A N H A N G

BEFEHLS - ABKÜRZUNGEN

Allgemeines Prinzip: wenn ein Schlüsselwort nicht ausgeschrieben wird, muß der zuletzt getippte Buchstabe mit <Shift> eingegeben werden. Zur Verdeutlichung ist die Schreibweise des Text-Modus gewählt. Der Typ des Befehls ist in der Tabelle wie folgt gekennzeichnet:

A = Anweisung, K = Kommando, O = Operator
NF = numerische Funktion, SF = String-Funktion
NR = reservierte numerische Variable, SR = reserv. String-Variable

Schlüsselwort	Abkürzung	Typ	Schlüsselwort	Abkürzung	Typ
abs	aB	NF	fn	-	NF
and	aN	O	for	fO	A
asc	aS	NF	fre	fR	NF
atn	aT	NF	get	gE	A
auto	aU	K	getkey	getkE	A
backup	bA	K	get#	gE#	A
box	bO	A	gosub	goS	A
char	chA	A	goto	gO	A
chr$	cH	SF	graphic	gR	A
circle	cI	A	gshape	gS	A
close	clO	A	header	heA	K
clr	cL	A	hex$	hE	SF
cmd	cM	A	if	-	A
collect	colL	K	input	-	A
color	coL	A	input#	iN	A
cont	cO	K	instr	inS	NF
copy	coP	K	int	-	NF
cos	-	NF	joy	jO	NF
data	dA	A	key	kE	K
dec	-	NF	left$	leF	SF
def	dE	A	len	-	NF
delete	deL	K	let	lE	A
dim	dI	A	list	lI	K
directory	diR	K	load	lO	K
dload	dL	K	locate	loC	A
do	-	A	log	-	NF
draw	dR	A	loop	loO	A
ds	-	NR	mid$	mI	SF
ds$	-	SR	monitor	mO	A
dsave	dS	K	new	-	K
el	-	NR	next	nE	A
end	eN	A	not	nO	O
er	-	NR	on	-	A
err$	eR	SF	open	oP	A
exp	eX	NF	or	-	O

Schlüsselwort	Abkürzung	Typ	Schlüsselwort	Abkürzung	Typ
paint	pA	A	save	sA	K
peek	pE	NF	scale	scA	A
poke	pO	A	scnclr	sC	A
pos	-	NF	scratch	scR	K
print	?	A	sgn	sG	NF
print#	pR	A	sin	sI	NF
printusing	?usI	A	sound	sO	A
pudef	pU	A	spc(	sP	nach PRINT
rclr	rC	NF	sqr	sQ	NF
rdot	rD	NF	sshape	sS	A
read	rE	A	st	-	NR
rem	-	A	stop	sT	A
rename	reN	K	str$	stR	SF
renumber	renU	K	sys	sY	A
restore	reS	A	tab(	tA	nach PRINT
resume	resU	A	tan	-	NF
return	reT	A	ti	-	NR
rgr	rG	NF	ti$	-	SR
right$	rI	SF	trap	tR	A
rlum	rL	NF	troff	troF	A
rnd	rN	NF	tron	trO	A
run	rU	K	until	uN	A
			usr	uS	NF
			val	vA	NF
			verify	vE	K
			vol	vO	A
			wait	wA	A
			while	wH	A

****
* FEHLERMELDUNGEN *
****

NR.	FEHLERMELDUNG	BESCHREIBUNG
1	TOO MANY FILES	Es dürfen höchstens 10 Dateien gleichzeitig geöffnet werden.
2	FILE OPEN	Es wurde versucht, eine bereits geöffnete Datei zu öffnen.
3	FILE NOT OPEN	Die in einer I/O-Anweisung angegebene Datei muß vorher geöffnet worden sein.
4	FILE NOT FOUND	Diskette: die gesuchte Datei existiert nicht, Kassette: Bandende-Markierung (EOT) gelesen.
5	DEVICE NOT PRESENT	Das Peripheriegerät ist nicht ansprechbar.
6	NOT INPUT FILE	Es wurde mit GET# oder INPUT# versucht, aus einer zum Schreiben geöffneten Datei zu lesen.
7	NOT OUTPUT FILE	Es wurde mit PRINT# versucht, in eine zum Lesen geöffnete Datei zu schreiben.
8	MISSING FILE NAME	Ein OPEN-, LOAD- oder SAVE-Befehl braucht immer einen Dateinamen.
9	ILLEGAL DEVICE NUMBER	Es wurde versucht, ein Gerät anzusprechen, das nicht paßt (z.B. SAVE auf Bildschirm).
10	NEXT WITHOUT FOR	Entweder sind Schleifen nicht korrekt verschachtelt oder nach NEXT steht ein nicht passender Variablenname.
11	SYNTAX	Ein BASIC-Befehl ist nicht korrekt geschrieben. Es kann sich um falsch geschriebene Schlüsselwörter, fehlende Klammern oder überzählige Zeichen handeln.
12	RETURN WITHOUT GOSUB	Eine RETURN-Anweisung wurde erreicht, obwohl vorher kein GOSUB-Befehl gegeben wurde.
13	OUT OF DATA	Eine READ-Anweisung wurde erreicht, ohne daß noch Daten in DATA-Zeilen vorhanden sind, die nicht bereits abgearbeitet wurden.
14	ILLEGAL QUANTITY	Eine Funktion wurde mit einem außerhalb des erlaubten Zahlenbereichs liegenden Argument aufgerufen.

NR.	FEHLERMELDUNG	BESCHREIBUNG
15	OVERFLOW	Das Ergebnis einer Berechnung ist betragsmäßig größer als die höchste zulässige Zahl (1.701411833E+38).
16	OUT OF MEMORY	Entweder ist kein Speicherplatz für Programme und Variablen vorhanden, oder es sind zu viele FN-, DO-, FOR- oder GOSUB-Anweisungen aktiv.
17	UNDEF'D STATEMENT	Eine angesprochene Zeilennummer existiert nicht im Programm.
18	BAD SUBSCRIPT	Ein Feldindex außerhalb des in der DIM-Anweisung festgelegten Bereichs wurde angesprochen.
19	REDIM'D ARRAY	Ein schon eingerichtetes Feld kann nicht ein zweitesmal DIMensioniert werden. Wenn ein Feld mit einem Index zwischen 0 und 10 angesprochen wird, bevor es DIMensioniert wurde, wird es automatisch mit 10 DIMensioniert.
20	DIVISION BY ZERO	Es darf nicht durch Null geteilt werden.
21	ILLEGAL DIRECT	INPUT- oder GET-Anweisungen dürfen nur im Programm-Modus verwendet werden.
22	TYPE MISMATCH	Anstelle einer Zeichenkette darf kein numerischer Ausdruck verwendet werden und umgekehrt.
23	STRING TOO LONG	Eine Zeichenkette darf nur bis zu 255 Zeichen enthalten. Programmzeilen und INPUT-Antworten dürfen nur 80 Zeichen enthalten.
24	FILE DATA	Entspricht TYPE MISMATCH bei Eingabe von Kassette oder Diskette.

