C64 SuperChart – Zeropage v2.0

geschrieben von Andreas Potthoff | 10. Juni 2023
Diese C64 Superchart v2.0 gibt einen Überblick über die CPU Opcodes, Zeropage Beschreibung, BASIC V2 Token, ASCII-Zeichen und die PETSCII-Zeichen bzw. Steuercodes.

Code

- Die Werte sind hier in dezimal (0-255) und hexadezimal ($00-$FF) angegeben.

CPU OpCodes

- Operation Codes der CPU (MOS 6502 / 6510 / 8500)
- Illegale bzw. undokumentierte OpCodes sind in eckigen Klammern […] geschrieben.
- Adressierungsarten:
    - abs – absolute Adressierung
    - abx – absolute X-indizierte Adressierung
    - aby – absolute Y-indizierte Adressierung
    - akk – Akkumulator Adressierung
    - imm – immediate – Unmittelbare Adressierung
    - imp – implizite Adressierung
    - ind – indirekte Adressierung
    - inx – indirekte X-indizierte Zeropage-Adressierung
    - iny – indirekte Y-nachindizierte Zeropage-Adressierung
    - rel – relative Adressierung
    - zp – Zeropage-Adressierung

- zpx – Zeropage X-indizierte Adressierung
- zpy – Zeropage Y-indizierte Adressierung

ZeroPage

- Ein Block von 256 Bytes (Seite Null) die für KERNAL- und BASIC-Routinen genutzt werden.
- Erfahrene Programmierer (z.B. Demo-Coder) nutzen diese Routinen in Maschinensprache.
- 1-Byte Adressierung

BASIC

- Das entsprechende Token (Kommando) für den Basic V2 Interpreter.

ASCII

- Das entsprechende ASCII-Zeichen.
- ASCII Codes liegen nur im Bereich von 0-127 ($00-$FF).

PETSCII / Fonts

- Das Zeichen (Font) oder der Steuercode im Upper- (Großschrift) und Lower-Modus (Kleinschrift).
- Die Codes für die Farben haben für eine bessere Sortierung einen . (Punkt) vor dem Wort.
- Ab Code 128 / $80 sind die Fonts alle invertiert. In dieser Tabelle kann ich das leider nicht grafisch darstellen.

Tabelle

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
0 $00	BRK impl	Prozessorport Datenrichtungsregister I/O-Data Bit 0 - 6; 0 = Eingang (read only), 1 = Ausgang (read/write) Bit 0: Direction of Bit 0 I/O on port at next address. Default = 1 (output) Bit 1: Direction of Bit 1 I/O on port at next address. Default = 1 (output) Bit 2: Direction of Bit 2 I/O on port at next address. Default = 1 (output) Bit 3: Direction of Bit 3 I/O on port at next address. Default = 1 (output) Bit 4: Direction of Bit 4 I/O on port at next address. Default = 0 (input) Bit 5: Direction of Bit 5 I/O on port at next address. Default = 1 (output) Bit 6: Direction of Bit 6 I/O on port at next address. Not used. Bit 7: Direction of Bit 7 I/O on port at next address. Not used. Default: $2F, %00101111.	0 = Ende Zeile 00 = Ende Prog.	{NUL}		

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
1 $01	ORA inx	Prozessorport I/O-Port Bit 0: LORAM signal. Selects ROM or RAM at 40960 ($A000). 1=BASIC, 0=RAM Bit 1: HIRAM signal. Selects ROM or RAM at 57344 ($E000). 1=Kernal, 0=RAM Bit 2: CHAREN signal. Selects character ROM or I/O devices. 1=I/O, 0=ROM Bit 3: Cassette Data Output line. Bit 4: Cassette Switch Sense. Reads 0 if a button is pressed, 1 if not. Bit 5: Cassette Motor Switch Control. A 1 turns the motor on, 0 turns it off. Bits 6-7: Not connected--no function presently defined. Default: $37, %00110111.		{SOH}		
2 $02	[STP]	Unbenutzt		{STX}		
3 $03	[SLO] inx	Jump Vector: Umwandlung von Fließkommazahl nach Ganzzahl ($B1AA)		{ETX}	RUN/STOP	RUN/STOP
4 $04	[NOP] zp	Jump Vector: Umwandlung von Fließkommazahl nach Ganzzahl ($B1AA)		{EOT}		
5 $05	ORA zp	Jump Vector: Umwandlung Ganzzahl nach Fließkommazahl ($B391)		{ENQ}	.WHITE	.WHITE

