Het MON65/DOS65 computersysteem

MON65/DOS65 is als aanvulling op een reeds bestaand 6502 systeem te leveren. Deze aanvulling bestaat uit een pakket software, eventueel een print voor de floppy disk controller, een nederlandstalige handleiding van de software en een hardware beschrijving van het systeem waarop MON65/DOS65 gaat draaien. Met dit pakket is het mogelijk uw 6502 systeem om te bouwen tot een professioneel 6502 ontwikkelsysteem. De begeleiding van dit systeem is optimaal doordat de ontwikkeling van dit systeem gedaan is door leden van de Kim gebruikersclub Nederland en daar nog steeds actief mee bezig zijn. Zijn er problemen wat betreft de hardware of de software dan kan men zich altijd tot de Kim gebruikersclub wenden. Ook op- en aanmerkingen zijn welkom omdat dit systeem in tegenstelling tot een heleboel andere systemen echt 'leeft'. Geregeld komen er nieuwe utilities en worden er aardigheden en opmerkingen geplaatst in het clubblad 'De 6502 Kenner'. Dit systeem zal volledig worden gesteund door de redactie van dit clubblad. Leden van de Kim gebruikerclub zullen dus steeds via het clubblad of via de landelijke bijeenkomsten kunnen vernemen wat voor extra's er leverbaar zijn. Als er een nieuwe versie van het operating system uitkomt zullen de gebruikers het via deze kanalen te weten komen. Het spreekt voor zich dat men de beste ondersteuning heeft van MON65/DOS65 als lid van de Kim gebruikerclub Nederland.

- Geschikt voor single/double sided, single/double density disk-drives, mits 40 tracks per kant.
- Maximale geformatteerde schijf-kapaciteit: 320 Kbyte. (DD,DS) (vrije data-kapaciteit 304 Kbyte)
- Ondersteunt: FULL SCREEN EDITOR (ED)
  ASSEMBLER (Moser, Micro-Ade)
  FORTH
  BASIC
- Uitgebreide input/output redirect mogelijkheden.
- Zeer gebruikersvriendelijk.
- Aan te passen voor vrijwel elk 6502 systeem.
- Zeer efficient schijfruimte gebruik.
- Volledige dokumentatie beschikbaar.
- Overdrachtsnelheid ca. 7 Kbyte/seconde.
- Eenvoudige hardware.

Hard- en software eisen:

- DOS65 floppy-controller kaart. De print hiervoor is leverbaar via de KIM Gebruikersclub Nederland.
- Minimaal 48 Kbyte RAM geheugen.
- Terminal eisen: 80 koloms, cursorbesturing. (Elekterminal is dus niet bruikbaar.)
- HAVISOFT monitorprogramma MON65 voor implementatie op de ELEKTUUR CPU/VDU konfiguratie.

Wat is er allemaal leverbaar?

- Nederlandstalige handleiding MON65, de monitor. 51 pagina's
- Nederlandstalige handleiding DOS65 en utilities. Hierbij horen ook 2 pagina's installatie handleiding. 28 pagina's
- Nederlandstalige handleiding screeneditor. 19 pagina's
- Nederlandstalige beschrijving van de hardware. 14 pagina's
- 2764 Eprom met MON65.
- 2732 Eprom met karaktergenerator voor VDU kaart.
- Diskette met DOS65 en utilities.
- Print voor FDC. Deze kaart kunnen we zeer bij u aanbevelen daar hiermee de bouw van de floppy disk controller kaart zeer vereenvoudigd wordt. Het is een dubbelzijdige doorgemetalliseerde van soldeermasker en tekstopdruk voorziene print. Bij de hardware beschrijving van het systeem is een speciaal hoofdstuk gewijd aan de bouw van deze kaart.
- Source listing van MON65. Ongeveer 80 pagina's met informatie over de 8 kbyte monitor in uw systeem. Deze source is in MOSER-formaat en voorzien van Engelstalig commentaar.
- Source listing van DOS65. Ongeveer 70 pagina's met informatie over het 8 kbyte disk operating system. Deze source is in MOSER-formaat en voorzien van Engelstalig commentaar.
- Source listing van de utilities. Deze source is in MOSER-formaat en voorzien van Engelstalig commentaar. Ongeveer 80 pagina's.

De diskette bevat de volgende software :

- BOOT, disk operating system.
- EDITOR, full screen editor, wordprocessor.
- eddhlp.edd, Nederlandstalige editor-help-file die vanuit de editor aangeroepen kan worden.
- eddhlp.edd(e), Engelstalige editor-help-file.
- MIADE , Micro ADE assembler.
- FORTH
- LOGIN.COM, command file die na opstarten van DOS65 automatisch wordt aangeroepen.
- LOGIN.DAT, datafile die een mededeling op het beeldscherm zet na het aanroepen van de LOGIN.COM file.
- Diverse utilities, deze worden beschreven in de DOS65 handleiding.

- >D	- DUMP	- RENAME
- APPEND	- FORMAT	- SDIR
- CAT	- LIST	- SETDRIVES
- CLEAR	- MEMFILL	- SETMODE
- COPY	- MEMMOVE	- SETSTAD
- CREATE	- PLIST	- TIME
- DELETE	- PRINT	- TYPE
		- UNEXPAND

Het MON65/DOS65 computersysteem

MON65/DOS65 is als aanvulling op een reeds bestaand 6502 systeem te leveren. Deze aanvulling bestaat uit een pakket software, eventueel een print voor de floppy disk controller, een nederlandstalige handleiding van de software en een hardware beschrijving van het systeem waarop MON65/DOS65 gaat draaien. Met dit pakket is het mogelijk uw 6502 systeem om te bouwen tot een professioneel 6502 ontwikkelsysteem. De begeleiding van dit systeem is optimaal doordat de ontwikkeling van dit systeem gedaan is door leden van de Kim gebruikersclub Nederland en daar nog steeds actief mee bezig zijn. Zijn er problemen wat betreft de hardware of de software dan kan men zich altijd tot de Kim gebruikersclub wenden. Ook op- en aanmerkingen zijn welkom omdat dit systeem in tegenstelling tot een heleboel andere systemen echt 'leeft'. Geregeld komen er nieuwe utilities en worden er aardigheden en opmerkingen geplaatst in het clubblad 'De 6502 Kenner'. Dit systeem zal volledig worden gesteund door de redactie van dit clubblad. Leden van de Kim gebruikerclub zullen dus steeds via het clubblad of via de landelijke bijeenkomsten kunnen vernemen wat voor extra's er leverbaar zijn. Als er een nieuwe versie van het operating system uitkomt zullen de gebruikers het via deze kanalen te weten komen. Het spreekt voor zich dat men de beste ondersteuning heeft van MON65/DOS65 als lid van de Kim gebruikerclub Nederland.

Het basispakket bestaat uit de volgende delen :

- Nederlandstalige handleiding MON65, de monitor.
- Nederlandstalige handleiding DOS65 en utilities.
- Nederlandstalige handleiding screeneditor.
- Nederlandstalige beschrijving van de hardware.
- 2764 Eprom met MON65.
- 2732 Eprom met karaktergenerator voor VDU kaart.
- Diskette met DOS65 en utilities.

De diskette bevat de volgende software :

- BOOT, disk operating system.
- EDITOR, full screen editor, wordprocessor.
- eddhlp.edd, Nederlandstalige editor-help-file die vanuit de editor aangeroepen kan worden.
- eddhlp.edd(e), Engelstalige editor-help-file.
- MIADE , Micro ADE assembler.
- FORTH
- LOGIN.COM, command file die na opstarten van DOS65 automatisch wordt aangeroepen.
- LOGIN.DAT, datafile die een mededeling op het beeldscherm zet na het aanroepen van de LOGIN.COM file.
- Diverse utilities, deze worden beschreven in de DOS65 handleiding.

- >D	- MEMMOVE
- APPEND	- PLIST
- CAT	- PRINT
- CLEAR	- RENAME
- COPY	- SDIR
- CREATE	- SETDRIVES
- DELETE	- SETMODE
- DUMP	- SETSTAD
- FORMAT	- TIME
- LIST	- TYPE
- MEMFILL	- UNEXPAND

Als extra's op het basispakket kan het volgende geleverd worden :

- Print voor FDC. Deze kaart kunnen we zeer bij u aanbevelen daar hiermee de bouw van de floppy disk controller kaart zeer vereenvoudigd wordt. Het is een dubbelzijdige doorgemetalliseerde van soldeermasker en tekstopdruk voorziene print. Bij de hardware beschrijving van het systeem is een speciaal hoofdstuk gewijd aan de bouw van deze kaart.
- Source listing van MON65. Ongeveer 100 pagina's met informatie over de 8 kbyte monitor in uw systeem. Deze source is in MOSER-formaat en voorzien van Engelstalig commentaar.
- Source listing van DOS65. Ongeveer 100 pagina's met informatie over het 8 kbyte disk operating system. Deze source is in MOSER-formaat en voorzien van Engelstalig commentaar.
- Source listing van de utilities. Deze source is in MOSER-formaat en voorzien van Engelstalig commentaar.

De prijzen van de diverse onderdelen :

- Handleiding MON65 Handleiding DOS65 Handleiding editor Hardware beschrijving Samen ongeveer 110 pagina's	Fl.	50,=
- 2764 Monitor Eprom	Fl.	35,=
- 2732 Karaktergenerator Eprom	Fl.	25,=
- Diskette met software als boven beschreven	Fl.	15,=
- Floppy Disk Controller kaart	Fl.	50,=
- Source MON65	Fl.	25,=
- Source DOS65	Fl.	25,=
- Source utilities	Fl.	25,=

Deze prijzen zijn exclusief verzendkosten en gelden alleen voor leden van de Kim gebruikersclub Nederland. Ieder lid komt slechts eenmalig voor deze prijzen in aanmerking.

Inhoudsopgave Blz.

Inhoudsopgave.....1
Inleiding.....3
1 Commando's.....4
1.01 Overzicht geldige commando's en karakters.....5
1.02 Hardware interrupts.....6
1.03 @.....7
1.04 A.....8
1.05 B.....9
1.06 C.....10
1.07 D.....11
1.08 E.....12
1.09 F.....13
1.10 H.....13
1.11 L.....14
1.12 M.....15
1.13 P.....15
1.14 Q.....16
1.15 S.....16
1.16 T.....17
1.17 V.....17
1.18 W.....18
1.19 X.....19
1.20 Y.....19
1.21 +.....19
1.22 -.....19
1.23 $.....20
1.24 #.....20
1.25 <.....20
1.26 Overige geldige karakters.....21
1.27 Foutmeldingen.....23
2 I/O devices.....24
3 Functie toetsen met hun functies.....26
3.01 ^^B.....26
3.02 ^^E.....26
3.03 ^^I.....27
3.04 ^^O.....28
3.05 ^^P.....28
3.06 ^^Q.....28
3.07 ^^S.....28
4 De statusregel.....29
5 De jumptabel.....31
6 Variabelen.....31
7 Monitor routines.....44
7.01 De beeldscherm output routine.....44
7.02 De break.....47
7.03 De Viacom routines.....48
7.04 De interrupt vectoren.....51
7.05 RS232 interface.....51

MON65 is ontwikkeld door : E.J.M. Visschedijk
Drakensteyn 299
7608 TR Almelo

Deze manual is geschreven door : E.J.M. Visschedijk

Voor de monitor programmatuur evenals deze manual geldt :
Copyright (c) 1985 Kim gebruikersclub Nederland

Inleiding

Deze 6502 monitor bevat de elementaire systeemsoftware voor uw 6502 systeem. Deze 8 Kbyte grote monitor bestaat voor 4 Kbyte uit I/O routines, waaronder de VDU-kaart routines, interrupt routines, printer routines, RS232 routines enz. De andere 4 Kbyte wordt gebruikt voor de monitor commando's als hexdump, disassembler, move-, fill-, check- en verify-instructies. Deze handleiding gaat in op het gebruik van de monitor op zich en op het gebruik van monitorroutines door gebruikers-programma's of door DOS65.

In hoofdstuk 1 worden de commando's en de bijbehorende syntax behandeld evenals de foutmeldingen. Hoofdstuk 2 gaat over I/O devices en het gebruik ervan. Ook in deze monitor zijn functietoetsen geimplementeerd. Het gebruik en de eigenschappen hiervan worden in hoofdstuk 3 nader uitgelegd. De statusregel die onderin het beeld staat wordt in hoofdstuk 4 behandeld. In hoofdstuk 5 wordt de monitor jumptabel beschreven. Hoofdstuk 6 bevat de uitleg van de in de monitor gebruikte variabelen en wat de gebruiker er eventueel mee zou kunnen doen. Hierin kunnen we ook de absolute adressen weervinden van de variabelen waarnaar door de hele manual heen verwezen wordt. In hoofdstuk 7 tenslotte worden een aantal monitor routines behandeld zodat de gebruiker een beter inzicht zal krijgen in het gebruik ervan.

1. Commando's

Als het systeem wordt aangezet, moet de monitor de text

MON65 1.XX
HaViSoft

laten zien. Waarbij XX een getal tussen de 00 en 99 mag zijn, die op de versie duidt. Op dat moment zit men in de command mode van de monitor en kunnen er commando's worden gegeven. De commando's bestaan uit een (1) karakter al dan niet met een of meerdere parameters. Het commando wordt door een spatie gescheiden van de parameters. De parameters onderling worden door een komma gescheiden. Bij opgave van een normale parameter, hieronder versta ik een parameter die een adres voorstelt, worden de 4 laatst ingetypte karakters als parameter gezien. Worden minder dan 4 karakters ingetypt, dan worden voor de parameter nullen gezet, zodat men toch op een adres van 4 nibbles uitkomt.
Voorbeeld: Typt men bv. 73EA7F dan is EA7F de parameter.
Typt men echter D4 dan wordt 00D4 als parameter gezien.
Er zijn ook commando's waarbij slechts 1 byte als parameter opgegeven dient te worden. Alleen de twee laatst ingetypte karakters worden genomen, en indien slechts 1 karakter wordt ingetypt, dan ziet de monitor een 0 voor het karakter.

Hieronder volgt een opsomming van de geldige commando's met hun syntax. AAAA, BBBB en CCCC moet men zien als willekeurige adressen. DD is een willekeurig byte. Na een apostrof (') mag een ascii karakter ingegeven worden. Dit wordt aangegeven met 'P. Soms mag er na ' een ascii string volgen. Deze string wordt aangegeven met 'PQR. Bij de gegeven voorbeelden is na een punt-komma (;) commentaar gegeven. Dit moet dus niet ingetypt worden!!

De monitor accepteert alleen hoofdletters als commando's. Als er een string ingetypt moet worden, bv. bij het commando W, dan mogen ook kleine letters gebruikt worden.

Als er een controlkarakter bedoeld wordt, dan wordt dit aangegeven met een circonflexe (^) voor het karakter. Een control karakter is een karakter met een asciiwaarde tussen $00 en $1F en wordt gegenereerd door de controltoets tegelijkertijd met de gevraagde toets in te typen.

Het is mogelijk meerdere commando's op een regel te geven. De commando's met bijbehorende parameters moeten dan worden gescheiden door een dubbele punt (:).

Om de vorige ingetypte commandoregel weer op het beeldscherm te toveren kan ^Y ingetypt worden. Met de DELETE toets kunnen vanaf het eind van de regel de karakters weggehaald worden. Na een tweede ^Y verschijnen de karakters rechts van de cursor weer tot aan het eind van de regel.

De monitor kent geen prompt. Dit is een karakter dat op het beeldscherm wordt gezet als de monitor een commando heeft uitgevoerd en de gebruiker een nieuw commando mag geven. De monitor leest karakters in als de cursor zichtbaar is. Tijdens de uitvoering van commando's is er geen cursor.

1.01 De 23 geldige commando's zijn :

1 @ - Haal adresinhoud, laat inhoud zien en wijzig data.
2 A - Laat de datum zien of wijzig de datum.
3 B - Laat breaktrap positie zien, set breaktrap, of haal breaktrap weg.
4 C - Check een geheugengebied op een bepaalde inhoud.
5 D - Disassembleer een geheugengebied of een beeldschermpagina.
6 E - Execute vanaf een opgegeven adres of vanaf het breakpoint.
7 F - Fill een geheugengebied met bepaalde data.
8 H - Hexdump van een geheugengebied.
9 L - List CPU registers.
10 M - Move een geheugengebied.
11 P - Maak het beeldscherm schoon. (formfeed)
12 Q - Verlaat de monitor met een RTS instructie.
13 S - Bepaal de checksum van een geheugengebied.
14 T - Laat de tijd zien of wijzig de tijd.
15 V - Verify twee geheugengebieden.
16 W - Zoek in een geheugengebied naar een string, of een bepaalde byte volgorde.
17 X - Spring naar een user gedefinieerde routine.
18 Y - Spring naar een user gedefinieerde routine.
19 + - Tel twee 16 bit hex getallen op.
20 - - Trek twee 16 bit hex getallen van elkaar af.
21 $ - Reken een decimaal getal om naar hex.
22 # - Reken een hex getal om naar decimaal.
23 < - Herhaal de commandoregel.