****
* DISK - FEHLERMELDUNGEN *
****

Diese Fehlermeldungen werden durch die reservierten Variablen DS (Fehler-Nr.) und DS$ (String aus: Fehler-Nr., Fehlermeldungstext, Spur, Sektor [, Drive-Nr.]) ausgegeben.

Fehlermeldungen mit Nummern unter 20 brauchen nicht berücksichtigt zu werden, mit Ausnahme von 01, die in der Spur-Nr. innerhalb DS$ angibt, wieviele Files durch einen SCRATCH-Befehl gelöscht wurden.

Das DOS ist das Disk Operating System (Disketten-Betriebssystem), das im Laufwerk über einen eigenen RAM-Speicher verfügt.

NR.	FEHLERMELDUNG	BESCHREIBUNG
20	READ ERROR	Blockheader nicht gefunden. Der Disk-Controller ist nicht in der Lage, den Header des benötigten Datenblocks zu finden. Wird hervorgerufen durch eine unzulässige Sektornummer oder durch Zerstörung des Headers.
21	READ ERROR	Kein Synchronisierungs-Zeichen. Der Disk-Controller ist nicht in der Lage, ein Synchronzeichen auf der verlangten Spur zu finden. Wird verursacht durch Dejustage des Schreib-/Lesekopfs, keine oder unformatierte Diskette im Laufwerk oder falsch eingelegte Diskette. Kann auch Hardware-Fehler anzeigen.
22	READ ERROR	Datenblock nicht vorhanden. Der Disk-Controller sollte einen Datenblock lesen oder prüfen, der nicht korrekt geschrieben wurde. Diese Fehlermeldung tritt in Verbindung mit Direktzugriffs-Befehlen auf und zeigt an, daß eine unzulässige Spur- und/oder Sektornummer angegeben wurde.
23	READ ERROR	Prüfsumme fehlerhaft im Datenblock. Dieser Fehler zeigt an, daß in einem oder mehreren Datenbytes ein Fehler vorliegt. Die Daten wurden in den DOS-Bereich geladen, aber die Prüfsumme ist falsch. Diese Meldung kann auch an schlechter Erdung liegen.
24	READ ERROR	Byte-Entschlüsselungsfehler. Die Daten wurden in den DOS-Bereich gelesen, aber aufgrund eines unzulässigen Bit-Musters im Datenbyte entstand ein Hardware-Fehler. Diese Meldung kann auch an schlechter Erdung liegen.
25	WRITE ERROR	Fehler bei der Überprüfung des Geschriebenen. Diese Meldung entsteht, wenn der Controller eine Abweichung zwischen den geschriebenen Daten und jenen im DOS-Bereich entdeckt.

NR.	FEHLERMELDUNG	BESCHREIBUNG
26	WRITE PROTECT ON	Es wurde versucht, auf eine Diskette mit aufgeklebtem Schreibschutz zu schreiben.
27	READ ERROR	Prüfsumme fehlerhaft im Header. Der Disk-Controller hat einen Fehler im Header des angesprochenen Datenblocks gefunden. Der Block wurde nicht in den DOS-Bereich eingelesen. Diese Mel- kann auch an schlechter Erdung liegen.
28	WRITE ERROR	Zu langer Datenblock. Der Controller versucht, das Synchronzeichen des nächsten Headers zu finden, nachdem ein Datenblock geschrieben wurde. Wenn das Synchronzeichen nicht innerhalb einer bestimmten Zeit auftaucht, entsteht die Fehlermeldung. Der Fehler wird durch schlecht formatierte Disketten (die Daten reichen bis in den nächsten Block) oder Hardwarefehler verursacht.
29	DISK ID MISMATCH	Diese Meldung tritt auf, wenn der Controller auf einen Datenblock zugreifen soll, dessen ID nicht mit der im Directory übereinstimmt. Die Meldung kann durch einen zerstörten Header entstehen.
30	SYNTAX ERROR	Das DOS kann den Befehl im Befehlskanal nicht verstehen (unzulässige Anzahl oder Länge von Dateinamen, falsches Muster z.B. im COPY-Befehl, usw.).
31	SYNTAX ERROR	Unzulässiger Befehl. Der Befehl muß an erster Stelle einer Befehlszeile stehen.
32	SYNTAX ERROR	Der Befehl ist länger als 58 Zeichen.
33	SYNTAX ERROR	Unzulässiger Dateiname, z.B. falsche Verwendung des 'Jokers' im OPEN- oder SAVE-Befehl.
34	SYNTAX ERROR	In einem Befehl wurde der Dateiname ausgelassen oder vom DOS nicht als solcher erkannt. Häufig wurde ein Doppelpunkt (:) ausgelassen.
39	SYNTAX ERROR	Dieser Fehler kann auftreten, wenn der über den Befehlskanal (Sekundäradresse 15) geschickte Befehl vom DOS nicht erkannt wird.