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
6 $06	ASL zp	Jump Vector: Umwandlung Ganzzahl nach Fließkommazahl ($B391)		{ACK}		
7 $07	[SLO] zp	Suchzeichen Texteingabe BASIC		{BEL}		
8 $08	PHP imp	Suchzeichen Befehlsende Hochkomma		{BS}		
9 $09	ORA imm	Spaltenposition (0-79) des Cursors nach dem letzten TAB- oder SPC-Befehl		{HT}	Unlock	Unlock
10 $0A	ASL akk	Flag: LOAD+VERIFY $00 = LOAD $01-$FF = VERIFY Load: POKE 10,0:SYS 57705		{LF}		
11 $0B	[ANC] imm	Flags: Pointer im Eingabepuffer; Anzahl der Dimensionen; AND/OR Switch $00 = AND $FF = OR		{VT}		
12 $0C	[NOP] abs	Flag: DIM $00 Operation nicht durch DIM-Befehl $40-$7F Operation durch DIM-Befehl		{FF}		
13 $0D	ORA abs	Flag: Datentyp $00 = numerisch $FF = String		{CR}	Carrige Return	Carrige Return
14 $0E	ASL abs	Flag: Zahlentyp $00 = Gleitkommazahl $80 = Ganzzahl		{SO}	UPPER/LOWER	UPPER/LOWER

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
15 $0F	[SLO] abs	Flags: LIST Quote; Garbage Collection; Tokenization $00-$7F = GC wurde noch nicht ausgeführt $FF = GC wurde bereits ausgeführt Konvertierung der Text-Zeilen in Tokens und Speicherung im Input Buffer ($0200, 512)		{SI}		
16 $10	BPL rel	Flags: Variablen-Namen; DEF FN $00 = Integer-Variablen werden akzeptiert $01-FF = Integer-Variablen werden nicht akzeptiert		{DLE}		
17 $11	ORA iny	Flag: INPUT, GET oder READ $00 = INPUT $40 = GET $98 = READ		{DC1}	CRSR DOWN	CRSR DOWN
18 $12	[STP]	Flags: Vorzeichen bei SIN, COS und TAN; Vergleichsoperator für Vergleichsoperationen Vorzeichen: $00 = Positive $FF = Negative Operator: $00 = < OR =, > OR = $01 = > $02 = = $03 = >= $04 = < $05 = <> $06 = <=		{DC2}	RVS ON	RVS ON
19 $13	[SLO] iny	Flag: Aktives I/O-Device $00 = Direkteingabe Default: $00		{DC3}	CRSR HOME	CRSR HOME

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
20 $14	[NOP] zpx	Temp: Integer-Wert Zeilennummer für LIST, GOTO, GOSUB und ON; Speichert höchste Zeilennummer bei LIST, Default: 65535 ($FFFF); Pointer der Adresse bei PEEK, POKE, SYS und WAIT		{DC4}	DEL	DEL
21 $15	ORA zpx	Temp: Integer-Wert Zeilennummer für LIST, GOTO, GOSUB und ON; Speichert höchste Zeilennummer bei LIST, Default: 65535 ($FFFF); Pointer der Adresse bei PEEK, POKE, SYS und WAIT		{NAK}		
22 $16	ASL zpx	Pointer: Stringstack Werte: $19; $1C; $1F; $22 Default: $19 BASIC: FORMULA TOO COMPLEX ERROR (Stack full)		{SYN}		
23 $17	[SLO] zpx	Pointer: Adresse des letzten Strings im String Stack		{ETB}		
24 $18	CLC imp	Pointer: Adresse des letzten Strings im String Stack		{CAN}		
25 $19	ORA aby	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{EM}		
26 $1A	[NOP] imp	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{SUB}		
27 $1B	[SLO] aby	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{ESC}	ESC	ESC
28 $1C	[NOP] abx	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{FS}	.RED	.RED

