

Vice Monitorbefehle

Hexadezimal
Binär
Dezimal
Oktal

\$A5 oder A5
%10100101
+165
&0020

Ohne Angabe des Zahlenformates werden alle Zahlen immer als Hexadezimal interpretiert.

~

~ 10

Ausgabe einer Zahl in Dezimal-, Hexadezimal-, Oktal- & Binärformat. Das Zahlenformat kann mit dem oben genannten Vorzeichen angegeben werden.

x / exit

Den Monitor verlassen.

cartfreeze

Startet nach verlassen des Monitors das Freezer-Menü eines Erweiterungsmoduls.

cpu

cpu 6502

Setzt die zu emulierende CPU. Die möglichen Prozessoren hängen vom genutzten Emulator ab und können meist nicht geändert werden.
6502 = Prozessor

exp / export

Zeigt eine Liste aller angeschlossenen Geräte am Expansionsport an.

? / help

help device / ? device

Listet alle Befehle auf. Wird ein Befehl zusätzlich angegeben, wird eine Erklärung für diesen angezeigt.

keybuf

keybuf poke53280,0\x0d

(funktioniert bis Version 2.3 und ab Version 2.4.18 R29640)

Übergibt den angegebenen String dem Tastaturpuffer. Es ist auch möglich Hexadezimal-Code für Steuerzeichen beginnend mit **\x** (Backslash x) einzugeben.

poke53280,0 = Tastatureingabe

\x0d = Hexadezimal-Code (hier = Return – weitere Kombinationen unter: <http://www.c64-wiki.de/index.php/Steuerzeichen>)

p / print

p ff

Zeigt eine beliebige Zahl in Dezimal an.

p %01101001

Umwandlung Binär → Dezimal

p &0200

Umwandlung Oktal → Dezimal

d / disass

d 1000 1010

Zeigt die Assemblerbefehle des angegebenen Speicherbereiches an. Bei Angabe der Startadresse wird die Standardanzahl an Instruktionen (40 Stück = \$28) disassembliert. Ohne Angabe einer Adresse wird das Disassemblieren an der Cursoradresse fort geführt.

1000 = Startadresse
1010 = Endadresse (optional)

```
.C:1000      .start:
.C:1000  A9 00      LDA #$00
.C:1002  EA              NOP
.C:1003      .play:
.C:1003  A0 02      LDY #$02
.C:1005  F0 FC      BEQ .play
.C:1007  20 00 10    JSR .start
```

Beim Disassemblieren werden ebenfalls definierte Labels angezeigt.

bank

bank ram

Zeigt den aktiven Bereich im Monitor durch ein Sternchen an.

bank ram Schaltet den kompletten Ram ein (z.B. um Bereiche unter dem ROM zu speichern).
bank cpu Normalzustand
bank cart Blendet Modulbereiche ein (ab \$8000) ein, wenn Modul installiert
Weiterhin möglich: rom io

c / compare

c 1000 2000 3000

Vergleicht einen Adressbereich und zeigt alle Adresse deren Werte sich unterscheiden.

1000 = Startadresse
2000 = Endadresse
3000 = Anfangsadresse Zielbereich

dev / device


dev c:

Wichtig: Doppelpunkt am Ende

Es wird das Gerät (C64 oder Floppylaufwerke) eingestellt, auf dem sich die Monitorbefehle beziehen.
(dev c: = C64 / dev 8: dev 9: dev 10: dev 11: = entsprechende Diskettenlaufwerke)

Das zur Zeit aktive Gerät erkennt man am Anfang der Adresszeile durch den Buchstaben bzw. Zahl nach dem Punkt.

```
.C:1008  A9 01      LDA #$01
.C:100a  8D 86 02    STA $0286
```



f / fill

f 1000 2000 ff

Füllt einen Speicherbereich mit einem bestimmten Wert.