NR.	FEHLERMELDUNG	BESCHREIBUNG
50	RECORD NOT PRESENT	Tritt auf, wenn auf einen Record einer relativen Datei zugegriffen wird, der noch nicht angelegt ist (Record-Nummer zu hoch). Kann ignoriert werden, wenn die Datei mit einem PRINT#-Befehl erweitert werden soll. GET# oder INPUT# dürfen nach dieser Meldung nicht ausgeführt werden!
51	OVERFLOW IN RECORD	Eine PRINT#-Anweisung überschreitet die Record-Grenze. Die Informationen werden abgeschnitten. Der Wagenrücklauf, der eine Eingabe abschließt und das Record-Ende markiert, muß bei der ausnutzbaren Länge berücksichtigt werden.
52	FILE TOO LARGE	Die für ein relatives File angegebene Record-Nummer ist zu groß und würde die Kapazität der Diskette überschreiten.
60	WRITE FILE OPEN	Diese Meldung entsteht, wenn eine nicht korrekt abgeschlossene Datei (kein CLOSE nach Beschreiben) wieder geöffnet werden soll.
61	FILE NOT OPEN	Es wurde versucht, eine nicht vorher geöffnete Datei zu benutzen. Manchmal wird die Meldung nicht ausgegeben und der Befehl ignoriert.
62	FILE NOT FOUND	Die gesuchte Datei ist nicht vorhanden.
63	FILE EXISTS	Es wurde versucht, ohne Überschreibungs-Kennzeichnung auf eine schon bestehende Datei zu schreiben.
64	FILE TYPE MISMATCH	Der Typ der angesprochenen Datei stimmt nicht mit dem im Directory vermerkten Typ überein.
65	NO BLOCK	Ein Block, der mit dem B-A-Befehl als belegt gekennzeichnet werden sollte, ist bereits belegt. Die Spur- und Sektornummer in DS$ zeigen den nächsten freien Block an; sind beide Werte Null, sind alle höhernumerierten Blocks belegt.

NR.	FEHLERMELDUNG	BESCHREIBUNG
66	ILLEGAL TRACK AND SECTOR	Das DOS hat versucht, eine Spur oder einen Sektor anzusprechen, die/der im gegebenen Format nicht vorhanden ist. Das kann von Problemen beim Lesen des Zeigers zum nächsten Block herrühren.
67	ILLEGAL SYSTEM T OR S	Diese besondere Fehlermeldung weist auf eine unzulässige Systemspur oder einen unzulässigen Systemsektor hin.
70	NO CHANNEL	Im RAM der Floppy sind alle Kanäle (interne Pufferspeicher) belegt. Je nach Floppy kann nur eine bestimmte Anzahl von Files (abhängig auch von deren Typ) gleichzeitig geöffnet sein.
71	DIRECTORY ERROR	Die BAM (Block Availability Map, Blockverfügbarkeitstabelle) kann nicht gelesen werden. Eventuell wurde sie im DOS-Speicher überschrieben. Um dieses Problem zu beseitigen, initialisieren Sie die Diskette, um die BAM wieder in den DOS-Speicher einzulesen. Vorher geöffnete Dateien können dadurch verstümmelt werden.
72	DISK FULL	Entweder sind alle Blocks der Diskette belegt, oder die Maximalzahl der Directory-Einträge ist erreicht. Die Meldung kommt, wenn z.B. auf der 1541 noch zwei Blocks verfügbar sind, um der momentanen Datei zu ermöglichen, korrekt abgeschlossen zu werden.
73	DOS MISMATCH	(73, CBM DOS V 2.6 1541) DOS 1 und 2 sind lese- aber nicht schreib-kompatibel. Disketten können zwar auf einem Laufwerk des anderen Formats gelesen werden. Bei einem Schreibversuch kommt jedoch diese Fehlermeldung. (Es gibt aber Hilfsprogramme, die ein Format in das andere umwandeln.) Die Meldung mit der DOS-Versionsnummer kann immer direkt nach dem Einschalten über den Fehlerkanal abgefragt werden.
74	DRIVE NOT READY	Es wurde versucht, ein Floppy-Laufwerk, das keine oder eine unformatierte Diskette oder eine mit einem total inkompatiblem Format enthält, anzusprechen.

```
****************************************
*  ABGELEITETE MATHEMATISCHE FUNKTIONEN  *
****************************************
```

Einige Funktionen, die in BASIC 3.5 nicht vordefiniert sind, können mit Hilfe der folgenden Formeln berechnet werden:

SEKANS	SEC(X) =1/COS(X)
COSEKANS	CSC(X) =1/SIN(X)
COTANGENS	COT(X) =1/TAN(X)
ARCUSSINUS	ARCSIN(X)=ATN(X/SQR(1-X*X))
ARCUSCOSINUS	ARCCOS(X)= π /2 - ATN(X/SQR(1-X*X))
ARCUSCOTANGENS	ARCCOT(X)= π /2 - ATN(X)
ARCUSSEKANS	ARCSEC(X)= π /2 - ATN(1/SQR(X*X-1))
ARCUSCOSEKANS	ARCCSC(X)=ATN(1/SQR(X*X-1))
SINUS HYPERBOLICUS	SINH(X) =(EXP(X)-EXP(-X))/2
COSINUS HYPERBOLICUS	COSH(X) =(EXP(X)+EXP(-X))/2
TANGENS HYPERBOLICUS	TANH(X) =1-2*EXP(-X)/(EXP(X)+EXP(-X))
COTANGENS HYPERBOLICUS	COTH(X) =1+2*EXP(-X)/(EXP(X)-EXP(-X))
SEKANS HYPERBOLICUS	SECH(X) =2/(EXP(X)+EXP(-X))
COSEKANS HYPERBOLICUS	CSCH(X) =2/(EXP(X)-EXP(-X))
AREASINUS HYPERBOLICUS	ARSINH(X)=LOG(X+SQR(X*X+1))
AREACOSINUS HYPERBOLICUS	ARCOSH(X)=LOG(X+SQR(X*X-1))
AREATANGENS HYPERBOLICUS	ARTANH(X)=LOG((1+X)/(1-X))/2
AREACOTANGENS HYPERBOLICUS	ARCOTH(X)=LOG((X+1)/(X-1))/2
AREASEKANS HYPERBOLICUS	ARSECH(X)=LOG((1+SQR(1+X*X))/X)
AREACOSEKANS HYPERBOLICUS	ARCSCH(X)=LOG((1+SQR(1-X*X))/X)
BRIGGS-LOG (Basis 10)	LG(X) =LOG(X)/LOG(10)
Runden ganzer Zahlen:	DEFFNI(X)=INT(X+.5)
Runden auf ganze Hundertstel:	DEFFNP(X)=INT(X*100+.5)/100

****
* NOTENWERTE *
****

NOTE	SOUNDREGISTER-WERT	FREQUENZ (Hz)
A	7	110
H	118	123.5
C	169	130.8
D	262	146.8
E	345	164.7
F	383	174.5
G	453	195.9
A	516	220.2
H	571	246.9
C	596	261.4
D	643	293.6
E	685	330
F	704	349.6
G	739	392.5
A	770	440.4
H	798	494.9
C	810	522.7
D	834	588.7
E	854	658
F	864	699
G	881	782.2
A	897	880.7
H	911	989.9
C	917	1045
D	929	1177
E	939	1316
F	944	1398
G	953	1575