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
29 $1D	ORA abx	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{GS}	CURSOR RIGHT	CURSOR RIGHT
30 $1E	ASL abx	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{RS}	.GREEN	.GREEN
31 $1F	[SLO] abs	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{US}	.BLUE	.BLUE
32 $20	JSR abs	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		{SPACE}		
33 $21	AND inx	Temp: Stringstack Drei Einträge mit jeweils 3 Byte		!	!	!
34 $22	[JAM]	Temp Pointer: First Utility Pointer (INDEX1)		"	"	"
35 $23	[RLA] inx	Temp Pointer: First Utility Pointer (INDEX1)		#	#	#
36 $24	BIT zp	Temp Pointer: Second Utility Pointer (INDEX2)		$	$	$
37 $25	AND zp	Temp Pointer: Second Utility Pointer (INDEX2)		%	%	%
38 $26	ROL zp	Register für Arithmetik (Multiplikation und Division)		&	&	&
39 $27	[RLA] zp	Register für Arithmetik (Multiplikation und Division)		´	´	´
40 $28	PLP imp	Register für Arithmetik (Multiplikation und Division)		(	(	(
41 $29	AND imm	Register für Arithmetik (Multiplikation und Division)		)	)	)
42 $2A	ROL akk	Register für Arithmetik (Multiplikation und Division)		*	*	*

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
43 $2B	[ANC] imm	Pointer: Beginn Speicherbereich BASIC Programme (Text) Default: $0801, 2049 Beginn Programmspeicher: PRINT PEEK (43) + PEEK (44)*256		+	+	+
44 $2C	BIT abs	Pointer: Beginn Speicherbereich BASIC Programme (Text) Default: $0801, 2049 Beginn Programmspeicher: PRINT PEEK (43) + PEEK (44)*256		,	,	,
45 $2D	AND abs	Pointer: Beginn Speicherbereich Variablen End of program +1		-	-	-
46 $2E	ROL abs	Pointer: Beginn Speicherbereich Variablen End of program +1		.	.	.
47 $2F	[RLA] abs	Pointer: Beginn Speicherbereich Arrays		/	/	/
48 $30	BMI rel	Pointer: Beginn Speicherbereich Arrays		0	0	0
49 $31	AND iny	Pointer: Ende Speicherbereich Arrays +1		1	1	1
50 $32	[JAM]	Pointer: Ende Speicherbereich Arrays +1		2	2	2
51 $33	[RLA] iny	Pointer: Ende Speicherbereich Textspeicher / Strings Default: $A000		3	3	3
52 $34	[NOP] zpx	Pointer: Ende Speicherbereich Textspeicher / Strings Default: $A000		4	4	4

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
53 $35	AND zpx	Pointer: Adresse des letzten Strings im Speicher		5	5	5
54 $36	ROL zpx	Pointer: Adresse des letzten Strings im Speicher		6	6	6
55 $37	[RLA] zpx	Pointer: Ende Speicherbereich BASIC-RAM Default: $A000, 40960		7	7	7
56 $38	SEC imp	Pointer: Ende Speicherbereich BASIC-RAM Default: $A000, 40960		8	8	8
57 $39	AND aby	Pointer: Aktuell ausgeführte Zeilennummer vom BASIC-Programm Zeilennummer: $00-$F9FF, 0-63999 Direktmodus: $FF00-$FFFF		9	9	9
58 $3A	[NOP] imp	Pointer: Aktuell ausgeführte Zeilennummer vom BASIC-Programm Zeilennummer: $00-$F9FF, 0-63999 Direktmodus: $FF00-$FFFF		:	:	:
59 $3B	[RLA] aby	Pointer: Letzte Zeilennummer bei Programmunterbrechung für CONT Nach STOP, END oder STOP-Taste		;	;	;
60 $3C	[NOP] abx	Pointer: Letzte Zeilennummer bei Programmunterbrechung für CONT Nach STOP, END oder STOP-Taste		<	<	<

