortdurend gebruikt worden en kan nooit door een transient-processor worden overschreven. Bij het aanpassen van software heeft men op deze wijze drie troeven in de hand:

1) Het monitorprogramma staat in een geheugengedeelte dat streng gescheiden is van het eigenlijke werkgeheugen. Overschrijven van het monitorprogramma door het werkprogramma is dus onmogelijk.
2) Met de monitor kan men elk programma bekijken dat van de diskette in het werkgeheugen wordt geladen. Als u wilt bekijken hoe de BASIC-interpreter van Microsoft werkt, dan kunt u met behulp van de tracer (zie hoofdstuk: Werken met de assembler) de handelingen van de interpreter precies volgen. Wij hebben op deze wijze de BASIC kunnen aanpassen.
3) Het monitorprogramma beschikt over een eenvoudige 6502-disassembler.

We gaan nu alle monitor-opdrachten behandelen die de Octopus 65 ter beschikking staan. Daartoe verdelen we de opdrachten in groepen, waardoor het eenvoudiger wordt ze te onthouden.

1) Melding van het monitorprogramma

De Octopus 65 kent twee meldingen na de uitvoering van een opdracht:
1) -OCTOPUS 65-
2) :

Telkens als de computer de eerste melding geeft, wordt de stack-pointer geinitialiseerd (wijst $1FF aan). Wordt een opdracht door drukken op de toets ↑C voortijdig afgebroken, dan meldt de monitor zich met — OCTOPUS 65 —.
Als de monitor zich meldt met het teken ":", dan wordt de stack-pointer niet teruggezet. Dit is belangrijk als de gebruiker een programma stap voor stap doorloopt met behulp van de tracer, enkele geheugenplaatsen wijzigt en dan weer in de tracer-kommandolus terugkeert. Natuurlijk mag in zo'n geval de stack-pointer niet worden gewijzigd.

2) Wijzigen van de geheugenplaatsen

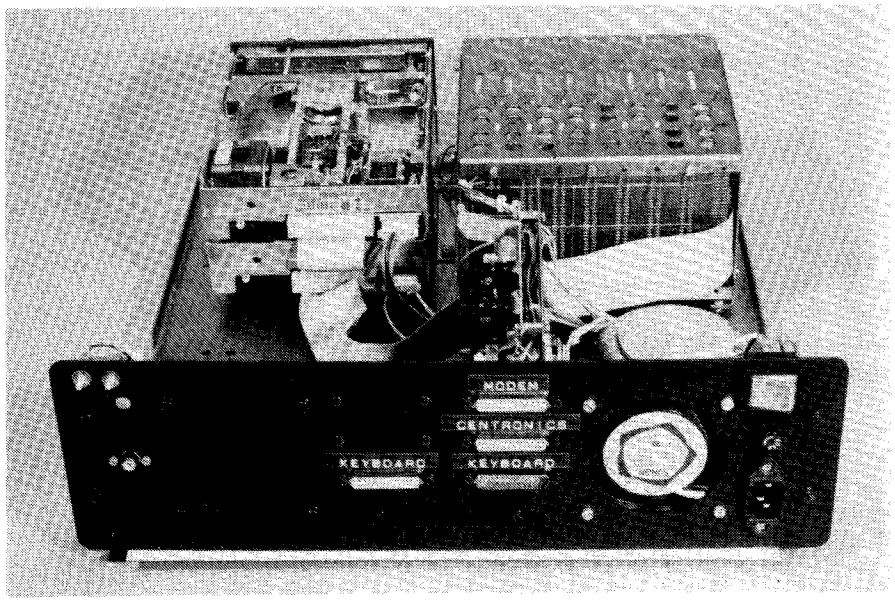
Bij de "standaard"-uitvoering van de Octopus 65 zijn 56 Kbytes RAM beschikbaar ($0000...DFFF). Niet elke geheugenplaats in dit bereik mag men wijzigen, omdat bepaalde geheugenplaatsen informatie voor de DOS en het monitorprogramma bevatten. Wijzig dus nooit de geheugenplaatsen $E0...$FF in page zero en $1C0...1FF in het stack-bereik. Is de DOS niet geladen, dan kan men $E400...$E7AF voor eigen programma's gebruiken. Is de DOS geladen, dan van dit geheugenbereik afblijven! De BASIC- en DOS-uitbreidingen vallen hier buiten.
Nu de behandeling van de kommando's waarmee de geheugeninhoud kan worden gewijzigd:

Space geef de inhoud van het ingetypte adres.
↑ geef de inhoud van het volgende adres.
Met "↑" wordt hier niet de controltoets bedoeld! Het is ook niet de cursor-stuurtoets "↑". Het is het ASCII-teken "↑" dat op vele keyboards met ^ is aangegeven.
— geef de inhoud van het vorige adres.
/ plaats het byte voor het teken "/" op het momentele adres en ga naar het volgende adres.
' zet het ASCII-teken achter het teken "'" op het momentele adres en ga naar het volgende adres.
om snel in te typen: lees twee toetsen en voeg die samen tot een byte. Zet het byte op het momentele adres en ga naar het volgende adres. Herhaal deze procedure zo lang totdat de gebruiker de toets @ nog eens indrukt. Er komt een foutmelding als de Octopus 65 een ongeldig hex-teken van het keyboard krijgt. Met de toets ↑H of "backspace" kan men de zojuist ingegeven byte korrigeren. Bij veel keyboards moet men het teken "§" ingeven. Op het beeldscherm verschijnt echter steeds het teken @.

Figuur 1 laat zien hoe simpel met deze vijf opdrachten het geheugen kan worden gewijzigd. De gebruiker geeft het adres F000 in en drukt op de spatiebalk. De Octopus 65 geeft de inhoud van het ingegeven adres. De gebruiker drukt de toets ↑ meerdere malen in en kan zo zien wat de volgende geheugenplaatsen bevatten. Door de toets — meerdere malen in te drukken kan de gebruiker weer bekijken wat in de voorgaande geheugenplaatsen is opgeslagen.
Nu kiest de gebruiker het adres 200. Na het indrukken van de spatie-toets geeft de gebruiker de data 1, 2, 3, 4, F1, F2, F3. Steeds als de toets / wordt gedrukt, slaat de Octopus 65 het ingegeven byte op in het momentele adres en maakt het volgende adres klaar voor ingave. Na het ingeven van deze bytes kiest de gebruiker het adres 300 en plaatst vanaf dit adres de string "OCTOPUS 65". Elk ASCII-teken na de opdracht ' wordt in het geheugen op het momentele adres opgeslagen.
Onderin figuur 1 kiest de gebruiker de snellaad-mode. Eerst wordt adres 400 gekozen. Door indrukken van de toets @ schakelt de Octopus 65 op snel-ingave. De gebruiker geeft nu de hexgetallen 01...14 in. Na de ingave van deze getallen drukt hij weer op kommando toets @ en verlaat zo de snellaad-mode.

3) Wijzigen van de CPU-registers

Bij het testen van zelfgeschreven programma's moeten vaak programmalussen (loops) onderzocht worden. Zoals we in het hoofdstuk "werken met de assembler" zullen zien, gebruikt men voor het testen van deze programma's de daar beschreven tracer. Wordt een programmalus met de tracer 20 maal of meer doorlopen, dan kan dat voor de gebruiker vervelend worden. Daarom zijn in de monitor opdrachten opgenomen waarmee de register-inhouden van de CPU



Fotoserie, figuur 7. Nu eens voor de afwisseling de achterkant van de kast. Links zitten twee kleine druktoetsen, één voor het resetten van de computer en één voor de tracer. Daaronder bevindt zich de video-uitgang.

1

```
-OCTOPUS 65-

: F000 <Space>
: F000 48 ^
: F001 8A ^
: F002 48 ^
: F003 98 ^
: F004 48 ^
: F005 AD ^
: F006 63 -
: F005 AD -
: F004 48 -
: F003 98 -
: F002 48 -
: F001 8A -
: F000 48 200 <Space>
: 0200 F0 1/
: 0201 34 2/
: 0202 5F 3/
: 0203 A7 4/
: 0204 EB F1/
: 0205 FD F2/
: 0206 FB F3/
: 0207 EC 300 <Space>
: 0300 50 'O
: 0301 20 'C
: 0302 47 'T
: 0303 85 'O
: 0304 01 'P
: 0305 34 'U
: 0306 42 'S
: 0307 48 '
: 0308 F0 '6
: 0309 36 '5
: 030A DF 400 <Space>
: 0400 BF @
0400: 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
0410: 11 12 13 14 @

-OCTOPUS 65-
```

kunnen worden gewijzigd. Lustellers die meestal door de CPU-registers geaktiveerd worden, kunnen zo vóór de sprong in de lus worden gewijzigd.

.RA wijzig de accu
.RX wijzig het X-register
.RY wijzig het Y-register
L toon de inhoud van alle CPU-registers (list-kommando)

Figuur 2 laat zien hoe deze kommando's worden gebruikt. De gebruiker schrijft in de accu 1, in het X-register 1C en in het Y-register F0. Het dan volgende L(ist)-kommando toont de inhoud van alle CPU-registers.

4) Weergeven van de geheugeninhoud (dump)

In de Octopus 65-monitor zijn twee mogelijkheden opgenomen om de inhoud van het geheugen weer te geven: hexdump en ASCII-dump. De opdrachten daartoe zijn:

.H of M maak een hexdump van de inhoud van het geheugen.
.A maak een ASCII-dump van de geheugeninhoud.

Na het geven van de dump-opdracht vraagt Octopus naar het start- en eindadres van het geheugendeel dat "gelist" moet worden. Figuur 3 laat een deel van de BASIC-interpreter zien, eerst als hexdump en daarna als ASCII-dump. Even een opmerking hierover. Het laatste karakter van de gereserveerde BASIC-woorden zoals END, FOR, NEXT, enz., kan in een ASCII-dump niet worden weergegeven, omdat bit 7 geset is. De BASIC-interpreter gebruikt bit 7 als eindaanduiding voor een gereserveerd BASIC-woord. Vanaf adres $200 staan de inspring-adressen voor de gereserveerde BASIC-kommando's (adres min één). Het END-statement wordt vanaf adres $082A en het NEXT-statement vanaf adres $0C4B afgewerkt.

5) Disassembleren van een programma in het geheugen

Het monitorprogramma van de Octopus 65 bevat een eenvoudige 6502-disassembler, niet te verwisselen met de symbolische disassembler in de Micro-Ware-assembler (zie het hoofdstuk "Werken met de assembler"). Figuur 4 toont een voorbeeld met deze eenvoudige disassembler. We disassembleren een deel van de BASIC-interpreter. Met het kommando ".D" starten we de disassembler. Het monitorprogramma gaat naar de kommando lus van de disassembler en wacht na de melding "DISASM:" op de ingave van start- en eindadres. Beide adressen geeft men gescheiden door een komma en sluit dan af met een CR-kommando. Na de ingave van de adresparameters geeft de disassembler vier mogelijkheden:

↑D start de disassembler opnieuw.
↑L disassembleer vanaf het startadres tot aan het eindadres (listing).
↑P disassembleer 16 instrukties en wacht op ↑D, ↑L, ↑P of ↑S (page).
↑S disassembleer een instruktie en wacht op ↑D, ↑L, ↑P of ↑S (step).

De kommandolus van de disassembler kan door indrukken van toets ↑C of door .. worden verlaten. In het eerste geval wordt de stack-pointer teruggezet, in het tweede geval niet.

6) Uitvoer naar een printer

Het monitorprogramma van de Octopus 65 kan printers zowel serieel als parallel sturen. De seriële uitvoer geschiedt via de ACIA 6551 en een RS232/V24-interface op de CPU-kaart. De parallel-uitvoer gaat via een VIA 6522, die elke printer met een Centronics-interface kan sturen. De opdrachten voor de printerbesturing zijn:

.P kies de parallelle printer-uitgang (Centronics)
.S kies de seriële printer-uitgang (RS232/V24)
.RF schakel de beide printer-uitgangen uit (reset output flags)

Let op, vóór via de seriële uitgang data kunnen worden gezonden, moet de ACIA 6551 op de CPU-kaart met kortsluitstekers worden geprogrammeerd. Hoe dat gedaan wordt, hebt u kunnen lezen bij de bouwbeschrijving van de CPU.

7) Starten van een machineprogramma

Een machineprogramma in het geheugen

2

```
-OCTOPUS 65-

.RA?1 <Cr>
: .RX?1C <Cr>
: .RY?F0 <Cr>
: L
A  X  Y  SP        PC   NV BDIZC
01 1C F0 01FF+1=68 DD00 00000100
:
```

3

```
-OCTOPUS 65-

: .H
HEXDUMP: 200,300
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200: 29 08 47 07 4A 0C F8 08 2B 0B 23 0F 57 0B A5 09 ).G.J...+.#.W...
0210: A5 08 7D 00 4C CB 25 08 88 08 D2 08 3B 09 27 08 ..}.L.%.....;.'.
0220: 4B 09 6F 37 9B 16 3B 22 52 22 34 12 92 16 58 20 K.o7..;"R"4...X
0230: 52 08 B8 06 7B 06 61 06 34 1B C7 1B 53 1B F7 22 R...{.a.4...S..*
0240: 04 12 25 12 45 1E 66 1F B3 18 C1 1E A2 1F A9 1F ..%.E.f.........
0250: F2 1F 56 20 88 16 F6 15 E9 12 27 16 05 16 66 15 ..V ......'...f.
0260: 7A 15 A6 15 B1 15 79 D8 16 79 C1 16 7B F3 18 7B z.....y..y..{..{
0270: 0C 1A 7F 4E 1E 50 88 0E 46 88 0E 7D 87 1E 5A E9 .. N.P..F..}..Z.
0280: 0D 64 B8 0E 45 4E C4 46 4F D2 4E 45 58 D4 44 41 .d..EN.FO.NEX.DA
0290: 54 C1 49 4E 50 55 D4 44 49 CD 52 45 41 C4 4C 45 T.INPU.DI.REA.LE
02A0: D4 47 4F 54 CF 52 55 CE 49 C6 52 45 53 54 4F 52 .GOT.RU.I.RESTOR
02B0: C5 47 4F 53 55 C2 52 45 54 55 52 CE 52 45 CD 53 .GOSU.RETUR.RE.S
02C0: 54 4F D0 4F CE 45 44 49 D4 54 52 41 D0 45 58 49 TO.O.EDI.TRA.EXI
02D0: D4 44 49 53 CB 44 45 C6 50 4F 4B C5 50 52 49 4E .DIS.DE.POK.PRIN
02E0: D4 43 4F 4E D4 4C 49 53 D4 43 4C 45 41 D2 4E 45 .CON.LIS.CLEA.NE
02F0: D7 54 41 42 A8 54 CF 46 CE 53 50 43 A8 54 48 45 .TAB.T.F.SPC.THE
0300: CE

: .A


ASCII DUMP: 200,300
     0123456789ABCDEF
0200: ).G.J...+.#.W...
0210: ..}.L.%.....;.'.
0220: K.o7..;"R"4...X
0230: R...{.a.4...S..*
0240: ..%.E.f.........
0250: ..V ......'...f.
0260: z.....y..y..{..{
0270: .. N.P..F..}..Z.
0280: .d..EN.FO.NEX.DA
0290: T.INPU.DI.REA.LE
02A0: .GOT.RU.I.RESTOR
02B0: .GOSU.RETUR.RE.S
02C0: TO.O.EDI.TRA.EXI
02D0: .DIS.DE.POK.PRIN
02E0: .CON.LIS.CLEA.NE
02F0: .TAB.T.F.SPC.THE
0300: .NO.STE......AN.
0310:
```

4

```
-OCTOPUS 65-

.D

DISASM: 3A1,3C0 <Cr>

^D,^L,^P,^S ?
<↑L>
03A1: BA          TSX
03A2: E8          INX
03A3: E8          INX
03A4: E8          INX
03A5: E8          INX
03A6: BD 01 01    LDA $0101,X
03A9: C9 81       CMP #$81
03AB: D0 21       BNE $03CE
03AD: A5 97       LDA $97
03AF: D0 0A       BNE $03BB
03B1: BD 02 01    LDA $0102,X
03B4: 85 96       STA $96
03B6: BD 03 01    LDA $0103,X
03B9: 85 97       STA $97
03BB: DD 03 01    CMP $0103,X
03BE: D0 07       BNE $03C7
03C0: A5 96       LDA $96

^D,^L,^P,^S ?
<Cr>

-OCTOPUS 65-
```

5

```
: I AT D047 <Cr>
READY
: I AT D12C <Cr>
READY
: I AT D200 <Cr>
READY
: H
01: D047
02: D12C
03: D200

: K
01: D047
02: D12C
03: D200
NR.? 2
: H
01: D047
02: D200

:
```

van de Octopus 65 kan met de G-opdracht worden gestart. Voor de programmastart moet eerst het startadres van het programma worden gegeven:

:XXXX space kies het startadres $XXXX
:XXXX YY XXXX=startadres; YY=data op het startadres
:XXXX YY G starten van het programma

8) Het zetten van breakpoints (programma-onderbrekingen)

Bij het testen van zelfgeschreven programma's komt men op een punt waarbij grote delen van het programma zonder fouten lopen, maar sommige delen willen niet werken zoals de bedoeling was. Meestal zitten die foutieve programmadelen midden in het programma en het zou dan lang duren om met de tracer het foutieve deel te bereiken. Het is daarom verstandig het programma tot aan de foutieve plaats normaal uit te voeren en op de foutieve plaats een "noodrem" (breakpoint) in te bouwen. Het programma loopt dan na het starten tot aan het breakpoint en keert dan terug naar het monitorprogramma van de Octopus 65. In het monitorprogramma worden na het bereiken van het breakpoint alle CPU-registers gesaved en de gebruiker kan vervolgens met het L-

6

```
: .D

DISASM: 0,3

^D,^L,^P,^S ?

0000: 20 5D F4    JSR $F45D
0003: 00          BRK



-OCTOPUS 65-

: .A


ASCII DUMP: 2200,22FF
      0123456789ABCDEF
2200: L.".^& .&Lg) T'.
2210: ..¥..... ."....
2220: 2..... .".....
2230: ..... .".......
2240: ..... .".t.2....
2250: .... .".........
2260: ... .".........
2270: ".....# .. -.
2280: Y. .. /. .......
2290: ...........!#."#
22A0:  s-...-OCTOPUS 6
22B0: 5- V2.8 SYSTEM
22C0:  LOYS SHORTHANDS
22D0:  & FULL SCREEN E
22E0: DITOR....L.¥....
22F0: )?I?.....)?..'$
2300:
```

kommando alle CPU-registers en de flags in het status-register tot aan het breakpoint kontroleren. Het breakpoint kan nu gewist worden en de gebruiker kan met de tracer het foutieve programmadeel stap voor stap doorlopen.

Wat gebeurt er nu eigenlijk bij het zetten van een breakpoint? De 6502 kent een BRK-instruktie ($00) die bij het zetten van breakpoints wordt gebruikt. BRK is een software-interrupt en kan worden beschouwd als een sprongopdracht met de lengte van een byte. De sprong naar het monitorprogramma geschiedt via de IRQ-vektor van de CPU. Zet de gebruiker op een bepaald adres een breakpoint, dan wordt het byte op dit adres op de breakpoint-stack gesaved en daarna door de break-instruktie overschreven. De breakpoint-stack kan maximaal vijf breakpoints bevatten. Is deze stack vol, dan geeft de Octopus 65 een foutmelding.

En wat gebeurt er bij het verwijderen van een breakpoint? Het breakpoint wordt dan door het byte overschreven dat oorspronkelijk op dit adres aanwezig was. Het monitorprogramma haalt het breakpoint-adres en het byte dat het breakpoint vervangt van de breakpoint-stack. De kommando's voor het zetten en verwijderen van breakpoints zijn:

I Zet een breakpoint. De Octopus 65 vraagt dan naar het adres waar een breakpoint gezet moet worden. Na ingave van dat adres wordt het breakpoint gezet en de breakpoint-stack opgevuld. De Octopus 65 meldt zich met "READY", wat betekent dat het breakpoint gezet is (I = insert breakpoint).

K Met het K-kommando kan een breakpoint in het programma worden verwijderd (K = kill breakpoint).

H Met het H-kommando kunnen alle breakpoints in de breakpoint-stack op het beeldscherm weergegeven worden (H = help to display breakpoints).

Verdere kommando's die tezamen met de breakpoint-opdrachten werken, zijn:

L Geeft de inhoud van alle CPU-registers tot aan het breakpoint.

P-- Zet de programmateller op het adres waarop zich het breakpoint bevond.