Overige geldige karakters zijn :

: - Scheiding tussen twee commando's.
, - Scheiding tussen parameters onderling.
<SPATIE> - De scheiding tussen een commando en de eerste parameter.
<DELETE> - Haalt het laatst ingetypte karakter weg.
^C - Break commando, breekt een lopend programma af.
^S - Stop een output naar het beeldscherm tijdelijk.
^Q - De met ^S gestopte uitvoer wordt weer hervat.
De toetsen ^S en ^Q zijn te vervangen door een toets die de waarde $80 kan afgeven. Dan werkt deze toets als een toggle. Na de eerste keer intypen stopt de uitvoer, na de tweede keer wordt de uitvoer hervat.
^Y - Roept de laatst ingetypte commandoregel nog eens op.
<RETURN> - Afsluiting van een commandoregel.

1.02 Hardware interrupts:

RESET - Wordt op de RESET-toets gedrukt, dan worden alle variabelen opnieuw in hun default toestand gezet. Het uurwerk blijft gewoon doorlopen en hoeft meestal (hangt af van de reden van de RESET) niet meer op tijd gezet te worden. Het toetsenbord en het beeldscherm worden weer geinitialiseerd zodat altijd de controle over het systeem terug is.

NMI - Na een NMI wordt er via de NMIVECTOR naar een routine gesprongen die de processor registers op het beeldscherm zet. Als een zelf- geschreven programma 'hangt' wil deze toets nog wel eens uitkomst bieden omdat men dan kan zien waar het programma vastzit. Deze toets mag niet denderen daar anders de processor register inhouden van de NMI routine op het beeldscherm worden gezet.

IRQ - Er wordt via de vector IRQVECTOR naar de IRQ routine gesprongen. Hierin wordt het uurwerk afgehandeld evenals het toetsenbord, dat ook onder interrupt draait. Het is erg gevaarlijk deze vector zelf te wijzigen omdat dan al gauw de controle over het toetsenbord wegvalt en de interrupt van het uurwerk niet goed afgehandeld wordt.

1.03 Commando @

Dit commando is er voor om een geheugenlocatie te openen. Er kan dan naar de inhoud van een adres gekeken worden. Het is tevens mogelijk ascii teken dat bij dit geheugenadres hoort, te bekijken. De syntax is alsvolgt:

@ AAAA

De monitor keert terug met het geheugenadres en de bijbehorende inhoud. Wil men de ascii betekenis van de inhoud weten, dan moet ' ingetypt worden. Direkt na het ' karakter wordt dan de ascii betekenis gegeven. Wil men de inhoud van dit adres wijzigen, dan moet eenvoudigweg het nieuwe byte ingetypt worden. Alleen karakters die een byte kunnen vormen worden hier geaccepteerd. Dat houdt in de karakters 0 t/m 9 en A t/m F. Om uit deze mode te komen en terug te keren naar de command mode moet een return getypt worden. Ook is het toegestaan de karakters te typen waarvan de ascii waarden $0A (linefeed) en $0B (vertical tab) zijn. Na een linefeed wordt de volgende geheugenlocatie geopend. Na een vertical tab wordt de vorige geheugenlocatie geopend. Na deze commando's kan men weer een nieuw byte intypen of een ' om de ascii waarde te bekijken. Na een return, linefeed of vertical tab wordt het eventueel veranderde geheugenadres pas werkelijk gewijzigd. Bij het onderstaande voorbeeld wordt ook het afsluitkarakter gegeven tussen < > dit om duidelijk te maken hoe men op een ander adres komt.

Voorbeeld :

```
@ CA00<return>                ;Geheugenadres wordt geopend
CA00 02 41<linefeed>          ;Inhouden overschrijven
CA01 FF 42<return>            ;Afsluiten met return


@ CA00<return>                ;Opnieuw openen
CA00 41 'A<linefeed>          ;Inhoud is veranderd
CA01 42 'B<vert. tab>         ;Na ' teken de ascii waarde
CA00 41<return>
```

1.04 Commando A

Met dit commando is het mogelijk de huidige datum te laten zien of de datum goed te zetten. De datum verschijnt ook op de statusregel. De datum en het uurwerk werken op interrupt basis. De datum wordt om middernacht verzet. Er wordt rekening gehouden met schrikkeljaren. Het is niet mogelijk iedere willekeurige datum in te voeren. Een datum voor 1985 bv. is niet mogelijk. Men moet zelf rekening houden met andere data die niet kunnen, zoals bv. 30 februari. Wordt dit wel gedaan, dan komt de datumroutine in de problemen. Is de datum correct ingegeven, dan wordt deze wel correct bijgehouden. Typt men alleen een A dan laat de monitor de datum zien. Wil men de datum wijzigen dan is de syntax:

A dd,mm,jj

Hierin zijn de kleine letters decimale getallen, waarbij dd de dag, mm de maand en jj het jaar is. De grenzen waartussen de in te voeren parameters moeten liggen zijn achtereenvolgens 1 en 31, 1 en 12, 85 en 99.

Voorbeeld :

A 28-05-85	;Hier werd alleen een A getypt, gevolgd ;door <return>
A 18,3,86	;Dit werd allemaal ingetypt
A 18-03-86	;Dit is het resultaat

1.05 Commando B

Met dit commando kan een breakpoint worden gezet. Er kan slechts een (1) breakpoint worden gezet. Dit kan alleen van een geheugenadres dat op een ramadres staat. Er kan dus geen breakpoint in een EPROM worden gezet. Ook is het mogelijk het breakpoint weer op te heffen, of het huidige breakpointadres te laten zien.

Een breakpoint wordt gebruikt om een machinetaal routine te debuggen (fouten eruit halen). Er wordt dan op een bepaald adres $00 neergezet. Dit is de opcode voor een break. Dit breakpoint moet wel een andere opcode vervangen en mag dus niet op de plaats van een operand worden gezet, daar anders alleen de operand wordt gewijzigd en er dus geen breakpoint is geplaatst. Loopt nu de processor op een $00 dan wordt het lopende programma afgebroken en wordt via een vector, de interrupt vector IRQVECTOR, naar een routine gesprongen die 'ziet' dat er een software break heeft plaatsgevonden. Op dat moment worden de register inhouden van de processor op het beeldscherm gezet. Zo kan er gekeken worden of die inhouden correct zijn. Typt men daarna als commando alleen een E dan wordt het breakpoint hersteld en wordt het programma vervolgd vanaf het adres waarop het breakpoint stond. De syntax is :

```
B AAAA
```

Zo werd een breakpoint op adres AAAA gezet. Dit moet een opcode adres zijn!!
Het is mogelijk een breakpoint weer weg te halen. De syntax is :

```
B R
```

Als men niet meer weet waar een breakpoint stond, dan wordt ingetypt :

```
B S
```

Met dit commando laat de monitor het breakpointadres zien met de inhoud die er oorspronkelijk stond.

Voorbeeld :

```
B CA10                      ;Op $CA10 wordt een breakpoint gezet

B S                         ;Laat het breakpointadres zien
Breakpoint: $CA10 02        ;Adres met oorspronkelijke inhoud

B R                         ;Haal het breakpoint weg

B S                         ;Laat het breakpointadres zien
No breakpoint
```

1.06 Commando C

Dit commando wordt gebruikt om een geheugengebied te checken op een bepaalde inhoud. Op ieder adres binnen dit gebied moet dus dezelfde waarde staan. De syntax is :

C AAAA,BBBB,DD

Er wordt binnen het gebied AAAA tot en met BBBB gekeken of er wel DD staat. Is dit niet het geval, dan worden de onjuiste adressen op het beeldscherm gezet met de inhoud die ze hebben.

C AAAA,BBBB,'P

Nu wordt naar de ascii waarde van het ingetypte karakter gekeken.
Het commando C kan heel handig gebruikt worden om te kijken of een EPROM die geprogrammeerd moet worden wel leeg is. Stel de EPROM staat op de adressen $B000 t/m $BFFF. Er wordt dan ingetypt :

C B000,BFFF,FF

Als de EPROM leeg is gaat de cursor naar de volgende regel en er gebeurt verder niets. Is de EPROM echter niet leeg, dan worden de foute adressen met de inhouden op het beeldscherm gezet.

Voorbeeld :

```
@ CA00                          ;Open een adreslocatie
CA00 40 41                      ;Zet er $41 in t/m $CA06
CA01 42 41
CA02 43 41
CA03 FF 42                      ;Zet hier $42 in
CA04 00 41
CA05 FF 41
CA06 00 41

C CA00,CA06,41                  ;Check het gebied op $41
CA03 42                         ;Hier staat geen $41

C CA00,CA06,'A                  ;Check het gebied op A
CA03 42                         ;Dit is geen A

@ CA03                          ;Open het foute adres
CA03 42 41                      ;Herstel in $41

C CA00,CA06,41                  ;Nu is het hele gebied goed
```

1.07 Commando D

De monitor is uitgerust met een dissassembler. Met het commando D wordt er een stuk geheugen gedisassembleerd volgens de codes van een 65C02. Met de inhoud van een vlag kan er gekozen worden of er volgens Rockwell of Synertek gedisassembleerd moet worden. Dit is de variabele C02TYP. Als de inhoud $01 is wordt er voor een Rockwell 65C02 gedisassembleerd, met de inhoud $00 is het een Synertek/GTE disassembler. De default waarde is $01. De syntax is alsvolgt :

D AAAA,BBBB

Er wordt vanaf AAAA tot en met BBBB gedisassembleerd. Nadat de laatste regel gedisassembleerd is komt men weer in de commando mode. Ook is het mogelijk een pagina te disassembleren. De syntax wordt dan :

D AAAA

Er wordt vanaf AAAA gedisassembleerd en wel 20 regels. Na het intypen van een willekeurige toets (behalve de spatie- of de returntoets) wordt er een regel vrijgelaten, waarna er weer 20 regels gedisassembleerd worden. Na een spatie wordt direct na de 20 regels een nieuwe regel gedisassembleerd. Na een return wordt de disassembler verlaten en komt men weer in de commando mode en kan een nieuw commando gegeven worden. Een listing is tijdelijk te stoppen met ^S. Er wordt weer vervolgd met ^Q. ^C om de listing af te breken werkt natuurlijk ook.

Het aantal regels dat gedisassembleerd wordt is instelbaar met de variabele DISLCT. Hierin staat default $14.

Voorbeeld :

```
D E048,E076                                ;Geef het commando
E048  4C 4E F3          JMP  $F34E         ;Er wordt continue
E04B  4C D4 E6          JMP  $E6D4         ;gedisassembleerd
E04E  D8                CLD
E04F  78                SEI
E050  A9 00             LDA  #$00
E052  A2 BB             LDX  #$BB
E054  9D FF CD          STA  $CDFF,X
E057  CA                DEX
E058  D0 FA             BNE  $E054
E05A  CA                DEX
E05B  9A                TXS
E05C  A0 00             LDY  #$00
E05E  B9 91 E0          LDA  $E091,Y
E061  F0 0A             BEQ  $E06D
E063  AA                TAX
E064  B9 C9 E0          LDA  $E0C9,Y
E067  9D 00 CE          STA  $CE00,X
E06A  C8                INY
E06B  D0 F1             BNE  $E05E
E06D  20 00 E1          JSR  $E100
E070  20 30 E1          JSR  $E130
E073  20 92 E1          JSR  $E192
E076  20 7F EB          JSR  $EB7F
```

```
D E048                                  ;Geef commando voor 20 regels.
E048  4C 4E F3          JMP  $F34E
E04B  4C D4 E6          JMP  $E6D4
E04E  D8                CLD
E04F  78                SEI
E050  A9 00             LDA  #$00
E052  A2 BB             LDX  #$BB
E054  9D FF CD          STA  $CDFF,X
E057  CA                DEX
E058  D0 FA             BNE  $E054
E05A  CA                DEX
E05B  9A                TXS
E05C  A0 00             LDY  #$00
E05E  B9 91 E0          LDA  $E091,Y
E061  F0 0A             BEQ  $E06D
E063  AA                TAX
E064  B9 C9 E0          LDA  $E0C9,Y
E067  9D 00 CE          STA  $CE00,X
E06A  C8                INY
E06B  D0 F1             BNE  $E05E
E06D  20 00 E1          JSR  $E100     ;Na een spatie...
E070  20 30 E1          JSR  $E130     ;Na een spatie...
E073  20 92 E1          JSR  $E192     ;Na een return...
```

1.08 Commando E

Om een machinetaalprogramma vanuit de monitor te runnen wordt het commando E(xecute) gegeven. Het is ook mogelijk om vanaf een breakpoint het programma te vervolgen. Is er namelijk een breakpoint gezet op een bepaald adres en wordt het programma gestart en het programma stopt op het breakpoint, dan wordt, na het commando E zonder verdere parameters, het programma vervolgt vanaf het breakpoint. Op dat moment wordt de inhoud van het breakpoint hersteld, oftwel het breakpoint wordt opgeheven. De normale syntax om een programma op te starten is:

E AAAA

Het programma dat op adres AAAA begint wordt hiermee opgestart. Als het programma wordt afgesloten met een RTS, dan wordt aan het eind van dit programma weer naar de monitor gesprongen. Het machinetaalprogramma wordt als een subroutine aangeroepen. Dit commando wordt verder niet met een voorbeeld verduidelijkt.

1.09 Commando F

Met dit commando is het mogelijk een stuk geheugen te vullen met dezelfde data. De syntax is :

```
F AAAA,BBBB,DD
```

Nu wordt het geheugengebied AAAA tot en met BBBB gevuld met de data DD. Het is ook mogelijk een ascii waarde in te voeren. De syntax is dan :

```
F AAAA,BBBB,'P
```

Het geselecteerde geheugengebied wordt nu gevuld met de ascii waarde van het teken P.

Voorbeeld :

```
F 800,80F,42                    ;Vul het geheugengebied

H 800,80F                       ;Kijk wat er staat
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB

F 804,807,'q                    ;Vul een geheugengebied

H 800,80F                       ;Kijk wat er staat
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 42 42 42 42 71 71 71 71 42 42 42 42 42 42 42 42  BBBBqqqqBBBBBBBB
```

1.10 Commando H

Met dit commando is het mogelijk een hex- en asciidump van een geselecteerd geheugengebied te geven. De syntax is :

```
H AAAA,BBBB
```

Er wordt geen rekening gehouden met een aantal regels per pagina. Een listing kan tijdelijk gestopt worden met ^S. Er wordt vervolgd met ^Q en een listing afbreken gebeurd met ^C. Helemaal rechts van de listing staat de ascii representatie van de hex inhoud.

Voorbeeld :

Bij het commando F is al een voorbeeld voor het gebruik van het hexdump commando gegeven.

1.11 Commando L

Als er een breakpoint is gezet en het programma waarin dat breakpoint staat wordt afgebroken dan wordt eigenlijk het commando L uitgevoerd. Hiermee worden inhouden van de CPU registers getoond. De waarden die getoond worden zijn de inhouden op het moment van de break. Alleen een break instuctie en een NMI kunnen deze waarden wijzigen.

Voorbeeld :

```
0800  A9 01              LDA  #$01     ;Een simpel programma
0802  A2 02              LDX  #$02
0804  A0 04              LDY  #$04
0806  00                 BRK           ;Afsluiten met break

E 800                                  ;Opstarten
A  X  Y  PC   SP NV BDIZC              ;Tegen een BRK aangelopen
01 02 04 0808 F1 00110000
L                                      ;Geef het commando
A  X  Y  PC   SP NV BDIZC
01 02 04 0808 F1 00110000
```

A is de accu inhoud
X is de inhoud het X register
Y is de inhoud van het Y register
PC is de programcounter
SP is de stackpointer

De volgende bits stellen het statusregister voor
N is het negative bit
V is het overflow bit
B is het break bit
D is het decimal bit
I is het interrupt bit
Z is het zero bit
C is het carry bit

Let erop dat de programcounter altijd 2 hoger staat dan het adres waarop de BRK staat!!

1.12 Commando M

Met dit commando wordt een stuk geheugen verplaatst. De monitor zoekt zelf uit of er van beneden naar boven of andersom verplaatst moet worden om te voorkomen dat er een stuk geheugen overschreven wordt voordat dit zelf verplaatst is. De syntax is :

M AAAA,BBBB,CCCC

Het stuk geheugen AAAA tot en met BBBB wordt verplaatst naar CCCC en verder.