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
61 $3D	AND abx	Pointer: Nächster BASIC-Befehl für CONT $00-$FF: CONT nicht möglich $0100-$FFFF: Nächster BASIC-Befehl		=	=	=
62 $3E	ROL abs	Pointer: Nächster BASIC-Befehl für CONT $00-$FF: CONT nicht möglich $0100-$FFFF: Nächster BASIC-Befehl		>	>	>
63 $3F	[RLA] abx	Pointer: Aktuell ausgeführte Zeilennummer bei DATA für READ		?	?	?
64 $40	RTI imp	Pointer: Aktuell ausgeführte Zeilennummer bei DATA für READ		@	@	@
65 $41	EOR inx	Pointer: Nächste Zeilennummer bei DATA für READ		a	A	a
66 $42	[JAM]	Pointer: Nächste Zeilennummer bei DATA für READ		b	B	b
67 $43	[SRE] inx	Pointer: Adresse Input Buffer ($0200, 512) bei GET, READ, INPUT		c	C	c
68 $44	[NOP] zp	Pointer: Adresse Input Buffer ($0200, 512) bei GET, READ, INPUT		d	D	d

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
69 $45	EOR zp	Variable: Name und Typ der aktuellen BASIC-Variable bits #0-#6: Erstes Zeichen des Variablennamens bit #7 %00: Gleitkomma %01: String %10: FN Funktion %11: Integer		e	E	e
70 $46	LSR zp	Variable: Name und Typ der aktuellen BASIC-Variable bits #0-#6: Zweites Zeichen des Variablennamens $00 = Variablenname hat nur ein Zeichen bit #7 %00: Gleitkomma %01: String %10: FN Funktion %11: Integer		f	F	f
71 $47	[SRE] zp	Pointer: Adresse des aktuellen Variablenwerts FN-Funktion		g	G	g
72 $48	PHA imp	Pointer: Adresse des aktuellen Variablenwerts FN-Funktion		h	H	h
73 $49	EOR imm	Temp Pointer: Adresse des aktuellen Variablenwerts; Zwischenspeicher bevor Stack ($0100, 256) bei FOR/NEXT, INPUT, GET, READ, LIST, WAIT, CLOSE, LOAD, SAVE, RETURN, GOSUB		i	I	i

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
74 $4A	LSR akk	Temp Pointer: Adresse des aktuellen Variablenwerts; Zwischenspeicher bevor Stack ($0100, 256) bei FOR/NEXT, INPUT, GET, READ, LIST, WAIT, CLOSE, LOAD, SAVE, RETURN, GOSUB		j	J	j
75 $4B	[ASR] imm	Temp Pointer: Zwischenspeicher für Pointer bei GET, INPUT, READ und Arithmetik		k	K	k
76 $4C	JMP abs	Temp Pointer: Zwischenspeicher für Pointer bei GET, INPUR, READ und Arithmetik		l	L	l
77 $4D	EOR abs	Maske: Vergleichsoperationen bit #1: 1 = > bit #2: 1 = = bit #3: 1 = <		m	M	m
78 $4E	LSR abs	Pointer: Adresse aktueller FN Descriptor		n	N	n
79 $4F	[SRE] abs	Pointer: Adresse aktueller FN Descriptor		o	O	o
80 $50	BVC rel	Pointer: Adresse aktueller String Descriptor		p	P	p
81 $51	EOR iny	Pointer: Adresse aktueller String Descriptor		q	Q	q
82 $52	[JAM]	Pointer: Adresse aktueller String Descriptor $xx Länge des Strings		r	R	r
83 $53	[SRE] iny	Flag: Garbage Collection Schrittweite $03: 3 Bytes $07: 7 Bytes		s	S	s