1000 = Startadresse
2000 = Endadresse
ff = Füllwert

f 1000 2000 20 e0 30 f0

Es können auch mehrere Werte getrennt durch Leerzeichen angegeben werden. (Füllt den angegebenen Bereich mit den Werten 20 10 e0 3f).

```
>C:1000  20 e0 30 f0  20 e0 30 f0  20 e0 30 f0
>C:100c  20 e0 30 f0  20 e0 30 f0  20 e0 30 f0
>C:1018  20 e0 30 f0  20 e0 30 f0  20 e0 30 f0
>C:1024  20 e0 30 f0  20 e0 30 f0  20 e0 30 f0
```

h / hunt

h 1000 2000 f0

Sucht nach einem Wert im angegebenen Speicherbereich. Alle Adressen die mit dem Wert übereinstimmen werden angezeigt.

1000 = Startadresse
2000 = Endadresse
f0 = Wert

h 1000 2000 +250 %11110000 a0 %0020

h 1000 2000 01 02 03

Es können auch mehrere Werte getrennt durch Leerzeichen angegeben werden. Dann werden die Speicherzellen angegeben, welche exakt mit den Werten und der Reihenfolge identisch sind.

i

i a000 a1000

Zeigt den angegebenen Speicherbereich im PETSCII Format an.

Startadresse und / oder Endadresse sind optional Ohne Angabe einer Adresse / Adressbereiches wird der aktuelle Bereich angezeigt (4 x 40 Zeichen)

ii

ii a000 a100

Zeigt den angegebenen Speicherbereich als Bildschirmtext an.

Startadresse und / oder Endadresse sind optional Ohne Angabe einer Adresse / Adressbereiches wird der aktuelle Bereich angezeigt (4 x 40 Zeichen)

m / mem

m B 1000 2000

Zeigt den angegebene Speicherbereich im vorgegeben Format.

B = Formattyp
1000 = Startadresse
2000 = Endadresse

Der Formattyp muss immer groß angegeben werden (B = Binär / D = Dezimal / H = Hexadezimal). Ohne Angabe des Formattyps wird der Standardtyp angenommen (Hexadezimal).

mc / memchar

mc d800 d900

Gibt den angegebenen Speicherbereich als Zeichensatzdarstellung aus. Ist nur die Startadresse angegeben, wird 1 Zeichen ausgegeben. Ohne Angabe einer Adresse wird die aktuelle Adresse ausgegeben und automatisch erhöht („Enter“ gibt das nächste Zeichen aus).

d800 = Startadresse
d900 = Endadresse

ms / memsprite

ms d800 d900

Verhält sich wie „mc“. Jedoch werden die Daten als Sprites angezeigt.

d800 = Startadresse
d900 = Endadresse

t / move

t 1000 2000 3000

Kopiert den angegebenen Adressbereich in den Zielbereich. Beide Bereiche dürfen sich überlappen.

1000 = Startadresse
2000 = Endadresse
3000 = Anfangsadresse Zielbereich

sc / screen

Zeigt den Bildschirminhalt im Monitor an.

sfx / sidefx**sfx on**

Steuerung der Lese-Seiteneffekte bei Speicherstellen die den Lese-Seiteneffekt besitzen (z.B. CIA-Interrupt Register). Ohne Argument wird der aktuelle Status angezeigt.

on = Effekt ein

off = Effekt aus

toggle = Effekt umschalten (von ein → aus / von aus → ein)

>**> 1000 20**

Speichert die angegebenen Werte in die angegebene Speicherstelle.

1000 = Adressen

20 = Wert

> 1000 25 36 98 +100 %00110011

Es können auch mehrere Werte getrennt durch Leerzeichen angegeben werden.

bt / backtrace

Zeigt die Adressen der JSR-Aufrufe (gespeichert in Stackpuffer). Der letzte Aufruf wird zuerst gezeigt. Der Stackoffset+1 wird in Klammern angegeben – diese ist aber nur eine Schätzung.

```
(C: $e5cf) bt
(2) e112
(4) a562
(6) a483
(8) a677
(10) e39a
(C: $e5cf)
```

dump**dump „test“**

Erstellt ein Snapshot und speichert diesen in das aktuelle Arbeitsverzeichnis. Es werden keine ROM-Abbilder mit gespeichert. Dieser Snapshot entspricht einem Snapshot, welcher über die grafische Oberfläche erstellt wurde (Menü → Snapshot). Der Dateiname muss in Anführungszeichen angegeben werden.