Door de 6502 wordt na het uitvoeren van een BRK-instruktie de programmateller-inhoud plus twee op de stack gezet. Daarom moet, nadat de BRK-instruktie heeft plaatsgevonden, de P-toets en daarna de —toets tweemaal worden ingedrukt om het breakpoint-adres weer te bereiken. Figuur 5 toont met een voorbeeld hoe de Octopus 65 op de diverse opdrachten reageert.

9) Kopiëren van datablokken (block move)

Het monitorprogramma heeft een kopieerprogramma dat datablokken vanuit het ene geheugengedeelte naar een ander kan kopiëren. Het kommando luidt:
ZZZZ=XXXX,YYYY cr Transporteer een datablok, beginnend op het adres $XXXX en eindigend op het adres $YYYY-1, naar adres $ZZZZ.
Opmerking: de datablokken mogen elkaar niet overlappen!

10) DOS en transient processor laden

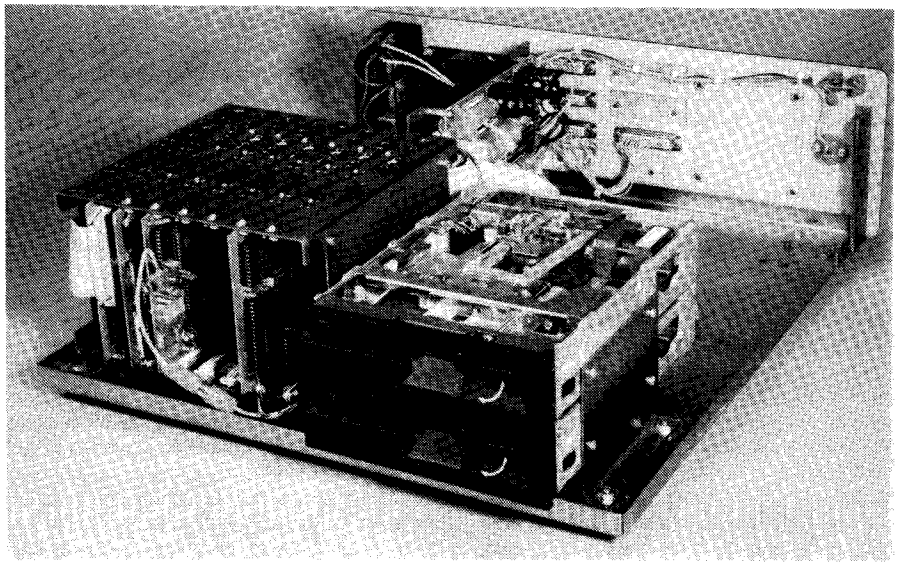
.B laad de DOS en de transient processor in.
Wat gebeurt er nadat u .B hebt ingetikt? De Octopus 65 doet dan het volgende:

* initialiseert de floppy-interface.
* zet de kop van disk drive op track nul.
* wacht op het begin van het index-gat.
* als het index-gat is gevonden wacht hij tot het einde van het indexgat.
* leest de laadvektor en het aantal te laden 256-byte-blokken van track nul en slaat deze data op.
* leest de desbetreffende 256-byte-blokken in het geheugen.
* springt naar het adres dat de laadvektor aangeeft (normaal is dat $2200). De belangrijkste BDOS-subroutines staan nu in het geheugen.
* laadt de CCP en de transient processor van de diskette in het geheugen.

Verdere informatie vindt men bij de beschrijving van het disk-operating-system. Krijgt u bij het laden van de diskette moeilijkheden, dan kan track nul toch in het geheugen worden geladen. Daartoe moet u het volgende, korte programma op page zero ingeven:
JSR $F45D...20 5D F4
BRK...00
Start dit programma. Als track nul vanaf het adres $2200 is geladen, meldt de Octopus 65 zich via de monitor. Figuur 6 laat de ASCII-dump zien die op uw scherm moet verschijnen. Als de Octopus 65 zich niet meldt of een afwijkende ASCII-dump toont, kan een van de volgende fouten zijn opgetreden:

* de schrijf/leeskop is onjuist afgesteld.
* de schrijf/leeskop is vuil.
* de as van de drive draait te snel of te langzaam.
* de index-puls is niet aanwezig.
* de PIA- en ACIA-adressen op de floppy-kaart liggen nog volgens het OSI-systeem op $C000 en niet volgens het Octopus-systeem op $E000.

Opmerking: slechts een vakman of -vrouw mag een disk drive afregelen! De ASCII-dump in figuur 6 kan van diskette tot diskette verschillen. Belangrijk is slechts dat ergens tussen $2200 en $2300 een string "OCTOPUS 65-V..." staat. Verder zijn in dit geheugendeel alle informaties aanwezig die de Octopus 65 nodig heeft om eerst de DOS en daarna de BASIC-interpreter en BEXEC* te laden.



Fotoserie, figuur 8. De kompleet opgebouwde Octopus. De kaartenhouder is zodanig geplaatst dat men de kaarten gemakkelijk aan de voorzijde van de kast kan uittrekken en insteken.

Door het zetten van breakpoints in het geheugendeel $2200...$2300 kunt u het systeem in delen "booten" en daardoor vaststellen hoe en waar de fouten bij het opstarten van het systeem ontstaan.
Bij OSI-diskettes vindt men in plaats van de string "-OCTOPUS 65-" een string "OS-65D"!
Aanwijzing: De eerste 256 bytes op track nul kunnen zo gewijzigd worden dat in plaats van de BASIC-interpreter de assembler of de wordprocessor geladen wordt.

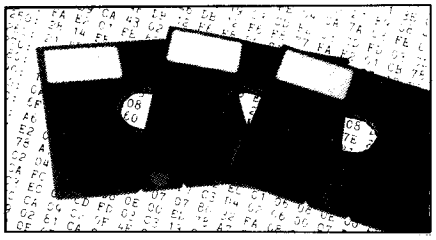
11) Verdere monitorkommando's

.OS65D Verlaat de monitor en spring naar de Console Command Processor (CCP). De DOS meldt zich met A* of B*, enz. Daartoe moet deze natuurlijk eerst met .B geladen zijn!

N Zet de NMI-vektor weer op het oorspronkelijke adres. Dit kommando is alleen van belang als de gebruiker de NMI-vektor gewijzigd heeft.

V Zet de vektor van de stoptoets ↑C weer in de oorspronkelijke toestand. Steeds als het monitorprogramma een karakter geeft, wordt getest of tijdens het uitgeven de stoptoets ↑C werd ingedrukt. Indien dat het geval was wordt de uitgave van karakters gestopt, waarna via de vektor van de stoptoets een sprong naar de monitor volgt. De computer meldt zich met "-OCTOPUS 65-" weer terug. Vaak wil de gebruiker de karakteruitgave-routine van het monitorprogramma in zijn eigen programma opnemen; bij het drukken van de stoptoets ↑C mag dan echter niet naar het monitorprogramma worden teruggesprongen, maar naar het eigen gebruikerprogramma. De stopvektor dient dan gewijzigd te worden. Belangrijke adressen:
* NMI-vektor $E7C9,E7CA
* IRQ-vektor $E7CB,E7CD
* stopvektor $E7C5,E7C6

wis het beeldscherm en initialiseer de computerperiferie (cold start).



Werken met de assembler

Op de Octopus 65 staan twee assemblers ter beschikking:
1) de MOS-technology-assembler
2) de Micro-Ware-assembler
De eerste van deze twee staat op de OSI Tutorial Disk 5 en kan vanuit BASIC met de opdracht DISK!"AS worden opgeroepen. Deze assembler staat beschreven in de OSI-dokumentatie. Wij vonden de MOS-assembler zeer gekompliceerd en ook de OSI-editor was verre van ideaal. Als bijv. in een regel een tikfout optreedt, dan moet de gehele regel opnieuw ingetypt worden.
Voor de Octopus 65 hebben we een nieuwe systeemdiskette gemaakt, die wordt aangeduid met:
--OCTOPUS 65--V2.8 SYSTEM LOYS
Op deze diskette staat de Micro-Ware-assembler, waarmee alle Octopus 65-programma's zijn geschreven. De assembler is afkomstig van "de 6502 kenners", de grootste 6502-computerclub in Nederland. Nadere informatie over deze club, die een internationaal karakter heeft, vindt u aan het einde van deze Special. Ook de editor van de oorspronkelijke Micro-Ware-assembler kon ons niet bekoren en daarom schreven we een nieuwe editor. Omdat de assembler oorspronkelijk bedoeld was voor gebruik met een cassette-recorder, werd eveneens nog een programma geschreven dat de Micro-Ware-assembler met de OHIO-DOS kan laten samenwerken. We hebben zolang aan deze assembler gewerkt totdat we een optimale opzet hadden gevonden. Sedert jaren is deze assembler nu onze Elektuur-huis-assembler. Zeer veel in Elektuur gepubliceerde computerprojekten, of het nu software of hardware was, werden inmiddels met deze assembler ontwikkeld en getest.
De assembler kan op verschillende manieren van de diskette in de Octopus 65 worden geladen. Gewoonlijk kiest men menu nummer 10, waarna BEXEC* vanaf de diskette "SYSTEM LOYS" in de computer wordt geladen. Vanuit BASIC kan de assembler worden opgeroepen met de opdracht DISK!"SE A":DISK!"AS". De regel-georiënteerde editor en de symbolische disassembler worden tegelijkertijd met de assembler in de Octopus 65 geladen. De editor geeft met een "–" aan dat de gebruiker kommando's kan geven. Het speciale teken @ aan het begin van een regel brengt de editor van de input-mode in de kommando-mode. Dit teken @ mag niet in de tekst worden gebruikt; dit kan op veel keyboards alleen met de §-toets worden opgeroepen. Op het beeldscherm staat echter steeds @ .
De assembler bestaat uit drie delen:
* editor (=tekstverwerkingssysteem)
* assembler
* symbolische disassembler (noodzakelijk bij software-aanpassingen)

1) De editor

Met de editor kunnen we een assemblerprogramma in de computer intypen. Staan er syntax-fouten in het programma, dan signaleert de assembler deze fouten, zodat de gebruiker die kan korrigeren. De editor is regel-geörienteerd, hetgeen betekent dat elke regel een regelnummer heeft. De regels kunnen in stappen van één, vijf of tien worden genummerd. Hoe eenvoudig met de editor een assemblerprogramma in de Octopus 65 kan worden getypt, gaan we nu ontdekken. Eerst beschrijven we alle stuur-kommando's van de editor en geven dan een demonstratievoorbeeld. Maar intussen merken we even op dat de editor niet alleen voor het intypen van assembler-programma's geschikt is, maar ook uitstekend kan worden gebruikt voor het schrijven van brieven. Maar daarvoor kan men beter het beeldscherm-georiënteerde tekstverwerkingssyteem gebruiken dat in deze Special beschreven wordt.
De kommando's van de editor zijn in drie groepen onder te verdelen:
* Kommando's voor het bewegen van de cursor
* Kommando's voor het wijzigen en toevoegen van tekst
* Kommando's voor de diskette (lezen en schrijven)

Kommando's voor het verplaatsen van de cursor

De cursor kan alleen in horizontale richting worden bewogen. In vertikale richting is dus onmogelijk. Er kunnen zeven opdrachten worden gebruikt:
* ↑B cursor naar links
* ↑N cursor naar rechts
* ↑T cursor naar het linker regel-einde
* ↑Y cursor naar het midden van de regel
* ↑U cursor naar het rechter regel-einde
* ↑L stel tabulator in (tabulator = momentele cursorpositie)
* "underline" = zet de cursor op de tabulator

Kommando's voor het wijzigen en toevoegen van tekst

Slechts de eerste letter behoeft te worden ingegeven. In de ingave-mode staan drie krachtige kommando's ter beschikking, waarmee men gemakkelijk wijzigingen in de file kan aanbrengen. De kommando's zijn:
* ↑O overstrike = overschrijf de letter onder de cursor
* ↑I insert = voeg een nieuwe letter in vóór de cursor en verschuif de tekst naar rechts
* ↑D delete = wis de letter onder de cursor en verschuif de tekst naar links

Overige kommando's

Alle in het nu volgende deel gebruikte regelnummers zijn slechts als voorbeeld op te vatten.
=CLEAR=
C(LEAR) wis het geheugen
Met de CLEAR-opdracht kan een file in het geheugen van de computer worden

gewist. Het werkgeheugen van de Octopus 65 wordt automatisch geformatteerd en de gebruiker kan een nieuw programma laden.

—C(LEAR) cr de computer vraagt dan:
NEW?Y(ES) cr de gebruiker geeft Y(ES) in
CLEAR het geheugen is vrij, de editor wacht op een nieuwe opdracht

Steeds als de editor opnieuw geladen wordt vraagt de computer of het werkgeheugen gewist en geformatteerd moet worden. De gebruiker dient dan in elk geval na de vraag "NEW?" met "YES" te antwoorden, om het geheugen te formatteren.

=ADD=
A(DD) voeg nieuwe regels toe aan het einde van de file
Indien geen file van de diskette in de computer is geladen, kan met het kommando A(DD) een nieuwe file worden geopend. Vóór het openen van een nieuwe file moet het geheugen met de CLEAR-opdracht geformatteerd worden. Na het ADD-kommando geeft de editor een nieuw regelnummer.

A(DD) cr ga in de ingave-mode
0000: dit is regel 0 cr
0010: dit is regel 10 cr
0020: dit is regel 20 cr
0030: dit is regel 30 cr
0040: @ cr verlaat de ingave-mode
- de editor wacht op een nieuw kommando.

=LIST=
L(IST) cr toon de inhoud van de gehele file
L(IST)10 cr laat regel 10 zien
L(IST)10,30 cr laat de regels 10...30 zien
Met het L(IST)-kommando kan de gebruiker de gehele file of een deel van de file op het beeldscherm zetten. Moet de file ook via de Centronics-interface naar de printer worden gezonden, dan staan daartoe de volgende opdrachten ter beschikking:

-* cr verlaat de editor-kommando-mode
A*IO ,09 cr schakel de printer in via de Centronics-interface
A*AS cr keer terug naar de editor van de assembler en wacht op een nieuw kommando

=DELETE=
D(ELETE)20 cr wis regel 20 in de file
D(ELETE)10,30 cr wis de regels 10...30 in de file
Met het D(ELETE)-kommando kunnen regels uit de file worden verwijderd. Worden in een grote file (= lang programma) vele regels gewist, dan kan het "deleten" wel een minuut duren voor de computer in de kommando-mode terugkeert.

=INSERT=
I(NSERT)20 cr voeg vóór regel 20 een of meer nieuwe regels in
-L(IST) geef de inhoud van
0000: dit is regel 0
0010: dit is regel 10
0020: dit is regel 20
0030: dit is regel 30
0040: @
-I(NSERT))20 cr voeg vóór regel 20 nieuwe regels in
0011: dit is regel xx
0012: dit is regel yy
0013: keer terug naar de kommando-mode
- wacht op een nieuw kommando

=NUMBER=
N(UMBER) cr nummer alle regels in stappen van tien
N(UMBER)5 cr nummer alle regels in stappen van vijf
N(UMBER)1 cr nummer alle regels in stappen van één
-N(UMBER) cr nummer alle regels in stappen van tien
-L(IST) laat de inhoud zien van
0010: dat is regel 0
0020: dat is regel 10
0030: dat is regel xx
0040: dat is regel yy
0050: dat is regel 20
0060: dat is regel 30
0070: wacht op een nieuwe opdracht

=FIX=
F(IX)10 cr haal regel 10 uit de file om te korrigeren en voeg zonodig nieuwe regels in
-F(IX)10 cr voer op regel 10 een FIX-kommando uit
0010: dat is regel 0
De editor maakt een kopie van regel 10 en wacht op een nieuwe ingave. De cursor-stuurtekens ↑T, ↑Z, ↑U,↑B, ↑N, evenals de editor-kommando's ↑I, ↑D, ↑O zijn operationeel.
Let op! Om het FIX-kommando stap voor stap te demonstreren is regel 10 meerdere keren na elkaar afgedrukt. Op het beeldscherm is echter slechts één regel 10 te zien!
0010: dat is regel 0 als ↑U is gedrukt
0010: dat is regel 0 als ↑T is gedrukt
0010: dat is regel 0 als 7 x ↑N is gedrukt
0010: dat is regel 0 ↑I schakelt om naar invoegen letters
0010: dat is niet regel 10
"niet" werd ingetikt en de tekst naar rechts verschoven
0010: dat is niet regel 0 ↑U kopieert tot aan het einde van de regel
0010: dat is niet regel 0 cr — zet de regel in het werkgeheugen
0011: cr ga terug in de kommando-mode
- de computer wacht op een nieuw kommando
-L(IST) cr toon de file op het scherm
0010: dat is niet regel 0
0020: dat is regel 10
0030: dat is regel xx
0040: dat is regel yy
0050: dat is regel 20
0060: dat is regel 30
0070: @ wacht op een nieuw kommando

=MOVE=
M(OVE)20,40 cr plaats regel 40 voor regel 20
M(OVE)20,40,60 cr plaats regel 40...60 voor regel 20
Met het MOVE-kommando kan de gebruiker een testregel voor een andere testregel plaatsen. Het is ook mogelijk een groep regels voor een bepaalde regel te plaatsen. Alle verplaatste regels krijgen het regelnummer 0000:. Na een MOVE-kommando kan men dus gemakkelijk zien of de regels op de juiste wijze werden verplaatst. Een NUMBER-opdracht nummert alle regels van de file in de juiste volgorde.
-M(OVE)20,40,60 cr plaats regel 40...60 voor regel 20
-L(IST) cr
0010: dat is niet regel 0
0000: dat is regel yy
0000: dat is regel 20
0000: dat is regel 30
0020: dat is regel 10
0030: dat is regel xx
0070: @
-N(UMBER) cr
-L(IST) cr
0010: dat is niet regel 0
0020: dat is regel yy
0030: dat is regel 20
0040: dat is regel 30
0050: dat is regel 10
0060: dat is regel xx
0070: @

=SCREEN=
S(CREEN) cr laat 16 regels zien en wacht tot een willekeurige toets wordt gedrukt
Met het SCREEN-kommando kan men een deel van de file op het beeldscherm oproepen, waarna de computer wacht op een toetsdruk om een volgend deel van de file te laten zien. Wordt nog een SCREEN-kommando gegeven, dan toont de computer weer de gehele file zoals na een LIST-opdracht. Het SCREEN-kommando werkt hier dus als een flipflop.

=WHERE=
W(HERE)10 cr de computer laat het eerste absolute adres van regel 10 zien. Ook de inhoud van die regel wordt getoond.

=END=
E(ND) cr de computer toont het eerste absolute adres van de laatste regel van de file.

2) De assembler

Op de diskette "SYSTEM LOYS" vinden we de Micro-Ware-assembler. Van de oorspronkelijke assembler hebben we de "parser- en analyzer-subroutines" gehandhaafd, maar het assembler-hoofdprogramma is geheel nieuw geschreven. Daardoor heeft de gebruiker op de Octopus 65 de mogelijkheid assemblerprogramma's in zeer korte tijd samen te stellen. Editor en assembler werken interaktief. Als de assembler bijvoorbeeld in een regel een programmafout ontdekt, dan wordt dat direkt gemeld en die regel met regelnummer getoond. Met het FIX-kommando kan de fout dan worden gekorrigeerd. Omdat bij de Octopus 65 uitermate zuinig wordt omgesprongen met de adresruimte van 64 Kbyte, mag een assembler-programma 44 Kbyte lang zijn. De assembler wordt vanuit de editor via een X-kommando opgeroepen. Aan het eind van dit hoofdstuk laten we zien hoe men met de assembler werkt.
Het programma dat de gebruiker via het keyboard of vanaf de diskette in de computer laadt, heet source code. De assembler vertaalt de source code in een voor de 6502-processor uitvoerbaar machinetaalprogramma of "object module". Voordat de assembler de source code omzet in een machinetaalprogramma kan de gebruiker kiezen vanaf welk geheugenadres het machinetaalprogramma moet worden opgeslagen. Wordt geen keuze gemaakt, dan slaat de assembler het machinetaalprogramma automatisch op vanaf het adres $D000. Indien zich bijvoorbeeld op het adres $D000 een EPROM-programmer bevindt, dan kan

Tabel 13.

Adresseringswijze "Immediate"

Micro-Ware	MOS-Technology	kommentaar
LDAIM $1A	LDA#$1A	laad de accumulator met $1A
LDXIM '?	LDX#'?'	laad het X-register met '?'