Die obige Tabelle enthält die Soundregisterwerte für die Noten von vier Oktaven. Die Werte werden als zweiter Parameter beim SOUND-Befehl benötigt. Um die erste Note in der Tabelle (A mit Soundregisterwert 7) erklingen zu lassen, geben Sie den Befehl SOUND 1,7,30 ein. Mit den folgenden Formeln können Sie die Soundregisterwerte für nicht in der Tabelle enthaltene Frequenzen bestimmen, die noch vom verwendeten Farbfernsehsystem abhängen (in Deutschland PAL, die Tabellenwerte stimmen unabhängig davon gut überein):

SOUNDREGISTERWERT = 1024 - (111860.781 / Frequenz[Hz]) (NTSC)
SOUNDREGISTERWERT = 1024 - (111840.45 / Frequenz[Hz]) (PAL)

****
* KODE - TABELLEN *
****

BILDSCHIRM - KODE

Die folgende Tabelle listet alle Zeichen auf, über die der COMMODORE 116 auf dem Bildschirm verfügt. Sie zeigt, welche Zahlen in den Bildschirmspeicher (3072 bis 4095) gePOKEt werden müssen, um ein gewünschtes Zeichen zu erhalten. Die Tabelle zeigt auch, welches Zeichen einer Zahl entspricht, die vom Bildschirm gePEEKt wurde.

Von den zwei Zeichensätzen ist jeweils nur einer zur Zeit benutzbar. Das bedeutet, Sie können nicht gleichzeitig Zeichen beider Sätze auf dem Bildschirm haben. Die Sätze werden durch gleichzeitiges Niederhalten der Tasten <Shift> und <C=> gewechselt. Im BASIC schaltet PRINT CHR$(142) in den Großbuchstaben-/Grafik-Modus und PRINT CHR$(14) in den Groß-/Kleinbuchstaben-Modus.

Jedes Zeichen in der Tabelle kann auch REVERS angezeigt werden. Den Kode des reversen Zeichen erhalten Sie, wenn Sie zum angegebenen Wert 128 addieren.

Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE
@		0	T	t	20	(		40
A	a	1	U	u	21	)		41
B	b	2	V	v	22	*		42
C	c	3	W	w	23	+		43
D	d	4	X	x	24	,		44
E	e	5	Y	y	25	-		45
F	f	6	Z	z	26	.		46
G	g	7	[		27	/		47
H	h	8	£		28	0		48
I	i	9	]		29	1		49
J	j	10	↑		30	2		50
K	k	11	←		31	3		51
L	l	12	SPACE		32	4		52
M	m	13	!		33	5		53
N	n	14	"		34	6		54
O	o	15	#		35	7		55
P	p	16	$		36	8		56
Q	q	17	%		37	9		57
R	r	18	&		38	:		58
S	s	19	'		39	;		59

Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE	Satz 1	Satz 2	POKE
<		60		T	84			108
=		61		U	85			109
>		62		V	86			110
?		63		W	87			111
		64		X	88			112
	A	65		Y	89			113
	B	66		Z	90			114
	C	67			91			115
	D	68			92			116
	E	69			93			117
	F	70			94			118
	G	71			95			119
	H	72	SPACE		96			120
	I	73			97			121
	J	74			98			122
	K	75			99			123
	L	76			100			124
	M	77			101			125
	N	78			102			126
	O	79			103			127
	P	80			104			
	Q	81			105			
	R	82			106			
	S	83			107			

Die Kodes 128 bis 255 bilden die reversen Zeichen zu den Kodes 0 bis 127.

ASC- UND CHR$-KODES

Diese Tabelle zeigt Ihnen alle möglichen Zeichen (inclusive Steuerzeichen), die abhängig von X erscheinen, wenn Sie PRINT CHR$(X) eingeben. Ebenfalls angegeben werden die Werte, die Sie erhalten, wenn Sie PRINT ASC("A") eingeben, wobei A jedes eintippbare Zeichen (auch Steuerzeichen) sein kann. Dies ist nützlich, wenn Sie das in einer GET-Anweisung erhaltene Zeichen auswerten, Groß-/Kleinbuchstaben-Wandlung durchführen oder auf Zeichen basierende Befehle (wie Schalten zum Groß-/Kleinbuchstaben-Modus), die nicht in Anführungszeichen geschlossen werden können, drucken wollen.

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	0		17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
DISABLES SHIFT C=	8		25	*	42	;	59
ENABLES SHIFT C=	9		26	+	43	<	60
	10	ESCAPE	27	,	44	=	61
	11	RED	28	-	45	>	62
	12		29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
SWITCH TO LOWER CASE	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
D	68		97		126		155
E	69		98		127		156
F	70		99		128		157
G	71		100		129		158
H	72		101		130		159
I	73		102		131		160
J	74		103		132		161
K	75		104		133		162
L	76		105		134		163
M	77		106		135		164
N	78		107		136		165
O	79		108		137		166
P	80		109		138		167
Q	81		110		139		168
R	82		111		140		169
S	83		112		141		170
T	84		113	SWITCH TO UPPER CASE	142		171
U	85		114		143		172
V	86		115		144		173
W	87		116		145		174
X	88		117		146		175
Y	89		118		147		176
Z	90		119		148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	184		186		188		190
	185		187		189		191

Kodes 192-223 identisch mit 96-127
Kodes 224-254 identisch mit 160-190
Kode 255 identisch mit 126

Bildschirm - Steuerzeichen

Bei der Eingabe einer in Anführungszeichen eingeschlossenen Zeichenkette (String) von der Tastatur befindet sich der Rechner im 'Anführungszeichen-Modus', der jeweils nach dem Eintippen weiterer Anführungszeichen aus- und eingeschaltet wird. In dieser speziellen Betriebsart werden die Bildschirm-Steuerbefehle von der Tastatur (z.B. Cursor Home) nicht ausgeführt, sondern als Steuerzeichen in die Zeichenkette eingefügt und gelangen erst beim Abruf des Strings durch einen PRINT-Befehl zur Ausführung. Im Programmlisting werden diese Steuerzeichen durch revers dargestellte Platzhalter-Symbole sichtbar gemacht. Sie finden diese Symbole in der ASC- und CHR$-Tabelle, wenn Sie die Kodezahl des Steuerzeichens (Cursor abwärts z.B. 17) nehmen, 64 aufaddieren (hier 81) und das dazugehörige Zeichen invers darstellen (hier im Groß-/Grafik-Modus ein großes 'Q', im Groß-/Kleinbuchstaben-Modus ein kleines 'q'). Man findet diese Symbole häufig in Programmlistings, die mit einem Matrix-Drucker erstellt wurden. Beim Eintippen eines solchen Programms ist also jeweils die dem Platzhalter-Symbol entsprechende Steuertaste (hier Cursor ab) zu betätigen.