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
84 $54	[NOP] zpx	Jump Vector: Adresse der BASIC-Funktionen Tabelle: $A052-$A07F, 41042-41087 JMP $4C, 76		t	T	t
85 $55	EOR zpx	Jump Vector: Adresse der BASIC-Funktionen Tabelle: $A052-$A07F, 41042-41087 Sprungvektor BASIC-Funktion		u	U	u
86 $56	LSR zpx	Jump Vector: Adresse der BASIC-Funktionen Tabelle: $A052-$A07F, 41042-41087 Sprungvektor BASIC-Funktion		v	V	v
87 $57	[SRE] zpx	Register: Arithmetik Akku #3 5 Bytes		w	W	w
88 $58	CLI imp	Register: Arithmetik Akku #3		x	X	x
89 $59	EOR aby	Register: Arithmetik Akku #3		y	Y	y
90 $5A	[NOP] imp	Register: Arithmetik Akku #3		z	Z	z
91 $5B	[SRE] aby	Register: Arithmetik Akku #3		[	[	[
92 $5C	[NOP] abx	Register: Arithmetik Akku #4 5 Bytes		\	£	£
93 $5D	EOR abx	Register: Arithmetik Akku #4		]	]	]
94 $5E	LSR abx	Register: Arithmetik Akku #4		^	↑	↑
95 $5F	[SRE] abx	Register: Arithmetik Akku #4		_	←	←

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
96 $60	RTS imp	Register: Arithmetik Akku #4		`	▬	▬
97 $61	ADC iny	Register: Arithmetik Gleitkomma FAC Akku #1 5 Bytes Exponent		A	♠	A
98 $62	[JAM]	Register: Arithmetik Gleitkomma FAC Akku #1 Mantisse		B	▌	B
99 $63	[RRA] inx	Register: Arithmetik Gleitkomma FAC Akku #1 Mantisse		C	▬	C
100 $64	[NOP] zp	Register: Arithmetik Gleitkomma FAC Akku #1 Mantisse		D	▬	D
101 $65	ADC zp	Register: Arithmetik Gleitkomma FAC Akku #1 Mantisse		E	▬	E
102 $66	ROR zp	Pointer: Sign Vorzeichen FAC Bit #7: 0 = Positive; 1 = Negative $00 = positiv $ff = negativ		F	▬	F
103 $67	[RRA] zp	Register: Counter für Polynomauswertung; Vorzeichenspeicher		G	▌	G
104 $68	PLA imp	Register: Arithmetik Gleitkomma Überlauf FAC Akku #1; Rundungsbyte Limit: 1,70141183 * 10^38		H	▌	H
105 $69	ADC imm	Register: Arithmetik Gleitkomma ARG Akku #2 5 Bytes Exponent		I	▝	I
106 $6A	ROR akk	Register: Arithmetik Gleitkomma ARG Akku #2 Mantisse		J	▘	J

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
107 $6B	[ARR] imm	Register: Arithmetik Gleitkomma ARG Akku #2 Mantisse		K		K
108 $6C	JMP ind	Register: Arithmetik Gleitkomma ARG Akku #2 Mantisse		L		L
109 $6D	ADC abs	Register: Arithmetik Gleitkomma ARG Akku #2 Mantisse		M		M
110 $6E	ROR abs	Pointer: Sign, Vorzeichen ARG Bit #7: 0 = Positive; 1 = Negative $00 = positiv $ff = negativ		N		N
111 $6F	[RRA] abs	Flag: Vorzeichenvergleich der Gleitkomma-Akkumulatoren #1 und #2 $00 = gleiche Vorzeichen $FF = ungleiche Vorzeichen		O		O
112 $70	BVS rel	Flag: Vorzeichenvergleich der Gleitkomma-Akkumulatoren #1 und #2; FAC Rundungsbyte Akku #1		P		P
113 $71	ADC iny	Pointer: FBUFFER FOUT; Polynomauswertung, Hilfspointer		Q		Q
114 $72	[JAM]	Pointer: FBUFFER FOUT; Polynomauswertung, Hilfspointer		R		R
115 $73	[RRA] iny	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen 24 Bytes		S		S
116 $74	[NOP] zpx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		T		T