„test“ = Dateiname

undump**undump „test“**

Lädt die angegebenen Snapshot-Datei aus vom aktuellen Arbeitsverzeichnis. Der Dateiname muss in Anführungszeichen angegeben werden.

test“ = Dateiname

g / goto**g 1000**

Setzt die aktuelle Adresse auf den angegebenen Wert und führt die Befehle von dort weiter aus.

n / next

Führt die nächste Instruktion aus. Nach Ausführung wird die aktuelle Adresse, der Befehl und alle Register angezeigt.

io

io d000

Zeigt alle I/O Register an. (CIA 1+2 / SID / VIC). Bei Angabe einer Adresse werden zusätzliche Information angezeigt.
D000 = Adresse für zusätzliche Informationen (DC00 = CIA1 / DD00 = CIA2 / D000 = VIC)

reset

reset 1

Führt einen Reset aus (SYS64738). Ohne Angabe einer Zahl wird eine Softreset durch geführt.
1 = Reset-Art / Gerät (0 = Softreset / 1 = Hardreset / 8-11 Laufwerksreset)

ret / return

Führt das Programm weiter aus und springt nach dem nächsten RTS oder RTI Befehl zurück in den Monitor.

stopwatch

Zeigt den Prozessor Zyklusähler an.
stopwatch reset setzt den Zähler auf 0

r / registers

Zeigt die Werte der einzelnen Register einschließlich Flags und Stack Pointer an.

```
(C:$e5d4) r
  ADDR A  X  Y  SP 00 01 NV-BDIZC LIN  CYC  STOPWATCH
.;e5d4 00 00 e0 21 2f 37 00100010 000 002    8255522
(C:$e5d4) █
```

A = Akku / Y = Y-Register / X = X-Register / SP = Stack Pointer /
00 / 01 Datenrichtungsregister CIA / NV-BDIZC = Flags
(LIN = Rasterzeile / CYC = Taktzyklus)

r A=e0

rA=e0, X=e1, Y=C1, SP=e0

Es ist auch möglich einzelne Register zu ändern (nur A / X / Y / SP). Mehrere Register können getrennt durch Komma geändert werden.

z / step

z 10

Einzelschrittausführung – ein Befehl wird ausgegeben und zurück in den Monitor gesprungen. Es wird der letzte Befehl / Akku / X- & Y-Register / Stackzähler und Flags angegeben. Wird ein Wert (siehe Befehl rad) angegeben, werden so viele Befehle abgearbeitet.

```
(C:$e5d4) z 05
Stepping through the next 5 instruction(s).
.C:e5cd  A5 C6      LDA $C6      - A:00 X:00 Y:E0 SP:21 ..-...Z.    8255554
(C:$e5cd)
```

rad / radix

rad H

Setzt das Zahlensystem für Eingaben bzw. zeigt das derzeit eingestellte Zahlensystem an. Vorzeichen für die Eingabe des Zahlenformates können weiterhin genutzt werden (\$ / % / + / &).
H = Abkürzung für Zahlensystem (H = Hexadezimal / D = Dezimal / O = Oktal / B = Binär)

cd

cd Test Ordner

Wechselt das (Arbeits)Verzeichnis. Ebenfalls erlaubt sind 2 Punkte (..) und Backslash (\) zum Verzeichniswechsel.