Adresseringswijze "Absolute"

Micro-Ware	MOS-Technology	kommentaar
LDA $E82C	LDA $E82C	laad accumulator met inhoud van $E82C
CMP OCTOP	CMP OCTOP	vergelijk accumulator met inhoud van OCTOP

Adresseringswijze "Zero Page"

Micro-Ware	MOS-Technology	kommentaar
LDA $00F2	LDA $F2	laad accumulator met inhoud van $00F2
LDAZ $00F2		identiek met LDA $00F2
LDA PTR LDAZ PTR	LDA PTR	$0 = PTR = $FF

Adresseringswijze "Absolute Indexed"

Micro-Ware	MOS-Technology	kommentaar
LDAX $E000	LDA $E000,X	laad accumulator met $E000 + X
ADCY OCTOP	LDA OCTOP,Y	voeg aan accumulator toe OCTOP + Y

Adresseringswijze "Zero Page Indexed"

Micro-Ware	MOS-Technology	kommentaar
LDXZY $00FE	LDX $FE,Y	laad X-register van $00FE + Y
STYZX PETER	STY PETER,X	sla het Y-register op in PETER + X
LDXY $00FE STYX PETER		$0 = PETER = $FF identiek met LDXZY en STYZX

Adresseringswijze "Indirect Indexed by Y"

Micro-Ware	MOS-Technology	kommentaar
LDAIY PTR	LDA (PTR),Y	laad accumulator van de geheugencel aangegeven door PTR + Y; PTR is een pointer in page zero

Adresseringswijze "Indexed Indirect by X"

Micro-Ware	MOS-Technology	kommentaar
LDAIX PETER	LDA (PETER,X)	laad de accumulator van de geheugencel aangegeven door "PETER"; X geeft pointer "PETER" aan in page zero

Adresseringswijze "Accumulator"

Micro-Ware	MOS-Technology	kommentaar
LSRA	LSR	schuif de accumulator een plaats naar rechts op

het machinetaalprogramma direkt in de EPROM worden geprogrammeerd.
De assembler vertaalt de source code in twee stappen. Eerst verzamelt de assembler alle door de gebruiker gedefinieerde symbolen, kent deze symbolen adressen toe en zet de symbolen met de adressen in de symbol stack. Daarna leest de assembler weer de gehele source code en geeft met behulp van de symbol stack elke machine-instruktie een adres en zo mogelijk een operand. Zo ontstaat het voor de 6502-processor uitvoerbare machinetaalprogramma. Indien gewenst kan de gebruiker het geassembleerde programma bekijken. Ook alle symbolen in de symbol stack kunnen worden opgeroepen.
Het programma dat de assembler vertaalt, moet in een bepaalde vorm in het geheugen zijn geplaatst. De regels moet men met behulp van de editor als volgt in het geheugen plaatsen.
cr nh nl Ø tot 64 karakters van de eerste regel
cr nh nl Ø tot 64 karakters van de tweede regel
... verdere regels
cr nh nl @ @ @ @... (laatste regel met file-eindmarkering)
Verklaring:
cr = $ØD
nh = meest signifikante deel van het regelnummer
nl = minst signifikante deel van het regelnummer
Ø tot 64 karakters = "source statement"
= $4Ø = eindmarkering van de file
De tussenruimtes tussen cr, nh, nl en het begin van de regel zijn hier ter verduidelijking gebruikt, in werkelijkheid mogen hier geen "spaces" worden geplaatst.
Een "source statement" is een programmaregel die uit ASCII-tekens bestaat. Een "program statement" is een deel van een source statement. Het program statement bestaat uit drie delen:
.mnemonic
.adresseringswijze
.operand
Bij het assembleren vertaalt de assembler een program statement in een machinetaalinstruktie. Een machinetaalinstruktie kan bij de 6502-processor één, twee of drie bytes lang zijn. De lengte van de instruktie wordt bepaald door het soort adressering. Het source statement bestaat uit vier kolommen:
LABEL
MNEMONIC-adresseringswijze
OPERAND
COMMENT
Opmerking: de kolommen moeten door spaties van elkaar worden gescheiden.
. LABEL = adresnaam
. MNEMONIC-adresseringswijze = 6502-instruktie en samenstelling van het adres van de operand
. OPERAND = byte dat door de adresseringswijze wordt geadresseerd
. COMMENT = kommentaar dat de gebruiker bij de instrukties van het programma kan geven

LABEL

geldig:	ongeldig:	
LOOP	LO OP	er zit een spatie in
OCTO	ptr	alleen hoofdletters zijn geldig

SIGMA	ASM	begint niet op de eerste positie van de regel
END	A*B	bevat een bijzonder teken
A	A12	betekent hetzelfde als AAB TST

Een LABEL kan maximaal uit 6 letters bestaan. Er mogen getallen in een label voorkomen, maar de assembler kan geen onderscheid maken tussen getallen en letters. Zo is voor de assembler SAMABC en SAM123 hetzelfde label. Symbolische adressen worden in gekomprimeerde vorm in de symbol stack geplaatst; daarvoor gebruikt de assembler alleen de vijf laagste bits van een ASCII-teken. Maar dat heeft het voordeel dat in de symbol stack, die maar 2,75 Kbyte lang is, ca. 470 symbolen een plaats kunnen krijgen.
Opmerking: een label staat altijd op de eerste positie van een regel. Wordt geen label gebruikt, dan moet die eerste positie een spatie bevatten.

MNEMONIC-adresseringswijze
Mnemonics beschrijven in korte, symbolische vorm de instrukties die de processor uitvoert. Bij de 6502-processor bestaan alle mnemonics uit drie hoofdletters. Alle instrukties die de processor kan uitvoeren, kunnen in enkele groepen worden verdeeld:
1) berekeningen: ADC-DEC-INC-SBC
2) laden/opslaan:
LDA-LDX-LDY-STA-STX-STY
3) booleaans: AND-BIT-EOR-ORA
4) vergelijken: CMP-CPX-CPY
5) verschuiven: ASL-LSR-ROL-ROR
6) springen: BCC-BCS-BEQ-BNE-BPL-BMI-BVC-BVS-JMP-JSR
7) diversen: BRK-CLC-SEC-CLD-SED-CLI-SEI-CLV-DEX-DEY-INX-INY-NOP-PHA-PLA-PHP-PLP-RTI-RTS-TAX-TXA-TAY-TYA-TSX-TXS
De adresseringswijze is, zoals hiervoor reeds gezegd, een deel van het program statement. De adresseringswijze wordt door een identifikatie vastgelegd. Deze identifikatie-marker kan direkt aan de mnemonics of aan de operand worden gekoppeld. De Micro-Ware-assembler koppelt de identifikatie-marker direkt achter de mnemonics, terwijl de MOS-technology-assembler ze aan de operands koppelt. Daardoor zien de program statements van de beide assemblers er iets verschillend uit. De voorbeelden in tabel 1 laten duidelijk het verschil zien.
Opmerking: alle in tabel 1 gebruikte namen zijn symbolische adressen!

OPERAND
De operand-kolom moet van de mnemonic-adresseringswijze-kolom worden gescheiden door een space. De operand-kolom kan een symbool, een ASCII-teken of een hexadecimaal getal bevatten.
Opmerking: Instrukties met de adresseringswijze "implied" (NOP, DEX, RTS, enz.) hebben geen operand-kolom nodig. Volgt na zo'n instruktie een comment-kolom, dan moet die comment-kolom door twee spaces van de mnemonic-adresseringswijze-kolom gescheiden zijn.

COMMENT
De comment-kolom is voor de assembler zelf van geen betekenis. Zij dient alleen voor de dokumentatie van een programma. Als een program statement slechts uit een comment-kolom bestaat, moeten voor die kolom drie spaties worden geplaatst zodat de assembler de comment-kolom van de overige kan onderscheiden.

Assembler-pseudo-instrukties
Pseudo-instrukties leiden de assembler door het programma. De Micro-Ware-assembler bezit slechts enkele, maar wel zeer krachtige pseudo-instrukties. De speciale tekens voor deze instrukties zijn: "ORG", "*", "=", "/", "+", "–".
ORG $D000
Het te assembleren programma krijgt als startadres $D000.
Let op de spatie vóór ORG!

PETER * $C000	geeft het adres $C000 aan het symbool PETER
SAM * PETER +05	geeft het adres $C005 aan het symbool SAM
PTR * SAM –02	geeft het adres $C002 aan het symbool PTR

TABLE = $0D	sla in geheugencel Table het byte $0D op
= $0A	sla in geheugencel Table+1 het byte $0A op
= 'E	sla in geheugencel Table+2 het teken 'E' op
= 'N	sla in geheugencel Table+3 het teken 'N' op

= 'D sla in geheugencel Table+4 het teken 'D' op
Zeer eenvoudig kunnen de door de assembler berekende adressen van symbolen als operands worden gebruikt. De micro-ware-assembler kent geen .BYTE-, .WORD- of .DBYTE-pseudo-instrukties. Het volgende voorbeeld toont aan dat die instrukties ook niet nodig zijn! Laten we aannemen dat de assembler het symbool TABLE uit het vorige voorbeeld het adres $D32F heeft. Het byte $0D bevindt zich dus op adres $D32F. Wil men in page zero het symbool TABLE laten aanwijzen, dan moet men als volgt handelen:

DEMO	LDAIM	TABLE laad accumulator met minst significante adresbyte

Fotoserie, figuur 9. De kap erop, het keyboard aangesloten, monitor op de kast, printer ernaast en klaar staat ons komplete systeem.

```
STAZ PTR
LDAIM TABLE /256    laad accumulator met meest signifikante adresbyte
STAZ PTR +Ø1
```

Het register met het adres PTR in page zero wijst nu het symbool TABLE aan. De gebruiker behoeft het adres van TABLE niet te kennen omdat de assembler dat automatisch definieert.

3) De symbolische disassembler

De in de assembler geïntegreerde disassembler roept diverse routines van de assembler op. Het is dus geen gebruikelijke disassembler, maar een "intelligente" disassembler. De disassembler wordt met de Z-opdracht door de editor opgeroepen:

Z F000,F050 disassembleer van $F000 tot $F050

Werd daarvoor een S-opdracht gegeven, dan worden 16 instrukties gedisassembleerd. Het indrukken van een willekeurige toets start de disassembler voor de volgende 16 instrukties.

1a

```
.D

DISASM: C000,C00A

^D,^L,^P,^S ?

C000: 18          CLC
C001: AD 00 D0    LDA $D000
C004: 6D 01 D0    ADC $D001
C007: 8D 02 D0    STA $D002
C00A: 00          BRK

^D,^L,^P,^S ?
```

1b

```
C000: 18          CLC
C001: AD 00 D0    LDA $D000
C004: 6D 01 D0    ADC $D001
C007: 8D 02 D0    STA $D002
C00A: 00          BRK
```

1c

```
: .OS65D
D*AS


*ASM*


NEW?

-Z C000,C010

C000 18       ADD   CLC
C001 AD 00 D0       LDA   PETER
C004 6D 01 D0       ADC   KARIN
C007 8D 02 D0       STA   SOM
C00A 00             BRK
C00B 40             RTI
C00C 40             RTI
C00D 40             RTI
C00E 40             RTI
C00F 40             RTI
-
```

De disassembler kan met de symbol stack van de assembler werken. De gebruiker kan met de editor een file met symbolen samenstellen en daarna op de diskette wegschrijven. Zo'n file kan er bijvoorbeeld zo uitzien:

```
PETER * $D000
KARIN * $D001
SOM * KARIN +01
ADD * $C000
```

Laad met de monitor van de Octopus 65 de volgende bytes in de computer:

C000 space
C000 XX @ willekeurige data bij $C000; schakel de snel-intoets-mode in
C000 18 AD 00 D0 6D 01 D1 8D 02 D0 00 @

(@ = einde van de snel-intoetsmode)

Figuur 1a laat zien hoe dit programma er uitziet, als u het met een gewone disassembler disassembleert (bijv. met de disassembler in de monitor van de Octopus 65). Figuur 1b toont hoe de intelligente disassembler het hierboven ingetikte programma disassembleert als er geen symbolen in de symbol stack van disassembler staan. Worden echter de symbolen PETER, KARIN, enz. met de editor in een file opgenomen en start men daarna de assembler met de X-opdracht, dan staan die symbolen in de symbol stack en kan de disassembler ze gebruiken. Figuur 1c laat zien hoe de symbolische disassembler met behulp van de symbol stack een programma overzichtelijk disassembleert. Door gericht gebruik van symbolen kan men bij het ontrafelen van software dus zeer overzichtelijke disassembly-listings verkrijgen. Probeer het maar eens met de OHIO-DOS; het hoofdstuk "Waar vinden we wat in de DOS" is daarbij een goede hulp!

4) De DOS en de editor

Heeft de gebruiker een programma in de computer getypt, dan verdient het aanbeveling dit op diskette op te slaan. De editor heeft kommando's waarmee het programma met een naam op de diskette kan worden geschreven of van de diskette in de computer kan worden geladen.

Het K(ERNEL)-kommando vertelt de grootte van het programma en hoeveel tracks op de diskette voor het programma worden gereserveerd. Figuur 2 laat zien hoe de Octopus 65 op de K-opdracht reageert. Met B* geeft de computer aan dat hij op een DOS-opdracht wacht. Wil men bijv. een programma met de naam PETER op diskette opslaan, dan tikt men achter B* in:

B*PUT PETER cr

Dit kommando zet het programma met de naam PETER op de diskette. De DOS schrijft de gekozen naam in de directory van de diskette. De DOS beantwoordt het kommando weer met B*. Nu geeft men een DI-kommando (directory-kommando):

B*DI cr

De DOS geeft nu een overzicht van de totale inhoud, zodat men kan kontroleren of de nieuw ingegeven programmanaam inderdaad is opgenomen. Wanneer men nu bijvoorbeeld een tweede programma met de naam KARIN op de diskette wil saven, tikt men:

B*PUT KARIN cr

2

```
-K


---DOS COMMANDS---

 KERNEL
 .LINK
 PUT FN

SOURCE FILE: 4000 - 4046
REQUIRED TRACK(S): 01
B*
```

3

```
-.LINK

HOW MANY FILES
DO YOU WANT TO LINK?  <MAX 9>1


ENTER A FILE NAME:  MUL

FOR *MUL* IS (ARE) 01 TRACK(S)
                          ON FLOPPY
SOURCE FILE: 4000 - 4304
REQUIRED TRACK(S): 01
MUL  IS LOADED

-
```

Beide programma's zijn nu op de diskette in drive B opgeslagen. Het laden van programma's vanaf de diskette is eveneens eenvoudig. De editor heeft hiertoe de opdracht ".LINK". De punt voor het woord LINK moet ook ingetikt worden. Zoals het woord LINK reeds uitdrukt kan met deze opdracht een of meerdere programma's na elkaar van de diskette in het geheugen van de Octopus 65 geladen worden. Figuur 3 toont hoe de Octopus 65 op een .LINK-opdracht reageert. Op de vraag "How many files would you like to link?" of "hoeveel programma's wilt u in het geheugen koppelen?" kan men met 1...9 antwoorden. Wenst men slechts één programma van de diskette te laden, dan antwoordt men met 1 cr; wil men twee programma's aan elkaar koppelen, dan antwoordt men met 2 cr.

Er bestaat ook een opdracht om van de editor direkt in de DOS te springen:

.* cr

Na deze opdracht meldt de DOS zich met A*. B* enz. afhankelijk van de op dat moment gekozen disk drive.

Vanuit de DOS kan weer BASIC, de assembler of de tekstverwerker worden gekozen:

B*RE M spring naar de monitor van de Octopus 65 (programma in de EPROM).
B*AS keer naar de assembler terug. Laad zonodig de assembler.
B*BA keer naar BASIC terug. Laad zonodig BASIC.
B*WP keer naar de tekstverwerker terug (wordprocessor = WP). Laad zonodig de tekstverwerker.

Het is ook mogelijk een naam in de inhoud te wissen. De E(RASE)opdracht zorgt hiervoor. Wil men bijvoorbeeld de naam van de file "KARIN" in de inhoud wissen, dan tikt men in:

B*ER KARIN

4

```
-L
0010:  ORG $D000
0020:
0030:
0040:    ********************
0050:    *8X8 MULTIPLICATION*
0060:    ********************
0070:
0080:
0090: MA * $0010
0100: MB * MA +01
0110: RES * MB +01
0120:
0130:
0140:
0150: MULPLY PLA  GET RETURN ADDRESS FROM STACK
0160:  STA RETADR
0170:  PLA
0180:  STA RETADR +01
0190:
0200:  PLA  GET MULTIPLIER FROM STACK
0210:  STA MB AND SAVE IT
0220:  PLA  GET MULTIPLICAND FROM STACK
0230:  STA MA AND SAVE IT
0240:
0250: MUL LDAIM $00 RESET RESULT BUFFER
0260:  STA RES
0270:  STA RES +01
0280:  LDYIM $08
0290:
0300: MULA LSR MA
0310:  BCC MULB
0320:  CLC
0330:  LDA MB
0340:  ADC RES +01
0350:  STA RES +01
0360:  LDAIM $00
0370:  ADC RES
0380:  STA RES
0390:
0400: MULB LSR RES +01
0410:  ROR RES
0420:  DEY  DECREMENT BIT COUNTER
0430:  BNE MULA
0440:
0450:  LDA RETADR +01 PUSH RETURN ADDRESS ON STACK
0460:  PHA
0470:  LDA RETADR
0480:  PHA
0490:
0500:  RTS
0510:
0520:
0530: RETADR = $00 STORE RETURN ADDRESS HERE
0540:  = $00
0550:
0560:
0570: @

-
```

5

```
JUNIOR'S ASSEMBLER          PAGE 01


0010: D000                  ORG   $D000
0020:
0030:
0040:                ********************
0050:                *8X8 MULTIPLICATION*
0060:                ********************
0070:
0080:
0090: D000           MA     *     $0010
0100: D000           MB     *     MA     +01
0110: D000           RES    *     MB     +01
0120:
0130:
0140:
0150: D000 68        MULPLY PLA          GET RETURN ADDRESS FROM STACK
0160: D001 8D 37 D0         STA   RETADR
0170: D004 68               PLA
0180: D005 8D 38 D0         STA   RETADR +01
0190:
0200: D008 68               PLA         GET MULTIPLIER FROM STACK
0210: D009 85 11            STA   MB    AND SAVE IT
0220: D00B 68               PLA         GET MULTIPLICAND FROM STACK
0230: D00C 85 10            STA   MA    AND SAVE IT
0240:
0250: D00E A9 00     MUL    LDAIM $00   RESET RESULT BUFFER
0260: D010 85 12            STA   RES
0270: D012 85 13            STA   RES   +01
0280: D014 A0 08            LDYIM $08
0290:
0300: D016 46 10     MULA   LSR   MA
0310: D018 90 0D            BCC   MULB
0320: D01A 18               CLC
0330: D01B A5 11            LDA   MB
0340: D01D 65 13            ADC   RES   +01
0350: D01F 85 13            STA   RES   +01
0360: D021 A9 00            LDAIM $00
0370: D023 65 12            ADC   RES
0380: D025 85 12            STA   RES
0390:
0400: D027 46 13     MULB   LSR   RES   +01
0410: D029 66 12            ROR   RES
0420: D02B 88               DEY         DECREMENT BIT COUNTER
0430: D02C D0 E8            BNE   MULA
0440:
0450: D02E AD 38 D0         LDA   RETADR +01 PUSH RETURN ADDRESS ON STACK
0460: D031 48               PHA
0470: D032 AD 37 D0         LDA   RETADR
0480: D035 48               PHA
0490:
0500: D036 60               RTS
0510:
0520:
0530: D037 00        RETADR =     $00    STORE RETURN ADDRESS HERE
0540: D038 00               =     $00
0550:
0560:
-T

    SYMBOL TABLE 3400 3430
    MA     0010    MB     0011    MULA   D016    MULB   D027
    MULPLY D000    MUL    D00E    RES    0012    RETADR D037
-T1

    SYMBOL TABLE 3400 3430
    MA     0010    MB     0011    RES    0012    MULPLY D000
    MUL    D00E    MULA   D016    MULB   D027    RETADR D037
-
```

De ruimte die de zojuist gewiste file op de diskette innam is nu vrij voor andere files of programma's.
Vanuit de editor kan men ook rechtstreeks naar de monitor van de Octopus 65 springen. De opdracht luidt ".M cr".
Zo, nu zijn alle opdrachten van de editor en de assembler bekend. We gaan nu aan de hand van een 8-bit-maal-8-bit-vermenigvuldiging aantonen hoe eenvoudig en snel machinetaalprogramma's op de Octopus 65 kunnen worden geschreven. Vóór de assembler een programma kan assembleren moet dit eerst met de editor in het geheugen van de Octopus 65 worden geladen. Figuur 4 laat zien hoe het vermenigvuldigingsprogramma er uit ziet.