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
117 $75	ADC zpx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		U		
118 $76	ROR zp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		V		
119 $77	[RRA] zpx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		W		
120 $78	SEI imp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		X		
121 $79	ADC aby	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen; Pointer: Aktuelles Byte im BASIC-Programm oder im Direktmodus		Y		
122 $7A	[NOP] imp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen; Programmzeiger: Anfangsadresse des nächste Befehls im BASIC-RAM		Z		
123 $7B	[RRA] aby	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen; Programmzeiger: Anfangsadresse des nächste Befehls im BASIC-RAM		{		
124 $7C	[NOP] abx	CHRGET-Routine: Nächstes Zeichen aus BAIC-Text holen		|		
125 $7D	ADC abx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		}		

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
126 $7E	ROR abx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		~		
127 $7F	[RRA] abx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen		{DEL}		
128 $80	[NOP] imm	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	END			
129 $81	STA inx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	FOR		.ORANGE	.ORANGE
130 $82	[NOP] imm	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	NEXT			
131 $83	[SAX] inx	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	DATA		LOAD+RUN	LOAD+RUN
132 $84	STY zp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	INPUT#			
133 $85	STA zp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	INPUT		F1	F1
134 $86	STX zp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	DIM		F3	F3
135 $87	[SAX] zp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	READ		F5	F5
136 $88	DEY imp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	LET		F7	F7
137 $89	[NOP] imm	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	GOTO		F2	F2

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
138 $8A	TXA imp	CHRGET-Routine: Nächstes Zeichen aus BASIC-Text holen	RUN		F4	F4
139 $8B	[XAA] imm	RND-Funktion: Wert als Gleitkommazahl, Seed (-/0/+) Good Random Hint: RND(-RND(0)) oder RND(-TI) 5 Bytes Es ist zunächst auf einen aus dem ROM kopierten Startwert eingestellt (die fünf Bytes sind 128, 79, 199, 82, 88 – $80, $4F, $C7, $52, $58).	IF		F6	F6
140 $8C	STY abs	RND-Funktion: Wert als Gleitkommazahl, Seed (-/0/+) Good Random Hint: RND(-RND(0)) oder RND(-TI)	RESTORE		F8	F8
141 $8D	STA abs	RND-Funktion: Wert als Gleitkommazahl, Seed (-/0/+) Good Random Hint: RND(-RND(0)) oder RND(-TI)	GOSUB		SHIFT RETURN	SHIFT RETURN
142 $8E	STX abs	RND-Funktion: Wert als Gleitkommazahl, Seed (-/0/+) Good Random Hint: RND(-RND(0)) oder RND(-TI)	RETURN		UPPER/GFX	UPPER/GFX
143 $8F	[SAX] abs	RND-Funktion: Wert als Gleitkommazahl, Seed (-/0/+) Good Random Hint: RND(-RND(0)) oder RND(-TI)	REM			

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
144 $90	BCC rel	Statusvariable: ST für I/O bei Kassette, Floppy, Drucker Serial bus bits: Bit #0: Transfer direction during which the timeout occured; 0 = Input; 1 = Output. Bit #1: 1 = Timeout occurred. Bit #4: 1 = VERIFY error occurred (only during VERIFY), the file read from the device did not match that in the memory. Bit #6: 1 = End of file has been reached. Bit #7: 1 = Device is not present. Datasette bits: Bit #2: 1 = Block is too short (shorter than 192 bytes). Bit #3: 1 = Block is too long (longer than 192 bytes). Bit #4: 1 = Not all bytes read with error during pass 1 could be corrected during pass 2, or a VERIFY error occurred, the file read from the device did not match that in the memory. Bit #5: 1 = Checksum error occurred. Bit #6: 1 = End of file has been reached (only during reading data files)	STOP		.BLACK	.BLACK