ls / dir	Zeigt den Inhalt des aktuellen Pfades am PC. Zusätzlich wird der Pfad des aktuellen Arbeitsverzeichnisses angegeben.
pwd	Zeigt den aktuellen Pfad des Arbeitsverzeichnisses an.
attach	<p style="text-align: center;">attach „Test Name.d64“ 8</p> <p>Die angegebene Datei (Disketten-Image) wird in das angegebene Laufwerk eingelegt. Laufwerksnummer 32 steht für Erweiterungsmodule. „Test Name.d64“ = Dateiname in Anführungszeichen 8: = Laufwerksnummer (32 entspricht Modul)</p>
detach	<p style="text-align: center;">detach 8</p> <p>Disketten-Image wird aus dem angegebenen Laufwerk entfernt. Laufwerksnummer 32 steht für Erweiterungsmodule.</p>
br / block_read	<p style="text-align: center;">br 10 05 c000</p> <p>Lädt die angegebene Spur und Sektor. Bei fehlender Angabe des Speicherbereiches wird der Block direkt auf dem Bildschirm im aktuellen Zahlenformat / Anzeigemodus ausgegeben (siehe rad). Es wird immer auf Laufwerk 8 zugegriffen. 10 = Spur 05 = Sektor c000 = Speicheradresse (optional)</p>
bw / block_write	<p style="text-align: center;">bw 10 05 c000</p> <p>Lädt eine Datei in den Speicher. Bei der Angabe des Laufwerks 0 wird die Datei direkt aus dem aktivem PC-Verzeichnis geladen. Die Adresse muss zwingend angegeben werden. 10 = Spur 05 = Sektor c000 = Speicheradresse</p>
bl / bload	<p style="text-align: center;">bl „test“ 8 c000</p> <p>Lädt eine Datei in den Speicher. Bei der Angabe des Laufwerks 0 wird die Datei direkt aus dem aktivem PC-Verzeichnis geladen. Die Ladeadresse muss zwingend angegeben werden. „test“ = Dateiname in Anführungszeichen 8 = Laufwerk c000 = Ladeadresse</p>
bs / bsave	<p style="text-align: center;">bs „test“ 8 c000 c100</p> <p>Speichert einen Bereich auf Diskette. Die Zwei-Byte Ladeadresse wird nicht mit abgespeichert (im Gegensatz zu save). Bei der Angabe des Laufwerks 0 wird die Datei direkt in das aktive PC-Verzeichnis gespeichert. test“ = Dateiname in Anführungszeichen 8 = Laufwerk c000 = Startadresse c100 = Endadresse</p>

I / load

I „test“ 8 c000

Lädt eine Datei von der Diskette. Sollte als Laufwerk 0 angegeben werden, wird die Datei direkt aus dem aktiven Verzeichnis des PCs geladen. (Zeiger für Basic werden gesetzt.)

„test“ = Dateiname in Anführungszeichen

8 = Laufwerk

c000 = Ladeadresse (optional)

Bei nicht angegebener Ladeadresse wird die Datei entsprechend der angegebenen Ladeadresse in den Speicher geladen (die ersten beiden Bytes des Programms auf der Diskette).

list

list 9

Diskettenverzeichnis wird angezeigt. Ohne Angabe eines Laufwerkes wird immer das Inhaltsverzeichnis von Laufwerk 8 verwendet.

9 = Laufwerksnummer (optional)

Bei Angabe der Laufwerksnummer wird das Inhaltsverzeichnis des entsprechenden Laufwerks angezeigt.

s / save

s „test“ 8 c000 c100

Speichert einen Bereich auf Diskette. Die Zwei-Byte Ladeadresse wird ebenfalls mit hinterlegt. Bei der Angabe des Laufwerks 0 wird die Datei direkt in das aktive Verzeichnis gespeichert.

„test“ = Dateiname in Anführungszeichen

8 = Laufwerk

c000 = Ladeadresse

c100 = Endadresse

@

@ N:test,01

Befehl an das Diskettenlaufwerk schicken über Kanal 15.

N:test,01 = Laufwerksbefehl Der Befehl wird nur angenommen, wenn dieser Großgeschrieben wird. (N = New für Diskette formatieren in diesem Beispiel). Gültige Befehle sind N / S / R / C / V / I / D. Quelle: <http://www.c64-wiki.de/index.php/Floppy-Befehle>

al / add_label

al 8: 1000 .test

Erstellt ein Labelname für die angegebene Speicheradresse. Der Labelname muss mit einem Punkt beginnen, damit der Monitors als Label erkennt.

8: = Nummer des Gerätes (optional)

1000 = Speicheradresse (Vorzeichen für Zahlenformate sind weiterhin nutzbar.)

.test = Labelname

Wenn ein Label mit gleichen Namen einer anderen Speicheradresse zu gewiesen wird. Übernimmt der Monitor die neue Adresse für das Label.