MASCHINENSPRACHEN-MONITOR TEDMON

Einführung

TEDMON ist ein eingebautes Maschinensprachenprogramm, mit dem Sie leicht Programme in Maschinensprache schreiben und testen können. TEDMON enthält einen Monitor für Maschinensprache, einen Mini-Assembler und einen Disassembler.

Programme in Maschinensprache, die mit TEDMON geschrieben werden, können eigenständig ablaufen, oder von BASIC aus als Unterprogramm aufgerufen werden. Dabei muß der letzte Befehl des Unterprogramms ein RTS (Return from Subroutine, $60) sein.

TEDMON-Befehlssatz:

A	Assemble	Wandelt ein Mnemonik (Klartext-Befehl) in den entsprechenden Maschinenkode des 6502 bzw. 7501.
C	Compare	Vergleicht zwei Speicherbereiche und zeigt die Unterschiede.
D	Disassemble	Wandelt Maschinenkode in Mnemoniks (Klartext).
F	Fill	Füllt einen Speicherbereich mit wählbarem Wert.
G	Go	Startet Maschinenprogramm an angegebener Adresse.
H	Hunt	Durchsucht einen Speicherbereich nach einem bestimmten Wert und zeigt alle Speicherplätze an, die diesen Wert enthalten.
L	Load	Lädt ein Programm von Kassette oder Diskette.
M	Memory	Zeigt alle Inhalte eines wählbaren Speicherbereichs in Hexdarstellung an.
R	Registers	Ausgabe der aktuellen Registerinhalte.
S	Save	Speichert ein Programm auf Kassette oder Diskette.
T	Transfer	Blockkopierbefehl, kopiert einen bestimmten Speicherbereich in einen anderen.
V	Verify	Vergleicht ein Programm im Arbeitsspeicher mit einem auf Kassette oder Diskette.
X	Exit	Zurück zu BASIC.
.	(Punkt)	Entspricht dem A (Assemble).
>	(größer als)	Ändert bis zu 8 Bytes ab bestimmter Speicherstelle (nach M-Befehl).
;	(Semikolon)	Ändert die 7501-Registerinhalte (nach R-Befehl).

Mit der Speicherstelle $07F8 ist es möglich, im Speicherbereich ab $8000 zwischen ROM (BASIC, Kernal) und RAM zu wählen. Hat die Speicherstelle den Wert 00, kann das ROM gelesen werden, beim Wert $80 das darunterliegende RAM. Das ist oft nützlich bei der Entwicklung von Maschinenprogrammen. Beachten Sie, daß die Speicherstelle $07F8 keinen Einfluß auf den Go-Befehl hat, dieser startet ein Programm abhängig von der aktuellen Speicherkonfiguration (RAM oder ROM aktiv) im jeweiligen Bereich, unabhängig vom Inhalt in $07F8. Beim COMMODORE 116 wird das nur 16 KByte große RAM dreimal in höhere Adreßbereiche 'gespiegelt', ein Zugriff per TEDMON auf z.B. RAM-Zelle $A000 wirkt daher auf $2000.

Benutzen von TEDMON:

Rufen Sie TEDMON auf, indem Sie eingeben:

MONITOR (oder die Abkürzung mO).

TEDMON antwortet, indem die Inhalte der Prozessor-Register angezeigt werden und der Cursor blinkt. Der Cursor ist Ihr 'Prompt', der Ihnen anzeigt, daß TEDMON auf Ihre Befehle wartet.

Beschreibung der einzelnen Befehle:

Befehl: A
Zweck: Eingeben einer Zeile Assemblerkode
Schreibweise: A <Adresse> <Mnemonischer Befehlskode> [<Operand>]

<Adresse> Eine Hexadezimalzahl, die die Speicherstelle angibt, wohin der Assemblerbefehl plaziert werden soll.
<Mnemonischer Befehlskode> Ein Klartext-Assemblerbefehl (z.B. LDA).
<Operand> Ob und was für ein Operand angegeben wird, legt die Adressierungsart des Befehls fest (z.B. sind für Zero-Page-Adressierung Werte zwischen 00 und $FF einzugeben, bei absoluter Adressierung 0000 bis $FFFF).

Die fertige Befehlszeile wird mit der Taste <Return> übergeben. Wenn die Zeile Fehler enthält, wird ein Fragezeichen angezeigt, und der Cursor geht zur nächsten Zeile. Der Fehler kann auf dem Schirm korrigiert werden.

Nachdem eine Zeile korrekt erstellt ist, wird ein Prompt angezeigt, der die nächste Speicherstelle schon bereitstellt, sodaß A und diese Speicherstelle nicht mehr eingetippt zu werden brauchen.

Beispiel: A 1200 LDX #$2A
A 1202

Anmerkung: Ein Punkt (.) ist dem A-Befehl gleichgestellt.
Beispiel: .2000 LDA #$23

Befehl: C
Zweck: Vergleichen von zwei Abschnitten des Arbeitsspeichers
Schreibweise: C <Adresse 1> <Adresse 2> <Adresse 3>

<Adresse 1> Hexadezimalzahl, die den Anfang des zu vergleichenden Abschnitts angibt.
<Adresse 2> Hexadezimalzahl, die das Ende davon angibt.
<Adresse 3> Hexadezimalzahl, die den Anfang des anderen zu vergleichenden Abschnitts angibt.

Wenn die beiden Abschnitte des Arbeitsspeichers den gleichen Inhalt haben, gibt TEDMON "RETURN" aus. Bemerkt er Unterschiede, werden die Adressen der abweichenden Bytes angezeigt.

Befehl: D
Zweck: Disassemblieren, Zurückübersetzen von Maschinensprache in mnemonische Assemblerbefehle und Operanden.
Schreibweise: D [<Adresse 1>] [<Adresse 2>]

<Adresse 1> Hexadezimalzahl, die die Startadresse für die Rückübersetzung angibt.
<Adresse 2> Hexadezimalzahl, die die Endadresse angibt.

Das Ausgabeformat des D-Befehls unterscheidet sich nur wenig von dem des A-Befehls. Das erste Zeichen einer Rückübersetzung ist ein Punkt statt eines "A". Zusätzlich werden die Hexadezimalzahlen der Kodes aufgelistet.