Voorbeeld :

```
H 800,80F                          ;We gaan uit van deze sitatie
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 42 42 42 42 41 41 41 41 41 41 41 41 41 41 41 41  BBBBAAAAAAAAAAAA

M 800,803,807                      ;Verplaats een stuk

H 800,807
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 42 42 42 42 41 41 41 42 42 42 42 41 41 41 41 41  BBBBAAABBBBAAAAA

M 800,806,801                      ;Verschuif een stuk 1 plaats

H 800,80F
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 42 42 42 42 42 41 41 41 42 42 42 41 41 41 41 41  BBBBBAAABBBAAAAA
```

1.13 Commando P

Hiermee is het mogelijk het beeldscherm schoon te maken. De syntax is :

P

1.14 Commando Q

Als de monitor vanuit de DOS wordt aangeroepen, dan wordt met dit commando terug naar DOS gesprongen. Ook als vanuit andere programma's de monitor wordt aangeroepen, dan is dit het commando om weer terug te komen. De monitor moet wel als subroutine zijn aangeroepen!!! Als men direct na aanzetten van de machine dit commando geeft, zal er een foutmelding komen die aangeeft dat dit een illegaal commando is. De syntax is:

Q

Voorbeeld :

```
$M                                      ;Vanuit DOS monitor aanroepen
MON65 1.01
 HaViSoft

D E000,E006                             ;Monitor commando
E000  4C 5C E7          JMP  $E75C
E003  4C 33 E7          JMP  $E733
E006  4C 4E E0          JMP  $E04E

Q                                       ;Verlaat monitor
$                                       ;Terug in DOS
```

1.15 Commando S

Met dit commando wordt de checksum bepaald van een bepaald geheugengebied. Deze checksum is een waarde tussen $0000 en $FFFF. Een checksum is een optelling van de inhouden van een geheugengebied. De checksum wordt vaak gebruikt om te kijken of een stuk geheugen foutloos is overgezonden van tape naar computer, of tussen twee computers onderling. Zo kan dit commando worden gebruikt om te kijken of de EPROM waar het monitorprogramma in staat nog de juiste inhoud heeft. De syntax van het checksum commando is :

S AAAA,BBBB

De checksum wordt bepaald voor de inhouden van adres AAAA tot en met BBBB.

Voorbeeld :

```
@ 800                           ;Wat staat er
0800 00
0801 01
0802 02
0803 03
0804 04
0805 05
0806 06

S 800,806                       ;Bepaal de checksum
0015                            ;Het resultaat
```

1.16 Commando T

Met dit commando is het mogelijk de juiste tijd te laten zien of de tijd goed te zetten. De tijd verschijnt ook op de statusregel. Typt men alleen een T dan laat de monitor de tijd zien. Het uurwerk werkt op interruptbasis. De tijd zal door het gebruik van diskroutines niet de gehele sessie gelijk blijven lopen, maar de tijd wordt meer als indicatie dan als exacte tijd gezien. Wil men de tijd wijzigen dan is de syntax:

T uu,mm

Hierin zijn de kleine letters decimale getallen, waarbij uu het uur en mm het aantal minuten is. De grenzen waartussen de in te voeren parameters moeten liggen zijn achtereenvolgens 0 en 23 voor de uren en 0 en 59 voor de minuten. Worden deze grenzen overschreden, dan volgt een foutmelding.

Voorbeeld :

```
T 21:07:34          ;Hier werd alleen een T getypt, gevolgd
                    ;door <return>

T 22,3              ;Dit werd allemaal ingetypt

T 22:03:05          ;Dit is het resultaat
```

1.17 Commando V

Met het commando V is het mogelijk twee geheugengebieden met elkaar te vergelijken. De vergelijking gaat byte voor byte. Alle adressen met hun inhouden die niet met elkaar overeenkomen worden op het beeldscherm weergegeven. De syntax is :

V AAAA,BBBB,CCCC

Het geheugengebied AAAA tot en met BBBB wordt vergeleken met CCCC en verder.

Voorbeeld :

```
M E000,E03F,800     ;Copieer een stuk geheugen

V E000,E03F,800     ;Vergelijk beide stukken

@ 807               ;Verander een paar inhouden
0807 4E E4
0808 E0 45

V E000,E03F,800     ;Vergelijk wederom
0807 E4  E007 4E    ;Dit zijn de verschillen
0808 45  E008 E0
```

1.18 Commando W

Om in een stuk geheugen naar een bepaalde bytevolgorde of naar een bepaalde string te zoeken gebruikt men het commando W. Er kunnen ook zogenaamde don't care karakters opgegeven worden. Dit kan echter alleen als men asciikarakters opgeeft en niet bij het zoeken naar bytes. De syntax is :

W AAAA,BBBB,DD

Er wordt nu in het geheugengebied van AAAA tot en met BBBB gezocht naar het byte DD. Er mogen 16 bytes opgegeven worden waarnaar gezocht moet worden. Als de sleutel gevonden is wordt het beginadres van de bytes weergegeven. Er worden evenzoveel adressen weergegeven als er sleutels gevonden worden. Als er naar een string gezocht moet worden is de syntax :

W AAAA,BBBB,'PQR

Er wordt nu in het gebied AAAA BBBB gezocht naar de string PQR. Hier mogen in totaal 16 asciikarakters staan. Op de plek van een asciikarakter mag ook het % teken staan. Dat is dan het don't care karakter. Er ontstaat echter een probleem als het % teken deel uitmaakt van de sleutel. In dat geval kan het don't care karakter gewijzigd worden. Dat gebeurd door de inhoud van variabele DNTCR te wijzigen. Deze staat default op $25 dat de hexwaarde voor het % teken is.

Voorbeeld :

```
W E000,FFFF,C953          ;Zoek naar C953
ED8B                      ;Komt tweemaal voor
F84C

@ ED8B                    ;Even kijken of het klopt
ED8B C9                   ;Klopt inderdaad
ED8C 53

@ F84C
F84C C9                   ;Klopt ook op dit adres
F84D 53

W E000,FFFF,'BOOT         ;Zoek naar de string 'BOOT
EE12                      ;Komt eenmaal voor

W E000,FFFF,'B%%T         ;Zoek naar B%%T, met don't care
EE12                      ;karakters
F1E6                      ;Komt tweemaal voor

@ EE12                    ;Even kijken of het klopt
EE12 42 'B
EE13 4F 'O                ;Klopt!!!
EE14 4F 'O
EE15 54 'T

@ F1E6
F1E6 42 'B
F1E7 3C '<                ;Klopt!!!
F1E8 51 'Q                ;De tekens < en Q waren don't care
F1E9 54 'T
```

1.19 Commando X

Als men zelf een machinetaalroutine heeft geschreven, dan is het mogelijk deze routine als commando te definieren. Men moet dan echter eerst de vector USERX naar het begin van deze machinetaalroutine laten wijzen. Er kunnen geen parameters worden meegegeven. De syntax voor deze userroutine aanroep is :

X

Na dit commando wordt direct de userroutine gestart. Als deze routine wordt afgesloten met een RTS dan komt men weer normaal in de commando mode. Als de vector USERX nog niet gedefinieerd is wijst deze naar een foutmelding.

1.20 Commando Y

Ook dit is een commando om een zelf geschreven routine aan te roepen. Men moet nu echter de vector USERY naar het begin van de routine laten wijzen. Zie verder commando X. De syntax is :

Y

1.21 Commando +

Dit commando wordt gebruikt om twee 16 bits getallen bij elkaar op te tellen. Er wordt alleen hex gerekend. De syntax is :

+ AAAA,BBBB

Hierin stellen AAAA en BBBB twee hex getallen voor.

Voorbeeld :

```
+ 24A2,4D2F                ;Tel twee hex getallen op
71D1                       ;Het resultaat

+ FFFF,2                   ;Er wordt geen rekening gehouden
0001                       ;met een overflow
```

1.22 Commando -

Dit commando wordt gebruikt om twee 16 bits getallen van elkaar af te trekken. Er wordt alleen hex gerekend. De syntax is :

- AAAA,BBBB

Hierin stellen AAAA en BBBB twee hex getallen voor.

Voorbeeld :

```
- 71D1,4D2F                ;Trek twee hex getallen van elkaar af
24A2                       ;Resultaat

- 1,2                      ;Dit gebeurt er bij een underflow
FFFF
```

1.23 Commando $

In de monitor zijn ook rekenroutines opgenomen om om te rekenen van decimaal naar hex en omgekeerd. Het commando $ wordt gebruikt om van een decimaal getal een hex getal te berekenen. Bij te grote getallen wordt er een foutmelding gegeven. De syntax is :

```
$ 12345
```

Hierin stelt 12345 een decimaal getal voor.

Voorbeeld :

```
$ 12345                    ;Bereken naar hex
3039                       ;Resultaat

$ 2023                     ;Nog een voorbeeld
07E7
```

1.24 Commando

Om van een hex getal de decimale waarde te bepalen wordt het commando # gebruikt. De syntax hiervan is:

```
# AAAA
```

Hierin is AAAA het hex getal waarvan de decimale waarde berekend dient te worden.

Voorbeeld :

```
# FFFF                     ;Bereken naar decimaal
65535                      ;Resultaat

# AE7F                     ;Nog een voorbeeld
44671
```

1.25 Commando <

Een ingetypte commandoregel kan continue herhaald worden door als laatste commando op die regel het commando < te geven. De commando's met hun parameters worden door : gescheiden, dus ook voor < komt altijd een : te staan. De syntax is :

```
<
```

Het commando moet wel als laatste commando van die commandoregel gegeven zijn, anders worden de commando's na < niet meer uitgevoerd.

Voorbeeld :

Voor een voorbeeld wordt verwezen naar het voorbeeld bij het ook geldige karakter :

1.26 Overige geldige karakters

Karakter :

Het is mogelijk meerdere commando's in een commandoregel te geven. Het karakter waarmee de commando's gescheiden worden is : De syntax wordt gegeven aan de hand van een voorbeeld.

Voorbeeld :

```
F 800,80F,41:H 800,80F:F 800,80F,42:H 800,80F    ;Vul geheugen en geef
                                                 ;hexdumpen

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA



      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0800: 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42  BBBBBBBBBBBBBBBB

                         ;Tot hier is het resultaat van de ingetypte
                         ;commandoregel
```

Het is ook mogelijk een simpele ramtest te maken als we gebruik maken van meerdere commando's op een regel. Tevens wordt als laatste commando de < gegeven om de ramtest oneindig lang door te laten lopen. De enige mogelijkheid om het programma te stoppen is een break geven met ^C.

```
F 800,8FF,AA:C 800,8FF,AA:F 800,8FF,55:C 800,8FF,55:<

Break                            ;Geef een break om te stoppen
```

Karakter ,

Dit karakter wordt gebruikt om parameters onderling te scheiden. De variabele DELIM staat default op $2C en kan eventueel gewijzigd worden als men liever aan ander karakter als scheiding ziet.

Karakter <SPATIE>

Het karakter dat gebruikt wordt om het comando van zijn parameters te scheiden is de spatie. Dit behoeft verder geen uitleg. Ook voorbeelden zijn er genoeg bij de uitleg van de commando's.

Karakter <DELETE>

De ascii waarde van DELETE is $7F. Het gevolg van het intypen van dit karakter is dat het daarvoor ingetypte karakter uit de commandoregel gewist wordt. Het geven van een voorbeeld hiervoor lijkt mij overbodig.

Karakter ^C

Dit karakter wordt gebruikt om een uitvoer naar beeldscherm af te breken. Ook het intypen van de BREAK-toets (als deze de waarde $03) afgeeft heeft hetzelfde gevolg. De monitor zal de melding 'Break' geven als op deze wijze een listing afgebroken wordt. Ook een eigengeschreven machinetaal programma kan met ^C afgebroken worden. Als echter in dat programma een commando SEt Interrupt gegeven wordt, zodat er geen interrupts meer mogelijk zijn, dan zal ^C niet meer werken. Er zijn ook nog andere situaties denkbaar waarin ^C niet zal werken, in samen werking met een zelfgeschreven programma maar die situatie zal zelden of nooit voorkomen en uitleg daarvan wordt in hoofdstuk 6 gegeven.

Karakters ^S en ^Q

Het tijdelijk stopzetten van een uitvoer naar het beeldscherm gebeurd met ^S. Het weer door laten gaan van deze listing wordt gedaan met het karakter ^Q. In plaats van ^S en ^Q die de ascii waarden $13 en $11 hebben is het ook mogelijk een listing tijdelijk stop te zetten met een toets die de ascii waarde $80 afgeeft. Op sommige toetsenborden is dit een extra functietoets of iets dergelijks. Voor de zelfbouwer moet het geen probleem zijn zoiets te implementeren. Dan geldt deze toets als een toggle. Als voor de eerste keer deze toets wordt ingedrukt zal de listing stoppen, na nog eens indrukken wordt de listing weer voortgezet.

Karakter ^Y

Met ^Y wordt de laatst ingetypte commandoregel nog eens teruggeroepen. Dit is erg handig als er eerst een stuk geheugen gemoved moet worden en daarna met dezelfde parameters een verify moet plaatsvinden.

Voorbeeld :

```
M 800,83F,840          ;Move een stuk geheugen

V 800,83F,840          ;Typ eerst een V gevolgd door ^Y
```

Karakter <RETURN>

Met dit karakter wordt een commandoregel afgesloten. Er bestaat geen andere mogelijkheid om een commandoregel af te sluiten. Verder wordt dit karakter gebruikt om uit het @ commando terug in de commando mode te komen. Ook bij het D commando, als men D AAAA intypt moet men met <RETURN> terug naar de commando mode. Wordt er een <RETURN> getypt op de eerste positie van een commandoregel, dan wordt er alleen naar de volgende regel gesprongen, zonder dat er verder iets gebeurd.

1.27 Foutmeldingen

De monitor kent drie soorten foutmeldingen, namelijk :

1 - Illegal command
2 - Illegal parameter(s)
3 - Device not ready

Als er een foutmelding 1 of 2 plaatsvindt, wordt tegelijkertijd met het circonflexe (^) teken aangegeven waar deze fout ontdekt werd. Het duidelijkst komt het een en ander tot uiting in voorbeelden. Het commando G is geen commando dus er moet een foutmelding volgen.

```
G
^
Illegal command
```

Als er na een illegaal commando een foute parameter opgegeven wordt, wordt alleen het foute commando gezien.

```
G AXBCD
^
Illegal command
```

Ook wordt er gecontrolleerd op de juistheid van de opgegeven parameters. Zo moet de eerste parameter altijd kleiner zijn dan de tweede als het om twee adressen gaat. De plaats van de foutmelding is dan tussen de twee parameters in.

```
H 200,100
    ^
Illegal parameter(s)
```

Bij het commando H verwacht de monitor twee parameters. Wordt er nog een parameter opgegeven, dan ziet de monitor de komma die tussen de twee laatste parameters staat als een illegaal karakter in een parameter.

```
H 100,200,20
        ^
Illegal parameter(s)
```

Ook als er slechts 1 parameters is opgegeven en de monitor verwacht er twee dan wordt deze foutmelding gegeven.

```
H 100
    ^
Illegal parameter(s)
```

Als er meerdere commando's op een commandoregel staan en er staat een fout in een van de commando's behalve de eerste, dan wordt de gehele commandoregel nog eens op het beeldscherm gezet met een ^ onder de plaats waar de fout ontdekt werd.

Een andere foutmelding is "device not ready". Deze melding komt gedurende enkele seconden op de statusregel te staan als geprobeerd wordt een printer op output device 2 aan te zetten als deze niet aangesloten is of niet aan staat. Ook als in- of output device 5 aan staat en er moet te lang op een handshake in het VIACOM programma gewacht worden, wordt deze foutmelding gegeven.

2. Input en output devices

Ieder computersysteem heeft een bepaalde invoer nodig. De invoer wordt door het systeem verwerkt met een bepaald programma. Het resultaat van de invoer en dit programma moet zichtbaar worden gemaakt op een randapparaat dat voor de uitvoer zorg draagt. We noemen nu alles wat voor die invoer kan zorgen input devices en alles wat voor uitvoer kan zorgen output devices. Zo is het toetsenbord een input device en het beeldscherm of printer een output device. Iedere machine kent een of meerdere I/O devices die door de software ondersteund moeten worden. MON65 houdt in principe rekening met 8 input en 8 output devices. Om bij te houden welk device nu actief is wordt voor zowel de input als de output een byte gereserveerd. In dit byte stelt ieder bit een device voor. Is een bit 1 dan is het bijbehorende device actief. Er moet altijd een (1) device actief zijn, zowel bij de input als bij de output. Met behulp van de functietoetsen worden de I/O devices geselecteerd. Als alle input en/of output devices worden uitgezet dan is dat iets wat niet mag voorkomen en dan worden automatisch de default I/O devices aangezet. De variabele waar de output device bits in staan is DEVMODOUT en de variabele waar de input device bits in staan is DEVMODINP. De default output device bits staan in de variabele OUTRET en de default input device bits staan in INPRET. De variabelen OUTRET en INPRET staan zo dat default het toetsenbord en het beeldscherm aan staan. OUTRET en INPRET zijn niet met de functietoetsen te wijzigen. Hieronder volgt een opsomming van de nummers van de I/O devices met het randapparaat dat ze aansturen.