Einer Speicheradresse können auch mehrere Labels zu gewiesen werden. Der Monitor weißt jedoch darauf hin.

```
(C:$e5d4) al 1000 .test
(C:$e5d4) al 1200 .test
Changing address of label .test from $1000 to $1200
(C:$e5d4) shl
$1200 .test
```

```
(C:$e5d4) al 1200 .doppeltes_Label
Warning: label(s) for address $1200 already exist.
(C:$e5d4) shl
$1200 .doppeltes_Label
$1200 .test
```


dl / delete_label

dl 8: .test

Löscht das angegebene Label.

.test = Labelname

8: = Laufwerksnummer (optional)

Wird keine Laufwerksnummer angegeben, bezieht sich der Befehl auf den Computer.

shl / show_labels

shl 8:

Zeigt alle Labels des aktuellen Gerätes an.

8: = Laufwerksnummer (optional)

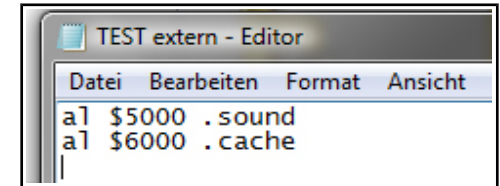
Wird keine Laufwerksnummer angegeben, bezieht sich der Befehl auf den Computer.

ll / load_labels

ll 1000 2000 „testlabels“

Lädt eine Datei mit definiert Labels und dem angegeben Dateinamen.

Eine Datei mit Labeldaten kann mittels eines Editors (z.B. Windows Notepad) auch selbst erstellt werden. In dieser Datei müssen die einzelnen Label mittels des add_label Befehls aufgeführt sein.



sl / save_labels

sl „testlables“

Speichert alle Labels in eine Datei. Diese kann mit einem Texteditor nachträglich editiert werden.

bk / break

bk load c000 if 8:A == 0

Setzt einen Breakpoint. Wird keine Adresse angegeben, werden alle aktiven Breakpunkte angezeigt. Bei Angabe einer Adresse wird der Breakpoint auf diese Adresse gesetzt (es wird bei Erreichen der Adresse und Operator in den Monitor gesprungen).

load = Parameter (optional)

Art bei welcher Operation zum Monitor gesprungen werden soll. Ohne diese Angabe wird der Breakpoint als exec gesetzt.
load = bei Load-Befehl / store = bei Store-Befehl / exec = bei ausführen der Speicheradresse

c000

Adresse des Break-Befehls

if c:A == 0 = Bedingung (optional)

Bedingung für Breakpoint (siehe hierfür den Befehl condition)

```
(C:$1005) break store c000
WATCH: 5 C:$c000 (Stop on store)
(C:$1005) x
#5 (Stop on store c000) 160 016
.C:1002 8D 00 C0 STA $C000 - A:00 X:00 Y:00 SP:f6 ..-B..Z. 195941104
.C:1005 8D 20 D0 STA $D020 - A:00 X:00 Y:00 SP:f6 ..-B..Z. 195941104
(C:$1005)
```

en / enable

enable 1

Aktiviert den angegebenen Kontrollpunkt (Breakpoints / Watchpoints / Tracepoints / Untilpoints). Ohne Angabe werden alle Kontrollpunkte aktiviert.

1 = Nummer des Breakpointes

dis / disable

dis 1

Deaktiviert den angegebenen Kontrollpunkt (Breakpoints / Watchpoints / Tracepoints / Untillpoints). Ohne Angabe werden alle Kontrollpunkte deaktiviert.

1 = Nummer des Breakpointes

command

command 1 „m 2000 2010“

Wenn der angegebene Kontrollpunkt erreicht wird springt Vice in den Monitor zurück und der angegebene Befehl wird ausgeführt. Der Befehl "x" wird nicht unterstützt.

1 = Kontrollpunktnummer

„m 2000 2010“ = Monitorbefehl (in Anführungszeichen)

del / delete

del 1

Löscht den angegebenen Breakpunkt. Wird keine Nummer angegeben werden alle gelöscht.