Eine Disassemblerliste kann über den Bildschirmeditor geändert werden. Ändern Sie den mnemonischen Befehlskode oder den Operanden und drücken Sie <Return>. Der A-Befehl wird aufgerufen und der geänderte Befehl in den Speicher geschrieben.

In der Disassemblerliste kann geblättert werden. Das Drücken eines D's (ohne Adreßangaben) bewirkt, daß die nächste Seite auf dem Schirm erscheint.

Beispiel:
```
D 3000 3004
. 3000  A9 00    LDA #$00
. 3002  FF       ???
. 3003  D0 2B    BNE $3030
```

Befehl: F
Zweck: Füllen eines Speicherbereichs mit einem bestimmten Bytewert.
Schreibweise: F <Adresse 1> <Adresse 2> <Byte>

<Adresse 1> Erste Adresse, die mit dem <Byte> zu füllen ist.
<Adresse 2> Letzte Adresse dieses Speicherbereichs.
<Byte> Ein- oder zweistellige Hexadezimalzahl angeben.

Beispiel: F 0400 0518 EA
Füllt den Speicherbereich von $0400 bis $0518 mit dem Wert $EA (einer NOP-Anweisung).

Befehl: G
Zweck: Startet die Ausführung eines Maschinenprogramms an der angegebenen Adresse.
Schreibweise: G [<Adresse>]

<Adresse> Eine Adresse (bis zu 4stellige Hexadezimalzahl), die den im R-Befehl angezeigten PC-Inhalt (PC = Program Counter = Befehlsadreßzeiger) verändert. Die Ausführung erfolgt ab bisherigem oder neu eingegebenen PC.

Der Go-Befehl stellt den alten Zustand der Register (die durch den R-Befehl angezeigt werden können) wieder her und beginnt die Ausführung an der (ggf. neu spezifizierten) PC-Adresse. Bei Anwendung des Go-Befehls ist Vorsicht geboten, damit der Rechner nicht durch ein fehlerhaftes Programm 'abstürzt'. Um nach Ausführen eines Maschinenprogramms zum TEDMON zurückzukehren, benutzen Sie die BRK-Anweisung ($00), nicht den RTS-Befehl!

Beispiel: G 1400
Die Ausführung beginnt mit der Speicherstelle $1400.

Befehl: H
Zweck: Durchsuchen eines Speicherbereichs nach angegebenen Inhalten.
Schreibweise: H <Adresse 1> <Adresse 2> <Daten>

<Adresse 1> Anfangsadresse des zu durchsuchenden Speicherbereichs.
<Adresse 2> Endadresse davon.
<Daten> Die gesuchten Daten können hexadezimal oder als ASCII-Zeichenkette angegeben werden. Bei der Angabe in ASCII muß dem ersten Zeichen ein Apostroph vorausgehen, z.B. 'ZEICHENKETTE. Die Daten können aus einem oder mehreren Elementen bestehen. Bei mehreren Elementen und hexadezimaler Schreibweise muß zwischen jeder Zahl eine Leerstelle stehen.

Beispiele:
```
H C000 FFFF 'READ
```
Suche im Bereich $C000 bis $FFFF nach der Zeichenkette "READ".

```
H A000 A101 A9 FF 4C
```
Suche im Bereich $A000 bis $A101 nach den Daten $A9, $FF und $4C.

Befehl: L
Zweck: Laden eines Programmfiles von Kassette oder Diskette.
Schreibweise: L <"Filename">,<Geräte-#>

<"Filename"> Ein beliebiger, zulässiger Dateiname in Anführungszeichen.
<Geräte-#> Eine Hexadezimalzahl, die die Gerätenummer angibt, 1 = Datassette, 8 = Floppy (oder 9, A, usw.).

Der LOAD-Befehl bewirkt, daß eine Programmdatei in den Arbeitsspeicher geladen wird. Die Anfangsadresse ist immer am Anfang eines Programms abgespeichert. Der LOAD-Befehl lädt ein Programm stets an dieselbe Stelle, von der es abgespeichert wurde. Dies ist für die Arbeit mit Maschinenprogrammen sehr wichtig, da nur wenige Programme 'relokierbar' sind, d.h. an anderer Stelle im Speicher ohne Änderungen lauffähig sind. Die Datei wird bis zu ihrer EOF-Marke (EOF = End Of File = Dateiendemarkierung) in den Arbeitsspeicher geladen.

Beispiele:
```
L "SCHIRM",1
```
Liest ein Programm "SCHIRM" von Kassette.

```
L "0:BLINKER",8
```
Lädt das Programm "BLINKER" vom Diskettenlaufwerk 0.

Befehl: M
Zweck: Anzeige des Speicherinhalts in Hexadezimaldarstellung und ASCII-Zeichen als Speicherauszug ('Dump').
Schreibweise: M [<Adresse 1>] [<Adresse 2>]

<Adresse 1> Startadresse des Speicherauszugs, frei wählbar. Wird sie nicht angegeben, wird von der um 96 erhöhten zuletzt verwendeten Startadresse ausgegangen.
<Adresse 2> Endadresse des anzuzeigenden Speicherbereichs, frei wählbar. Wird sie nicht angegeben, so wird die aktuelle Speicheradresse +96 angenommen.

Der Speicherinhalt wird in folgendem Format angezeigt (Beispiel):

>A048 41 E7 00 AA AA 00 98 56 :A!.**..V

Der Speicherinhalt kann durch den Bildschirmeditor geändert werden. Bringen Sie den Cursor zu den zu ändernden Hex-Daten, geben Sie die Korrektur ein und drücken Sie <Return>. Falls eine defekte RAM-Zelle gefunden oder ein Versuch gemacht wird, eine ROM-Speicherstelle zu ändern, wird das Fehlerflag (?) angezeigt.

Der ASCII-Speicherauszug der Daten wird REVERS (um sich von den anderen Daten abzuheben) rechts von den hexadezimalen Daten angezeigt. Wenn ein Zeichen nicht druckbar ist, wird es als reverser Punkt (.) angezeigt.

Wie beim DISASSEMBLER-Befehl können Sie auch hier durch Eingabe von M allein und Drücken der Taste <Return> weiterblättern.

