MONITOR HANDLEIDING MON65, monitor versie 2.01 1986

Inhoudsopgave Blz.

MONITOR
MAP: 9000-9EC0
9F50-9F55.
9000

EPROMMER ZIT
VAN 3000-4FFF.

EDITOR:
MAP: 1000-2FFF.
0400-04FF.
0600-06FF.
0580-05FF.
0700-07FF.
1000

AS:
MAP: 0A00-3AC3
0A00

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

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

Inleiding

MON65 is een utility die vanuit de DOS65 commandmode kan worden aangeroepen. Dit in tegenstelling tot MON65 van versie 1.01. In versie 1.01 is de monitor verwerkt in de eprom waarin ook de IO routines in zijn ondergebracht. Aangezien de Octopus comptabiliteit een eprom van slechts 4 kbyte toestaat is besloten de monitor uit deze eprom te halen en onder te brengen op de systeemschijf. Bijna alle mogelijkheden zijn hetzelfde gebleven. Een tweetal commando's zijn verdwenen omdat deze ook al in de DOS zijn ondergebracht (A en T). Het IO gedeelte is wel in de eprom gebleven en hiervoor is een aparte manual geschreven. Dat houdt in dat de beschrijvingen van de IO devices, de functietoetsen, de statusregel, de jumptabel, de variabelen en enkele IO routines in die aparte manual worden behandeld.

In deze handleiding zijn voorbeelden gegeven. Deze voorbeelden zijn als illustratie bedoeld. Als u dezelfde commando's geeft is het mogelijk dat er andere waarden als resultaat gegeven worden.

1. Installatie

Door MONITOR of de afkorting MON in te typen vanuit de DOS commandmode wordt de monitor van schijf gelezen en gestart. Default wordt er een prompt gegeven die bestaat uit de string 'MON> '. Wil men deze prompt wijzigen dan moet met het @ commando de string op adres $9F50 gewijzigd worden. Er mogen maximaal 16 karakters voor genomen worden. Na het laatste karakter moet een $00 staan om het eind van de prompt aan te geven. Wil men geen prompt dan wordt op adres $9F50 al $00 gezet.

De monitor kan disassembleren voor twee verschillende 65C02 types. Default staat de variabele op adres $9F2E op $01. Dat wil zeggen voor de Rockwell 65C02. Wil men echter disassembleren voor een Synertek/GTE dan moet op adres $9F2E met het commando @ een $00 gezet worden.

Het aantal regels dat de disassembler op het beeldscherm zet als na aanroep van het commando D slechts 1 parameter wordt opgegeven is default 20. Dit is te wijzigen door het gewenste aantal in hex op adres $9F2F te zetten met het commando @.

Wil men een andere delimiter dan de komma, het karakter dat twee parameters van elkaar scheidt, dan moet de asciiwaarde van dat karakter op adres $9F10 gezet worden. Dat kan ook met het commando @.

In het commando W is het mogelijk om bij het zoeken naar een string een don't care karakter in te geven. Dit karakter is standaard het % teken. Moet hiervoor een ander karakter komen dan moet de variabele op adres $9F46 de gewenste inhoud krijgen.

De boven genoemde zaken zijn naar eigen keuze in te stellen. Wil men echter ook bij een volgende aanroep van MON deze instelling weer zo hebben, dan is het verstandig de monitor opnieuw op de systeemschijf te zetten. De procedure is alsvolgt :

- Verlaat de monitor met het commando Q.
- Save de monitor op de systeemschijf :

SAVE S:MONITOR 9000,9FFF <return>

Wordt nu de monitor opnieuw aangeroepen dan heeft men de nieuwe instelling. Eventueel kan de tweede versie een andere naam hebben. Vervang dan MONITOR door een andere naam. Vergeet niet deze file weer als commando te zetten met 'SETMODE -C S:filename'

2. Commando's

Vanuit de DOS commandmode wordt ingetypt MONITOR (of MON), de monitor wordt van schijf geladen en meldt zich met :

MON65 2.XX
HaViSoft

Hier is XX een getal tussen de 00 en 99, dat 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 zowel hoofd als kleine letters bij zowel de commando's als de adressen voor parameters als bij een string voor bv. het commando W.

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 als prompt de string 'MON> '. Dit is een string die 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. De prompt is instelbaar (zie hoofdstuk 1).

2.01 De 21 geldige commando's zijn :

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

NMI - Na het indrukken van de NMI toets volgt er een interrupt. Op dat moment worden de inhouden van Accu Xreg en Yreg evenals de processorstatus en de programcounter op het beeldscherm gezet. MON65 komt terug in de command mode. Als een programma dat vanuit MON65 wordt aangeroepen vastloopt, dan kan men zeer handig deze toets gebruiken. Vanuit DOS65 werkt dat niet. Op dat moment wijst de NMI vector namelijk naar een dummy RTI en er gebeurt niets.