OUTPUT DEVICE NUMMER	DEVICE OMSCHRIJVING	INPUT DEVICE NUMMER	DEVICE OMSCHRIJVING
1	beeldscherm	1	toetsenbord
2	printer centr.par.	2	niet gebruikt
3	RS232 output	3	RS232 input
4	VIACOM output	4	VIACOM input
5	memory output	5	memory input
6	vrij (DVO6VEC)	6	vrij (DVI6VEC)
7	vrij (DVO7VEC)	7	vrij (DVI7VEC)
8	vrij (DVO8VEC)	8	vrij (DVI8VEC)

Output device 1 is het beeldscherm dat meestal aan staat. In MON65 staan alle routines die nodig zijn om het beeldscherm aan te sturen. Er wordt van uit gegaan dat als hardware voor het beeldscherm een Elektuur VDU kaart gebruikt wordt met 16 MHz kristal. Er is een andere karakter generator gebruikt, dit om invers video te krijgen. Deze karakter generator wordt ook door de KIM gebruikers club geleverd.

Output device 2 wordt gebruikt om een printer aan te sturen. De printer moet een centronics parallel interface hebben. Als dit device wordt aangezet en de printer is niet aanwezig, wordt er na een aantal seconden een foutmelding gegeven. Tegelijkertijd wordt dan dit device weer uitgezet om te voorkomen dat er op een handshake wordt gewacht van een printer die niet aanwezig is.

Output device 3 is een seriele verbinding via de 6551 ACIA die op de CPU kaart aanwezig is. Deze ACIA staat default op 2400 baud. Na het selecteren van in of output device 3 worden de variabelen ACCTL en ACCMD in het control- en com- mandregister van de ACIA geladen. Door nu deze variabelen te wijzigen kan een andere baudrate geselecteerd worden. ACCTL staat default op $BA en ACCMD staat default op $05.

Output device 4 is de zogenaamde VIACOM output. MON65 is uitgerust met een speciaal communicatie programma. Deze communicatie gaat met behulp van 8 datalijnen en 2 handshakelijnen van een via. Hiermee kunnen 2 computers, die beide over VIACOM beschikken met elkaar communiceren. Als er enkele seconden geen handshake wordt ontvangen door de VIA, dan wordt automatisch dit device uitgeschakeld en het default output device ingeschakeld.

Output device 5 is het memory. MON65 ziet ook het geheugen als een output device. Zo kan alle output die normaal naar het beeldscherm zou gaan, nu ook naar het geheugen gestuurd worden. Als dit device wordt aangezet wordt er eerst een geheugenbereik gevraagd waarbinnen de informatie neergezet moet worden. Wordt dit bereik overschreden, dan wordt automatisch output device 5 uitgezet.

De output devices 6, 7 en 8 zijn vrij. Als men zelf output routines wil schrijven dan kunnen die via deze output devices aangeroepen worden. Bij ieder output device hoort een vector die naar de output routine moet wijzen. De namen van deze vectoren zijn in de tabel weer te vinden.

Input device 1 is het toetsenbord. Dit is ook het default input device. Als alle input devices worden uitgezet wordt automatisch dit input device weer aangezet om te voorkomen dat de controle over het systeem wegvalt.

Input device 2 wordt niet gebruikt. Deze staat eigenlijk naast de centronics parallel output. Om hier een centronics parallel input van te maken heeft niet zoveel zin omdat ook VIACOM input als een centronics parallel input gezien kan worden.

Input device 3 is de RS232 input. Deze input gaat via dezelfde ACIA als de RS232 output. Ook deze baudrate staat default op 2400 baud. Om de baudrate en het aantal stopbits en dergelijke te wijzigen, moeten de variabelen ACCTL en ACCMD gewijzigd worden. Na een system reset staan deze waarden weer default en moeten dan opnieuw gevuld worden.

Input device 4 is de VIACOM input. De bedoeling van VIACOM is bij output device 4 al gegeven. Het gebruik van VIACOM wordt in hoofdstuk 6 uit de doeken gedaan. Als er enkele seconden geen handshake wordt ontvangen door de VIA, dan wordt automatisch dit device uitgeschakeld en het default input device ingeschakeld.

Input device 5 is het geheugen als input device. Uit het geheugen wordt gelezen en het systeem "denkt" dat de invoer gewoon van het toetsenbord komt. Als in een computer input device 5 actief is samen met output device 4 (VIACOM) en in een andere computer output device 5 en input device 4, dan stelt dit de gebruiker in staat m.b.v. een 11 aderige kabel zeer snel files of stukken geheugen over te zenden van de ene naar de andere computer. In hoofdstuk 6 wordt hieraan meer aandacht besteed.

De input devices 6, 7 en 8 zijn net als bij de output devices vrij door de gebruiker te definieren.

3. Functietoetsen met hun functies

Het is mogelijk, zelfs als u een toetsenbord hebt zonder functietoetsen, functietoetsen te gebruiken. Het gebruik van deze toetsen is namelijk een bepaalde volgorde van gewone ascii waarden en niet het gebruik van toetsen met een vreemde code. Om bij een functietoets te komen moet eerst de ascii waarde $1E weggestuurd worden. Op de meeste toetsenborden is dat control tegelijk met een circonflexe (^). Na deze $1E wordt een gewone toets ingedrukt. Als $1E niet gegenereerd kan worden door het toetsenbord, dan moet de variabele MONESC gewijzigd worden. De default waarde is $1E. Omdat de control waarde van ^ volgens onze notatie ^^ is worden de functietoetsen in deze handleiding voorafgegaan door ^^. Het verschil tussen functietoetsen en commando's is dat commando's alleen vanuit de monitor en in de commando mode gegeven kunnen worden. Functietoetsen daarentegen kunnen ten allen tijde gebruikt en gegenereerd worden, ook vanuit DOS. Hieronder volgt een tabel met korte opsomming van de betekenis van de functietoetsen.

^^B - opstarten van de bootstrap voor DOS65
^^E - activeren van de echomode
^^I - active input devices laten zien en wijzigen
^^O - active output devices laten zien en wijzigen
^^P - user functie P, zelf te definieren
^^Q - user functie Q, zelf te definieren
^^S - aan en uitzetten van de statusregel

Let op dat de gewone toetsen van de functietoets volgorde hoofdletters zijn. Typt men kleine letters dan wordt dit niet als functietoets gezien.

3.01 Functie ^^B

Als deze functietoets wordt gebruikt, wordt de bootstrap voor DOS65 opgestart. De systeemschijf waarop het programma BOOT moet staan moet in drive 0 zitten. Kan BOOT niet gelezen worden, dan volgt de foutmelding 'Boot read error' of 'BOOT not found'.

3.02 Functie ^^E

Met deze functietoets wordt de zogenaamde echomode opgestart. Normaal gaat de input via de monitor naar de output. De monitor verwerkt de input. In de echomode echter gaat de input rechtstreeks door naar de output, zonder dat ingevoerde karakters als commando's voor de monitor worden gezien. Als de echomode wordt opgestart, dan gebeurt dit terwijl de software op een toets van het toetsenbord zit te wachten. Het gevolg is dan ook dat nadat alle I/O goed is gezet en de echomode wordt ingeschakeld er eerst een dummy toets moet worden ingedrukt. Het is afhankelijk van het actieve output device en een eventuele computer waarmee gecommuniceerd wordt wat een dummy toets is, want deze dummy toets wordt ook naar het actieve output device gestuurd. Aan de hand van onderstaand voorbeeld wordt een stuk geheugen gevuld met ascii waarden die we gewoon via het toetsenbord intypen.

Voorbeeld :

```
- Typ eerst ^^0                  ;Dit om output devices te activeren
- Typ dan een 5                  ;Nu wordt output device 5 ook aangezet
- Typ nu de toets <RETURN>       ;Om output device selectiemode te
                                 ;verlaten
                          ;Op de statusregel komt 'Write mem. from $'
                          ;te staan. De bedoeling is om een stuk geheugen
                          ;te selecteren.
- Typ 4000                       ;Dit is het startadres.
                          ;Dan wordt het eindadres gevraagd.
- Typ 400F                       ;Dit is het eindadres
                          ;Men komt automatisch in de commando mode terug
- Typ ^^E                        ;Activeer de echomode
                          ;De echomode is nu actief zoals te zien is aan
                          ;de statusregel.
- Typ ABCDEFGHIJKLMNOPQ          ;Dit is een willekeurige text.
- Typ ^C of de BREAK toets       ;Beeindig de echomode
- Typ <RETURN>                   ;Om in de commando mode te komen
```

Als we nu een hexdump geven van 4000 tot 4020 dan zien we dat de ingetypte karakters in het geheugen zijn gekomen. Het laatste karakter dat in het geheugen is gezet is de P, die op adres $400F terecht is gekomen. Als het eind van het geheugengebied niet bereikt wordt, doordat een te hoog eindadres is opgegeven, dan wordt het output device niet automatisch uitgezet. Dit moet dan weer met ^^O gebeuren nadat men weer in de commandomode is gekomen.

We kunnen nu een stuk geheugen vullen met commando's die we eigenlijk aan de monitor willen geven. Nadat dan het geheugen is gevuld, moeten we input device 5 actief zetten en input device 1 uitzetten. Als we dan de vraag 'Read mem. from $.... to $....' beantwoord hebben met een adresbereik dat overeenkomt met de plaats waar de commando's in het geheugen neer zijn gezet, dan leest de monitor de commando's uit het geheugen. Als dummy toets moet er de toets <RETURN> ingedrukt worden, dit omdat de monitor nog steeds op een toets zit te wachten.

3.03 Functie ^^I

Deze functietoets wordt gebruikt om te zien welke input devices actief zijn en om input device aan en uit te zetten. Nadat ^^I is ingetypt komt er een andere statusregel. Op deze statusregel staat dat we naar de input devices aan het kijken zijn. Er kan nu een getal ingetypt worden tussen de 1 en de 8. Op de statusregel zien we dat een eventuele 'on' achter een nummer verandert in een 'off'. Het device overeenkomend met het ingetypte nummer is dan uitgezet. Het is onzinnig alle input devices uit te zetten. De software zet dan ook het default input device weer aan. Als er twee input devices tegelijkertijd aangezet worden, dan wordt de meest linkse genomen. Dit in tegenstelling tot de output devices, hier kunnen er wel meerdere tegelijkertijd actief zijn. Input device 2 is zoals reeds eerder vermeld inactief.

Nadat men input device 4 heeft gekozen en 1 heeft uitgezet, moet de input van de viapoort komen volgens het VIACOM protocol. Is er echter niets aangesloten, dan volgt na enkele seconden de foutmelding 'Device not ready' en input device 4 wordt uitgezet, input device 1 gaat dan automatisch weer aan. Men moet er rekening mee houden dat wanneer er een ander input device wordt aangezet, er vaak nog op een karakter van het vorige input device gewacht werd, zodat er, als dit vorige input device b.v. het toetsen- bord was, nog een dummy toets moet worden ingedrukt. De inhoud van deze dummy toets wordt wel naar het actieve output device gestuurd.

De input devices 6, 7 en 8 zijn vrij door de gebruiker te definieren. De bijbehorende vectoren moeten naar de input routines wijzen, willen deze werken. Default wijzen deze vectoren naar een routine die het default input device weer aanzetten en het zojuist aangezette device weer uitzetten.

3.04 Functie ^^O

Deze functietoets wordt gebruikt om te zien welke output devices actief zijn en om output devices aan of uit te zetten. Deze functie werk analoog aan de ^^I functietoets. Het is echter wel mogelijk meerdere output devices tegelijk aan te zetten.

3.05 Functie ^^P

De functietoets ^^P is net zoals ^^Q door de gebruiker zelf te definieren. Het is de vector USERP die naar het begin van de zelfgeschreven routine moet wijzen. Deze routine moet afgesloten worden met een RTS. Wordt ^^P aangeroepen terwijl de monitor of DOS o.i.d. op een toetsingave zit te wachten, dan wordt eerst deze functietoets behandeld waarna er wordt vervolgd zonder dat de monitor of DOS er iets van merkt. De toetsingave kan dan gewoon plaatsvinden. Default wijst de vector USERP naar een dummy RTS zodat er niets wordt uitgevoerd.

3.06 Functie ^^Q

Idem als voor ^^P maar dan is de vector USERQ.

3.07 Functie ^^S

Deze functietoets wordt gebruikt om de statusregel die onderin het beeld verschijnt uit te zetten. Door nog eens deze functietoets te gebruiken wordt de statusregel weer aangezet. Als de statusregel is uitgezet is het niet mogelijk toch een 25e regel op het beeldscherm te krijgen.

4. De statusregel

Het videoram op de VDU-kaart is 2 Kbyte groot. Het is hiermee mogelijk 25 regels van elk 80 karakters op te slaan. Omdat 24 regels van 80 karakters een standaard beeldscherm formaat is werd besloten de overige ruimte te gebruiken om een statusregel in het beeld te brengen. De statusregel is optioneel. Dat houdt in dat deze aan of uit gezet kan worden. Het is niet mogelijk een ander beeldscherm formaat te kiezen, bv. 25 regels van 80 karakters en geen statusregel. De statusregel kan men uitzetten met ^^S. Normaal staat de statusregel in invers video. Dit kan gewijzigd worden, door de variabele INVST die default op $80 staat op $00 te zetten. Als deze variabele gewijzigd is vanuit de monitor dan moet even het commando P gegeven worden, anders blijven enkele karakters nog invers staan. Dat zijn de karakters die niet iedere seconde of na iedere toetsingave gewijzigd worden.

Er zijn diverse soorten statusregels. De belangrijkste is de statusregel die verschijnt bij het aanzetten of resetten van het systeem. Van links naar rechts is er op die statusregel het volgende te zien.

Time uu:mm:ss - Hier staat de tijd in uren, minuten en seconden. Met het commando T kan deze tijd goed gezet worden.

Date dd-mm-yy - De datum is zichtbaar in dag, maand en jaar. De datum kan gewijzigd worden met het commando A. Alleen om 00:00:00 uur wordt de datum door de monitor gewijzigd. Als de bijbehorende geheugen adressen van de datum door een zelfgeschreven programma gewijzigd worden dan is dat niet direct zichtbaar op het beeldscherm. Men moet dan even de vlag DATUPD op $FF zetten. Dan wordt de datum goed gezet op het moment dat de monitor naar de keyboard input routine springt.

Col: xx - Hier staat de kolom positie van de cursor. Dus de positie in horizontale richting. Deze waarde is 1 als de cursor geheel links staat. De maximale waarde is 80.

Row: yy - Dit is de positie van de cursor in verticale richting. De bovenste regel is regel 1, de onderste regel is regel 24.

Ln: z - Deze informatie is alleen relevant als men in de editor bezig is. Tijdens een edit sessie is altijd de positie van de cursor gegeven in Col en Row. Dit is de positie op het beeldscherm. Men weet dan echter niet welke regel dat in de gehele file is. Als een edit file uit 200 regels bestaat en de cursor staat op de onderste regel, dan is z 200.

Fn: filename - Ook deze informatie is alleen belangrijk in de editor. Als men een file opent om hierin te gaan editen, dan staat op deze plaats de filenaam. Deze filenaam blijft staan ook als men uit de editor gaat. Alleen als opnieuw een file wordt geopend wordt de naam gewijzigd.

Er bestaan nog een aantal statusregels. Als de functie ^^O of ^^I wordt gekozen om out- of input devices te wijzigen dan verschijnt er op de statusregel informatie over de huidige situatie van die devices. Men kan dan zien welke devices actief zijn. Bij het aanzetten van out- of input device 5, het memory, verschijnt na de <RETURN>, die ons uit het wijzigen van I/O devices haalt, een statusregel waarop gevraagd wordt het begin- en het eindadres te geven van het geheugen gebied dat gebruikt moet worden voor dit I/O device 5. Nadat de adressen zijn ingevoerd komt de normale statusregel weer terug.

Als de echomode met behulp van ^^E wordt geactiveerd verschijnt er een statusregel die ons erop attendeerd dat we in de echomode zitten en hier alleen via een BREAK uit kunnen komen. Na deze BREAK en een <RETURN> verschijnt de normale statusregel weer.

Willen we de printer op output device 2 activeren terwijl deze niet aanwezig is of niet aan staat, dan verschijnt er een foutmelding op de statusregel "Device not ready". Deze melding blijft een aantal seconden staan en de normale statusregel verschijnt weer. Dit output device wordt dan afgeschakeld. Ook als de handshake van de VIA met het programma VIACOM te lang uitblijft komt deze foutmelding en wordt ook dit I/O device afgeschakeld.