Befehl: > (Größer-als-Zeichen)
Zweck: 1 bis 8 Speicherstellen können auf einmal gesetzt werden.
Schreibweise: > <Adresse> <Datenbyte 1> [<Datenbytes 2...8>]

<Adresse> Erste Speicherstelle, die gesetzt werden soll.
<Datenbyte 1> Bis zu zweistellige Hex-Daten, die in die erste Speicherstelle gesetzt werden sollen.
<Datenbytes 2...8> Hex-Daten, die in die folgenden Speicherstellen abgelegt werden sollen.

Dieser Befehl wird automatisch ausgeführt, wenn nach dem M-Befehl mit dem Bildschirmeditor Speicherinhalte geändert werden.

Beispiele: >2000 08
Setzt eine 08 in Speicherstelle $2000.

>3000 23 45 65
Setzt eine $23 in Speicherstelle $3000, eine $45 in $3001 und eine $65 in $3002.

Befehl: R
Zweck: Anzeige der Prozessor-Registerinhalte, und zwar: Programmzähler PC, Prozessorstatusregister SR, Akkumulator AC, X- und Y-Indexregister XR und YR und den Stackpointer SP.
Schreibweise: R

Beispiel:
```
R
   PC  SR AC XR YR SP
; 3020 00 20 00 10 F8
```

Anmerkung: Das Semikolon (;) kann in gleicher Weise zur Änderung der Registerinhalte benutzt werden wie das ">" beim M-Befehl.

Befehl: S
Zweck: Abspeichern des Speicherinhalts als Programmfile auf Kassette oder Diskette.
Schreibweise: S <"Filename">,<Geräte-#>,<Adresse 1>,<Adresse 2>

<"Filename"> Beliebiger zulässiger Dateiname, der in Anführungszeichen eingeschlossen sein muß.
<Geräte-#> Die beiden möglichen Gerätetypen sind Kassette und Diskette. Für Kassette geben Sie hier 1 an, für das Diskettenlaufwerk normalerweise 8 (bei mehreren auch 9 oder A, usw., siehe Floppy-Handbuch).
<Adresse 1> Anfangsadresse des abzuspeichernden Speicherbereichs.
<Adresse 2> Endadresse + 1 davon. Alle Daten bis ausschließlich dieser Adresse werden abgespeichert.

Mit diesem Befehl wird eine Programmdatei erzeugt, die Byte für Byte den Speicherinhalt wiedergibt. Als erste zwei Bytes wird die Anfangsadresse <Adresse 1> in LO-HI-Reihenfolge abgespeichert (Ladeadresse). Das Programm kann mit dem L-Befehl wieder geladen werden.

Beispiel:
```
S "SPIEL",8,0400,0BFF
```
Speichert den Inhalt von $0400 bis $0BFE unter dem Namen "SPIEL" als Programmfile auf die Diskette.

Befehl: T
Zweck: Kopieren von Speicherinhalten in einen anderen Bereich des Arbeitsspeichers.
Schreibweise: T <Adresse 1> <Adresse 2> <Adresse 3>

<Adresse 1> Anfangsadresse der zu verschiebenden Daten.
<Adresse 2> Endadresse davon.
<Adresse 3> Anfangsadresse der neuen Plazierung, ab der die Daten abgespeichert werden sollen.

Daten können von einem niedrigen in einen höheren Speicherbereich übertragen werden und umgekehrt. Überlappen sich Ausgangs- und Zielbereich, so ist allerdings nur Rückwärtsverschieben möglich. Notfalls ist der Umweg über einen ganz anderen Speicherbereich zu nehmen.

Beispiel: T 1400 1600 3400
Der Speicherbereich von $1400 bis einschließlich $1600 wird nach $3400 bis $3600 kopiert.

Befehl: V
Zweck: Vergleichen einer Programmdatei auf Kassette oder Diskette mit dem Inhalt des Arbeitsspeichers.
Schreibweise: V <"Filename">,<Geräte-#>

<"Filename"> Beliebiger zulässiger Dateiname, der in Anführungszeichen eingeschlossen sein muß.
<Geräte-#> Hexadezimalzahl, die angibt, von welchem Gerät die Datei gelesen werden soll: 1 für Kassette, 8 (oder 9 oder A, usw.) für Diskette.

Der Verify-Befehl vergleicht eine Programmdatei mit dem Inhalt des Arbeitsspeichers. Der COMMODORE 116 zeigt auf dem Bildschirm "VERIFYING" an. Wenn eine Abweichung gefunden wird, wird "ERROR" ausgegeben. Wenn kein Fehler gefunden wurde, erscheint einfach wieder der blinkende Cursor.

Beispiel: V "TESTPRG",8
Vergleicht das Disketten-Programm "TESTPRG" mit dem aktuellen Speicherinhalt.

Befehl: X
Zweck: Rückkehr zu BASIC.
Schreibweise: X

Bei Ausführung des X-Befehls wird der Stackpointer SP auf den Wert gesetzt, wie er im R-Befehl angezeigt wurde. Wenn dieser irgendwie verändert wurde, benutzen Sie nach der Rückkehr ins BASIC den BASIC-Befehl CLR, um alle Zeiger auf korrekte Werte zurückzustellen.

****
* BITS UND BYTES *
****

Der Prozessor im COMMODORE 116 heißt 7501, er ist ein Nachfolgetyp des 6502 und ein 8-Bit-Prozessor. Mit jeder Speicherstelle werden also vom Prozessor gleichzeitig 8 Bits angesprochen, ein 'Byte'.

Mit 8 Bits lassen sich Zahlenwerte von 0-255 darstellen, nämlich 2*2*2*2 * 2*2*2*2 = 2 hoch 8 verschiedene, weil jedes Bit unabhängig von den anderen die 2 Zustände 0 oder 1 annehmen kann. Dieser Zahlenbereich ist bei PEEK- und POKE-Befehlen zu beachten, die den Inhalt einzelner Speicherstellen auslesen bzw. neu setzen.

Ein 8-Bit-Prozessor kann maximal 65536 = 2 hoch 16 Speicherstellen verwalten. Der Adreßbereich umfaßt die Adressen 0 bis 65535 (bei PEEK, POKE und SYS zu beachten). Dieser Umfang kommt dadurch zustande, daß der Rechner zur Speicheradressierung jeweils 2 Bytes, also insgesamt 16 Bits, verwendet, die er ggf. nacheinander aus dem Programmspeicher holen muß (z.B. bei Adressierungsart 'absolut').