Bij het intikken moet men elke regel met cr (RETURN-toets) afsluiten. Let er op, dat na mnemonics die geen operand hebben (implied addressing) twee spaties moeten worden gegeven! Een regel die slechts één kommentaar bevat, begint met drie spaties. Een lege regel bevat slechts één cr-teken. In de regels 530 en 540 worden twee bytes voor de variabele RETADR gereserveerd. Tik het programma van figuur 4 eens in uw Octopus 65 en zet het daarna onder de naam "MUL8*8" op de diskette.
Het vermenigvuldigingsprogramma kan men nu met de assembler assembleren. De X-opdracht van de editor start de assembler, die in twee "runs" het ingegeven programma in een voor de processor uitvoerbaar programma (object module) vertaalt. Na de eerste "run" (pass 1) meldt de assembler zich om op eventuele syntax-fouten te wijzen. De tweede "run" (pass 2) begint zodra men de cr-toets drukt, waarbij de assembler vraagt of een listing moet worden gemaakt. Antwoordt men op de vraag "LISTING?" met "Y(ES) cr", dan produceert de assembler een listing zoals in figuur 5 is te zien. Moet de listing ook geprint worden via de Centronics-uitgang, dan moet men voor de start de volgende opdracht geven:
-.* cr spring naar de DOS
B*IO, 09 cr schakel beeldscherm en Centronics in.
B*AS cr keer terug naar de editor
-X cr start de assembler

Zijn in het programma fouten geslopen, dan geeft de assembler de foutieve regels aan. Met de F(IX)-opdracht kan men de fouten vlug korrigeren. De assembler geeft de volgende foutmeldingen:

F6 bij een relatieve sprong (BCC, BEQ, enz.) is de beschikbare ruimte overschreden.

E2 er werd een niet bestaande 6502-adresseringswijze gekozen.

A8 een symbool is niet gedefinieerd.

6F een symbool is tweemaal gebruikt.

23 de adresseringswijze past niet bij de mnemonic.

07 de 6502-processor kent de gebruikte instruktie niet.

De tracer

Als men een assembler-programma schrijft, zal dit meestal niet onmiddellijk goed lopen. Vaak geeft het programma allerlei onzin en in het ergste geval stort het programma in elkaar (hang up). De computer moet dan met de RESET-toets worden gestart.

Om zulke fouten op te kunnen sporen, heeft de Octopus 65 in zijn EPROM een software-pakket, genaamd "tracer-software". De tracer-software is een van de krachtigste systeemprogramma's die thans op 8-bit-computers voor het opsporen van programmafouten bestaan.

De tracer bestaat uit twee delen:

1) tracer-hardware

2) tracer-software

Figuur 1 toont het schema van de tracer. N1 is een adresdekoder, waarvan de uitgang in het adresbereik $F000...$FFFF logisch nul is. NAND-poort N2 krijgt het sync-signaal van de CPU en Φ2. N2 wordt door de adresdekoder N1 of door de flipflop FF2 geblokkeerd. De adresdekoder blokkeert N2 als de EPROM van de Octopus 65 door de CPU wordt geadresseerd. Daardoor kan de sync-lijn van de 6502-processor geen "non maskable interrupt" geven. Doorloopt de processor ergens in de RAM een machinetaalprogramma, dan geeft N2 een interrupt zodra de processor een opcode leest, omdat de sync-lijn logisch één wordt. De NMI voert de 6502-processor naar een interruptprogramma in het monitorprogramma. In dit interrupt-programma worden alle CPU-registers opgeslagen en daarna op het beeldscherm getoond. Tegelijk disassembleert de processor de volgende uit te voeren instruktie. De gebruiker kan dus vóór het uitvoeren van een instruktie precies zien onder welke omstandigheden de processor met de uitvoering van de volgende opcode begint. Omdat ook het status-register met alle flags op het beeldscherm verschijnt, kan de gebruiker gemakkelijk zien of een flag foutief is geset in het status-register.

In het onderste deel van het schema in figuur 1 staan twee flipflops. FF1 werkt als anti-denderschakeling en FF2 klapt om zodra hij een positieve flank van FF1 ontvangt. Als de tracer is ingeschakeld brandt de rode LED. R4 en C1 zorgen voor een "power-on reset" en schakelen de tracer uit als de Octopus 65 wordt ingeschakeld.

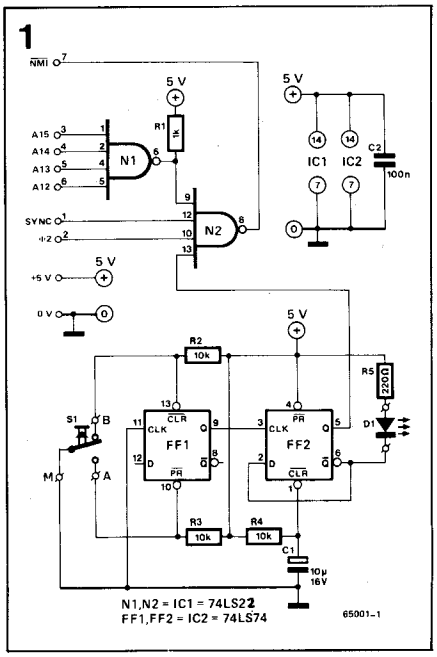
We willen nu laten zien hoe men met de tracer kan werken. Daartoe assembleren we met de assembler het programma "MUL8*8, dat we tevoren op de diskette hebben gezet. We laden dus de assembler en aansluitend het programma "MUL8*8". Met de X-opdracht wordt de assembler gestart. Het door de assembler geproduceerde machinetaalprogramma plaatsen we op het adres $D000, (bij elke vraag van de assembler een cr geven). Neem nu de listing van het vermenigvuldigingsprogramma en bestudeer dat. Zoals men gemakkelijk zal herkennen, worden eerst de terugkeer-adressen en aansluitend de twee te vermenigvuldigen getallen van de stack gehaald. De stack geeft dus de parameter aan het vermenigvuldigingsprogramma door. Zodra de parameters van de stack zijn gehaald, kan de processor er mee werken. Aan het eind van de vermenigvuldiging zendt de processor de terugkeer-adressen weer naar de stack terug en dè dan volgende RTS-opdracht stuurt de 6502-processor weer terug naar het hoofdprogramma. Vóór men het vermenigvuldigingsprogramma start, dient men de te vermenigvuldigen getallen op de stack te plaatsen. Figuur 2 laat zien hoe de getallen 5 en 3 worden vermenigvuldigd. De uitkomst van de vermenigvuldiging ($0F = 15) vindt men in de geheugenposities $11 en $12.

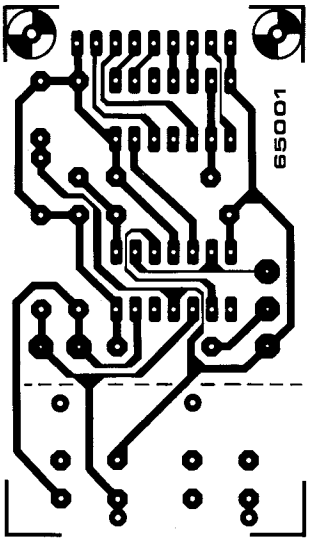
We geven nu met de monitor het kleine demoprogramma van figuur 2 in. We disassembleren daarna het ingetikte programma, om er zeker van te zijn dat er geen fouten in zitten. We schakelen de tracer-hardware met S1 in. De rode LED moet nu branden. Stel met de monitor het startadres $C000 in en initialiseer met de T-opdracht de tracer-software. Met ↑S (step) kan men van instruktie tot instruktie stappen. Met ↑P worden 16 instrukties achter elkaar afgewerkt en met ↑L wordt het gehele programma doorlopen totdat een BRK-instruktie verschijnt. Figuur 2 laat in het kort de werkwijze van de tracer zien. In de geheugenposities $12 en $13 is, zoals te verwachten was, het resultaat van de vermenigvuldiging korrekt gegeven: $000F. Het resultaat is een 16-bit-getal.

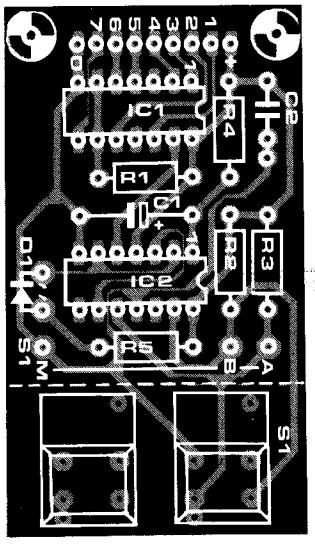
2

```
D000
: D000 A9 T
^L.^P.^S ?

A  X  Y  SP       PC   NV BDIZC
1F 06 5F 01FF+1=D8 D002 00100100   D002: 48          PHA
1F 06 5F 01FE+1=1F D003 00100100   D003: A9 2C       LDA #$2C
2C 06 5F 01FE+1=1F D005 00100100   D005: 48          PHA
2C 06 5F 01FD+1=2C D006 00100100   D006: 20 0C D0    JSR $D00C
2C 06 5F 01FB+1=00 D00C 00100100   D00C: 68          PLA
00 06 5F 01FC+1=D0 D00D 00100100   D00D: 8D 43 D0    STA $D043
00 06 5F 01FC+1=D0 D010 00100100   D010: 68          PLA
D0 06 5F 01FD+1=2C D011 10100100   D011: 8D 44-D0    STA $D044
D0 06 5F 01FD+1=2C D014 10100100   D014: 68          PLA
2C 06 5F 01FE+1=1F D015 00100100   D015: 85 11       STA $11
2C 06 5F 01FE+1=1F D017 00100100   D017: 68          PLA
1F 06 5F 01FF+1=D8 D018 00100100   D018: 85 10       STA $10
1F 06 5F 01FF+1=D8 D01A 00100100   D01A: A9 00       LDA #$00
00 06 5F 01FF+1=D8 D01C 00100110   D01C: 85 12       STA $12
00 06 5F 01FF+1=D8 D01E 00100110   D01E: 85 13       STA $13
00 06 5F 01FF+1=D8 D020 00100110   D020: A0 08       LDY #$08
00 06 08 01FF+1=D8 D022 00100100   D022: 46 10       LSR $10

A  X  Y  SP       PC   NV BDIZC
00 06 08 01FF+1=D8 D024 00100101   D024: 90 0D       BCC $D033
00 06 08 01FF+1=D8 D026 00100101   D026: 18          CLC
00 06 08 01FF+1=D8 D027 00100100   D027: A5 11       LDA $11
2C 06 08 01FF+1=D8 D029 00100100   D029: 65 13       ADC $13
2C 06 08 01FF+1=D8 D02B 00100100   D02B: 85 13       STA $13
2C 06 08 01FF+1=D8 D02D 00100100   D02D: A9 00       LDA #$00
```

1



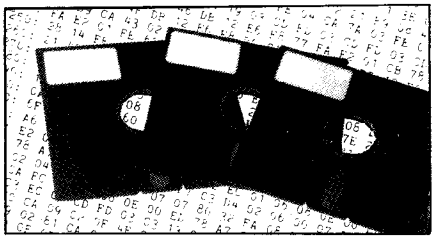




Tenslotte schakelen we nogmaals met de monitor naar het adres $C000; ook de tracer is ingeschakeld (rode LED brandt). We geven nu de W-opdracht (WHERE). De Octopus 65 geeft nu op het beeldscherm alle adressen waar een opcode bij de uitvoering van de vermenigvuldiging staat. Met deze W-opdracht kunnen de volgende programmafouten gemakkelijk worden gevonden:

* eindeloze lus: de processor blijft in een programmalus lopen.
* ongeoorloofde sprongen in een programma.
* ongeoorloofde input/output-procedures.

Opmerking: Dit overzicht was kort maar krachtig, wegens gebrek aan plaatsruimte. In de volgende computer-special meer over de tracer.



Waar vinden we wat in de DOS?

Het Disk Operating System (DOS) is modulair opgebouwd. Op de Octopus 65 zijn twee DOS-versies beschikbaar: OS-65D V3.2 en OS-65D V3.3. De assembler, de tekstverwerker en de OSI-tutorial-disks 2 tot 4 gebruiken de oudere maar kompaktere versie 3.2. OSI-tutorial-disk 5 werkt met V3.3, terwijl de door ons ontwikkelde diskette "SYSTEM LOYS" met beide versies werkt. De gebruiker merkt geen verschil tussen de DOS-versies, zolang hij slechts kant-en-klare programma's draait. Bij het schrijven van BASIC-programma's moet echter V3.3 worden gebruikt, omdat op de diskette "SYSTEM LOYS" een full-screen-editor en het steno-BASIC ter beschikking staan. Ook bij toepassing van sekwentiële en random files moet V3.3 worden toegepast, omdat de diskette-verwerkingstijden veel korter zijn.
De DOS is hiërarchisch opgebouwd en bestaat uit de volgende delen:

1) BIOS = Basic Input Output System
2) BDOS = Basic Disk Operating System
3) CCP = Console Command Processor
4) TPA = Transient Processor Area
5) BOOT = parameter bij het opstarten van het systeem

1) BIOS

De BIOS ontvangt alle karakters die in het geheugen van de computer moeten worden opgeslagen en verzendt alle karakters die van de computer naar alle aangesloten randapparatuur worden gezonden. De BIOS herkent de in tabel 1 genoemde invoer-apparatuur en de in tabel 2 genoemde uitvoer-apparatuur. De BIOS verwerkt alle input- en output-operaties via 2 input/output-bytes en 2 device-tables. Het I/O-byte voor input bevindt zich op het adres $2321 en het I/O-byte voor output op adres $2322. In het I/O-byte voor input mag slechts één bit worden geset, in het I/O-byte voor output echter meerdere. Zoals reeds in de device-tabellen te zien is, is aan elk bit in het I/O-byte een apparaat toegewezen. Apparaat 1 korrespondeert met bit 0, apparaat 2 met bit 1, enz. Worden bijvoorbeeld in het I/O-byte voor output bit 0 en bit 3 geaktiveerd, dan worden het beeldscherm en de Centronics-printeruitgang gelijktijdig van output voorzien. Voor elk input/output-apparaat zijn in de devicelijst 2 bytes gereserveerd voor het sprongadres van de bijbehorende subroutine. De video-output-routine bevindt zich bijvoorbeeld in de Octopus-EPROM op $F78C. Op de adressen $2311, $2322 moet $d8B, $F7 worden geplaatst om een weer te geven karakter aan de videoroutine door te geven. $F78B=$F78C-1 moet hier worden gebruikt, omdat de input/output-routines niet via een JSR-instruktie worden opgeroepen, maar via een RTS-instruktie (het sprong-adres wordt voor de oproep van de subroutine op de stack geplaatst).

Tabel 1.

input	keyboard, parallel aangesloten	apparaat 1 — $2301
	keyboard, serieel aangesloten	apparaat 2 — $2303
	modem, serieel aangesloten	apparaat 3 — $2305
	nul-input	apparaat 4 — $2307
	input byte voor byte van geheugen	apparaat 5 — $2309
	input van sekwentiële disk-buffer	apparaat 6 — $230B
	input van random disk-buffer	apparaat 7 — $230d
	algemene seriële ingang	apparaat 8 — $230F

Tabel 2.

output	beeldscherm-output	apparaat 1 — $2311
	beeldscherm-output (vrij voor gebruiker)	apparaat 2 — $2313
	printer, serieel aangesloten	apparaat 3 — $2315
	Centronics-printer (parallel)	apparaat 4 — $2317
	output byte voor byte naar geheugen	apparaat 5 — $2319
	output naar sekwentiële disk-buffer	apparaat 6 — $231B
	output naar random disk-buffer	apparaat 7 — $231D
	algemene seriële uitgang	apparaat 8 — $231F

BIOS-subroutine DOINP * $2339
JSR DOINP
Ontvangt een karakter van een input-apparaat. Het karakter staat in de accumulator en in geheugenplaats AHOLD= $2363. Het karakter wordt niet op het beeldscherm weergegeven.

BIOS-subroutine INECHO * $2340
JSR INECHO
Werkt op gelijke wijze als DOINP, maar het karakter van het input-apparaat wordt wel op het beeldscherm weergegeven.

BIOS-subroutine PRINT * $2343
JSR PRINT
Zendt het karakter in de accumulator naar alle in het I/O-byte ($2322) geselekteerde output-apparaten. De accumulator en de beide indexregisters worden op de stack opgeslagen.

BIOS-subroutine MEMIN * $2389
JSR MEMIN
Deze routine wordt bij input-apparaat 5 en door de BASIC bij het aftasten van indirekte files gebruikt. Op de adressen $238A en $238B bevindt zich een 16-bits pointer, (IPTRL, IPTRH). Deze pointer moet voor het oproepen van de routine op dat adres in het geheugen worden gezet, waar de karakters moeten worden ontvangen.

BIOS-subroutine MEMOUT * $2390
JSR MEMOUT
Heeft dezelfde funktie als MEMIN, maar dan voor verzenden van karakters uit het geheugen van de computer. De output-pointer moet voor het aanroepen van de routine worden gezet. Deze pointer bevindt zich op de adressen $2891 en $2892 (OPTRL, OPTRH).

BIOS-subroutine DK1IN * $23A1
JSR DK1IN
Deze routine leest sekwentieel data uit de disk-buffer. Programma's die meer data bevatten dan in het geheugen van de computer kunnen worden opgeslagen, gebruiken indirekte files. Denk maar eens aan een verkiezing voor de tweede kamer, als met een computer de berekeningen worden gemaakt van het aantal uitgebrachte stemmen. Lang niet alle data

kunnen tegelijk in het geheugen van de computer worden opgeslagen. Men gebruikt dan externe geheugens, zoals een diskette. Alle binnenkomende gegevens worden sekwentieel op de diskette opgeslagen. Met de DK1IN-routine ziet de Octopus 65 data die op de diskette staan op gelijke wijze alsof ze zich in het geheugen van de computer zouden bevinden. De file op de diskette mag vele tracks lang zijn; bij het lezen van de file van de diskette zorgt DK1IN er voor dat de leespointer steeds naar het juiste byte op het juiste spoor van de diskette wijst! De laadpointer wordt automatisch door BASIC ingesteld. Als men in een eigen programma een "disk open"-funktie wil inbouwen, dan moet de laad-pointer voor het oproepen van de routine worden ingesteld (adres $23AC, $23AD = SLPTRL, SLPTRH).

BIOS-subroutine DK1OUT * $23B2
JSR DK1OUT
Deze routine schrijft data sekwentieel in de disk-buffer en werkt tegengesteld aan DK1IN. Het teken "line feed" (= $0A) wordt bij het schrijven onderdrukt. Wil men line feeds schrijven, dan moet men het byte op het adres $23B4 vervangen door $00. Als BASIC deze routine gebruikt, mag deze wijziging niet worden gemaakt! De schrijf-pointer wordt automatisch door BASIC ingesteld. Wil men in een eigen programma een "disk close"-funktie inbouwen, dan moet de schrijfpointer voor het oproepen van de routine met DK1OUT worden ingesteld (adres $23C3, 23C4 = SWPTRL, SWPTRH).

BIOS-subroutine DK2IN * $23F0
(Apparaat 7, V3.2) Deze routine leest data uit indirekte random files en mag uitsluitend door BASIC worden opgeroepen. Het is binnen het raam van deze computer-special onmogelijk de komplexe machinetaalroutines van een random file te beschrijven.

BIOS-subroutine DK2OUT * $2403
(Apparaat 7, V3.2) Deze routine schrijft data in indirekte random files en is dus de tegenhanger van DK2IN.

BIOS-subroutine VIDEO * $F000
STA AHOLD
JSR VIDEO
Deze routine toont het karakter in de geheugencel AHOLD ($2363) op het beeldscherm. De beeldscherm-pointer wordt automatisch met één verhoogd. Wordt de regel te lang, dan maakt VIDEO een cr en lf.

BIOS-subroutine RESET * $F32F
JSR RESET
De routine VIDEO ziet het videogeheugen (E800...EFFF) als "wrap around"-geheugen. Daarom verandert het adres van het karakter, dat links boven op het beeldscherm staat, telkens. Het oproepen van de subroutine RESET heeft tot gevolg dat de CRT-controller 6845 geïnitialiseerd wordt en dat het eerste karakter links boven in het beeld op adres $E800 blijft staan totdat een scroll-up of scroll-down volgt.

BIOS-subroutine INIKBD * $F707
JSR INIKBD
De routine INIKBD initialiseert de poortlijnen waarop het parallelle keyboard is aangesloten.