5. De jumptabel

In hoofdstuk 1 zijn de commando's van de monitor uitgelegd. Dit beslaat ongeveer de helft van de gehele monitor. De andere helft bestaat uit routines die we vanuit andere programma's aanroepen en nodig hebben voor I/O en dergelijke. De meeste routines die we nodig hebben in gebruikers programma's en DOS65 zijn bereikbaar via de jumptabel die voorin de monitor staat. Hierin vinden we een aantal I/O, breaktest en cursormode routines. Hieronder volgt een opsomming van de adressen van de jumptabel en wat de bedoeling is van de routine die er achter zit.

$E000 - Algemene output routine. Zet een karakter in de accu en roep deze routine aan. Alle actieve output devices krijgen dit karakter voor uitvoer. De registers A, X en Y blijven intact.

$E003 - Algemene input routine. Als naar deze routine wordt gesprongen keert de routine terug met in de accu de waarde van de invoer van het actieve input device. Als er meerdere input devices actief staan, wordt alleen de meest significante genomen. Het X en Y register blijven intact.

$E006 - Startadres voor de reset routine. De reset systeemvector op adres $FFFC wijst naar dit adres. Alle variabelen worden in de default stand gezet. Het uurwerk blijft onaangetast. Na afloop wordt er naar de monitor gesprongen.

$E009 - Er wordt naar een routine gesprongen die het karakter dat in de accu staat alleen op het beeldscherm staat. Dit onafhankelijk van de actieve output devices. A, X en Y blijven onaangetast.

$E00C - Deze routine keert terug met in de accu de waarde van de ingetypte toets. Dit onafhankelijk van het actieve input device. X en Y zijn onaangetast.

$E00F - Springt men naar deze routine met achter de aanroep een ascii string afgesloten door $00, dan wordt deze string naar alle actieve output devices gestuurd. Deze string moet afgesloten worden met $00. In de string mag het ascii karakter $0C niet voorkomen. Wil men toch eerst het beeldscherm schoonmaken, spring dan eerst naar de routine op adres $E04E.

$E012 - Deze routine keert terug met de ascii waarde in de accu van een eventueel ingetypte toets, dit onafhankelijk van het actieve input device. Het verschil met de routine op $E00C is dat wanneer er geen toets is ingedrukt deze routine terug komt met $00 in de accu terwijl $E00C blijft wachten tot er een toets wordt ingedrukt. X en Y blijven onaangetast.

$E015 - Deze routine voert een break uit als de breakvlag gezet was. Het uitvoeren van een break houdt in, het springen naar een routine die aangewezen wordt door de vector BRKVECTOR.

$E018 - Er wordt alleen gekeken of de breakvlag BRKFLG gezet is. Als dit het geval wordt er teruggekeerd met de carry gezet. A, X en Y blijven onaangetast.

$E01B - Roept men deze routine aan, dan gaat de cursor uit. Dat wil zeggen als de cursor aan was. Dit verdient enige uitleg. Normaal staat de cursor namelijk uit. Alleen de keyboard input routine zet de cursor aan. Nadat een toets is opgehaald, gaat de cursor uit. Schrijft men dus een routine die niet naar de keyboard input routine springt dan zal de cursor altijd uit blijven. Wil men ook bij de keyboard input routine geen cursor zien, dan moet $20 in variabele COUD worden gezet. Het wederom aanzetten gaat door de routine op $E01E aan te roepen.

$E01E - Zet de cursor weer aan. Ook te gebruiken om een knipperende cursor weer permanent aan te zetten.

$E021 - Laat de cursor snel knipperen.

$E024 - Laat de cursor traag knipperen.

$E027 - De inhoud van de accu wordt dan als byte geprint, dus als twee karakters naar de actieve output devices gestuurd. Alleen X en Y blijven onaangetast.

$E02A - Als de cursor op knipperen stond en daarna werd uitgezet door de routine $E01B aan te roepen wordt deze mode weer hersteld na aanroepen van deze routine.

$E02D - Geeft een carriage return en een linefeed op alle actieve output devices. In de accu staat na afloop $0A. X en Y blijven onaangetast.

$E030 - Print een aantal spaties. Het aantal staat in het X register. Na afloop staat er $00 in het X register en $20 in de accu. Het Y register blijft onaangetast.

$E033 - Dit is de entry die de monitor als subroutine aanroept. Men komt weer uit de monitor door het commando Q te geven. Alle registers worden dan weer teruggezet. Ook het statusregister en de stackpointer worden hersteld, evenals de breakvector.

$E036 - Dit is de indirecte jump naar de NMI vector. De systeemvector NMI op adres $FFFA wijst ook naar $E036.

$E039 - Dit is de indirecte jump naar de IRQ vector. De systeemvector IRQ op adres $FFFE wijst ook naar $E039.

$E03C - Het karakter in de accu wordt na aanroepen van deze routine naar de centronics printer gestuurd, onafhankelijk van de actieve output devices. A, X en Y blijven onaangetast.

$E03F - Deze routine probeert de centronics printer te initialiseren. Als de printer niet aanwezig is wordt "Device not ready" gedurende enkele seconden als foutmelding op de statusregel gezet.

$E042 - Dit is een entry voor de interrupt routine. Er wordt niet naar een interrupt van het toetsenbord gekeken. Deze entry wordt gebruikt door DOS65 omdat deze zelf het toetsenbord behandeld en niet bv. het uurwerk.

$E045 - Dit is een interrupt routine die alleen het uurwerk en de datum behandeld. Deze entry werd gecreeerd voor DOS65.

$E048 - Deze routine converteert een ascii waarde in de accu naar de bijbehorende hex waarde. Bv. $32 wordt $02 en $43 wordt $0C. De uitkomst staat alleen in het rechter nibble. Bij het aanroepen van deze routine moet de waarde in de accu tussen $30 en $39 liggen of tussen $41 en $46. Is dat niet het geval dan keert deze routine terug met het negative bit gezet (N=1).

$E04B - Deze routine zet een filenaam in de statusregel. De ascii string die de filenaam bevat moet worden aangewezen door een pointer. Het highbyte van deze pointer moet in het X register staan, het lowbyte in het Y register. Alle registers blijven onaangetast. (A, X, Y, processor status). Deze routine wordt aangeroepen door de editor. Het karakter waarmee de string moet worden afgesloten is een van de volgende: $00, $0D, $20. De maximale lengte van de string is 14 karakters. Als de string langer is worden de resterende karakters weggelaten.

$E04E - Maakt het beeldscherm schoon. Geeft geen formfeed op andere devices.

6. Variabelen

De bedoeling van dit hoofdstuk is om de monitor gebruiker duidelijk te maken welke variabelen waar staan en waarvoor hij deze eventueel zelf zou kunnen gebruiken. Door de hele manual heen werd er naar variabelen verwezen en werd vermeld wat er zou gebeuren als deze variabelen gewijzigd zouden worden. Hieronder volgt een opsomming van alle in de monitor gebruikte variabelen. Als eerste wordt het absolute adres genoemd van de variabele, daarna volgt de naam en de uitleg. De naam die gebruikt wordt is dezelfde als in de source listing van de monitor die ook verkrijgbaar is.

Er worden een aantal zeropage variabelen gebruikt. Dat wil zeggen tijdelijk gebruikt. De variabelen op adres $F6 t/m $FB worden door de bootstrap routine gebruikt. De variabelen PTA, PTB en TMPO worden door de monitor zelf gebruikt, maar na gebruik wordt de oorspronkelijke inhoud teruggezet. Deze adressen mogen door de gebruiker dus best ergens anders voor gebruikt worden. Dat houdt in, in gewone programma's. Het gaat fout als deze variabelen in interrupt routines gebruikt worden.

0000- PTA	Wordt gebruikt om de statusregel te copieren bij de scroll routine, zowel bij scroll up als scroll down. (2 bytes)
0002- PTB	Ook deze variabelen worden gebruikt bij de scroll routines. Er is hier voor zeropage adressen gekozen uit snelheids overwegingen. (2 bytes)
0004- TMPO	De disassembler routine gebruikt dit adres om zijn huidige disassembler adres bij te houden. (2 bytes)

Onderstaande variabelen worden door de bootstrap routine gebruikt. Afgeraden wordt deze adressen voor programma's te gebruiken. Als men alleen in de monitor blijft en niet van plan is de DOS te gebruiken kan het echter geen kwaad.

00F6- POIN	Pointer naar system blok (input buffer). (2 bytes)
00F8- SECSID	Variabele voor sector en side.
00F9- SECC	TSL plaats.
00FA- ERCNT	Aantal leesfouten.
00FB- ERC1	Density indikator.

De nu volgende variabelen zijn alleen voor de monitor. Met enkele daarvan mag naar hartelust geexperimenteerd worden. Dit wordt aangegeven door (exp) achter de uitleg.

CE00- HOLDVR	Als holdscreen actief is door ^S dan staat hier $FF. Na een ^Q staat er $00.
CE01- OFFSET	De out- en input routines, de initialisatie van output en input devices verlopen via dezelfde routines. Een offset op dit adres echter geeft aan in welk gedeelte van een bijbehorende tabel het adres moet worden gehaald voor een bepaalde routine. $00=output $08=input $10=init.output $18=init.input
CE02- CHRNR	Counter die bijhoudt hoeveel karakters er zijn ingetypt als input of output device 5 vraagt wat de begin en eind adressen zijn.

CE03- STATTOG Variabele die bijhoudt welke statusregel actief is. Het meest significante bit is aan of uit. 1=aan.

CE04- SAVSTACKP Dit is een software stackpointer voor een routine die A, X en Y register saved.

CE05- ECIRQ Adres waar de IRQ vector wordt opgeslagen tijdens de echomode.

CE07- CRPX Variabele waarin de inhoud van CURPX wordt gezet voordat er naar de routine wordt gesprongen die Col en Row print in de statusregel.

CE08- CRPY Variabele waarin de inhoud van CURPY wordt gezet voordat er naar de routine wordt gesprongen die Col en Row print in de statusregel.

CE09- CURPX De positie in de X (horizontale) richting van de cursor. De waarde van deze variabele is 1 lager dan wat Col aangeeft. Deze waarde loopt van $00 tot max. $4F

CE0A- CURPY De positie in de Y (verticale) richting van de cursor. De waarde van deze variabele is 1 lager dan wat Row aangeeft. Deze waarde loopt van $00 tot max. $17 Als er iets op de statusregel wordt geprint, dan is deze variabele $18.

CE0B- CURP Vlag die bijhoudt of er direct cursor adressing plaatsvindt. Ook is hieraan te zien of de eerste binnengekomen coordinaat binnen de grenzen viel.

CE0C- SCURPX Tijdelijk bewaar adres van CURPX.

CE0D- SCURPY Tijdelijk bewaar adres van CURPY.

CE0E- DEVCHOIC Als functie ^^0 wordt gekozen is deze variabele tijdelijk $00. Bij ^^I is deze variabele $01.

CE0F- RES Bij rekenkundige bewerkingen wordt hierin een tijdelijke uitkomst opgeslagen.

CE10- VDUS Save adres voor de variabele VDUONLY in bepaalde routines.

CE11- VDUONLY Vlag die bijhoudt waar bepaalde output naar toe moet. Zo is $00 naar alle active devices en de default waarde. $80 is alleen de centronics printer en $C0 is alleen naar het beeldscherm.

CE12- DEVMODOUT Hierin staan de active output device bits.

CE13- DEVMODINP Hierin staan de active input device bits.

CE14- SDEVMODOUT Save adres voor DEVMODOUT

CE15- SDEVMODINP Save adres voor DEVMODOUT

CE16- OUTRET Default setting voor DEVMODOUT. Als VIACOM te lang op een handshake moet wachten, als het memory output device niet meer actief is of wanneer output device 6, 7, of 8 wordt geactiveerd zonder dat deze geinitialiseerd is, dan wordt deze waarde in DEVMODOUT gezet en de initiele toestand is terug.

CE17- INPRET Default setting voor DEVMODINP. Hiervoor geldt hetzelfde verhaal als voor OUTRET maar dan met betrekking tot de input devices en DEVMODINP.

CE18- COUD Als de keyboard routine om input vraagt, dan wordt de cursor aangezet. De soort cursor wordt bepaald door een register in de CRTC. De waarde COUD wordt in de CRTC gezet. Als de cursor van mode wordt veranderd, dan wordt die nieuwe CRTC waarde ook in COUD gezet. (exp)

CE19- CLSEC Alleen tijdens de input routine die wacht op het indrukken van een toets wordt op het beeldscherm het uurwerk bijgehouden. Deze variabele krijgt de waarde van de seconden als het uurwerk "op tijd gezet" is. Er wordt daarna gekeken of CLSEC nog gelijk is aan de seconden teller. Is dit het geval dan wordt nog even gewacht voordat het uurwerk weer gelijk wordt gezet.

CE1A- BUFPOINT	Pointer die wijst naar het eerstvolgende karakter dat uit het commando inputbuffer gelezen moet worden.
CE1B- PARAL	Low byte van parameter A die bij verschillende commando's wordt ingelezen uit het commando buffer.
CE1C- PARAH	High byte van parameter A.
CE1D- PARBL	Low byte van parameter B die bij verschillende commando's wordt ingelezen uit het commando buffer.
CE1E- PARBH	High byte van parameter B.
CE1F- PARCL	Low byte van parameter C die bij verschillende commando's wordt ingelezen uit het commando buffer.
CE20- PARCH	High byte van parameter C.
CE21- @GEW	Vlag die bijhoudt of er bij het @ commando een adresinhoud gewijzigd is.
CE22- QCOMVR	Stackpointer saveadres bij aanroep van monitor als subroutine.
CE23- FUNENA	Vlag die geset wordt als ^^ ingedrukt wordt. Oftewel er wordt een functietoets verwacht.
CE24- FNCEN	Als hier een waarde ongelijk aan $00 in staat zijn de functietoetsen uitgeschakeld. In de echo-mode zijn de functietoetsen uitgeschakeld.
CE25- STMPO	Deze twee bytes worden door de disassembler gebruikt om de zeropage adressen TMPO tijdelijk in op te slaan zodat na gebruik van de disassembler zeropage niet is aangetast.
CE27- INSLEN	Hierin staat de instructielengte-1 tijdens disassembleren.
CE28- TMP4	Deze twee bytes gebruikt de disassembler om compressed ascii te converteren naar normaal formaat. De opcodes die allemaal uit 3 bytes bestaan staan als 2 bytes in het geheugen opgeslagen.
CE2A- TMP6	Tijdelijke variabele in disassembler.
CE2B- HSFLG	Vlag die voor de handshake procedure ^S en ^Q zorgt in de RS232 routines.
CE2C- DELIM	Default staat hier $2C, oftewel het scheidingsteken tussen twee parameters in de commando regel is een komma. Dit is zo gekozen omdat DOS65 ook een komma vraagt tussen de parameters. Deze waarde is bv. op $20 te zetten zodat een spatie het scheidingsteken wordt. (exp)
CE2D- VIAVRA	Samen met VIAVRB zorgen deze variabelen voor de timing bij de routines van VIACOM. Als er een tijdje geen handshake is geweest wordt er uit VIACOM gesprongen. VIAVRA houdt bij of de verstreken tijd nog in dezelfde seconde zit. VIAVRB telt het aantal verstreken seconden. Deze wordt bij opstarten van de wachtloop VIALOOP op 10 gezet, zodat er ongeveer 10 seconden gewacht wordt op een handshake.
CE2E- VIAVRB	Zie boven.
CE2F- GRFLG	Grafics vlag. Hierin staat $46 als grafics aan staat. Op het beeldscherm worden dan de grafics karakters in plaats van de normale ascii karakters geprint. Deze vlag wordt gezet door de opeenvolging van de karakters <ESCAPE> en F naar de beeldscherm output routine te sturen, dus $1B $46. De grafics mode wordt uitgezet door <ESCAPE> en G naar de beeldscherm routine te sturen, dus $1B $47. Dan worden er weer gewone ascii karakters geprint. In de variabele GRFLG staat dan $00.
CE30- INL	De input routine die parameter waarden uit het commando buffer leest gebruikt de variabelen INL en INH.
CE31- INH	Zie INL.

CE32- SVSTCK Als het hoofdprogramma wordt opgestart, dan wordt de stackpointer hierin opgeslagen. Dit is gedaan omdat sommige fouten door de commando interpreter diverse subroutines diep pas ontdekt worden. Er wordt dan een foutmelding gegeven en er kan vanuit die subroutine direkt naar een bepaald adres in de mainflow gesprongen worden die dan de stack weer goed zet.