2.02 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. Ook is het mogelijk ascii karakters in te voeren. Deze worden echter alleen geaccepteerd na een \ karakter. Typt men hierna een A in dan wordt $41 op die geheugen locatie gezet. 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 :

```
@ 2000<return>                  ;Geheugenadres wordt geopend
2000 02 41<linefeed>            ;Inhouden overschrijven
2001 FF 42<return>              ;Afsluiten met return


@ 2000<return>                  ;Opnieuw openen
2000 41 'A<linefeed>            ;Inhoud is veranderd
2001 42 'B<vert. tab>           ;Na ' teken de ascii waarde
2000 41<return>
```

2.03 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 software interrupt uit I065, naar een software interrupt afhandelings routine gesprongen. 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. Denk erom de breakpoints alleen te zetten op adressen waar opcodes staan en niet op de plaatsen van de operanden.
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 2010                          ;Op $2010 wordt een breakpoint gezet

B S                             ;Laat het breakpointadres zien
Breakpoint: $2010 02            ;Adres met oorspronkelijke inhoud

B R                             ;Haal het breakpoint weg

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

2.04 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 $D000 t/m $DFFF. Er wordt dan ingetypt :

```
C D000,DFFF,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 :

```
@ 2000                          ;Open een adreslocatie
2000 40 41                      ;Zet er $41 in t/m $2006
2001 42 41
2002 43 41
2003 FF 42                      ;Zet hier $42 in
2004 00 41
2005 FF 41
2006 00 41

C 2000,2006,41                  ;Check het gebied op $41
2003 42                         ;Hier staat geen $41

C 2000,2006,'A                  ;Check het gebied op A
2003 42                         ;Dit is geen A

@ 2003                          ;Open het foute adres
2003 42 41                      ;Herstel in $41

C 2000,2006,41                  ;Nu is het hele gebied goed
```

2.05 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. Zie hoofdstuk 1 voor het wijzigen van deze intelling. 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 op adres $9F2F (zie hoofdstuk 1). Default is $14.

Voorbeeld :

```
D F000                                  ;Geef het commando
F000  4C EF F6          JMP  $F6EF
F003  4C AC F6          JMP  $F6AC
F006  4C 08 F7          JMP  $F708
F009  4C BA F6          JMP  $F6BA
F00C  4C B7 F0          JMP  $F0B7
F00F  4C AE F7          JMP  $F7AE
F012  4C 82 FF          JMP  $FF82
F015  4C A0 F3          JMP  $F3A0
F018  4C DB F3          JMP  $F3DB
F01B  4C 18 F4          JMP  $F418
F01E  4C 60 F4          JMP  $F460
F021  4C 90 F4          JMP  $F490
F024  4C A1 F4          JMP  $F4A1
F027  4C BB F4          JMP  $F4BB
F02A  4C 71 F9          JMP  $F971
F02D  4C 69 F9          JMP  $F969
F030  6C C2 CE          JMP  ($CEC2)
F033  6C C6 CE          JMP  ($CEC6)
F036  4C 39 F0          JMP  $F039
F039  D8                CLD             ;Tot hier gedisassembleerd
F03A  78                SEI             ;Na een spatie
F03B  A2 0F             LDX  #$0F       ;Na een spatie
```

```
F03D  8A              TXA           ;Na een willekeurige toets
F03E  49 0F           EOR  #$0F
F040  9D F0 FF        STA  $FFF0,X
F043  CA              DEX
F044  10 F7           BPL  $F03D
F046  A9 00           LDA  #$00
F048  A2 61           LDX  #$61
F04A  9D FF CD        STA  $CDFF,X
F04D  CA              DEX
F04E  D0 FA           BNE  $F04A
F050  CA              DEX
F051  9A              TXS
F052  A0 00           LDY  #$00
F054  B9 5D F1        LDA  $F15D,Y
F057  F0 0A           BEQ  $F063
F059  AA              TAX
F05A  B9 A8 F1        LDA  $F1A8,Y
F05D  9D 00 CE        STA  $CE00,X
F060  C8              INY
F061  D0 F1           BNE  $F054    ;Na een return terug naar
                                    ;command mode
```

2.08 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. Om de registers accu, xreg, yreg en de processor status te kunnen vullen voordat het programma gestart wordt, moeten de adressen $9F24 ev. gevuld worden.

Accu - $9F24
X reg - $9F25
Y reg - $9F26
Pr. Stat - $9F29

2.09 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
```

2.10 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.

2.11 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!!

2.12 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
```

2.13 Commando P

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

P

2.14 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!!! De syntax is:

Q

2.15 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 io-programma 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
```

2.17 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 F000,F03F,800                ;Copieer een stuk geheugen

V F000,F03F,800                ;Vergelijk beide stukken

@ 807                          ;Verander een paar inhouden
0807 4F F4
0808 F0 45

V F000,F03F,800                ;Vergelijk wederom
0807 F4  F007 4F               ;Dit zijn de verschillen
0808 45  F008 F0
```

2.18 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. Zie hiervoor hoofdstuk 1.

Voorbeeld :

```
W F000,FFFF,C953                ;Zoek naar C953
FD8B                            ;Komt tweemaal voor
F84C

@ FD8B                          ;Even kijken of het klopt
FD8B C9                         ;Klopt inderdaad
FD8C 53

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

W F000,FFFF,'BOOT               ;Zoek naar de string 'BOOT
FE12                            ;Komt eenmaal voor

W F000,FFFF,'B%%T               ;Zoek naar B%%T, met don't care
FE12                            ;karakters
F1E6                            ;Komt tweemaal voor

@ FE12                          ;Even kijken of het klopt
FE12 42 'B
FE13 4F 'O                      ;Klopt!!!
FE14 4F 'O
FE15 54 'T

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

2.19 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. De vector USERX staat op adres $9F69 voor het low byte en op $9F6A voor het high byte.

2.20 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. De vector USERY staat op adres $9F6B voor het low byte en op $9F6C voor het high byte. Zie verder commando X. De syntax is:

```
Y
```

2.21 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
```

2.22 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
```

2.23 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
```

2.24 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
```

2.25 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 :

3. 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. Eventueel kan deze delimite gewijzigd worden. Zie hiervoor hoofdstuk 1.

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.

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.

4. Foutmeldingen

De monitor kent twee soorten foutmeldingen, namelijk :

1 - Illegal command
2 - Illegal parameter(s)

Als er een foutmelding 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.