BIOS-subroutine RECCHA * $F71D
JSR RECCHA
De routine RECCHA ontvangt een karakter van het keyboard. Het ontvangen karakter wordt in de accumulator en in de geheugencel AHOLD ($2363) opgeslagen.

BIOS-subroutine CTLCMD * $F72D
JSR CTLCMD
De routine CTLCMD leest de kortsluitstekers op de CPU-kaart, waarmee het control- en het command-register in de ACIA 6551 worden geprogrammeerd.

BIOS-subroutine INICEN * $F6F1
JSR INICEN
De routine INICEN initialiseert de I/O-lijnen voor de Centronics-interface.

BIOS-subroutine CENTRO * $F7B0
JSR CENTRO
De routine CENTRO zendt het karakter in de accumulator naar de Centronics-interface.

BIOS-subroutine ACIIN * $F770
JSR ACIIN
De routine ACIIN ontvangt een karakter van de ACIA (seriële ingang). Het ontvangen karakter staat in de accumulator en in de geheugencel AHOLD ($2363).

BIOS-subroutine ACIOUT * $F77E
JSR ACIOUT
De routine ACIOUT zendt een karakter naar de ACIA (seriële uitgang). Het te verzenden karakter moet in de accumulator staan.

BIOS-subroutines
BASVID * $F78C
BASCEN * $F7A4
BASACI * $F7B0
Deze drie subroutines worden door BASIC opgeroepen. BASVID zendt een karakter naar het beeldscherm, BASCEN naar de Centronics-interface en BASACI naar de ACIA 6551 (seriële uitgang). Deze routines kontroleren of tijdens de uitgave van een karakter ↑C werd gedrukt. Indien ja, dan wordt ↑C = $03 in de geheugencel KPDO*$2325 opgeslagen en de verzending van verdere karakters naar de momenteel geselekteerde output-apparaten gestopt; BASIC meldt zich dan met "BREAK" of "BREAK IN LINE...".

2) BDOS

Het Basic Disk Operating System zendt data die in het geheugen van de Octopus 65 staan naar de diskette of leest data die op de diskette staan in het geheugen. De BDOS werkt niet, zoals de BIOS, met enkele bytes, maar met bytes die tot blokken zijn samengevoegd. De kleinste dataeenheid die de BDOS kent is 256 bytes.

BDOS-subroutine HOME * $2663
JSR HOME
Dit programma plaatst de schrijf/leeskop van de disk drive op track zero (track 0).

BDOS-subroutine STEPIN * $2683
JSR STEPIN
Dit programma beweegt de schrijf/leeskop één stap in de richting van track nul.

BDOS-subroutine STEPOT * $268A
JSR STEPOT
Dit programma beweegt de schrijf/leeskop van de disk drive één stap in de richting van track 39 (of track 79).

BDOS subroutine SETTK * $26BC
LDA $27
JSR SETTK
Dit programma beweegt de schrijf/leeskop van de disk drive naar de track die in de accumulator staat. Het programma kontroleert of in de accumulator een toegestaan tracknummer staat en kan twee foutmeldingen geven:
* ERR6 disk drive is niet bedrijfsklaar
* ERR8 foutief tracknummer
In dit voorbeeld wordt de schrijf/leeskop op track 27 geplaatst.

BDOS-subroutine LDHEAD * $2754
JSR LDHEAD
Dit programma plaatst de schrijf/leeskop van de disk drive op het oppervlak van de diskette.

BDOS-subroutine UNLDHD * $2761
JSR UNLDHD
Dit programma trekt de schrijf/leeskop van de disk drive weer van het oppervlak van de diskette.

BDOS-subroutine INIT * $2768
LDA $14
STA TKNUM
JSR INIT
Dit programma initialiseert een track op de diskette. In het voorbeeld wordt track 14 geïnitialiseerd. Bij het initialiseren worden de volgende gegevens op de diskette geschreven: enkele stuurbytes (track heading), het track-nummer (track number), 8 maal 256 = 2048 bytes nullen en enkele stuurbytes die het eind van de track (track trailing) aangeven. Het gewenste track-nummer moet vóór de oproep van INIT in de geheugencel TKNUM ($265D) geplaatst zijn. INIT kan twee foulmeldingen geven:
* ERR3 track nul (zero) mag wegens haar bijzondere taak (system track) niet worden geïnitialiseerd.
* ERR4 de diskette is beschermd tegen overschrijven.

BDOS-subroutine DSKWRT * $27E1
LDA $39
JSR SETTK
LDA $02
STA PGCNT
LDY $00
LDA F0
LDY 00
STA MEMHI
JSR DSKWRT
Dit programma schrijft een sektor op de diskette. In het voorbeeld wordt op sektor 2 van track 39 de inhoud van de Octopus-EPROM op de diskette geschreven, beginnend bij adres $F000 en eindigend bij adres F1FF. Eerst wordt de schrijf/leeskop op track 39 geplaatst. Dan wordt het aantal 256-byte-pages in de geheugencel PGCNT opgeslagen. In het voorbeeld worden twee pages = 512 bytes op de diskette geschreven. Het startadres van waar-

uit op de diskette moet worden geschreven, staat in de pointer MEMHI, MEMLO in page zero.
MEMLO * $00FE MEMHI * $00FF
Het programma DSKWRT kan vier foutmeldingen geven:
* ERR2 niet alle bytes zijn korrekt op de diskette geschreven.
* ERR5 de schrijf/leeskop is niet op de juiste track geplaatst.
* ERR9 de sektor kan op de nu gekozen track niet gevonden worden.
* ERRB Fout bij de keuze van de sektorlengte (voorbeeld: op een sektor die 2 pages lang is kunnen geen drie pages worden geschreven; de sektor is te kort).

BDOS-subroutine READDK * $2967
LDA $12
JSR SETTK
LDA $02
STA SECTNM
LDY $00
LDA $42
STY MEMLO
STA MEMHI
JSR READDK
Dit programma leest een sektor van de diskette in het geheugen van de Octopus 65. In het voorbeeld wordt van track 12 de sektor 2 naar het adres $4200 geladen. De informatie over de sektorlengte ontvangt de computer van de diskette. Belangrijke adressen zijn:
SECTNM * $265E
MEMLO * $00FE
MEMHI * $00FF
Het programma READDK kan drie foutmeldingen geven:
* ERR1 de sektor kan wegens een pariteitsfout niet worden gelezen.
ERR5 en ERR9, zie hiervoor DSKWRT.

BDOS-subroutine SETDRV * $29C6
LDA 'B' JSR SETDRV
Dit programma kiest een bepaalde disk drive. Het nummer van de drive moet in de accumulator staan. In het voorbeeld wordt drive B gekozen. Let op; dit programma vernielt de inhoud van geheugencel TKNUM!

BDOS-subroutine INCTKN * $2C83
JSR INCTKN
Dit programma plaatst de schrijf/leeskop van de disk drive op de volgens track. Voordat dit programma kan worden opgeroepen, moet in de geheugencel TKNUM * $265D het momentele track-nummer staan en in de geheugencel HSTTK * $00ES het hoogste track-nummer (normaal 39).

BDOS-subroutine SWAP * $2CF7
JSR SWAP
Dit programma is een van de belangrijkste sub-programma's in de BDOS. Het verwisselt de inhoud van page zero en van het stack-bereik met de inhoud van een 512-byte-buffer. Gebruikt de programmeur in BASIC de opdracht DISK!"GO XXXX", dan slaat BASIC alle variabelen en pointers in page zero en in het stack-bereik op en stelt de gebruiker een eigen page zero en een eigen stack-bereik ter beschikking. Bij de terugkeer van het user-programma naar BASIC worden de gebruiker-variabelen in page zero en in het stack-bereik weer automatisch verwisseld met de variabelen van BASIC. De gebruiker behoeft zich dus niet steeds af te vragen "welke adressen mag ik gebruiken en welke niet?". De volgende geheugenbereiken worden door SWAP verwisseld:
page 0 -- $2F79 page zero
page 1 -- $3079 stack-bereik

3) Console Command Processor CCP

De console command processor dekodeert alle kommando's die via het keyboard worden ingegeven. De DOS-kommando's zijn elders in deze special exakt beschreven. De CCP bestaat uit twee delen:
1) kommando-lus
2) subroutines die door de kommando-lus worden opgeroepen.
De kommando-lus is een lus zonder einde die door een sprong-opdracht kan worden opgeroepen (JMP-instruktie). De subroutines van de CCP kunnen in eigen programma's worden ingebouwd en zijn bij de ontwikkeling van eigen BASIC- of assembler-programma's bijzonder praktisch. De CCP-subroutines worden vanuit BASIC met de opdracht DISK!"GO XXXX" opgeroepen. De RTS-opdracht aan het eind van elke CCP-subroutine schakelt weer terug naar BASIC (XXXX is een hexadecimaal adres).
Opmerking: het doorgeven van parameters aan de subroutines van de CCP geschiedt via de kommando-buffer. Deze buffer bevindt zich gewoonlijk op het adres $2E1E. De pointer OSIBAD in de page-zero-adressen $E1, $E2 wijst naar de kommando-buffer. DOS-parameters zijn ASCII-tekens die door een CR-teken zijn afgesloten (bijv. SA 39,2 = B000/3 cr).

CCP-programma DOSLP * $2A51
JMP DOSLP
Op het beeldscherm verschijnt de typische melding van de CCP: A* of B*, enz. De CCP wacht op een kommando. Die kommando's kunnen via het keyboard worden gegeven (input distributor $2321 = $01) of in het geheugen van de Octopus 65 zijn opgeslagen (input distributor $2321 = $10). Een cr aan het eind van het kommando geeft de verdere sturing weer terug aan de CCP.

CCP-subroutine EXCOM * $2A7D
JSR EXCOM
Als men DOS-kommando's vanuit een zelf geschreven programma wil oproepen, kan men gebruik maken van het hulpprogramma EXCOM. De uit te voeren opdracht moet in de DOS-kommando-buffer staan en door een CR-teken zijn afgesloten. De kommando-buffer bevindt zich op $2E1E. De pointer OSIBAD in de geheugencellen $E1, $E2 wijst de kommando-buffer aan (wordt door DOSLP ingesteld). Als men een DOS-kommando in de eigen input-buffer wil schrijven, moet men de pointer OSIBAD het begin van het DOS-kommando in de eigen input-buffer laten aanwijzen. ERR7 wordt gegeven indien er een foutief kommando in de kommando-buffer staat.

CCP-subroutine LDCMN * $2AEE
LDA $02
JSR LOCMN
Dit programma laadt bij versie V3.2 drie tracks van de diskette in het geheugen van de computer (bij de versie V3.3 vier tracks). Het nummer van de eerste track staat in de accumulator. De data die van de diskette in de computer worden geladen, worden opgeslagen vanaf adres $0200. In het voorbeeld worden track 2, 3 en 4 vanaf adres $0200 opgeslagen.

CCP-subroutine CALL * $2B11
JSR CALL
Dit programma leest een sektor van de diskette in het geheugen. Voordat men CALL kan oproepen moet het kommando CA XXXX=TT,S cr in de opdrachtbuffer staan (TT = tracknummer, S = sektornummer, XXXX = hexadecimaal adres).

CCP-subroutine DIR * $2B29
JSR DIR
Dit programma toont op het beeldscherm de opbouw van een bepaalde track (print sector map). Voordat DIR kan worden opgeroepen, moet het kommando DI TT cr in de kommando-buffer worden gezet. Let op! Op de diskette "SYSTEM LOYS" is het DIR-kommando uitgebreid. Het kommando DI cr geeft de inhoud (directory) van de momenteel gekozen drive. Het adres van het DIR-kommando op deze diskette is $E5E4.

CCP-subroutine ERASE * $2599
JSR ERASE
Dit programma staat alleen op de diskette "SYSTEM LOYS" en wist de naam van een file in de directory van de diskette. Voordat ERASE kan worden opgeroepen, moet het kommando ER NAME cr in de kommando-buffer staan. ERASE staat ter beschikking van de gebruiker bij het werken met de Micro-Ware-assembler en de tekstverwerker (die V3.2 toepast). ERASE vervangt de OSI-Video-routine die bij de Octopus 65 in de EPROM staat. Bij de versie V3.3 staat ERASE niet als "built-in"-kommando ter beschikking. Het "DELETE program" in BEXEC* wist bij V3.3 de naam van een file in de directory.

CCP-subroutine EXAM * $2B73
JSR EXAM
Dit programma laadt alle sektoren die op een bepaalde track staan in het geheugen van de computer. Ook alle formatteringsinformatie (track heading, sector heading, sector trailing) wordt tegelijk met de data in het geheugen geladen. Voordat EXAM kan worden opgeroepen, moet de opdracht EX XXXX=TT cr in de kommando-buffer staan (XXXX = hexadecimaal adres, TT = track-nummer).

CCP-subroutine GO * $2B46
JSR GO
Dit programma biedt de gebruiker de mogelijkheid de DOS te verlaten en naar een user-programma te springen. Is aan het eind van dat toepassingsprogramma een RTS-instruktie opgenomen, dan meldt de CCP zich met A* of B*, enz., nadat het user-programma is uitgevoerd. Voordat GO kan worden opgeroepen, moet het kommando GO XXXX cr in de kommando-buffer staan (zie ook onder DIS-kommando's).

CCP-subroutine INIT * $2B55
JSR INIT
Dit programma initialiseert een of alle tracks op de diskette. Voordat INIT kan worden opgeroepen, moet het kommando IN cr of IN TT cr in de kommando-buffer staan.

CCP-subroutine IO * $2B83
JSR IO
Dit programma wijzigt de input/output-distributor (= input/output-verdeler). Voordat IO kan worden opgeroepen moeten de volgende kommando's in de kommando-buffer staan: IO II,OO cr of IO ,II cr of IO ,OO cr (II = input distributor byte; OO = output distributor byte). Zie ook de beschrijving van de DOS-opdrachten en deel 1 van dit hoofdstuk!

CCP-subroutine LOAD * $2BA7
JSR LOAD
Dit programma laadt een file met een bepaalde naam in het geheugen van de computer. De file wordt bij versie V3.2 op adres $3279 en bij versie V3.3 op adres $3A79 geladen. De eerste vijf bytes van de file geven aan hoe lang de file is en hoeveel tracks op de diskette in beslag worden genomen:
$3279 (3A79): SAL = startadres van de file, minst signifikante byte
$327A (3A7A): SAH = startadres van de file, meest signifikante byte
$327B (3A7B): EAL = eindadres van de file, minst signifikante byte
$327C (3A7C): EAH = eindadres van de file, meest signifikante byte
$327D (3A7D): TRK = aantal gebruikte tracks
$327E (3A7E): eerste byte van de file
Voordat LOAD kan worden opgeroepen, moet het kommando LO NAME cr in de kommando-buffer staan. Een foutmelding ERRC wordt gegeven als NAME niet in de directory staat.
Let op: om bij het werken met de Micro-Ware-assembler het geheugen van de Octopus 65 zo ekonomisch mogelijk te benutten, werden voor het laden van assembler-files nieuwe laadroutines geschreven. Deze routines zijn in de editor ingebouwd. Het LOAD-kommando kan bij de Micro-Ware-assembler niet worden gebruikt, omdat assembler-files met het .LINK-kommando op het adres $4000 worden geplaatst (chained file structure).

CCP-subroutine PUT * $2BDD
JSR PUT
Dit programma slaat een file in het geheugen van de computer onder een naam op de diskette op. PUT kan slechts een file op de diskette schrijven als de naam van de file reeds in de directory is opgenomen. Een naam voor een file kan met BEXEC* worden vastgelegd. Voordat PUT kan worden opgeroepen, moet in de kommando-buffer het kommando PU NAME cr staan. Er wordt een foutmelding ERRC gegeven als NAME niet in de directory staat.
Let op: We vonden het erg lastig dat voor gebruik van het PUT-kommando een naam voor de file in de directory moet worden opgenomen. Dit werk zou PUT eigenlijk automatisch moeten kunnen uitvoeren. Daarom hebben we het PUT-kommando op de diskette "SYSTEM LOYS" uitgebreid. PUT plaatst nu automatisch de naam van de file in de directory als deze nog niet bestaat. Ook alle berekeningen over lengte en aantal tracks worden automatisch door PUT uitgevoerd.

CCP-subroutine MEM * $2BC6
JSR MEM
De DOS kan het geheugen van de Octopus 65 ook als input- en output-apparaat gebruiken. Het maakt geen verschil of de kommando-buffer door het keyboard of door het geheugen wordt bediend. Hetzelfde geldt voor de output-apparaten. De DOS kan ASCII-tekens naar het beeldscherm of de printer zenden, maar ook naar het geheugen. De lees- en schrijf-pointers worden met behulp van het MEM-kommando gedefinieerd. Voordat MEM kan worden opgeroepen, moet in de kommando-buffer het kommando ME IIII,OOOO cr staan (IIII = hexadecimaal ingangsadres, OOOO = hexadecimaal uitgangsadres).

CCP-subroutine SAVE * $2C28
JSR SAVE
Dit programma schrijft data vanuit het geheugen van de computer naar een sektor op de diskette. Voordat SAVE kan worden opgeroepen, moet in de kommando-buffer het kommando SA TT,S = XXXX/P cr staan (TT = tracknummer, S = sektor-nummer, XXXX = hexadecimaal adres, P = aantal 256-byte-blokken = pages).

CCP-routine SELECT * $2C43
JSR SELECT
Dit programma selekteert een disk drive. Voordat SELECT kan worden opgeroepen, moet in de kommando-buffer het kommando SE A cr of SE B cr, enz., staan. Als de gekozen disk drive niet bedrijfsklaar staat, wordt de foutmelding ERR6 gegeven.

CCP-subroutine GETTK * $2C60
JSR GETTK
Dit programma leest het track-nummer en het sektornummer uit de kommando-buffer. Track- en sektornummer zijn als ASCII-tekens (string) in de buffer geplaatst. Na het lezen van de kommando-buffer wordt de schrijf/leeskop in positie gebracht. Voordat GETTK kan worden opgeroepen, moet een index in geheugencel $2CE5 wijzen naar het track-nummer in de kommando-buffer.

CCP-subroutine SETPGM * $2C70
JSR SETPGM
Dit programma plaatst de laad/schrijf-pointer op het adres $3279 bij V3.2 en op $3A79 bij V3.3. Het sektornummer is SECTNM = 1. Voordat SETPGM kan worden opgeroepen, moet in de accumulator het track-nummer staan van de track waarvan een sektor moet worden gelezen. SETPGM laat de schrijf/leeskop van de disk drive op de diskette zakken.

CCP-subroutine OSINP * $2C9B
JSR OSINP
Dit programma laadt een kommando in de DOS-kommando-buffer. Staat $01 in het input-distributor-byte ($2321), dan wordt de kommando-buffer door het keyboard geladen; staat $10 in het input-distributor-byte dan wordt de kommando-buffer van-uit het geheugen van de computer geladen. Belangrijke adressen in dit programma: $2C9B — bevat de waarde van de maximale bufferlengte (normaal $11). $2CA5 — bevat het stuurteken voor backspace ($5F = underline).

CCP-subroutine BUFBYT * $2CE4
JSR BUFBYT
Dit programma laadt een byte uit de kommando-buffer in de accumulator. Is de kommando-buffer vol, dan gaat BUFBYT met $0D = cr in de accumulator terug. Belangrijke adressen in dit programma: $2CE5 — bevat de momentele index in de kommando-buffer. De inhoud van geheugencel $2C9B (zie OSINP) wordt naar $2CED gebracht.
V3.3 gebruikt bij de disk-kommando's tevens de subroutine BUFBYT. De kommando-buffer wordt niet met cr afgesloten, maar de momentele bufferlengte in $2CED geeft het einde van de buffer aan ($2CEC, 2CED is een zelf-modificerende kode).

CCP-subroutine GETADR * $2D23
JSR GETADR
Dit programma leest ASCII-tekens uit de kommando-buffer en maakt daarvan een hexadecimaal adres. Het resultaat wordt naar de laad/schrijf-pointer in page zero opgeslagen ($FE,$FF). Voordat GETADR kan worden opgeroepen, moet de buffer-index in geheugencel $2CE5 het begin van het ASCII-hexgetal in de kommando-buffer aanwijzen.

CCP-subroutine BLDHEX * $2D2E
JSR BLDHEX
Dit programma leest twee ASCII-tekens uit de kommando-buffer en voegt die tot een byte samen. Een voorbeeld:
1. byte: "A" = $41
2. byte: "4" = $34
wordt in de accumulator $A4.
Voordat BLDHEX kan worden opgeroepen, moet de buffer-index in geheugencel $2CE5 het begin van het ASCII-hexgetal in de kommando-buffer aanwijzen.