CE33- PNTXLSAVE Save adres voor PNTXL

CE34- PNTXHSAVE Save adres voor PNTXH

CE35- PNTX Doordat de ontwerper zo weinig mogelijk gebruik wilde maken van zeropage adressen vervallen bepaalde adresserings mogelijkheden. Bv. LDA (ADR),Y. Om nu toch dat soort adresseringen te simuleren worden elders in het geheugen self modifying programma's toegepast. Op adres PNTX wordt een opcode neergezet. PNTXL en PNTXH zijn dan absolute adressen die gewijzigd kunnen worden. Op het eerste adres na PNTXH staat een RTS. De opcodes die op PNTX worden gezet hangen af van het gebruik. Zo worden PNTX en PNTY gebruikt om stukken geheugen te verplaatsen of met elkaar te vergelijken. Deze twee self modifying programma's worden door commando's van de monitor gebruikt.

CE36- PNTXL Low byte van het absolute adres van het self modifying programma PNTX.

CE37- PNTXH Low byte van PNTX.

CE39- PNTY Opcode adres voor selfmodifying programma PNTY.

CE3A- PNTYL Low byte absolute adres van PNTY.

CE3B- PNTYH High byte absolute adres PNTY.

CE3D- DMPRAM Self modifying programma dat gebruikt wordt om karakters op het beeldscherm te zetten. Dit is het opcode adres, er staat altijd STA. ($8D)

CE3E- DUMPL Low byte van VDU ram adres dat gevuld gaat worden bij het eerstvolgende karakter dat naar de beeldscherm routine gaat.

CE3F- DUMPH High byte van boven genoemde.

CE41- LDALETTER Self modifying programma dat gebruikt wordt om een hele rij van karakters naar de actieve output devices te sturen. Hier staat altijd LDA. ($AD)

CE42- LETTERADRL Low byte van het absolute adres van het karakter dat deze routine ophaalt.

CE43- LETTERADRH High byte van boven genoemde.

CE45- FRWR Self modifying programma dat gebruikt wordt om een karakter ergens in het geheugen te zetten bij output device 5, het memory als output device. Er wordt hier geen gebruik gemaakt van andere self modifying programma's als PNTX of PNTY omdat deze eventueel door een commando gebruikt worden waarvan de uitvoer in het geheugen gezet moet worden.

CE46- FRWRL Low byte van absolute adres van FRWR.

CE47- FRWRH High byte van absolute adres van FRWR.

CE49- FRRE Self modifying programma dat gebruikt wordt om een karakter ergens uit het geheugen te lezen bij input device 5, het memory als input device. Er wordt hier geen gebruik gemaakt van andere self modifying programma's als PNTX of PNTY omdat deze eventueel door een commando gebruikt worden waarvan de invoer uit het geheugen gelezen wordt.

CE4A- FRREL Low byte van absolute adres van FRRE.

CE4B- FRREH High byte van absolute adres van FRRE.

CE4D- PRBFFG Vlag die bijhoudt of het commando buffer na een foutmelding nog eens op het beeldscherm gezet moet worden. Worden er namelijk meerdere commando's op een regel gegeven, dan wordt de commando regel nog eens op het beeldscherm gezet als de fout niet tijdens het eerste commando is opgetreden.

CE4E- QSFLG ^S, ^Q en ^C enable vlaggen. Als het meest significante bit wordt geset dan kan een output naar beeldscherm niet meer tijdelijk worden stilgezet met ^S en niet meer worden gestart met ^Q. Als bit 6 is geset, dan is een ^C niet meer mogelijk. Oftewel het systeem reageert niet meer op een break. (exp)

CE4F- MONESC Hierin staat default $1E. Oftewel het karakter waarna de monitor het volgende karakter als functie toets ziet. (exp)

CE50- ACCTL Deze waarde wordt bij het aanzetten van I/O device 3 de RS 232 routines in het acia control register gezet. Default staat deze variabele op $BA wat inhoudt dat er twee stop bits en 7 data bits gebruikt worden. Tevens wordt de interne baudrate generator gebruikt die ingesteld staat op 2400 baud. Na een reset wordt de default waarde weer in ACCTL gezet. (exp)

CE51- ACCMD Bij het aanzetten van I/O device 3 wordt de inhoud van deze variabele in het command register van de acia gezet. Default staat deze waarde op $05. Dat houdt in dat er geen parity wordt verstuurd en dat er niet naar wordt gekeken, tevens wordt de echo mode gedisabled. Er komt een transmitter control interrupt en het RTS level is laag actief. Ook is de IRQ interrupt van de acia ingeschakeld. Bit 0 is ook 1 om de gehele acia te laten werken. (exp)

CE52- TIMDAT Er is in de monitor een routine die zowel de datum als de tijd kan printen. Als de tijd geprint moet worden is deze variabele gevuld met het scheidings karakter ':'. Als de datum geprint wordt is deze variabele '-'.

CE53- SCHEI Er is in de monitor een routine die decimale getallen uit de commando regel leest. Deze getallen moeten worden gescheiden door een scheidings karakter. De ascii waarde van dit karakter staat in SCHEI. Soms is deze waarde $2C (de komma), maar als de laatste parameter gelezen wordt is dit een $3A (dubbele punt) of een $0D (carriage return).

CE54- ONDERG Bij bovengenoemde routine moet tevens gekeken worden of de ingelezen waarde wel binnen de grenzen valt. Bij de datum is de ondergrens bv. 1, de minimale waarde.

CE55- BOVENG Er moet bij bovengenoemde routine ook een bovengrens opgegeven worden. Deze is 1 hoger dan de maximale waarde. Voor de datum is dat dus 32.

CE56- SEC1/20 Er loopt een interrupt via een timer in de VIA. Deze timer geeft iedere 1/20e seconde een interrupt. In deze variabele worden de 1/20ste delen van 1 seconde geteld. Deze waarde loopt tot maximaal $13. De minimale waarde is 0. Dus deze teller kan tot 20 tellen.

CE57- EINDVLAG Als de hexdump aan zijn einde is wordt deze vlag gezet.

CE58- TREMP Als het transmit register van de acia empty is wordt er een interrupt gegeven. Dan wordt er $00 in deze vlag geschreven zodat de acia output routine hieraan kan zien of het transmit register leeg is.

CE59- COUNT In dit register houdt de disassembler bij hoeveel regels er al gedisassembleerd zijn.

CE5A- ACC	Na een NMI of een breakpoint wordt hier de accu in opgeslagen.
CE5B- XREG	Na een NMI of een breakpoint wordt hier het X register in opgeslagen.
CE5C- YREG	Na een NMI of een breakpoint wordt hier het Y register in opgeslagen.
CE5D- PCL	Na een NMI of een breakpoint wordt hier het low byte van de program counter in opgeslagen.
CE5E- PCH	High byte van program counter.
CE5F- SPUSER	Na een NMI of een breakpoint wordt hier de stackpointer in opgeslagen.
CE60- PREG	Na een NMI of een breakpoint wordt hier de processor-status in opgeslagen.
CE61- TOWRL	Low byte van het eindadres van het geheugen gedeelte dat gevuld moet worden bij memory als output device.
CE62- TOWRH	High byte van boven genoemde.
CE63- TOREL	Low byte van het eindadres van het geheugen gedeelte waaruit wordt gelezen bij memory als input device.
CE64- TOREH	High byte van boven genoemde.
CE65- DECI1	Deze drie variabelen worden gebruikt om om te rekenen van hex naar decimaal. Het uiteindelijke decimale getal staat als decimale bytes in deze variabelen.
CE66- DECI2	Gedeelte van decimaal getal. (Zie boven)
CE67- DECI3	Gedeelte van decimaal getal. (Zie boven)
CE68- C02TYP	Kies het soort 65C02 tijdens het disassembleren. Deze waarde staat default op $01 zodat er volgens een ROCKWELL 65C02 gedisassembleerd wordt. Moet er echter voor een GTE/SYNERTEK gedisassembleerd worden, dan moet op dit adres $00 staan. (exp)
CE69- DISLCT	Hierin staat het default aantal regels die de disassembler disassembleerd als er slechts een parameter na het commando wordt opgegeven. Default staat deze waarde op 20. ($14) (exp)
CE6A- NMB	In het commando W (zoek) wordt deze variabele gebruikt om het aantal tekens waarnaar gezocht wordt in op te slaan.
CE6B- WHBUF	Dit is een buffer bestaande uit 16 adressen om de string waarnaar gezocht dient te worden in op te slaan.
CE7B- DATUPD	Als er een interrupt van de timer voor het uurwerk komt, worden bijbehorende registers bijgehouden. Alleen tijdens de keyboard input routine wordt de tijd in de statusregel ingevuld. De datum wordt niet telkens vanuit zijn registers gecopieerd naar de statusregel. Dat gebeurt alleen als er een dag-overgang is, dus 's nachts om 12 uur. Op dat moment wordt de vlag DATUPD op $FF gezet en de routine die de tijd op de statusregel zet weet dan dat ook de datum veranderd is.
CE7C- LNRL	Low byte van een 16 bit getal dat omgerekend wordt en als decimaal getal op de statusregel wordt gezet achter Ln: De screen editor maakt hier gebruik van om aan te geven in welke regel van de file de cursor staat.
CE7D- LNRH	High byte van boven genoemde.
CE7E- DISFLG	Vlag die geset is als de disassembler actief was. Als er namelijk een break komt tijdens een listing van de disassembler, dan moet de 16 bit variabele TMP0 weer goed gezet worden. Anders zou de monitor een zeropage adres vernielen na een break.

CE7F- BRKVS Dit is het save adres voor de break vector als de monitor als subroutine wordt aangeroepen. Na het commando Q wordt de waarde uit BRKVS weer in BRKVECTOR teruggezet.

CE81- BKPTFL Hierin staat $FF als er een breakpoint geplaatst is.

CE82- BKPTL Low byte van adres waar het breakpoint staat.

CE83- BKPTH High byte van adres waar het breakpoint staat.

CE84- BKPTVR Oorspronkelijke data van het adres waar het breakpoint op is gezet.

CE85- TABVEC Dit is een 2 byte vector die default wijst naar de foutmelding 'illegal command'. Dit is het eind van de tabel die alle geldige commando's langs gaat. Door deze vector naar een speciale routine te laten wijzen zou het aantal commando's van de monitor uitgebreid kunnen worden.

CE87- DNTCR Hierin staat het don't care karakter voor het commando W. Default staat hier $25, het '%' teken. (exp)

CE88- COMP Door de ascii waarde $19 naar de beeldscherm routine te sturen wordt het beeldscherm schoon gemaakt vanaf de cursor tot het eind van het scherm. In deze 2 byte variabele staat op dat moment het adres van het laatste karakter op het beeldscherm dat gewist moet worden.

CE8A- KEYPNT Het toetsenbord werkt onder interrupt. Dwz. dat wanneer er een toets ingedrukt wordt de waarde van deze toets wordt opgeslagen in het keyboard buffer. Bij iedere toetsingave wordt de variabele KEYPNT een opgehoogd. Als de input routine nu om een toets vraagt wordt het eerst ingetypte karakter uit dit buffer gelezen, KEYPNT wordt een verlaagd en de inhoud van het buffer schuift een plaats op.

CE8B- NOBRK Omdat het toetsenbord onder interrupt werkt kan ook tijdens het lopen van een programma gereageerd worden op bepaalde toetsen. Bv. de break toets oftewel ^C. Toch is het niet altijd wenselijk dat er een break komt. Stel de output routine is een adres aan het berekenen waarop het volgende karakter neergezet moet worden, of de break komt tijdens het printen van de tijd in de statusregel (de invers mode staat dan aan). Op dat moment mag er geen immediate break komen. Op die momenten is de NOBRK vlag geset ($FF). Als er dan toch een break wordt gegeven, wordt er niet direkt op gereageerd, maar wordt de breakvlag gezet en wordt er later op deze break gereageerd. Ook als er bij zelfgeschreven programma's geen immediate break is toegestaan moet deze vlag gezet worden. Door naderhand naar de routine op adres $E015 te springen wordt als de breakvlag BRKFLG gezet is toch nog een break uitgevoerd.

CE8C- BRKFLG Breakvlag (Zie boven).

CE8D- ESCFLG Deze vlag wordt gezet als er een <ESCAPE> naar de beeldscherm routine is gestuurd. ($1B)

CE8E- INVERS Om karakters in invers video op het beeldscherm te krijgen moet deze vlag gezet zijn. Dat gebeurd door een <ESCAPE> en i naar de beeldscherm routine te sturen. De zogenaamde escape sequence wordt dus $1B $69. Om invers weer uit te zetten wordt <ESCAPE> en n, dus $1B $6E naar de beeldscherm routine gestuurd.

CE8F- INVERSS Dit is het save adres voor INVERSS tijdens bepaalde routines.

CE90- INVST Hierin staat default $80 omdat de statusregel default op invers staat. Wil men de statusregel gewoon, dan moet hier $00 staan. Geef naderhand wel even het commando P om het beeldscherm schoon te maken. Niet alle tekst op de statusregel zal namelijk veranderen na wijzigen van deze variabele, maar na het commando P wel. (exp)

CE91- NRVRP Dit adres wordt gebruikt om de processor status in op te slaan als de monitor als subroutine wordt aangeroepen.

CE92- NVRA Zie boven, maar dan voor de accu.

CE93- NVRX Zie boven, maar dan voor het X register.

CE94- NVRY Zie boven, maar dan voor het Y register.

CE95- ECXSAV Stackpointer save adres tijdens de echomode.

CE96- ECSSP Software stackpointer save adres tijdens echo mode.

CE97- ECHVR1 Save adres voor QSFLG tijdens de echo mode.

CE98- ECHVR2 Save adres voor STATTOG tijdens de echo mode.

CE9D- FLBUF Dit is een buffer bestaande uit 14 karakters nodig voor de filenaam die op de statusregel staat achter Fn:

CEBA- USERP Deze vector (2 bytes) kan door de gebruiker zelf gedefinieerd worden om een extra functie toets te krijgen. Default staat deze vector naar een RTS te wijzen, zodat na ^^P niets gebeurd. De routine waarnaar de vector dan wijst moet afgesloten worden met een RTS. Zie ook de uitleg in hoofdstuk 3 bij de functietoets P.

CEBC- USERQ Deze vector (2 bytes) kan door de gebruiker zelf gedefinieerd worden om een extra functie toets te krijgen. Default staat deze vector naar een RTS te wijzen, zodat na ^^Q niets gebeurd. De routine waarnaar de vector dan wijst moet afgesloten worden met een RTS. Zie ook de uitleg in hoofdstuk 3 bij de functietoets Q.

CEBE- USERX Deze vector (2 bytes) kan door de gebruiker zelf gedefinieerd worden om een extra commando te krijgen. Default wijst deze vector naar de foutmelding 'illegal command'. De routine waarnaar de vector dan wijst moet afgesloten worden met een RTS.
Zie ook de uitleg in hoofdstuk 1 bij het commando X.

CEC0- USERY Deze vector (2 bytes) kan door de gebruiker zelf gedefinieerd worden om een extra commando te krijgen. Default wijst deze vector naar de foutmelding 'illegal command'. De routine waarnaar de vector dan wijst moet afgesloten worden met een RTS.
Zie ook de uitleg in hoofdstuk 1 bij het commando Y.

CEC2- DAY Variabele waarin de datum wordt bewaard. Deze waarde wordt om middernacht door de interrupt routine gewijzigd. Na een RESET blijft deze waarde ongewijzigd. (Wordt ook gezet door commando A) (exp)

CEC3- MONTH Variabele waarin de maand wordt bijgehouden. Deze waarde wordt door de interrupt routine om middernacht en alleen op de laatste dag van de maand gewijzigd. Er wordt rekening gehouden met 30 of 31 dagen en met schrikkeljaren. Na een RESET blijft deze waarde ongewijzigd. (Wordt ook gezet door commando A) (exp)

CEC4- YEAR Variabele waarin het jaartal wordt opgeslagen. Wordt alleen op middernacht op 31 december door de interrupt routine gewijzigd. Na een RESET blijft deze waarde ongewijzigd. (Wordt ook gezet door commando A) (exp)

CEC5- HOURS Hierin worden de uren opgeslagen van het uurwerk dat op interrupt basis loopt. Na een RESET blijft deze waarde ongewijzigd. (Wordt ook gezet door commando T) (exp)

CEC6- MINUTES Hierin worden de minuten opgeslagen van het uurwerk dat op interrupt basis loopt. Na een RESET blijft deze waarde ongewijzigd. (Wordt ook gezet door commando T) (exp)

CEC7- SECONDS Hierin worden de seconden opgeslagen van het uurwerk dat op interrupt basis loopt. Na een RESET blijft deze waarde ongewijzigd. (Wordt ook gezet door commando T) (exp)

CEC8- DVO4VEC Output device 4 gooit nadat het eerste karakter verzonden is zijn vector om. Dit is omdat er na het eerste karakter gewacht moet worden op een handshake voordat het volgende karakter verzonden wordt. Anders zou dit output device wachten op een handshake voordat er een karakter verzonden is.