1 = Breakpoint Nummer

Ohne Angabe der Nummer werden alle Breakpoints gelöscht.

cond / condition

cond 1 if X == Y

Bei jeder Überprüfung des Breakpunktes (Kontrollpunkt) wird die Bedingung verglichen. Wenn das Ergebnis „wahr“ ergibt, wird der Breakpoint aktiviert, ansonsten ignoriert. Bei Nutzung von Registern wird der Zeitpunkt der Überprüfung zu Grunde gelegt.

1 = Nummer des Breakpoints

if X == Y = Bedingung

Als Bedingung können Register benutzer und gegen andere Register oder Konstanten verglichen werden. Ebenfalls sind Geräteregister von Laufwerken erlaubt. Konstanten: == != < > <= >=

(8:X == X= X-Register des Laufwerkes 8)

Geräteadressen: c: / 8: / 9: / 10: / 11:

tr / trace

tr load c000 c00a if c: A == Y

Der trace-Befehl verhält sich sehr ähnlich wie der break-Befehl. Wird keine Adresse angegeben, werden alle aktiven Tracepunkte angezeigt. Bei Angabe einer Adresse wird der Tracepoint auf diese Adresse gesetzt. Jedoch wird das Programm nicht unterbrochen. Es werden nach Rücksprung in den Monitor alle Register angezeigt.

load = Parameter (optional)

Art bei welcher Operation zum Monitor gesprungen werden soll. Ohne diese Angabe wird der Tracepoint als exec gesetzt.

c000

load = bei Load-Befehl / store = bei Store-Befehl / exec = bei ausführen der Speicheradresse

if c:A == 0 = Bedingung (optional)

Adresse des Break-Befehls

Bedingung für Breakpoint (siehe hierfür den Befehl condition)

```
(C:$1005) tr store c000
TRACE: 6 C:$c000 (Trace store)
(C:$1005) x
#6 (Trace store c000) 261 002
.C:1002 8D 00 C0 STA $C000 - A:00 X:00 Y:00 SP:f6 ..-B..Z. 199701749
(C:$e5d4)
```

w / watch

w load c000 if 8:A == 0

Der watch-Befehl verhält sich sehr ähnlich wie der break-Befehl. Wird keine Adresse angegeben, werden alle aktiven Watchpunkte angezeigt. Bei Angabe einer Adresse wird der Watchpoint auf diese Adresse gesetzt (es wird bei Erreichen der Adresse und Operator in den Monitor gesprungen). Nach Rücksprung in den Monitor werden alle Register angezeigt.

load = Parameter (optional)

Art bei welcher Operation zum Monitor gesprungen werden soll. Ohne diese Angabe wird der Tracepoint als exec gesetzt.

load = bei Load-Befehl / store = bei Store-Befehl / exec = bei ausführen der Speicheradresse

c000

Adresse des Break-Befehls

if c:A == 0 = Bedingung (optional)

Bedingung für Breakpoint (siehe hierfür den Befehl condition)

```
(C:$1012) w store c000
WATCH: 4 C:$c000 (Stop on store)
(C:$1012) x
#4 (Stop on store c000) 058 020
.C:1002 8D 00 C0 STA $C000 - A:00 X:00 Y:00 SP:f6 ..-B..Z. 190883090
.C:1005 8D 20 D0 STA $D020 - A:00 X:00 Y:00 SP:f6 ..-B..Z. 190883090
```

ignore

ignore 1 5

Der angegebene Kontrollpunkt (Breakpoint / Watchpoint / Tracepoint) wird für die angegebene Anzahl von Aktivierungen (Durchläufen) ignoriert. Ohne Angabe des Zählers wird 1 als Standard gesetzt.

1 = Nummer des Breakpoints

5 = Anzahl der Durchläufe

un / until

un 1000

Setzt einen temporären Breakpoint. Wird die angegebene Adresse erreicht, springt der Emulator in den Monitor, zeigt Adresse und Register an. Der Breakpoint wird danach gelöscht. Ohne Adressangabe werden alle Breakpoints / Watchpoints / Tracepoints / Untilpoints angezeigt.

```
(C:$100d) until 100d
UNTIL: 9 C:$100d (Stop on exec)
#9 (Stop on exec 100d) 025 018
.C:100d 8D 86 02 STA $0286 - A:01 X:00 Y:00 SP:f6 ..-B.... 216944865
(C:$100d)
```

stop

Stoppt die Aufnahme der Eingabebefehle bzw. kennzeichnet das in einer aufgenommenen Datei.

pb / playback

pb „test“

Monitor Kommandos aus der angegebenen Datei werden gelesen und ausgeführt. Dieses Kommando stoppt, wenn das Ende der Datei erreicht wurde, oder das Kommando "STOP" gelesen wurde.