CCP-subroutine GETHEX * $2D3D
JSR GETHEX
Dit programma leest een ASCII-teken uit de kommando-buffer en vertaalt dit in een hex-getal. Voorbeeld:
"A" = $41 wordt $0A
"3" = $33 wordt $03
in de accumulator.

CCP-subroutine CKEQL * $2D58
JSR CKEQL
Dit programma test of de operatoren "=" of "," of "/" zich in de kommando-buffer bevinden. Voor elke operator heeft het programma een eigen sprong-adres:
test op "=" : $2D58
test op "," : $2D5B
test op "/" : $2D5E
Voordat dit programma kan worden opgeroepen, moet de buffer-index in geheugencel $2CE5 de te testen operator aanwijzen. Wijst de index een teken aan dat niet de gezochte operator is, dan wordt een foutmelding ERR7 gegeven.

CCP-subroutine CRLF * $2D6A
JSR CRLF

Dit programma zendt een " cr lf "-serie (carriage return gevolgd door line feed) naar alle momenteel geselekteerde outputapparaten. Let op! Indien u de Octopus 65 via een modem met een data base (mail box) verbindt, dan kan de computer wel eens vreemd reageren op het ASCII-teken lf. Dat is eenvoudig te verhelpen door in geheugencel $2D6C het byte $20 te vervangen door $4C.

CCP-subroutine STROUT * $2D73
JSR STROUT
= $0D
= $0A
= 'O
= 'C
= 'T
= 'O
= 'P
= 'U
= 'S
= $00
Dit programma zendt naar alle momenteel geselekteerde output-apparaten een boodschap (string), die direkt volgt op de JSR-instruktie. De te verzenden boodschap kan maximaal uit 255 tekens bestaan en moet met $00 zijn afgesloten. Het voorbeeld geeft: cr OCTOPUS.

CCP-subroutine PRTAHX * $2D92
JSR PRTAHX
Dit programma geeft de inhoud van de accumulator als twee ASCII-tekens.

CCP-subroutine PRTHEX * $2D9B
JSR PRTHEX
Dit programma geeft het minst signifikante nibble in de accumulator als een ASCII-teken.

CCP-subroutine FNDFL * $2DA6
JSR FNDFL
Dit programma is vooral erg praktisch als men zelf assembler-programma's schrijft. FNDFL zoekt de naam van een file in de directory. Voordat FNDFL kan worden opgeroepen, moet aan de volgende voorwaarden zijn voldaan:
* De naam van de file die wordt gezocht moet in de kommando-buffer staan, hetzij in de DOS-buffer op adres $2E1E of in een andere buffer waarnaar de bufferpointer OSIBAD ($E1, $E2) wijst.
* De buffer-index in het adres $2CE5 moet het begin van de naam van de gezochte file aangeven.
Bevat de directory de naam van de file, dan berekent FNDFL het aantal tracks waarop de file staat. Na terugkeer uit FNDFL staat de start-track van de file in de accumulator en de laatste track in de geheugencel $E5. De lengte van de file kan als volgt worden berekend.
Inhoud van $E5 minus accumulator = aantal tracks minus één.

CCP-DISPATCH TABLE * $2E30
Op dit adres begint een tabel die alle DOS-kommando's met de bijbehorende sprong-adressen bevat. Elke registratie van een DOS-kommando bestaat uit vier bytes. De eerste twee bytes representeren de eerste twee karakters van een DOS-kommando (SA, CA, LO, PU, enz.). Byte 3 en byte 4 bevatten de sprongadressen minus één, omdat via een RTS-instruktie wordt gesprongen.

4) TRANSIENT PROCESSOR AREA

Transient processor area noemt men het geheugengedeelte waarin vanaf diskette een programma wordt geladen en dat afhankelijk van het gebruik op een ander geheugenbereik kan staan. Wil men in BASIC programmeren, dan laadt men de BASIC-interpreter in dit geheugengedeelte; wil men in assembler programmeren, dan laadt men de assembler in dit geheugenbereik. Programma's die in de transient processor area worden geladen, noemt men transient processors. Deze processoren heten transient omdat ze niet zoals de DOS in het geheugen blijven, maar kunnen wisselen. Alle transient processors hebben kommando's waarmee files op de diskette kunnen worden geschreven of waarmee files van de diskette kunnen worden gelezen. De transient processor area ligt bij de Octopus 65 in het geheugenbereik $0200...2300. Meteen daarna volgt de DOS, die ca. 4 Kbytes beslaat. Achter de DOS begint het werkgeheugen waarin de programma's (source code) worden opgeslagen.

5) BOOT, BOOTSTRAP

Na het inschakelen van de Octopus 65 staat nog geen BASIC of assembler in het geheugen. Slechts de monitor (die in de EPROM is opgeslagen) kan worden opgeroepen. Funkties als hexdump, ASCII-dump, wijzigen van een geheugencel of disassembleren zijn allemaal mogelijk. Wil men in een hogere programmeertaal werken, dan moet eerst de DOS en daarna de betreffende programmeertaal geladen worden in het geheugen van de Octopus 65. Het monitorkommando ".B" = .B(OOT) vervult deze taak. Welke aktiviteiten vinden in de Octopus 65 plaats nadat men dit kommando heeft gegeven?
1) De programmeur geeft het kommando ".B". De Octopus 65 gaat dan naar een programma in de EPROM.
2) Dit programma initialiseert de 6821-PIA en de 6850-ACIA op de FCU-kaart en stuurt de schrijf/leeskop van disk drive "A" zover naar buiten tot een lichtstraal wordt onderbroken of een schakelaar wordt gesloten. De Octopus 65 weet dan dat de kop in disk drive "A" op track zero staat.
3) De Octopus 65 wacht op de indeximpuls die gegeven wordt als het index-gat in de diskette een lichtstraal doorlaat. Is de index-puls voorbij, dan leest de Octopus 65 na een korte vertragingstijd track zero. Elk byte wordt begeleid door een startbit, een pariteitsbit en een stopbit.
4) De eerste drie bytes achter het index-gat hebben een bijzondere betekenis.
De eerste twee bytes vormen de laadvektor en het derde byte geeft aan hoeveel 256-byte-stukken (pages) moeten worden geladen. De laadvektor die van de diskette wordt gelezen, wijst gewoonlijk naar adres $2200; bij minidiskettes worden acht pages (dus 2 Kbytes) vanaf adres $2200 in het geheugen van de Octopus 65 geladen.
5) Is track zero in het geheugen geplaatst, dan staan de belangrijkste hulpprogramma's van de BDOS in de computer. De schrijf/leeskop kan worden neergelaten op de schijf op een bepaalde track, en er kan een sektor van de diskette in het geheugen worden geladen. Daarom springt de Octopus 65 na het laden van track zero naar adres $2200. Vanaf dit adres staan de instrukties die de Octopus 65 aangeven welke sektoren van welke tracks verder in het geheugen moeten worden geladen. Hiervoor zijn 256 bytes (1 page) gereserveerd ($2200...22FF). Wanneer het gehele programma in de trancient processor area geladen is, wordt het geheugenbereik $2200...22FF niet meer gebruikt en kan het worden overschreven.
Aangezien de Octopus 65 werkt met meerdere DOS-versies dienen de instrukties in het geheugenbereik $2200...22FF te worden aangepast aan die verschillende versies. Dat lijkt moeilijker dan het is. We beschrijven nu stap voor stap hoe de verschillende DOS-versies van de diskette in de computer worden geladen!

Laden van een diskette bij V3.2
1) Laad track zero op adres $2200 en spring naar dit adres. In de Octopus 65 staan nu alle belangrijke BIOS- en BDOS-hulpprogramma's.
2) Laad track 1, sektor 1 op het adres $2A00. In de Octopus 65 bevinden zich nu alle BDOS-hulpprogramma's en de CCP. De gehele DOS is geladen.
3) Laad in de geheugenposities $FE, $FF in page zero het adres $DF00 = hoogste 256-byte-page.
4) Spring naar het hulpprogramma MEMCHK * $22EC en doe een geheugentest. Na de terugkeer van MEMCHK staat in het Y-register de hoogste 256-byte-page, waar RAM-geheugen aanwezig is.
5) Schrijf de inhoud van het Y-register in de geheugencel HIMEM * $2300.
6) Zet de input-distributor ($2321) op $10 en de output-distributor ($2322) op $01 (lees "RUN BEXEC*" uit het geheugen en gebruik het beeldscherm voor output.
7) Geef zonodig een boodschap en initialiseer zonodig de Octopus-I/O.
8) Spring naar adres $2AE6. Hierdoor wordt BASIC in het geheugen geladen.

Laden van een diskette bij V3.3
1) Laad track zero op adres $2200 en spring naar dit adres. In de Octopus 65 staan nu alle belangrijke BIOS- en BDOS-hulpprogramma's.
2) Laad track 1, sektor 1 op het adres $2A00. In de Octopus 65 bevindt zich nu een deel van de BDOS-hulpprogramma's en de CCP.
3) Laad track 6, sektor 2 op het adres $3200. De gehele DOS is nu in de Octopus 65 geladen.
4) Laad track 6, sektor 3 op het adres $0000. Page zero wordt voorbereid voor BASIC en DOS.
5) Laad track 13, sektor 1 op het adres $3200. De BASIC-editor wordt in het geheugen geladen.
6) Laad in de heugenplaatsen $FE, $FF in page zero het adres $DF00 = hoogste 256-byte-page.
7) Spring naar het hulpprogramma MEMCHK * $22EC en doe een geheugentest. Na de terugkeer van MEMCHK staat in het Y-register de hoogste 256-byte-page waar RAM-geheugen aanwezig is.
8) Schrijf de inhoud van het Y-register in

de geheugencel HIMEM * $2300.
9) Zet de input-distributor op $10 en de output distributor op $01.
10) Geef indien gewenst een boodschap en initialiseer zonodig de Octopus I/O.
11) Spring naar adres $2AE6. Hierdoor wordt BASIC in het geheugen geladen.

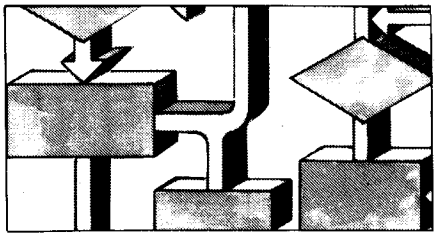
Laden van een systeemdiskette "SYSTEM LOYS".

De diskette "SYSTEM LOYS" is een wezenlijke uitbreiding van de OSI-diskette V3.3. Daarom moet bij het "booten" nog meer software in de Octopus 65 geladen worden. Het laden van een "SYSTEM LOYS"-diskette verloopt op dezelfde wijze als bij een diskette V3.3, waarbij na punt 5) het volgende wordt toegevoegd:

5a) Laad track 6, sektor 4 op het adres $E400. De PUT- en DIR-uitbreidingen zijn nu geladen, evenals de CCP-uitbreidingen.
5b) Laad track 12, sektor 5 op adres $DE00. De full screen editor voor BASIC en de "steno-software" zijn nu geladen.
5c) Laad track 13, sektor 1 op adres $3200. Zie V3.3
6) Laad in de geheugenplaatsen $FE, $FF in page zero het adres $DC00 = hoogste 256-byte-page.
. . .verder zoals onder V3.3

Wie de Octopus 65 gebouwd heeft en de diskette "SYSTEM LOYS" bezit, kan deze aanwijzingen verder weer vergeten. Maar voor Octopus 65-fans die met de DOS experimenteren willen en voor alle bezitters van een OHIO-scientific-computer, die onze uitbreidingen willen toepassen, zijn deze aanwijzingen zeer nuttig.

PROGRAMMA'S



ADRES

Een adresbeheer- en sorteerprogramma in samenwerking met N. Ludwig.

ADRES bestaat uit drie delen: het ADRES-hoofdprogramma en de beide subprogramma's CREATE en DELETE. Alle drie zijn zelfstandige programma's die samen 10 tracks op de diskette beslaan. Er blijven dus nog genoeg tracks vrij voor het opslaan van de adressen-files. Voor elk adres zijn 128 bits gereserveerd, zodat op een track maximaal 16 adressen kunnen staan.
Het programma behoeft geen uitgebreide verklaring, omdat het de gebruiker steeds vraagt wat de volgende stap moet zijn. We kunnen dus volstaan met een handleiding in telegramstijl. Er wordt gestart met RUN"ADRES. Het programma biedt de gebruiker nu drie mogelijkheden:
1. nieuwe file opzetten
2. file bewerken
3. file wissen
Kiest men voor "1" (in de listing zijn dat de programmaregels 5000...6070), dan wordt vervolgens gevraagd naar de nieuwe file-naam en het maximale aantal adressen in de file. Deze beide opgaven dient de gebruiker te onthouden (eventueel opschrijven!), want als hij een file wil wijzigen (keuze 2), dan zal het programma hem vragen naar de naam van de te wijzigen file èn naar het maximale aantal adressen. Noemt de gebruiker een niet aanwezige file-naam, dan kan het ADRES-hoofdprogramma niet starten. Is de filenaam bekend, dan start het hoofdprogramma en geeft een menu met 9 keuzemogelijkheden.

Keuze 1:
Nieuwe adressen ingeven (in de listing de programmaregels 460...850). Op de monitor verschijnt een adresblok met vier regels: titel, naam, adres en postkode met plaatsnaam. Wil men geen adres meer inladen, dan verlaat men met * in de titelregel deze mode en het menu verschijnt weer. Let op: in het adresblok 0 geen adres opnemen, maar een benaming voor de adres-file. Daarbij moet in elk van die vier regels een informatie staan. Een werkelijk adres in blok 0 zou bij sorteren niet worden meegenomen.
Dubbele namen, zowel voor- als achternamen, dienen door een teken, bijv. "-", te worden verbonden.

Keuze 2:
Adres wissen (in de listing de programmaregels 910...1100). Het adresbloknummer dat gewist moet worden, wordt gevraagd. De inhoud daarvan wordt op de monitor getoond. Na een tweede vraag of deze inhoud moet worden gewist, gebeurt dit pas na het indrukken van de J-toets en RETURN.

Keuze 3:
Het drukken van adressen in PTT-vorm (in de listing de programmaregels 1130...1410). De aangesloten printer drukt de adressen van blok X tot blok Y.

Keuze 4:
Adressen horizontaal van X tot Y listen (in de listing de programmaregels 1450...1830). De aangesloten printer drukt de adressen van blok X tot blok Y; de titel wordt niet mee gedrukt.

Keuze 5:
Adres-etiketten in de printer justeren (in de listing de programmaregels 1870...2070). Deze keuze kan drie verschillende adresblok-afmetingen produceren, zodat men de etiketten exakt kan justeren.

Keuze 6:
Adres wijzigen (in de listing de programmaregels 2110...2510). Verandert iemand uit de file zijn woonplaats, dan heeft men de mogelijkheid het adres aan te passen. Het adres moet nu geheel nieuw worden ingegeven, ook al zou alleen het huisnummer wijzigen.

Keuze 7:
Lege file-blokken listen (in de listing de programmaregels 2550...2710). Hoeveel adresblokken zijn nog vrij in de file? Deze keuze geeft daarop het antwoord.

Keuze 8:
File sorteren (in de listing programmaregels 3060...4420). Er kan worden gesorteerd op titel, naam, voornaam, straat, postkode en plaatsnaam. Het programma sorteert echter niet alle 6 mogelijkheden gelijktijdig, maar maximaal 3. Daarom verschijnt de vraag "Sorteren op hoeveel groepen? (1...3)". Vervolgens verwacht het programma de prioriteitsvolgorde. Er moet bijvoorbeeld een sortering komen die in alfabetische volgorde de plaatsnamen geeft, binnen elke plaats de straten alfabetisch sorteert en in de straten de namen, natuurlijk eveneens alfabetisch gerangschikt.
De sorteerroutine leest de adressen van een file tot aan het eerste lege blok. Dat heeft konsekwenties, want als er in een file een adres is gewist en er is geen nieuw adres op die plaats gezet, dan leest het programma alleen de adressen tot aan dat eerste lege blok en vergeet de volgende adressen. Geef daarom na het wissen van een adres steeds een nieuw adres op die plaats, of plaats op elke regel bijvoorbeeld enkele Z's.

Keuze 9:
Bewerking van de file beëindigen (in de listing de programmaregels 4460...4570). Hiermee keert het programma uit de bewerking-mode terug naar het hoofdmenu.