CECA- DVI4VEC Voor de uniformiteit loopt ook het input device 4 via een vector.

CECC- INPVEC Deze vector wordt door DOS65 omgelegd om ook met de monitor input redirect te kunnen plegen. Als de DOS is opgestart lopen alle input routines via DOS.

CECE- OUTVEC Deze vector wordt door DOS65 omgelegd om ook met de monitor output redirect te kunnen plegen. Als de DOS is opgestart lopen alle output routines via DOS.

CED0- UNRINT Als er een interrupt wordt gegenereerd die door de interrupt routine niet herkend wordt, springt de interrupt routine via deze vector naar de foutmelding 'interrupt ignored'. Als er dus een eigen programma wordt geschreven waarbij door een randapparaat of een timer een interrupt gegenereerd wordt, dan kan deze vector omgelegd worden naar een eigen interrupt routine die kijkt of de interrupt van dat apparaat kwam. Is dat niet het geval, dan moet alsnog naar de foutmelding 'interrupt ignored' gesprongen worden. (exp)

CED2- FREE Deze vector is nog vrij.

CED4- DVO6VEC Output device 6 vector. Door de gebruiker zelf te definieren. Als dit output device wordt geactiveerd zonder dat de vector gedefinieerd is, dan worden de input en output devices in de default stand gezet. Zie ook hoofdstuk 2 over I/O devices. (exp)

CED6- DVI6VEC Input device 6 vector. Door de gebruiker zelf te definieren. Als dit input device wordt geactiveerd zonder dat de vector gedefinieerd is, dan worden de input en output devices in de default stand gezet. Zie ook hoofdstuk 2 over I/O devices. (exp)

CED8- DVO7VEC Output device 7 vector. Zie bij DVO6VEC. (exp)

CEDA- DVI7VEC Input device 7 vector. Zie bij DVI6VEC. (exp)

CEDC- DVO8VEC Output device 8 vector. Zie bij DVO6VEC. (exp)

CEDE- DVI8VEC Input device 8 vector. Zie bij DVI6VEC. (exp)

CEE0- ASAVESTACK Software stack waar accu inhouden worden opgeslagen. 8 bytes.
CEE8- XSAVESTACK Software stack waar Xreg inhouden worden opgeslagen. 8 bytes.
CEF0- YSAVESTACK Software stack waar Yreg inhouden worden opgeslagen. 8 bytes.

CEF8- START Pointer die het begin van het videoram op het beeldscherm bijhoudt. Het videoram loopt van $D000 tot en met $D7FF. Echter het adres links boven op het beeldscherm (positie 1,1) is alleen adres $D000 na een reset. Na een scroll up is dit al $D000+$50=$D050

CEFA- NMIVECTOR De systeemvector NMI op adres $FFFA wijst naar dit adres. De NMIVECTOR wijst weer naar de nmi routine die als het ware het commando L uitvoert op het moment van de NMI. Als een programma 'hangt' dan kan een NMI wel eens uitkomst bieden over de plaats waar het programma 'hangt'.
CEFC- BRKVECTOR Als er een break wordt gedetecteerd hetzij een immediate break hetzij een break via de BRKFLG dan wordt de routine uitgevoerd waarnaar BRKVECTOR wijst.
CEFE- IRQVECTOR De systeemvector IRQ op adres $FFFE wijst naar dit adres. Komt er dus een IRQ en is de interrupt vlag in de processor niet gezet dan wordt de interrupt routine uitgevoerd waarnaar IRQVECTOR wijst.

CF00- INPBUFFER Het commando regel input buffer is 80 karakters groot en staat op deze adressen.
CF50- KEYBUF Het keyboard input buffer is 80 karakters groot en staat op deze adressen.

CC00- SYSB De DOS65 bootstrap heeft twee buffers van 256 bytes nodig. Dit buffer wordt gebruikt als 'system block'
CD00- TSLB Het tweede buffer wordt gebruikt als 'tsl block'

De volgende adressen worden gebruikt door de PIA op de floppy disk controller kaart.

C000- PAD Data and datadirection A.
C001- PAC Command register
C002- PBD Data and datadirection B
C003- PBC Command register

De volgende adressen worden gebruikt door de floppy disk controller zelf.

C004- CMR Command register
C004- STR Status register
C005- TKR Track register
C006- SCR Sector register
C007- DTR Data register

De volgende adressen worden gebruikt door VIA 1.

C100-	VAPBD	Port B data
C101-	VAPAD	Port A data
C102-	VAPBDD	Port B data direction
C103-	VAPADD	Port A data direction
C104-	VATACL	T1, latch-low, counter low
C105-	VATACH	T1, counter high
C106-	VATALL	T1, latch low
C107-	VATALH	T1, latch high
C108-	VATBCL	T2, latch low, counter low
C109-	VATBCH	T2, counter high
C10A-	VASR	Shift register
C10B-	VAACR	Auxiliary control register
C10C-	VAPCR	Peripheral control register
C10D-	VAIFR	Interrupt flag register
C10E-	VAIER	Interrupt enable register
C10F-	VAADN	Port A data, no handshake

De volgende adressen worden gebruikt door VIA 2.

C110-	VBPBD	Port B data
C111-	VBPAD	Port A data
C112-	VBPBDD	Port B data direction
C113-	VBPADD	Port A data direction
C114-	VBTACL	T1, latch-low, counter low
C115-	VBTACH	T1, counter high
C116-	VBTALL	T1, latch low
C117-	VBTALH	T1, latch high
C118-	VBTBCL	T2, latch low, counter low
C119-	VBTBCH	T2, counter high
C11A-	VBSR	Shift register
C11B-	VBACR	Auxiliary control register
C11C-	VBPCR	Peripheral control register
C11D-	VBIFR	Interrupt flag register
C11E-	VBIER	Interrupt enable register
C11F-	VBADN	Port A data, no handshake

De volgende adressen worden gebruikt door de acia.

C130-	RECREG	Receiver register
C130-	TRAREG	Transmitter register
C131-	ACIASR	Status register
C132-	ACICMD	Command register
C133-	ACICTL	Control register

De volgende adressen worden gebruikt door de CRT controller.

C140-	CRTCAR	Address register
C141-	CRTCRF	Register file

7. Monitor routines

In dit hoofdstuk wordt nader ingegaan op bepaalde monitor routines. De werking van de routines wordt beschreven en tevens waarvoor de gebruiker deze routines kan gebruiken.

7.01 De beeldscherm output routine

Alle output gaat default naar het beeldscherm. Alleen als er output devices worden omgezet kan de output naar andere devices gaan. Het beeldscherm blijft echter het belangrijkste output device. Vandaar dat hiervoor een aantal zeer speciale mogelijkheden geschapen zijn.

- Cursor besturing.
- Invers video.
- Grafics.

Het is mogelijk de cursor via 'direct cursor addressing' te besturen. Dat wil zeggen dat de cursor door een bepaalde volgorde van karakters op iedere willekeurige plaats op het beeldscherm geplaatst kan worden. Door eerst het karakter $14 (dat is control T) naar het beeldscherm te sturen wordt de routine er op geattendeerd dat de volgende twee karakters de positie van de cursor voorstellen. Het beeldscherm bestaat uit 80 kolommen en 24 regels. (De statusregel even niet meegeteld.) De positie die opgegeven wordt moet dus vallen binnen dat veld van 80 bij 24. De waarden moeten echter hexadecimaal worden opgegeven, zodat de grenzen van het veld worden: $01...$50 en $01...$18. (In de routine zelf wordt gerekend met waarden die met $00 beginnen, dus 1 lager, maar daar merkt de gebruiker niets van.) Na de $14 moet eerst de kolom coordinaat en daarna de rij coordinaat worden opgegeven. Als nu een '*' op de 40e kolom en de 10e regel geprint moet worden, moeten achtereenvolgens de volgende karakters naar de beeldscherm output routine gestuurd worden: $14 $28 $0A $2A. Dit kan eenvoudig door gebruik te maken van de routine die een string print die afgesloten wordt met $00. Zet op de adressen $2000 en verder de volgende hex bytes:
20 0F E0 14 28 0A 2A 00 60
Als nu E 2000 wordt ingetypt komt er een '*' op positie 40,10 te staan. De cursor komt op regel 11 te staan omdat na ieder commando, dus ook het commando E een carriage return line feed geprint wordt. De eerste 3 bytes zijn om de print textstring routine aan te roepen (zie de uitleg van de jumptabel). De $60 aan het eind is om terug te komen in de commando mode. De $00 ervoor is voor het afsluiten van een textstring. Het is zo natuurlijk ook mogelijk meerdere tekens te printen. Ook meerdere cursor besturingen mogen in een string staan. Wil men iets op de statusregel printen dan moet de regel coordinaat $19 zijn. Zorg er dan voor dat dit karakter niet wordt overschreven door bv. het uurwerk. Dat kan worden voorkomen door op variabele STATTOG de waarde $FF te zetten. De statusregel wordt dan niet meer bijgehouden. Roep dan wel even een routine aan die de statusregel schoon maakt. In de monitor is dat de routine VLSTSP. Deze routine moet in de source listing opgezocht worden omdat deze niet in de jumptabel is opgenomen. De routine staat ongeveer op adres $E6D0. Men kan ook het gehele beeldscherm schoonmaken. Als een karakter op de statusregel gezet wordt met behulp van de beeldscherm output routine, zorg er dan voor dat naderhand geen carriage return line feed oid. gegeven wordt, want de scroll routine houdt geen rekening met een cursor op de 25e regel.

Er zijn ook nog andere speciale (cursor besturings) karakters of karakters die het wissen van het gehele scherm of gedeeltes hiervan tot gevolg hebben. Deze karakters met hun uitwerking staan hieronder opgesomd.

$07 - Bell. Geeft een piepje als de juiste hardware aanwezig is.
$08 - Back space. Zet de cursor een positie terug. Als de cursor op positie 1,1 stond volgt een scroll down.
$09 - Horizontal tab. De cursor komt op de eerstvolgende tabpositie te staan. Dat is om de 8 posities. Als de laatste positie op de laatste regel geweest is volgt er een scroll up.
$0A - Line feed. De cursor gaat een regel lager. Als de cursor op de laatste regel stond volgt er een scroll up.
$0B - Vertical tab. De cursor gaat een regel omhoog. Als de cursor op de bovenste regel stond volgt een scroll down.
$0C - Form feed. Het beeldscherm wordt gewist. De statusregel blijft onveranderd. De cursor gaat naar positie 1,1
$0D - Carriage return. De cursor gaat naar het begin van de regel.
$14 - Direct cursor addressing. Na dit karakter worden twee coordinaten verwacht. Indien een of beide coordinaten niet binnen de grenzen vallen, dan wordt de besturing niet uitgevoerd. Als de eerste waarde al fout was wordt toch de tweede waarde binnengelezen, maar er wordt niets mee gedaan.
$19 - Clear to end of screen. Maakt schoon vanaf de cursor tot en met het eind van het beeldscherm. De cursor blijft staan.
$1A - Clear to end of line. Maakt schoon vanaf de cursor tot en met het eind van de regel. De cursor blijft staan.
$1B - Escape. Na dit karakter kan het beeldscherm in invers of grafics gezet worden. Ook met een escape sequence wordt die mode weer hersteld.
$1C - Home. De cursor gaat naar positie 1,1. Het beeld blijft onveranderd.

Invers video is een andere mogelijkheid van het beeldscherm. Hiermee kunnen stukken tekst geaccentueerd worden. Dit is oa. gedaan met de statusregel. Om een karakter in invers op het beeldscherm te krijgen moet of het meest significante bit van dit karakter geset zijn, of het karakter moet voorafgegaan worden door een escape sequence. Deze escape sequence is <ESCAPE> en i oftewel $1B $69. De karakters die hierop volgen staan allemaal invers. Na de escape sequence <ESCAPE> en n oftewel $1B $6E worden de karakters weer normaal afgedrukt. Probeer bv. hetvolgende weer op adres $2000 in hex:
20 0F E0 0D 0A 41 42 43 1B 69 41 42 43 1B 6E 41 42 43 C1 C2 C3 00 60
Start dit weer door E 2000 in te typen.

Het is ook mogelijk een beperkt aantal grafic karakters op het beeldscherm te toveren. Door vooraf invers video in te schakelen kunnen deze karakters ook invers weergegeven worden. De escape sequence voor grafics is <ESCAPE> en F oftwel $1B $46. Om de grafics mode weer uit te zetten geldt de sequence <ESCAPE> en G oftewel $1B $47. Probeer het onderstaande programma op adres $2000:

```
                0005            .BA $2000
                0010 ;
                0015 OUT        .DE $E000
                0020 CRLF       .DE $E02D
                0025 ;
2000- 20 2D E0  0030 BEGIN      JSR CRLF
2003- 20 35 20  0035            JSR PRLOOP   ;Print normal characters
2006- 20 2D E0  0040            JSR CRLF
2009- 20 21 20  0045            JSR STGRF    ;Set grafics
200C- 20 35 20  0050            JSR PRLOOP   ;Print grafics
200F- 20 2D E0  0055            JSR CRLF
2012- 20 24 20  0060            JSR STINV    ;Set invers
2015- 20 35 20  0065            JSR PRLOOP   ;Print invers grafics
2018- 20 27 20  0070            JSR RSTGRF
201B- 20 2A 20  0075            JSR RSTINV
201E- 4C 2D E0  0080            JMP CRLF
                0085 ;
2021- A0 46     0090 STGRF      LDY #'F
2023- 2C        0095            .BY $2C      ;Skip next instruction
2024- A0 69     0100 STINV      LDY #'i
2026- 2C        0105            .BY $2C      ;Skip next instruction
2027- A0 47     0110 RSTGRF     LDY #'G
2029- 2C        0115            .BY $2C      ;Skip next instruction
202A- A0 6E     0120 RSTINV     LDY #'n
202C- A9 1B     0125            LDA #$1B
202E- 20 00 E0  0130            JSR OUT      ;Print escape
2031- 98        0135            TYA
2032- 4C 00 E0  0140            JMP OUT      ;Print character
                0145 ;
2035- A2 40     0150 PRLOOP     LDX #$40
2037- 8A        0155 BEG1       TXA
2038- 20 00 E0  0160            JSR OUT
203B- E8        0165            INX
203C- E0 80     0170            CPX #$80
203E- D0 F7     0175            BNE BEG1
2040- 60        0180            RTS
                0185 ;
                0190            .EN
```

7.02 De break

Het gebruik van BREAK levert nogal eens problemen op. Vandaar dat we hier even iets dieper op ingaan. Omdat het keyboard onder interrupt werkt kan direct op een toetsingave gereageerd worden. Stel we hebben een programma geschreven met daarin een eindeloze loop. Het programma 'hangt'. Drukken we echter op de BREAK toets (^C) dan wordt deze loop onderbroken en is men terug in de commando mode van de monitor. We testen dit met het volgende programma op adres $2000:
4C 00 20
Start het programma met E 2000. De cursor verdwijnt en komt niet terug. Het programma 'hangt'. Druk nu op ^C en we hebben weer controle over de machine.
Op bovenstaande wijze wil ^C nogal eens uitkomst bieden. Er zijn natuurlijk ook haken en ogen aan. Stel we zijn een nieuwe positie van de cursor op het beeldscherm aan het berekenen. Als halverwege de berekening een interrupt komt die het rekenproces afbreekt kan de cursor op een vreemde plaats op het beeldscherm komen. Dit is oa. het geval als de monitor de tijd bijhoudt op de statusregel. Op dat moment mag er niet direct gereageerd worden op een BREAK. We zeggen dan dat er geen immediate break mag komen. Om te voorkomen dat er een immediate break komt moet de variabele NOBRK de waarde $FF bevatten. Staat hier $00 dan zal zodra ^C ingedrukt wordt naar een break routine gesprongen worden die aangewezen wordt door de vector BRKVECTOR. Wat gebeurt er als NOBRK de waarde $FF heeft? Als er dan een break wordt gegeven met ^C dan wordt de breakvlag (BRKFLG) gezet. Hierin komt dan de waarde $FF te staan en het programma vervolgd zijn normale weg alsof er niets aan de hand is. Dan moet echter de programmeur ergens in zijn programma, op een plaats waar hem dat uitkomt, een subroutine aanroep hebben geplaatst die kijkt of de breakvlag gezet is. Op dat moment kan er op een break gereageerd worden. Dit kan gedaan worden door op die plaats de routine BRKTEST op adres $E015 aan te roepen. Deze routine kijkt of de breakvlag gezet is, zo ja dan wordt deze vlag gereset en enkele andere noodzakelijke dingen afgehandeld en wordt er naar de break routine gesprongen die aangewezen wordt door de vector BRKVECTOR. Werd de routine BRKTEST aangeroepen en was de vlag niet gezet, dan vervolg het programma zijn normale loop.
Er zijn natuurlijk situaties denkbaar waarin een immediate break geen uitkomst kan bieden. Het onderstaande programma dat slecht 1 byte langer is dan het vorige zal niet stoppen met een ^C omdat de processor interrupt vlag gezet wordt en het toetsenbord 'dood' wordt gemaakt. De enige uitkomst hier is een RESET of een NMI. In het laatste geval kan men aan de program counter zien waar het programma verlaten is. Denk erom dat de program counter altijd twee hoger staat dan het werkelijke adres. Probeer dit programma op adres $2000
78 4C 00 20
Start het met E 2000. ^C werkt nu niet meer. Na een NMI zal het systeem nog steeds 'dood' zijn omdat de interrupts nog steeds niet werken, dus een RESET moet alsnog gegeven worden.