Z.B. bei Verwendung der USR-Funktion muß man diese beiden Bytes (die die Anfangsadresse der eigenen Maschinenroutine angeben) selbst auf 2 aufeinanderfolgende Speicherstellen ('Pointer') schreiben. Die beiden Adreßbytes werden unterschieden in das LO-Byte, das die unteren 8 Binärstellen der insgesamt ja 16stelligen Adresse aufnimmt, und das HI-Byte mit den oberen 8 Binärstellen. Dadurch hat das HI-Byte gegenüber dem LO-Byte eine 256fache Wertigkeit. Allgemein muß bei diesem Prozessortyp die sog. 'LO-HI-Reihenfolge' eingehalten werden. Dabei liegt das LO-Byte im Speicher direkt vor dem HI-Byte!

```
Zerlegung einer Adresse AD in LO- und HI-Byte:
HI = INT( AD / 256 ): LO = AD - 256 * HI              .
Ermittlung einer vollen Adresse AD aus LO- und HI-Byte:
AD = LO + 256 * HI                                    .
Setzen eines Pointers in P und P+1 auf den Wert AD:
HI=INT(AD/256):LO=AD-256*HI:POKE P,LO:POKE P+1,HI     .
Lesen eines Pointers aus P und P+1 in die Variable AD:
AD = PEEK( P ) + 256 * PEEK( P+1 )                    .
```

Wie man leicht erkennt, ist die dezimale Schreibweise für Binärzahlen sehr umständlich. Daher wurden die 'Hexadezimal'-Zahlen eingeführt. In der dezimalen Stellenschreibweise ist ja eine Stelle jeweils 10mal mehr wert als die rechts davon, weil eine Stelle 10 verschiedene Werte (0-9) annehmen kann. Im Hexadezimalsystem werden je 4 Bits (ein 'Nibble') zu einer Stelle, einer Hexadezimalziffer, zusammengefaßt. Durch 4 Bits können aber 16 verschiedene Zahlenwerte (0-15) dargestellt werden, d.h. die 10 herkömmlichen Dezimalziffern reichen nicht aus. Für die neuen Hex-Ziffern mit den Wertigkeiten 10-15 wurden daher die Buchstaben A-F ausgewählt. In der hexadezimalen Stellen-Schreibweise ist folgerichtig eine Stelle jeweils 16mal mehr wert als die rechts davon (16*16=256, 16*256=4096, 16*4096=65536). Hex-Zahlen werden zur Unterscheidung mit einem vorangestellten Dollarzeichen ($) gekennzeichnet, z.B. $0F = 15 (dez.!) oder $51 = 81 = 5*16+1.

Ein Byte umfaßt 8 Bits, also zwei Hexstellen, und kann Werte von $00 bis $FF (= 0 bis 15*16+15 = 255) annehmen. Eine Adresse aus zwei Bytes umfaßt vier Hexstellen mit Werten von $0000 bis $FFFF (= 0 bis 15*4096+15*256+15*16+15*1 = 65535).

```
Umrechnung Hex nach Dezimal:
z.B. $7D95          =
          5 * 1
        +9  * 16
      +13   * 256
      +7    * 4096  = 32149 = DEC( "7D95" )  .

Umrechnung Dezimal nach Hex:
Zuerst ggf. wie oben in LO- und HI-Byte zerlegen. Innerhalb jedes Bytes
BY Zerlegung in LO-Nibble LN und HI-Nibble HN:
HN = INT( BY / 16 ): LN = BY - 16 * HN
Umwandlung in Hexziffer:   PRINT HEX$(Dezimalzahl)
```

Bei reinen Binärzahlen ist eine Stelle jeweils nur 2mal mehr wert als die rechts davon (Kennzeichnung durch vorangestelltes %). In BASIC können Zahlen, die bis zu 16stelligen Binärzahlen entsprechen, durch die Operatoren AND, OR und NOT binärverknüpft werden. Zur Wirkungsweise dieser Befehle muß immer die Binärdarstellung der beteiligten Zahlen klar sein:

```
Umrechnung Binär nach Dezimal:
z.B.    %1111 = 1*8 + 1*4 + 1*2 + 1*1 = 15
        %1001 = 8 + 1 = 9
Verknüpfung durch Operatoren:
z.B.    5 OR  7 = %101 OR  %111 = %111 = 7
        3 AND 6 = %011 AND %110 = %010 = 2
Bei der Negation NOT wird das 'Zweierkomplement' gebildet (bitweise
Invertierung und Addition von1):
        NOT  10 = NOT %0000 0000 0000 1010
                =     %1111 1111 1111 0101 + 1
                =     %1111 1111 1111 0110
                =     -11  (neg. Zahl wegen gesetztem höchsten Bit),
        NOT  -1 = 0        (BASIC-Werte für 'wahr' und 'falsch').
```

```
Einheit KByte:   1 KByte = 1024 Bytes = (2 hoch 10) Bytes
Das K ist groß zu schreiben, um es vom herkömmlichen k (für kilo,
Faktor 1000) zu unterscheiden:
1 KByte = 1,024 kByte    und   64 KByte = 65,536 kByte
```

```
************************************
*  SPEICHERBELEGUNG  (MEMORY MAP)  *
************************************

Adresse          RAM                                    ROM
-------------------------------------------------------------------------
$FFFF 65535                                          * ROM-BANK-HIGH      *
$FFFE 65534                                          IRQ-Vektor
$FFFC 65532                                          RES-Vektor
$FFFA 65530                                          NMI-Vektor

                                                     Kernal-Sprungtabelle
$FF81 65409
            *  I/O-Adressen (bis $FEFF 65279) und TED-CHIP-Register *
$FD00 64768
                                                     ROM-BANKING-Routinen
$FC00 64512
                                                     Betriebssystem
$D800 55296
                                                     Character-Tabelle
$D000 53248
                                                     BASIC-Erweiterungen
$C000 49152                                          **********************
$BFFF 49151
                                                     * ROM-BANK-LOW       *
                                                     BASIC
$8000 32768                                          **********************
            ***************************
$3FFF 16383 *     Ende 16-KByte-RAM     *
            BASIC-RAM           (normal)
            Bildschirmspeicher (Grafik)
$2000  8192
            Farbtabelle (Grafik)
$1C00  7168
            Luminanz (Grafik)
$1800  6144

$17FF  6143 Ende BASIC-RAM bei Grafik

            BASIC-RAM
$1000  4096
            Bildschirmspeicher (Text)
$0C00  3072
            Farbspeicher (Text)
$0800  2048
            Systemspeicher
$0000     0
```