De derde mogelijkheid van het hoofdmenu is het wissen van een file (in de listing de programmaregels 7000...7400). Het programma haalt de gewenste filenaam uit de directory van de diskette, schrijft op die plaats het teken "#" en zet de tracknummers op 00.

```
10 REM ADRES-hoofdprogramma
20 POKE10081,169:DISK!"go 2761"
30 GOSUB2900
40 PRINTTAB(T)"1>  Nieuwe file opzetten"
50 PRINTTAB(T)"2>  File bewerken"
60 PRINTTAB(T)"3>  File wissen"
70 PRINT:PRINT
80 INPUT"Uw keus";A:IF A = 1 THEN RUN"CREATE"
90 IF A=3 THEN RUN"DELETE
100 PRINT:PRINT
110 INPUT"Welke file wilt u bewerken";NA$
120 NA=LEN(NA$):IFNA<10RNA>6GOTO110
130 NA=ASC(NA$):IFNA<65ORNA>90GOTO110
140 NA$=NA$+"       ":NA$=LEFT$(NA$,6)
150 INPUT"Bevindt de file zich in drive A, B, C of D ";D$
160 IFD$=""THEND$="A"
170 IFD$<"A"ORD$>"D"ORLEN(D$)<>1THENPRINT:GOTO150
180 DISK!"SE "+D$
190 S=0:DISK!"CA 2E79=12,1":GOSUB2980
200 IFS=1GOTO260
210 DISK!"CA 2E79=12,2":GOSUB2980
220 IFS=1GOTO260
230 PRINT:PRINT
240 PRINT"*** ";CHR$(34);NA$;CHR$(34);" bestaat niet. ***"
250 PRINT:PRINT:GOTO110
260 INPUT"Maximaal aantal adresblokken van deze file";MAX
265 DIMC$(6,MAX):DIMY$(6):DIMSM(3):DIMSG(MAX,2):REM SORTEER BLOKKEN
270 PRINT:PRINT
280 E$="<empty>"
290 FI$="........................."
300 FJ$="          "
310 PRINTTAB(T)"1>  Nieuw adres ingeven
320 PRINTTAB(T)"2>  Adres wissen
330 PRINTTAB(T)"3>  Adres in PTT-formaat in blokken of
340 PRINTTAB(T)"    per stuk drukken
350 PRINTTAB(T)"4>  Adressen horizontaal van X tot Y listen
360 PRINTTAB(T)"5>  Adress-etiketten in de printer justeren
370 PRINTTAB(T)"6>  Adres wijzigen
380 PRINTTAB(T)"7>  Lege file-blokken listen
390 PRINTTAB(T)"8>  File sorteren
400 PRINTTAB(T)"9>  Bewerking van file stoppen
410 PRINT:INPUT"Welke bewerking wilt u uitvoeren ";I
420 IFI<1ORI>9THENRUN
430 ON I GOTO 460,910,1110,1420,1870,2080,2550,3060,4460
440 :
450 :
460 GOSUB2870
470 FOR RN=0 TO MAX-1: DISKGET,RN: INPUT#6,B$
480 IF LEFT$(B$,1)="#" THEN 520
490 NEXT
500 PRINT:PRINT">> Alle plaatsen in de adres-file zijn bezet <<
510 PRINT:POKE10081,169:END
520 PRINTCHR$(27);"1";:FORX=1TO10:PRINT:NEXT
530 PRINT"Adresblok-nummer -- ";RN:PRINT
540 PRINTTAB(T)"Titel:          ";"........................."
550 PRINTTAB(T)"Naam:           ";"........................."
560 PRINTTAB(T)"Adres:          ";"........................."
570 PRINT:
580 PRINTTAB(T)"Postkode Plaats:";".........................
590 FOR X=1 TO 5: PRINTCHR$(27);"5";:NEXT
600 FOR X=1 TO T+14:PRINTCHR$(16);:NEXT: INPUT B$
610 IF B$="<*>" THEN780
620 FOR X=1 TO T+14: PRINTCHR$(16);:NEXT: INPUT N$
630 FOR X=1 TO T+14: PRINTCHR$(16);:NEXT: INPUT A$:PRINT
640 FOR X=1 TO T+14:PRINTCHR$(16);:NEXT: INPUT T$
650 IF LEN(B$)+LEN(N$)+LEN(A$)+LEN(T$)>120 THENGOSUB800: GOTO540
660 PRINT#6,B$
670 PRINT#6,N$
680 PRINT#6,A$
```

```
690 PRINT#6,T$
700 DISKPUT
710 IF RN=MAX-1 THEN500
720 RN=RN+1
730 FOR RN=RN TO MAX-1
740 DISKGET,RN: INPUT#6,B$
750 IF LEFT$(B$,1)="#" THEN DISKGET,RN: GOTO520
760 NEXT
770 GOTO 500
780 DISKCLOSE,6
790 GOSUB2900:GOTO310
800 PRINT:PRINT:
810 PRINTTAB(T)"Adres is te lang,"
820 PRINTTAB(T)"opnieuw proberen!"
830 PRINTTAB(T)"Het adres mag maximaal 120 karakters lang zijn!"
840 FOR Z=1TO3000:NEXT
850 PRINT:PRINT:PRINT: RETURN
860 :
870 :
880 :
890 :
900 :
910 DISK!"GO F32F"
920 PRINT:PRINT:PRINTTAB(T)"=== Adresblok wissen ==="
930 PRINT:PRINT:PRINT
940 PRINTTAB(T)"Geef adresblok-nummer (RN) ";:INPUT"RN = ";RN
950 GOSUB2870
960 DISKget, RN
970 INPUT#6,B$: IF LEFT$(B$,1)="#" THEN 1100
980 INPUT#6,N$
990 INPUT#6,a$: INPUT#6,t$
1000 PRINT:PRINTTAB(T)"De inhoud van adresblok ";rn;" is:"
1010 PRINT:PRINTB$:PRINTn$:PRINTa$:PRINT:PRINTt$:PRINT:PRINT
1020 PRINT"Wilt u dit adresblok werkelijk wissen,"
1030 INPUT"Druk dan op <J> ";I$
1040 IF LEFT$(i$,1)<>"J" THENGOSUB2900:GOTO310
1050 FOR y=1 TO 127
1060 PRINT#6,CHR$(35);
1070 NEXT
1080 PRINT#6,CHR$(13);
1090 DISKput
1100 DISKclose,6:GOSUB2900:GOTO310
1110 :
1120 :
1130 PRINTCHR$(27);"1"
1140 FORX=1TO10:PRINT
1150 PRINTTAB(T)"==== Adres drukken in PTT-formaat ====":PRINT
1160 PRINTTAB(T+7)"Eerste adresblok-nummer  ";:INPUTF
1170 PRINTTAB(T+7)"Laatste adresblok-nummer  ";:INPUTL
1180 IF L<F THEN 1130
1190 IFL>MAX-1THENGOSUB2920:GOTO1170
1200 IFL>F THEN 1240
1210 IFL=FTHENRN=F
1220 DISK!"IO ,09
1230 GOSUB2870:GOSUB2760:GOTO1330
1240 POKE10081,96:DISK!"IO ,09
1250 GOSUB2870
1260 FOR X=F TO L:GOTO1290
1270 POKE10081,96:DISK!"IO ,09"
1280 FOR X=RN+1 TO RN
1290 RN=X:IFRN>MAX-1THENPRINT:GOSUB2960:GOTO270
1300 PRINT:GOSUB2760
1310 NEXT
1320 POKE10081,169:DISK!"GO 2761":DISK!"IO ,01"
1330 DISK!"IO ,01
1340 PRINT" Indien meer adressen achter elkaar, druk <J - RETURN>;"
1350 PRINT"bij volgende adresblokken alleen <B - RETURN>;"
1360 INPUT"Terug naar het menu met <M - RETURN> ";Y$
```

```
1370 PRINT
1380 IF Y$ ="J" THEN 1270
1390 IF Y$ ="B" THEN 1130
1400 IF Y$ ="M" THEN GOSUB2900:GOTO310
1410 IFY$<>"J"ORY$<>"B"ORY$<>"M"THENPRINT:GOTO1340
1420 :
1430 :
1440 :
1450 PRINTCHR$(27);"1"
1460 FORX=1TO10:PRINT:NEXT
1470 PRINTTAB(18)"=================================================="
1480 PRINTTAB(18)"= Horizontaal drukken van adressen zonder titel ="
1490 PRINTTAB(18)"=                  van X tot Y                   ="
1500 PRINTTAB(18)"=================================================="
1510 PRINT:PRINT:PRINT
1520 POKE10081,96: GOSUB2870
1530 PRINTTAB(T+7)"Eerste adresblok-nummer ";:INPUTX
1540 PRINTTAB(T+7)"Laatste adresblok-nummer ";:INPUTY:PRINT
1550 IFY>MAX-1THENGOSUB2920:GOTO1540
1560 IFY<XTHEN1450
1570 IFX=YTHENRN=Y
1580 FOR RN = X TO Y
1590 DISKGET,RN
1600 INPUT#6,B$
1610 IF LEFT$(B$,1)="#" THEN 1700
1620 INPUT#6,N$
1630 INPUT#6,A$
1640 INPUT#6,T$
1650 A$=A$+LEFT$(FI$,(27-LEN(A$)))
1660 N$=N$+LEFT$(FI$,(27-LEN(N$)))
1670 rn$=STR$(rn):rn$=rn$+LEFT$(fj$,(5-LEN(rn$)))
1680 DISK!"IO ,09:PRINT
1690 PRINTrn$;N$;A$;T$
1700 NEXT
1710 POKE10081,169:DISK!"GO 2761":DISK!"IO ,01"
1720 PRINT
1730 PRINT"Indien meer adressen achter elkaar, druk <J - RETURN>;"
1740 PRINT"bij volgende adresblokken alleen <B - RETURN>;"
1750 PRINT"terug naar bewerkingsmenu met <M - RETURN>."
1760 PRINT
1770 INPUT"Uw keuze";Z$
1780 IF Z$="J"THEN X=RN:Y=X:PRINT:GOTO1580
1810 IF Z$="B"THEN 1450
1820 IF Z$ = "M" THEN GOSUB2900:GOTO270
1830 IFZ$<>"J"ORZ$<>"B"ORZ$<>"M"THENPRINT:GOTO1730
1840 :
1850 :
1860 :
1870 DISK!"GO F32F":PRINT:PRINT:PRINT:PRINT
1880 PRINTTAB(T)"Is de printer ingeschakeld en zijn de"
1890 INPUT"adres-etiketten ingevoerd, druk dan op <J> ";I$
1900 IF I$ <>"J" THEN 1880
1910 DISK!"IO ,09"
1920 FOR K=1TO3
1930 PRINT"+";:FOR M=1TO28:PRINT"-";:NEXTM:PRINT"+"
1940 PRINT"!";SPC(28):PRINT"!"
1950 PRINT"!";SPC(28):PRINT"!"
1960 PRINT"!";SPC(28):PRINT"!"
1970 PRINT"+";:FOR M=1TO28:PRINT"-";:NEXTM:PRINT"+"
1980 PRINT:PRINT:PRINT:NEXTK
1990 DISK!"IO ,01"
2000 PRINTTAB(T)"Als nog een test-run nodig is, justeer"
2010 PRINTTAB(T)"dan opnieuw de etiketten en druk op"
2020 PRINTTAB(T)"<J - RETURN>. Is alles OK, druk dan"
2030 INPUT"op <M - RETURN> voor terugkeer naar menu ";X$
2040 DISK!"IO ,01"
2050 IF X$ = "J" THEN 1910
2060 IF X$ = "M" THEN GOSUB2900:GOTO310
2070 IFX$<>"J"ORX$<>"M"THENPRINT:GOTO2000
```

```
2080 :
2090 :
2100 :
2110 DISK!"GO F32F"
2120 PRINT:PRINT:PRINTTAB(T+14)"=== Adres wijzigen ===
2130 PRINT:PRINT:PRINT
2140 PRINTTAB(T+4)"Geef het nummer van het te wijzigen"
2150 PRINTTAB(T+4)"adresblok ";:INPUT RN
2160 IFRN<0ORRN>MAXTHENDISK!"GO F32F":GOTO2180
2170 GOTO2200
2180 PRINT"Onjuiste ingave !!!"
2190 PRINT:PRINT:FORx=1TO2000:NEXT:GOTO2110
2200 GOSUB2870
2210 DISKGET,RN
2220 INPUT#6,B$
2230 IF LEFT$(B$,1)="#" THEN 2270
2240 INPUT#6,N$
2250 INPUT#6,A$
2260 INPUT#6,T$
2270 PRINT:PRINT"De momentele inhoud van blok ";RN;" is:":PRINT
2280 PRINTTAB(T)B$:PRINTTAB(T) N$:PRINTTAB(T) A$:PRINT:PRINTTAB(T) T$
2290 PRINT:PRINT"Wijziging van het adres in blok ";rn;":":PRINT
2300 PRINTTAB(T)"Titel:";SPC(10);:PRINTB$
2310 PRINTTAB(T)"Naam:";SPC(11);:PRINTN$
2320 PRINTTAB(T)"Adres:";SPC(8);:PRINTA$
2330 PRINT
2340 PRINTTAB(T)"Postkode Plaats:";SPC(8);:PRINTT$
2350 FOR X=1 TO 5: PRINTCHR$(27);"5";:NEXT
2360 FOR X=1 TO T+14: PRINTCHR$(16);:NEXT: INPUT B$
2370 IF B$="<X>" THENGOSUB2900:GOTO310
2380 FOR X=1 TO T+14: PRINTCHR$(16);:NEXT:INPUT N$
2390 FOR X=1 TO T+14: PRINTCHR$(16);:NEXT: INPUT A$:PRINT
2400 FOR X=1 TO T+14: PRINTCHR$(16);:NEXT: INPUT T$
2410 IF LEN(B$)+LEN(N$)+LEN(A$)+LEN(T$)>120 THENGOSUB800: GOTO2080
2420 PRINT#6,B$
2430 PRINT#6,N$
2440 PRINT#6,A$
2450 PRINT#6,T$
2460 DISKput:DISKclose,6:PRINT:PRINT
2470 PRINT"Indien meer wijzigingen, druk op <J - RETURN>."
2480 INPUT"Geen wijzigingen meer, dan <M - RETURN> ";Z$
2490 IF LEFT$(Z$,1)="M" THEN GOSUB2900:GOTO310
2500 IF Z$ = "J" THEN 2110
2510 IFZ$<>"M"ORZ$<>"J"THENPRINT:GOTO2470
2520 :
2530 :
2540 :
2550 PRINTCHR$(27);"1":DISK!"IO ,09"
2560 PRINTTAB(T)"=========================================
2570 PRINTTAB(T)"=  Welke adresblokken zijn nog leeg?  =
2580 PRINTTAB(T)"=========================================":PRINT:PRINT:
2590 POKE10081,96:GOSUB2870: CO=0:Z=0
2600 FOR X=0 TO MAX-1
2610 DISKGET,X
2620 INPUT#6,B$
2630 IF LEFT$(B$,1)="#" THENPRINT"Blok";X,:CO=CO+1:IFCO=6THENPRINT:CO=
2640 IF LEFT$(B$,1)="#"THENZ=Z+1
2650 NEXT
2660 IFZ=0THENPRINT">>> Alle blokken van file ";NA$;"zijn vol. <<<"
2670 POKE10081,169:DISK!"GO 2761":DISK!"IO ,01":PRINT:PRINT
2680 PRINT:PRINT"Wilt u terug naar het bewerkingsmenu,"
2690 INPUT"druk dan op <M - RETURN> ";Z$
2700 IFZ$<>"M"ANDZ$<>""GOTO2680
2710 GOSUB2900:GOTO310
2720 :
2730 :
2740 :
2750 REM ADRESGEGEVENS VOOR PTT-DRUKFORMAAT
2760 DISKGET,RN
```

```
2770 INPUT#6,B$
2780 IF LEFT$(B$,1)="#" THEN2830
2790 INPUT#6,N$
2800 INPUT#6,A$
2810 INPUT#6,T$
2820 PRINTB$:PRINTN$:PRINTA$:PRINT:PRINTT$:PRINT:PRINT:PRINT
2830 RETURN
2840 :
2850 :
2860 REM FILE OPENEN
2870 DISKOPEN,6,NA$:RETURN
2880 :
2890 :
2900 T=10:DISK!"GO F32F":FORX=1TO10:PRINT:NEXT:RETURN
2910 :
2920 PRINT:PRINT"De file heeft slechts ";MAX;" adresblokken, maar u"
2930 PRINT"wilt meer adresblokken laten drukken. Geef het laatste"
2940 PRINT"bloknummer opnieuw in.":PRINT:PRINT
2950 RETURN
2960 PRINT">>> Er zijn niet meer adresblokken ge-initialiseerd. <<<"
2970 POKE10081,169:DISK!"IO ,01":RETURN
2980 FORI=11897TO12152STEP8:IFPEEK(I)=35GOTO3010
2990 N$="":FORI1=0TO5:N$=N$+CHR$(PEEK(I+I1)):NEXTI1
3000 IFN$=NA$THENS=1
3010 NEXTI
3020 RETURN
3030 :
3040 :
3050 :
3060 REM ONDERPROGRAMMA DATA SORT
3080 GOSUB 4150
3100 FORJ=1TO3:SM(J)=0:NEXTJ
3110 DISK!"GO F32F"
3120 INPUT"Sorteren op hoeveel groepen (1...3)"; QM:PRINT
3130 IF QM<1 OR QM>3 THEN 3110
3140 PRINTTAB(10)"1>  Titel"
3150 PRINTTAB(10)"2>  Voornaam"
3160 PRINTTAB(10)"3>  Achternaam"
3170 PRINTTAB(10)"4>  Straat"
3180 PRINTTAB(10)"5>  Postkode"
3190 PRINTTAB(10)"6>  Plaats"
3200 PRINT:PRINT"Kies nu in gewenste prioriteitsvolgorde."
3210 FORJ=1TO QM
3220 PRINTJ;". Prioriteit ":INPUTSM(J):IF SM(J)<1 ORSM(J) >6 THEN 3220
3230 NEXTJ
3240 LG=1:RG=EN:KY=SM(1)
3250 GOSUB 3580
3260 IF QM = 1 THEN 3280
3270 GOTO 3340
3280 REM S1 (1x SORTEREN)
3290 INPUT"Oplopend (1) of aflopend (2)";QA
3300 IF QA =1 THEN N1=1: N2=EN:N3=1:GOTO 3320
3310 N1=EN:N2=1:N3=-1
3320 GOSUB 3900
3330 GOTO270
3340 REM S2 (2x SORTEREN)
3350 GOSUB 3790
3360 KY=SM(2)
3370 FOR J =1 TO Z
3380 LG=SG(J,1):RG=SG(J,2)
3390 IF RG=LG THEN 3410
3400 GOSUB 3580
3410 NEXT J
3420 IF QM=3 THEN 3460
3430 N1=1:N2=EN:N3=1
3440 GOSUB 3900
3450 GOTO270
3460 REM S3 (3x SORTEREN)
3470 GOSUB 3790
```

```
3480 KY=SM(3)
3490 FOR J=1 TO Z
3500 LG=SG(J,1):RG=SG(J,2)
3510 IF LG=RG THEN 3530
3520 GOSUB 3580
3530 NEXT J
3540 N1=1:N2=EN:N3=1
3550 GOSUB 3900
3560 GOTO270
3570 REM EINDE DATA SORT MAIN
3580 REM SUB SORT
3590 PRINT"SORTERING";KY
3600 FOR J1=LG+1 TO RG
3610 FOR J3=1 TO 6
3620 Y$(J3)=C$(J3,J1)
3630 NEXT J3
3640 L=LG:R=J1-1
3650 IF L>R THEN 3690
3660 M=INT((L+R+0.6)/2)
3670 IF Y$(KY)<C$(KY,M) THEN R=M-1:GOTO 3650
3680 L=M+1:GOTO 3650
3690 FOR J2=J1-1 TO L STEP -1
3700 FOR J3=1 TO 6
3710 C$(J3,J2+1)=C$(J3,J2)
3720 NEXT J3
3730 NEXTJ2
3740 FOR J3=1 TO 6
3750 C$(J3,L)=Y$(J3)
3760 NEXT J3
3770 NEXT J1
3780 RETURN
3790 REM SUB GR
3800 FOR J=1 TO EN:SG(J,1)=0:SG(J,2)=0:NEXT J
3810 Z=1:SG(Z,1)=1
3820 Q$=LEFT$(C$(KY,1),6)
3830 FOR J=2 TO EN
3840 IF LEFT$(C$(KY,J),6)=Q$ THEN 3870
3850 SG(Z,2)=J-1:Z=Z+1:SG(Z,1)=J
3860 Q$=LEFT$(C$(KY,J),6)
3870 NEXT J
3880 SG(Z,2)=EN
3890 RETURN
3900 REM SUB DO  DISKOUT
3910 POKE 10081,96: GOSUB 2870
3920 RN=1
3930 FOR N4=N1 TO N2 STEP N3
3940 DISKGET,RN
3950 B$=C$(1,N4)
3960 N$=C$(2,N4)+" "+C$(3,N4)
3970 A$=C$(4,N4)
3980 T$=C$(5,N4)+" "+C$(6,N4)
3990 PRINT#6,B$
4000 PRINT#6,N$
4010 PRINT#6,A$
4020 PRINT#6,T$
4030 DISKPUT:RN=RN+1
4040 NEXT N4
4050 FOR N4=RN TO MAX-1
4060 DISKGET,N4
4070 FOR Y=1 TO 127
4080 PRINT#6,CHR$(35);
4090 NEXTY
4100 PRINT#6,CHR$(13);
4110 DISKPUT
4120 NEXT N4
4130 POKE 10081,169:DISK!"GO 2761"
4140 RETURN
4150 REM SUB DI  DISKIN
4160 POKE10081,96:GOSUB 2870
```

```
4170 AN=0
4180 FOR RN=1 TO MAX-1
4190 PRINT RN;
4200 DISKGET,RN
4210 INPUT#6,B$
4220 IF LEFT$(B$,1)="#" THEN 4390
4230 AN=AN+1
4240 INPUT#6,N$
4250 INPUT#6,A$
4260 INPUT#6,T$
4270 C$(1,AN)=B$
4280 FOR I1=1 TO LEN(N$)
4290 IF MID$(N$,I1,1)=" " THEN 4310
4300 NEXT I1
4310 C$(2,AN)=LEFT$(N$,I1-1)
4320 C$(3,AN)=MID$(N$,I1+1)
4330 C$(4,AN)=A$
4340 FOR I1 =1 TO LEN(T$)
4350 IF MID$(T$,I1,1)=" " THEN 4370
4360 NEXT I1
4370 C$(5,AN)=LEFT$(T$,I1-1)
4380 C$(6,AN)=MID$(T$,I1+1)
4390 NEXT RN
4400 EN=AN
4410 POKE 10081,169: DISK!"GO 2761
4420 RETURN
4430 :
4440 :
4450 :
4460 DISK!"GO F32F":GOSUB2900
4470 PRINTTAB(15)"============================="
4480 PRINTTAB(15)"=     Het werk aan file     ="
4490 PRINTTAB(15)"=    ";CHR$(34);NA$;CHR$(34);" is klaar.     ="
4500 PRINTTAB(15)"============================="
4510 FORI=1TO5:PRINT:NEXT
4520 PRINT"Als u nog een file wilt bewerken, druk dan op"
4530 PRINT"<W - RETURN>; bent u klaar met het adressenbestand,"
4540 INPUT"druk dan op <E - RETURN> ";A$
4550 IFA$="W"THENRUN
4560 IFA$="E"THENDISK!"GO F32F":DISK!"DI":END
4570 IFA$<>"W"ORA$<>"E"THENPRINT:GOTO4520
4580 :
4590 :
4600 REM VERNIETIGDE ADRESBLOK OPNIEUW INITIALISEREN
4610 INPUT"In welke file is het adresblok vernietigd ";NA$
4620 INPUT"Geef het nummer van het vernietigde adresblok ";RN
4630 POKE 10081,96:DISKOPEN,6,NA$:DISKGET,RN
4640 FOR Y=1 TO 127:PRINT#6,CHR$(35):NEXT
4650 PRINT#6,CHR$(13):DISKPUT:DISKCLOSE,6
4660 POKE 10081,169:DISK!"GO 2761":RUN


5000 REM FILE OPZETTEN
5010 DISK!"GO F32F"
5020 NT=79:NP=8:D=65:NL=30
5030 DIMT(NT),FL(NL),FL$(NL)
5040 DEFFNA(X)=16*INT(X/10)+X-10*INT(X/10)
5050 DEFFNB(X)=10*INT(X/16)+X-16*INT(X/16)
5060 PRINTTAB(10)"=== Nieuwe adres-file opzetten. ===":PRINT:PRINT
5070 T(0)=1:GOSUB5820
5080 INPUT"Geef de naam van de nieuwe file (max. 6 karakters) ";NA$
5090 NA=LEN(NA$):IFNA<10RNA>6GOTO5080
5100 NA=ASC(NA$):IFNA<65ORNA>90GOTO5080
5110 NA$=NA$+"      ":NA$=LEFT$(NA$,6)
5120 S=0:GOSUB5880:IFS=1THENPRINT:GOTO5150
5130 FORI=0TONT:IFT(I)=0GOTO5180
5140 NEXT:PRINT"Diskette is vol - geen vrije tracks meer.":GOTO6020
5150 PRINT"File-naam ";CHR$(34);NA$;CHR$(34);" is reeds aanwezig."
5160 PRINT:PRINT"Kies een andere file-naam en/of een"
```

```
5170 PRINT"andere drive met een andere diskette.":PRINT:GOTO5070
5180 PRINT:PRINT
5190 PRINT"Hoeveel adresblokken moet file ";CHR$(34);NA$;CHR$(34)
5200 INPUT"bevatten ";NR
5210 PRINT:PRINT:IFNR<1GOTO5190
5220 B=128:TB=2048
5230 AN=(NR-1)*B
5240 TR=INT(AN/TB)+1
5250 MA=TR*TB/B
5260 PRINT:PRINT"Voor de ";CHR$(34);NR;CHR$(34);" opgegeven adres-"
5270 PRINT"blokken zijn ";CHR$(34);TR;CHR$(34);" tracks nodig."
5280 PRINT"U kunt hierin maximaal ";CHR$(34);MA;CHR$(34);" adres-"
5290 PRINT"blokken opslaan."
5300 PRINT:PRINT
5310 PRINT"    1>  Het aantal adresblokken is juist."
5320 PRINT"    2>  Wijzig het aantal adresblokken."
5330 PRINT"    3>  Stop het opzetten van de adres-file. "
5340 PRINT:PRINT
5350 INPUT"Uw keus ";A
5360 IFA<1ORA>3GOTO5300
5370 PRINT:ONAGOTO5400,5390
5380 GOTO6020
5390 PRINT:INPUT"Het nieuwe aantal adresblokken is ";NR:GOTO5220
5400 FORI=0TONT-TR+1:IFT(I)=1GOTO5430
5410 FORJ=ITOI+TR-1:IFT(J)=1GOTO5430
5420 NEXTJ:GOTO5540
5430 NEXTI
5440 PRINT:PRINT"Op de diskette is voor de file"
5450 PRINTCHR$(34);NA$;CHR$(34);" niet meer genoeg"
5460 PRINT"geheugenruimte vrij.":PRINT:PRINT
5470 PRINT"    1>  Wijzig het maximale aantal"
5480 PRINT"        adresblokken."
5490 PRINT"    2>  Het opzetten van de nieuwe adres-file afbreken."
5500 PRINT:PRINT
5510 INPUT"Uw keus ";A:IFA=1GOTO5390
5520 IFA<>2GOTO5440
5530 GOTO6020
5540 T0=FNA(I):T9=FNA(I+TR-1):GOSUB5970
5550 IFS<>0GOTO5600
5560 IFS=0THENPRINT"Geen plaats meer in de inhoud (directory)."
5570 PRINT"U kunt eventueel een andere file wissen."
5580 PRINT"Zoniet, dan stop een andere diskette in de drive ."
5590 GOTO6020
5600 FORJ=1TOLEN(NA$)
5610 POKEI+J-1,ASC(MID$(NA$,J)):NEXT
5620 POKEI+6,T0:POKEI+7,T9
5630 A$="SA 12,"+MID$(STR$(S),2)+"=2E79/1":DISK!A$
5640 T0=FNB(T0):T9=FNB(T9)
5650 FOR I=T0 TO T9
5660 B$=RIGHT$(STR$(I),2):IF VAL(B$)<10 THEN B$="0"+RIGHT$(B$,1)
5670 A$="SA "+B$+",1=317E/8"
5680 DISK!A$
5690 NEXT
5700 POKE10081,96
5710 DISKOPEN,6,NA$
5720 FORX=0TONR-1:PRINT"Adresblok -- ";X+1;" -- van";NR
5730 DISKGET,X
5740 FORY=1TO127
5750 PRINT#6,CHR$(35);
5760 NEXT
5770 PRINT#6,CHR$(13);
5780 DISKPUT
5790 NEXT
5800 DISKCLOSE,6
5810 POKE10081,169:DISK!"GO 2761":GOTO6020
5820 PRINT"Zit de diskette waarop de file moet worden geplaatst"
5830 INPUT"in drive A, B, C of D ";D$:PRINT:PRINT:D1=LEN(D$)
5840 IFD1<1GOTO5820
5850 IFD1=0THEND$=CHR$(D):D1=ASC(D$):IF D1<65ORD1>68GOTO5820
```

```
5860 IFD<>D1THENDISK!"SE "+D$:D=D1
5870 RETURN
5880 DISK!"CA 2E79=12,1"
5890 GOSUB5910
5900 DISK!"CA 2E79=12,2"
5910 FORI=11897TO12152STEP8:IFPEEK(I)=35GOTO5950
5920 N$="":FORI1=0TO5:N$=N$+CHR$(PEEK(I+I1)):NEXT
5930 IFN$=NA$THENS=1
5940 T0=FNB(PEEK(I+6)):T9=FNB(PEEK(I+7))
5950 FORJ=T0TOT9:T(J)=1:NEXT
5960 NEXTI:RETURN
5970 S=1:DISK!"CA 2E79=12,1"
5980 GOSUB6000:IFS=1THENRETURN
5990 S=2:DISK!"CA 2E79=12,2
6000 FORI=11897TO12152STEP8:IFPEEK(I)=35THENRETURN
6010 NEXT:S=0:RETURN
6020 PRINT:PRINT"Kontroleer of de diskette met het ADRES-programma"
6030 PRINT"in drive A zit en druk dan op <M>"
6040 INPUT"voor terugkeer naar het hoofdmenu ";M$
6050 IFM$<>"M"ANDM$<>""GOTO6020
6060 IFD<>65THENDISK!"SE A"
6070 RUN"ADRES"

7000 REM FILE WISSEN
7010 DISK!"GO F32F"
7020 PRINTTAB(15)"==== File-naam wissen. ====":PRINT:PRINT:PRINT
7030 PRINT"Zit de diskette waarop de file moet worden gewist"
7040 INPUT"in drive A, B, C of D ";D$
7050 IFD$=""THEND$="A"
7060 IFD$<"A"ORD$>"D"ORLEN(D$)<>1THENPRINT:GOTO7030
7070 DISK!"SE "+D$
7080 D1=ASC(D$)
7090 PRINT:PRINT
7100 PRINT"Geef de naam van de file die gewist"
7110 INPUT"moet worden (max. 6 karakters) ";NA$
7120 IFLEN(NA$)>6THENPRINT:GOTO7100
7130 IFNA$=""THENRUN
7140 IFLEN(NA$)<6THENNA$=NA$+" ":GOTO7140
7150 DISK!"CA 2E79=12,1
7160 S=0:GOSUB7320
7170 IFS=1THENDISK!"SA 12,1=2E79/1":GOTO7230
7180 DISK!"CA 2E79=12,2"
7190 GOSUB7320:IFS=1THENDISK!"SA 12,2=2E79/1":GOTO7230
7200 PRINT:PRINT"***  ";CHR$(34);NA$;CHR$(34);" bestaat niet.  ***"
7210 PRINT:PRINT
7220 GOTO7260
7230 PRINT:PRINT
7240 PRINT"De file ";CHR$(34);NA$;CHR$(34);" is gewist in"
7250 PRINT"de inhoud (directory).":PRINT:PRINT
7260 PRINT"Kontroleer of de diskette met het ADRES-programma"
7270 PRINT"in drive A zit en druk dan op <M>"
7280 INPUT"voor terugkeer naar het hoofdmenu ";M$
7290 IFM$<>"M"ANDM$<>""GOTO7230
7300 IFD1<>65THENDISK!"SE A"
7310 RUN"ADRES"
7320 FORI=11897TO12152STEP8
7330 FORJ=ITOI+5
7340 IFPEEK(J)<>ASC(MID$(NA$,J-I+1,1))THEN7390
7350 NEXTJ
7360 FORJ=ITOI+5:POKEJ,ASC("#"):NEXTJ
7370 POKEI+6,0:POKEI+7,0
7380 S=1:RETURN
7390 NEXTI
7400 S=0:RETURN
```

```
1 REM ######################################################################
2 REM ######################################################################
3 REM ##                                                                  ##
4 REM ##                              INDISK                              ##
5 REM ##                                                                  ##
6 REM ##                van N. Hagemann - 20.05.1984                      ##
7 REM ##                                                                  ##
8 REM ######################################################################
9 REM ######################################################################
10 REG=57664:DAT=REG+1:POKE REG,10:POKE DAT,32:PRINT CHR$(27);"1"
20 PRINT TAB(24);"Diskette initialiseren""
30 PRINT TAB(24);:FOR I=1 TO 22:PRINT CHR$(135);:NEXT I:PRINT
40 PRINT:PRINT:PRINT
50 PRINT "Plaats de diskette in drive A en";
60 PRINT " druk op de RETURN-toets!"
70 PRINT TAB(22);:FOR I=1 TO 7:PRINT CHR$(135);:NEXT I
80 PRINT TAB(43);:FOR I=1 TO 12:PRINT CHR$(135);:NEXT I:PRINT
90 DISK!"GO F71D":IF PEEK(9059) <> 13 THEN 90
100 PRINT:PRINT:PRINT
110 PRINT "0";TAB(19);"10";TAB(39);"20";TAB(59);"30";TAB(78);"40"
120 PRINT CHR$(216);:FOR J=1 TO 3:FOR I=1 TO 19:PRINT CHR$(148);:NEXT I
130 PRINT CHR$(219);:NEXT J:FOR I=1 TO 18:PRINT CHR$(148);:NEXT I
140 PRINT CHR$(218)
150 FOR I=1 TO 7:PRINT:NEXT I
160 PRINT TAB(20);"De diskette wordt nu ge-initialiseerd...."
170 FOR I=1 TO 11:PRINT CHR$(27);"5";:NEXT I:PRINT
180 PRINT CHR$(233);:DISK!"SE A"
190 FOR I=1 TO 39:PRINT CHR$(233);CHR$(233);
200 DISK!"IN "+RIGHT$(STR$(100+I),2):NEXT I:PRINT CHR$(233)
210 PRINT TAB(5);"Directory wordt nu opgezet...."
220 FOR I=11897 TO 12145:POKE I,35:NEXT I
230 POKE 11897,ASC("D"):POKE 11898,ASC("I"):POKE 11899,ASC("R")
240 POKE 11900,ASC("E"):POKE 11901,ASC("C"):POKE 11902,ASC("T")
250 POKE 11903,18:POKE 11904,18
260 PRINT TAB(5);"Directory deel 1":DISK!"SA 12,1=2E79/1"
270 FOR I=11897 TO 11904:POKE I,35:NEXT I
280 PRINT TAB(5);"Directory deel 2":DISK!"SA 12,2=2E79/1"
290 PRINT:PRINT TAB(10);"Moet nog een diskette worden bewerkt ";
300 PRINT "(J/N) ?";CHR$(27);"4"
310 DISK!"GO F71D":I$=CHR$(PEEK(9059)):IF I$<>"J" AND I$<>"N" THEN 310
320 IF I$="J" THEN RUN
330 POKE REG,10:POKE DAT,0:PRINT CHR$(27);"1":END
340 REM ######################################################################
350 REM ######################################################################
360 REM ##                                                                  ##
370 REM ##                        Variabelen-lijst                          ##
380 REM ##                                                                  ##
390 REM ##      REG......Adresregister van CRT-Controller                   ##
400 REM ##      DAT......Dataregister van CRT-Controller                    ##
410 REM ##      I........Variabele in lus                                   ##
420 REM ##      J........Variabele in lus                                   ##
430 REM ##      I$.......Karakter van ingedrukte toets                      ##
440 REM ##                                                                  ##
450 REM ######################################################################
460 REM ##                                                                  ##
470 REM ##                      Programmabeschrijving                       ##
480 REM ##                                                                  ##
490 REM ##    Regel 1-9........Titel                                        ##
500 REM ##    Regel 10.........Beeldscherm en variabelen init.              ##
510 REM ##    Regel 20-80......Inleiding                                    ##
520 REM ##    Regel 90.........Wacht op RETURN-toets                        ##
530 REM ##    Regel 100-170....Balken tekenen                               ##
540 REM ##    Regel 180-200....Disk init. en balken weer tekenen            ##
550 REM ##    Regel 210-250....Directory opzetten                           ##
560 REM ##    Regel 260-280....Directory op diskette zetten                 ##
570 REM ##    Regel 290-320....Einde programma?                             ##
580 REM ##    Regel 330........Einde programma, terug naar BASIC            ##
590 REM ##                                                                  ##
600 REM ######################################################################
610 REM ######################################################################
```