7.03 De Viacom routines

De VIACOM I/O routines zijn ontwikkeld om twee systemen die over een dergelijk protocol beschikken snel met elkaar te kunnen laten communiceren. De communicatie gaat met 8 bits parallel met behulp van een via. Verder zijn er twee handshake lijnen nodig en een aarde zodat een 11 aderige kabel minstens nodig is om de twee systemen met elkaar te verbinden. De bedoeling van VIACOM is om programma's, stukken geheugen, enz. van het ene naar het andere systeem te sturen. De VIACOM routines zijn ondergebracht bij de I/O devices en wel nummer 4. Zet men output device 4 aan tegelijk met het beeldscherm dan zal de data die naar het beeldscherm gaat ook naar de output VIACOM routine gestuurd worden. Als er geen andere computer op de via is aangesloten dan zal het systeem ogenschijnlijk 'hangen'. Echter na ongeveer 10 seconden zal er een foutmelding komen 'Device not ready'. Hetzelfde geldt voor input device 4. Dit kan echter niet tegelijk met het keyboard werken daar de input routine alleen het meest significante input device kiest en dat is het keyboard. Het keyboard moet dus uitgeschakeld worden. Dan zet men VIACOM input aan en weer lijkt het alsof het systeem 'hangt' als er geen ander systeem aan de andere kant van de via is aangesloten. De beide systemen moeten volgens fig. 1 op elkaar worden aangesloten. Bij gebruik van de CPU kaart van Elektuur is dat de A zijde van IC 2. Als de hardware in orde is moet volgens onderstaande beschrijvingen een stuk geheugen kunnen worden overgestuurd vanuit de monitor en een programma vanuit DOS65. We beginnen met het oversturen van het stuk geheugen dat staat op adres $4000 t/m $4FFF. We gaan gebruik maken van I/O device 5 om uit het geheugen te lezen of ernaar te schrijven. Op de systemen moet hetvolgende worden gedaan:

Zender:

1 - Zet output device 1 uit en 4 aan. (^^O 1 4)
2 - Zet input device 1 uit en 5 aan. (^^I 1 5)
3 - Geef als beginadres bij input device 5 op $4000 en als eindadres $5000. Het eindadres dat de zender opgeeft is altijd 1 hoger dan het werkelijke eindadres.
4 - Zet het systeem in de echo mode. (^^E)

Ontvanger:

1 - Zet output device 1 uit en 5 aan. (^^O 1 5)
2 - Geef als beginadres bij output device 5 op $4000 en als eindadres $4FFF.
3 - Zet input device 1 uit en zet 4 aan. (^^I 1 4)
4 - Zet het systeem in de echomode. (^^E)

Zorg ervoor dat de punten 4 voor beide systemen bijna tegelijkertijd worden uitgevoerd. VIACOM blijft namelijk maximaal 10 seconden op een handshake wachten. Komt er geen handshake van de andere kant dan volgt de foutmelding 'Device not ready.'. Ook moeten de grenzen bij beide systemen correct worden ingevuld. Als dit het geval is dan zal nadat het blok is overgestuurd de cursor weer verschijnen. Op dat moment zijn de I/O devices weer gereset en moet alleen nog uit de echomode gesprongen worden. Dit gebeurt door op ^C te drukken en een <RETURN> te geven. Worden de grenzen niet goed opgegeven, dan kan een van de systemen nog op een handshake staan te wachten. Dat duurt dan 10 seconden en de I/O devices staan weer goed. Spring dan uit de echomode zoals boven beschreven. Het is altijd verstandig met het S commando een checksum in beide machines op te vragen van het stuk geheugen en deze met elkaar te vergelijken.

Het is ook mogelijk vanuit DOS65 een file van het ene naar het andere systeem te sturen met behulp van VIACOM. De procedure is dan alsvolgt:

Zender:

1 - 'Type' de file met het commando TYPE op de volgende manier:
    TYPE -N filenaam
2 - Voordat de <RETURN> achter de filenaam gegeven wordt moet eerst output device 4 aangezet worden. (^^O 4)
3 - Geef dan de <RETURN>. Soms is nog een <RETURN> nodig.

Ontvanger:

1 - 'Create' een file met het CREATE commando op de volgende manier:
    CREATE filenaam
2 - Er verschijnt een & teken als prompt. Zet nu input device 1 uit en input device 4 aan. (^^I 1 4)
3 - Soms moet er een <RETURN> gegeven worden.

Als de file is overgestuurd duurt het 10 seconden en bij de ontvanger is het contact over het toetsenbord terug. Er moet dan nog een ^D en <RETURN> gegeven worden om de CREATE correct af te sluiten. Er moet nog wel in de file geedit worden om een ^J (Linefeed) die aan het begin van de file terecht is gekomen weg te halen, evenals een $ teken aan het eind van de file. De zender heeft nog steeds contact over het toetsenbord en kan dus gelijk het output device 4 uitzetten. (^^O 4)

De werking van VIACOM is gebasseeerd op twee via's die met full auto handshake met elkaar communiseren. Fig. 2 moet hierover enige duidelijkheid verschaffen. De zender staat klaar om data te verzenden en kijkt of de CA1 lijn nul wordt, dat houdt in dat hij mag gaan zenden. Als deze lijn nul is zet de zender de data op de viapoort en maakt CA2 laag die dan de CA1 lijn in de ontvanger nul maakt. De data op de zender viapoort is nu valid. Een CA1 lijn die nul wordt zet automatisch de CA2 lijn weer hoog. Bij de ontvanger wordt CA1 nu laag die dus zijn CA2 en dus CA1 van de zender weer hoog maakt. De ontvanger kijkt ook naar zijn CA1 lijn en zodra deze laag wordt leest de ontvanger de data viapoort. Zodra de data bij de ontvanger gelezen wordt maakt deze zijn CA2 lijn laag wat tot gevolg heeft dat de CA1 lijn bij de zender ook laag wordt. Als de ontvanger nog niet klaar was met de vorige data te verwerken dan zal zijn CA1 lijn al een tijdje laag zijn voordat de nieuwe data gelezen kon worden. Als de CA1 bij de zender weer laag wordt mag de zender weer iets op de viapoort zetten. Als de zender nog bezig is met data ophalen dan is de CA1 lijn bij de zender al laag voordat deze kon zenden. Zodra de zender dus klaar is wordt direct de data verzonden.

Het bovenstaande werkt prima als alles een keer op gang is. De routine is heel simpel: Kijk naar CA1, is deze laag schrijf dan naar de viapoort of lees ervan. Is CA1 hoog blijf dan wachten, maximaal 10 seconden. Het grote probleem doet zich echter voor als de zender het eerste karakter moet versturen. Op dat moment heeft de ontvanger namelijk nog niets ontvangen en is CA1 bij de zender nog hoog. Het gevolg is dat het eerste karakter nooit verstuurd zal worden. De oplossing is echter vrij simpel. De output routine verloopt via een vector. Als het output device geinitialiseerd wordt, wordt de vector zo gezet dat het eerste karakter dat verstuurd moet worden direkt, dus zonder dat er naar CA1 gekeken wordt, op de uitgangspoort van de via wordt gezet. Als dit karakter verstuurd is wordt de vector zo gezet dat voordat de volgende karakters verstuurd worden eerst naar CA1 wordt gekeken.

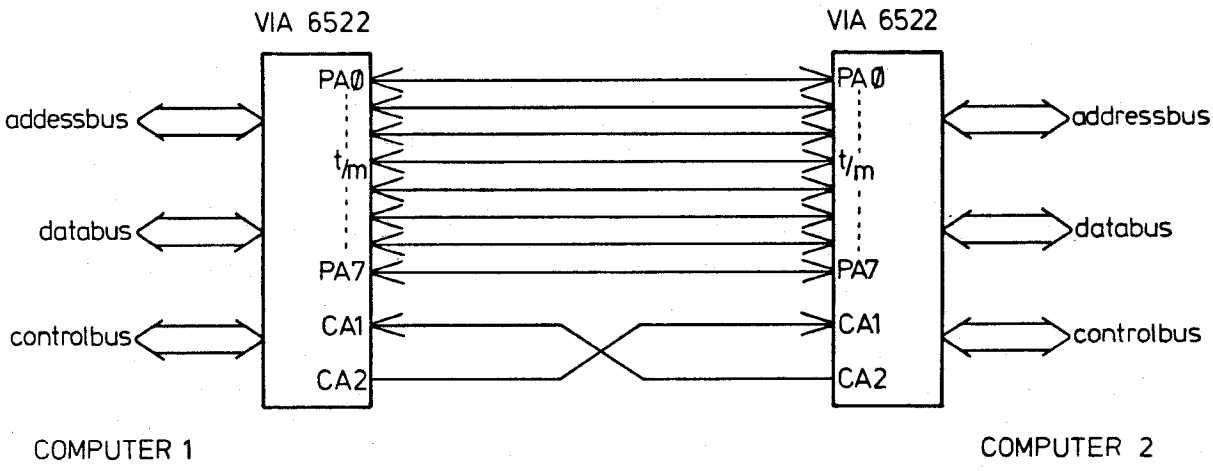
VIA 6522
VIA 6522
addessbus
databus
controlbus
PAØ
t/m
PA7
CA1
CA2
PAØ
t/m
PA7
CA1
CA2
addressbus
databus
controlbus
COMPUTER 1
COMPUTER 2

FIG. 1

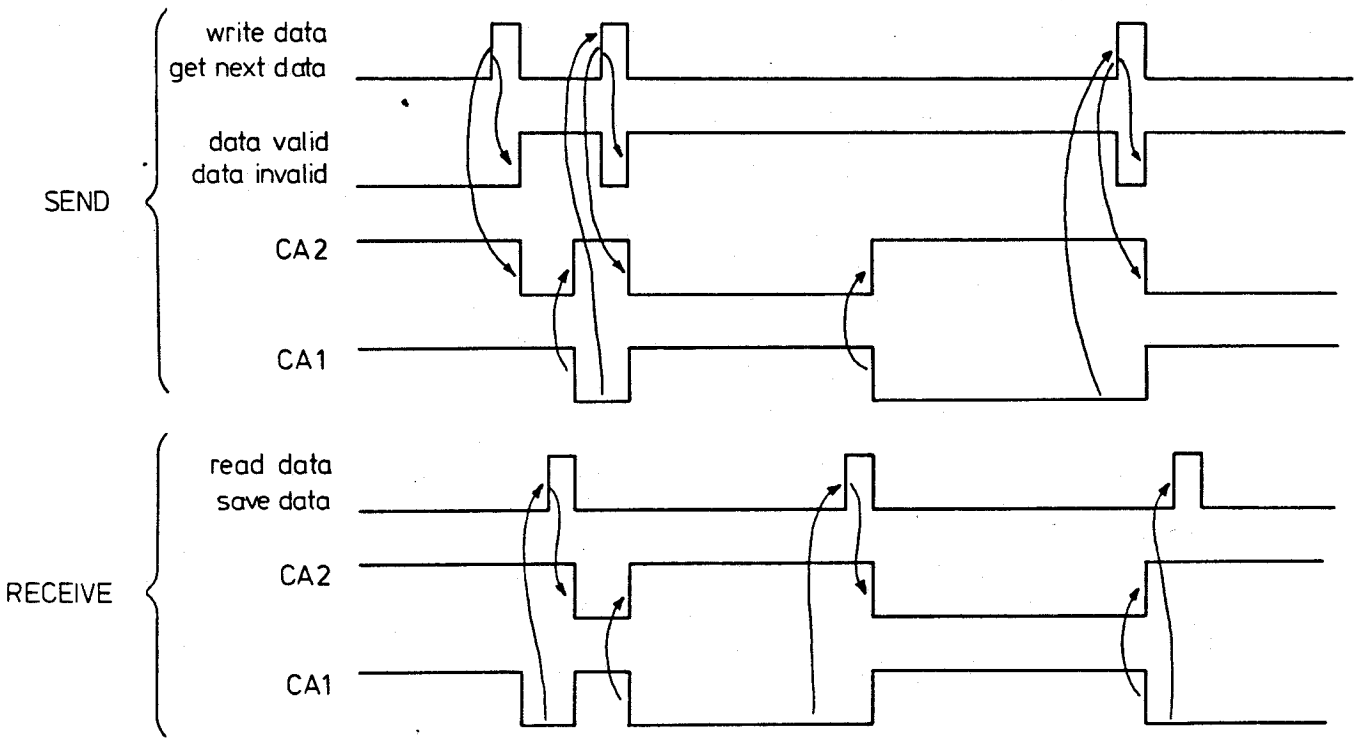
SEND
write data
get next data
data valid
data invalid
CA2
CA1
RECEIVE
read data
save data
CA2
CA1

FIG. 2

7.04 De interrupt vectoren

Als er een IRQ interrupt komt doordat de IRQ-lijn laag wordt, veroorzaakt door het aflopen van een timer, het ontvangen van een handshake puls oid., dan wordt indien de interrupt vlag in de processor niet is geset het lopende programma afgebroken en wordt er vervolgd met het programma dat aangewezen wordt door de IRQ vector. Deze vector heet IRQVECTOR en wijst naar de normale IRQ routine. Deze routine kijkt vanwaar de interrupt kwam en zal dan het bijbehorende programma afwerken. Echter soms wordt de lijn laag door een fout van de programmeur. Hij activeerd bv. een timer die na een bepaalde tijd een interrupt geeft en het interrupt programma kan niet vinden waar deze interrupt vandaan kwam. In dat geval wordt aan het eind van de interrupt routine via de vector UNRINT gesprongen naar de foutmelding 'Interrupt ignored.'. Moet er naar een eigengeschreven interrupt routine gesprongen worden dan kan deze vector omgelegd worden en hierheen laten wijzen. In deze routine moet dan getest worden of de interrupt inderdaad voor deze routine bedoeld is, zo ja dan kan deze naar behoren behandeld worden. Zo nee dan moet alsnog naar de foutmelding gesprongen worden. Soms moet men echter ook IRQVECTOR omleggen. In dat geval wordt eerst de eigen interrupt routine doorlopen en daarna pas de interrupt routine van de monitor en de DOS. Het is niet op de voorhand te zeggen welke vector het best kan worden omgelegd. Dit moet worden bekeken in samenhang met de interrupts die de DOS moet afhandelen.

Na een 'Interrupt ignored' wordt het lopende programma normaal vervolgd.

7.05 RS232 Interface

Er is zowel een in- als output RS232 routine in de monitor opgenomen. Bij de input routine moet input device 3 aan- en 1 uitgezet worden. Bij het ontvangen van een receiver register full interrupt wordt het ontvangen karakter in het keyboard buffer gezet. De RS232 ontvang routine leest deze karakters uit het keyboard buffer. Het gevolg hiervan is dat er tijdens het ontvangen via RS232 ook de karakters worden ontvangen die via het toetsenbord worden gegeven. Hiermee moet even rekening worden gehouden. Toch blijkt dit wel een handige bijkomstigheid te zijn. Als namelijk input device 1 uit- en 3 is aangezet, kan men niet meer over functietoetsen beschikken en is het niet meer mogelijk de devices om te wisselen. Toch moet men de controle over het keyboard helemaal terug hebben en moet input device 3 weer uitgezet kunnen worden. Doe dit dan met behulp van de monitor. Zet op de variabele DEVMODINP de waarde $80 en de controle is terug. Tevens is dan 3 uitgezet. De RS232 input routine werkt zonder handshake procedure.

De RS232 output routine kan bv. voor het aansturen van een seriele printer gebruikt worden. Vanuit de DOS werkt deze printer dan niet met het PRINT commando daar dat commando de centronics uitgang neemt. De RS232 uitgang werkt wel met een handshake procedure. Als namelijk karakters worden verstuurd dan kan deze uitvoer geblokkeerd worden door een ^S naar de RS232 ingang te sturen. Na een ^Q gaat de uitvoer verder. Deze software handshake wordt door veel printers toegepast. Ook diverse modem verbindingen werken met dit protocol. Het uitzetten van dit device levert geen problemen op daar men normale controle heeft over het toetsenbord.