„test“ = Dateiname (in Anführungszeichen)

rec / record

rec „test“

Mit diesem Kommando werden alle Eingaben in eine Datei gespeichert bis das Kommando "STOP" erfolgt. Die Datei wird in das aktuelle Arbeitsverzeichnis gespeichert.

„test“ = Dateiname (in Anführungszeichen)

Eingabe

```
(C:$1011) rec "test befehle"  
(C:$1011) a 1000  
.1000 lda #0  
.1002 sta c000  
.1005 sta d021  
.1008 sta d020  
.100b lda #1  
.100d sta 0286  
.1010 rts  
.1011  
(C:$1011) stop  
Closed file test befehle.  
(C:$1011)
```

Dateiname

```
test befehle  
1 a 1000  
2 lda #0  
3 sta c000  
4 sta d021  
5 sta d020  
6 lda #1  
7 sta 0286  
8 rts  
9  
10 stop  
11
```

chis / cpuhistory

chis 0a

Nur möglich wenn Vice mit `--enable-memmap` kompiliert wurde.

Zeigt die zu Letzt ausgeführten Befehle. Wenn ein Zähler angegeben wird, wird die nur die angegebene Anzahl an Befehlen angezeigt.

0A = Anzahl der angezeigten Befehle (optional)

Jeder Speicherstelle des emulierten Computers wird ein Byte in der memmap (vom Emulator / Monitor verwaltet) zu geordnet. Dadurch ist eine bildliche Darstellung der Speicherzugriffe auf IO-Bereich, ROM und RAM möglich.

mmsh / memmapshow

mmsh 6 0000 0100

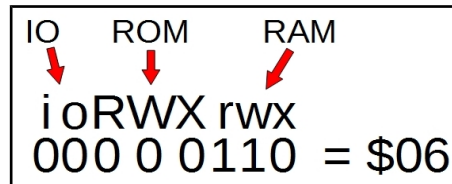
Nur möglich wenn Vice mit `--enable-memmap` kompiliert wurde.

Zeigt die Adressen an, welche sich seit dem Start bzw. dem Löschen der Memmap für den angegebenen Bereich geändert haben.

0000 = Startadresse

0100 = Endadresse

6 = Maskenwert (in Hexadezimal)



Der Beispielbefehl zeigt alle Speicheradresse im Bereich von \$0000 bis \$0100 an, bei dem ein Lese- & Speicherzugriff im angegebenen Bereich statt fand.

Der Maskenwert ist eine Hexadezimale Zahl, die aus den Bits der vorgegebenen Maske „ioRWXrwx“ erstellt wird.

io = IO-Bereich; RWX = ROM; rwx = RAM

R / r = Lesezugriff; W / w = Schreibzugriff; X / x = ausgeführt

```
(C:$e5d4) mmsh a 0000 0010  
addr: IO ROM RAM  
0000: -- -- rw-  
0001: -- -- rw-  
0002: -- -- rw-  
0003: -- -- rw-  
0004: -- -- rw-  
0005: -- -- rw-  
0006: -- -- rw-  
0007: -- -- rw-  
0008: -- -- rw-
```

mmsave / memmapsave

mmsave „test“ 0

Nur möglich wenn Vice mit `–enable-memmap` kompiliert wurde.

Speichert die geänderten Adressen als Bilddatei in das Arbeitsverzeichnis. Der Dateiname muss in Anführungszeichen angegeben werden.

„test“ = Dateiname

0 = Bildformat

0 = BMP, 1 = PCX, 2 = PNG, 3 = GIF, 4 = IFF

mmzap / memmapzap

Löscht die Memmap.

Nur möglich wenn Vice mit `–enable-memmap` kompiliert wurde.