Hoe komt men aan software?

Voordat we spreken over de verkrijgbaarheid van de software, willen we hier eerst iedereen danken, die heeft bijgedragen tot de suksesvolle software-ontwikkeling voor de Octopus 65. Met groot enthousiasme is hieraan gewerkt met de gedachte deze software aan alle geïnteresseerden ter beschikking te kunnen stellen, die er op niet-kommerciële basis mee willen werken. Let wel, dit is geen vrijbrief voor software-piraten, want de legale auteursrechten blijven gehandhaafd en kommerciële aktiviteiten met deze software zijn uit den boze!
Onze bijzondere dank gaat naar de Nederlandse computerclub "De 6502 kenners". Uit de hoofdstukken in deze special waar de Octopus 65 als ontwikkelingssyteem wordt besproken, blijkt zonneklaar dat de Micro-Ware-assembler hiertoe de grondslag vormt. De auteursrechten voor deze assembler zijn in handen van "De 6502 kenners", welke club de assembler voor niet-kommercieel gebruik op de Octopus 65 ter beschikking heeft gesteld.
Het lidmaatschap van deze club is voor iedere "6502-fan" een zeer interessante zaak. De club werkt met computers zoals de Octopus 65, Apple II, VIC20, Commodore 64...kortom alle computers met een 6502-CPU. Ieder clublid ontvangt elke twee maanden het clubtijdschrift "De 6502 kenner" dat boordevol staat met assembler- en BASIC-programma's, interessant voor elke programmaverzamelaar. Ook uit geheel andere overwegingen is deze club interessant voor nabouwers van de Octopus 65. Zij beschikt namelijk over een eigen software-"winkel", waar clubleden zeer goedkoop interessante software kunnen betrekken. Als voorbeeld noemen we een aan de Octopus 65 aangepast figForth, een Pascal en nog veel meer. Maar daarover kan de club zelf u nog veel beter inlichten. Zij werkt internationaal en heeft naast vele leden in Europa eveneens leden in Israel en Australië. Het lidmaatschap is dus een welkome aanvulling aan dat van uw plaatselijke club. Het adres is:

DE 6502 KENNERS
Jacob Jordaensstraat 15
2923 CK Krimpen a.d. IJssel
tel: 01807 — 19881 (werkdagen tussen 19 en 20 uur)

Met de 6502-club is de volgende afspraak gemaakt: Elke officiële bezitter van een set OHIO-DOS V3.3-diskettes (tutorial disk 1...5) kan bij "De 6502 kenners" een set aangepaste kopieën laten maken tegen een schappelijke prijs. Dat gaat als volgt: men stuurt 10 lege enkelzijdige of 5 lege dubbelzijdige diskettes, samen met een kopie van de rekening van de originele set OHIO-diskettes of een kopie van de originele labels op deze diskettes, naar het adres van "De 6502 kenners", met de mededeling "Octopus 65" en de vermelding "enkelzijdig" of "dubbelzijdig". Voorlopig kan men alleen 40-track-diskettes bestellen. We geven nog even het adres van de Nederlandse leverancier van de OHIO-DOS-diskettes:

Ingenieursburo Koopmans
Sluisweg 2h
Postbus 176
3370 AD Hardinxveld-Giessendam
tel. 01846-6833

In verband met juridische redenen moeten we er hier nog op wijzen dat men beslist geen aanspraak kan doen gelden op het verkrijgen van deze software, alsmede op volmaakte technische kwaliteit. Diverse lezers kunnen alleen via de post kopieën aanvragen (denk aan het insluiten van retourporto!). Gaat zo'n zending op een of andere wijze verloren, dan heeft men pech! Komt de zending bij de sorteerwerkzaamheden van de posterijen onverhoeds in een sterk magnetisch veld, dan heeft men ook pech! Hoewel dit alles onwaarschijnlijk klinkt, is het niettemin mogelijk en kunnen zich vergelijkbare situaties voordoen.

Literatuuroverzicht

6502-CPU-kaart
Elektuur nov. '83, pag. 11-36 e.v.
RS232 Centronics-interface
Elektuur okt. '84, pag. 10-34 e.v.
RS232/V24: de besturingssignalen
Elektuur nov. '84, pag. 11-70 e.v.
VIA-boek
uitgave Elektuur B.V.
VDU-kaart
Elektuur sept. '83, pag. 9-58 e.v.
Paperware 3
uitgave Elektuur B.V.
Dynamische RAM-kaart
Elektuur april '82, pag. 4-50 e.v.
64 K op de dynamische RAM-kaart
Elektuur sept. '83 pag 9-74 e.v.
Floppy-disk-interface, deel 1
Elektuur nov. '82 pag. 11-58 e.v.
Floppy-disk-interface deel 2
Elektuur dec. '82 pag. 12-70 e.v.
Motorschakeling voor floppy-drives
Elektuur apr. '84 pag. 4-71 e.v.
Busuitbreiding
Elektuur dec. '83 pag. 12-70 e.v.
Computervoeding
Elektuur ju/aug. '84 pag 7-60
ASCII-keyboard
Elektuur mei '83 pag 5-24 e.v.
SCART-adapter
Elektuur sept. '84 pag. 9-68 e.v.
VHF/UHF-modulator
Elektuur jan. '85 pag. 1-26 e.v.
Mini-printer
Elektuur nov. '84 pag. 11-34 e.v.
Kleine X-Y-plotter
Elektuur apr. '85 pag. 4-54 e.v.
Data per telefoon
Elektuur sept. '84 pag. 9-32 e.v.
Telektor
Elektuur sept. '84 pag. 3-38 e.v.
Enkeltoets -BASIC-kommando's
Elektuur dec. '84 pag. 12-44 e.v.
Bij het OHIO-DOS-pakket wordt meegeleverd:
* 65D Tutorial and Reference Manual, Ohio Scientific, august 1981
* BASIC Reference Manual, Ohio Scientific, 1981
* Assembler, Editor and Extended Monitor Reference Manual, Ohio Scientific,

Lijst van tot nu toe beschikbare diskettes

Disk 1: "SYSTEM LOYS"'
Track 0...1 OS-65D V3.3
track 2...5 BASIC V3.3
track 7...9 Micro-Ware-assembler & editor & DOS-link
track 10...11 OS-65D V3.2

Disk 2: "OSI-tutorial-disk II"

Disk 3: "OSI-tutorial-disk III"

Disk 4: "OSI-tutorial-disk IV"

Disk 5: "OSI-tutorial-disk V"
De diskettes 2...5 zijn aan Octopus 65 aangepast. Op disk 5 bevindt zich de MOS-Technology-assembler.

Disk 6: OCTOPUS-source, deel 1
SAMVID: Video-driver in de Octopus-EPROM
FLOPPY: Track-zero lader in de Octopus-EPROM
SAMDIS: 6502-disassembler in de Octopus-EPROM
SAMMON: Debugger in de Octopus-EPROM
SAM2.2: Schrijft OS-65D V3.2 over OS-65D V3.3 (overlay)
SA1BAW: Transient-processor-handler
SHOASM: Source van de Steno-BASIC
NEWDOS: Source van de DOS-uitbreidingen

Disk 7: OCTOPUS-source, deel 2
SAMASM: Editor van de Micro-Ware-assembler & DOS-link
BTLOYS: Boot-strap-loader voor "System LOYS"
BTDSK2: Boot-strap-loader voor "OS-65D V3.2
BTOSI: Boot-strap-loader voor OS-65D V3.3
MERBAS: BASIC-source van het file-merger-programma
MERASM: Assembler-source van het file-merger-programma
ERASE: Source van de ERASE-utility
EDIASM: Source van de BASIC-full-screen-editor

Disk 8: Wordprocessor-source deel 1
ED.ARI: Arithmetic-subroutines van de wordprocessor
ED.SUB: Screen-handler-subroutines van de wordprocessor

Disk 9: Wordprocessor-source deel 2
ED.MAI: Word-processor hoofdprogramma en DOS-interface.

Disk 10fig-Forth