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
145 $91	STA iny	Flag: STOP-Taste, Keyboard-Matrix Updated every 1/60 second $7F: Stop key is pressed $FF: Sop key is not pressed	ON		CRSR UP	CRSR UP
146 $92	[JAM]	Konstante: Zeitkonstante Tape Reads; Servo Control	WAIT		RVS OFF	RVS OFF
147 $93	[AHX] iny	Flag: LOAD oder VERIFY $00: LOAD $01-$FF: VERIFY	LOAD		CLR	CLR
148 $94	STY zpx	Flag: IEC Output Cache Status, LISTEN-Zustand Bit #7: 1 = Output cache dirty, must transfer cache contents upon next output to serial bus.	SAVE		INST	INST
149 $95	STA zpx	IEC: Zeichen im Ausgabepuffer	VERIFY		.BROWN	.BROWN
150 $96	STX zpy	Flag: EOT; Cassette Block Synchronization Number; Buffer	DEF		.LIGHT RED	.LIGHT RED
151 $97	[SAX] zpy	Temp: Zwischenspeicher X, Y Register X = Tape, Y = RS232	POKE		.GREY1	.GREY1
152 $98	TYA imp	Open Files: Anzahl offener Files; Index Filetable Werte: $00-$0A, 0-10.	PRINT#		.GREY2	.GREY2
153 $99	STA aby	Nummer: Aktives Eingabegerät Default: $00, keyboard $00 = Tastatur $01 = Datasette $02 = RS232 und User-Port $03 = Bildschirm $04-$05 = Drucker $08-$011 = Laufwerke Default: $00, keyboard.	PRINT		.LIGHT GREEN	.LIGHT GREEN

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
154 $9A	TXS imp	Nummer: Aktives Ausgabegerät Default: $03, screen $00 = Tastatur $01 = Datasette $02 = RS232 und User-Port $03 = Bildschirm $04-$05 = Drucker $08-$011 = Laufwerke Default: $03, screen.	CONT		.LIGHT BLUE	.LIGHT BLUE
155 $9B	[TAS] aby	Fehlerkontrolle Parität Tape I/O Quersummenbildung	LIST		.GREY3	.GREY3
156 $9C	[SHY] abx	Flag: Tape Byte Received Quersumme des empfangenen Bytes korrekt oder nicht	CLR		.PURPLE	.PURPLE

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
157 $9D	STA abx	Flag: Kernal Message Control Bit #6: 0 = Suppress I/O error messages 1 = Display them Bit #7: 0 = Suppress system messages 1 = Display them # MELDUNG (ERROR) 1 TOO MANY FILES 2 FILE OPEN 3 FILE NOT OPEN 4 FILE NOT FOUND 5 DEVICE NOT PRESENT 6 NOT INPUT FILE 7 NOT OUTPUT FILE 8 MISSING FILE NAME 9 ILLEGAL DEVICE NUMBER 10 NEXT WITHOUT FOR 11 SYNTAX 12 RETURN WITHOUT GOSUB 13 OUT OF DATA 14 ILLEGAL QUANTITY 15 OVERFLOW 16 OUT OF MEMORY 17 UNDEF'D STATEMENT 18 BAD SUBSCRIPT 19 REDIM'D ARRAY 20 DIVISION BY ZERO 21 ILLEGAL DIRECT 22 TYPE MISMATCH 23 STRING TOO LONG 24 FILE DATA 25 FORMULA TOO COMPLEX 26 CAN'T CONTINUE 27 UNDEF'D FUNCTION 28 VERIFY 29 LOAD	CMD		CRSR LEFT	CRSR LEFT

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
158 $9E	[SHX] aby	Zwischenspeicher Tape, Temp1 Bandpass 1 Checksumme Werte: $00-$3E, 0-62	SYS		.YELLOW	.YELLOW
159 $9F	[AHX] aby	Zwischenspeicher Tape, Temp2 Bandpass 2 Fehlerkorrektur Werte: $00-$3E, 0-62	OPEN		.CYAN	.CYAN
160 $A0	LDY #	Jiffy Clock: TI, TI$, Softwareclock 1 jiffy = 0.1667 second (1/60) 24h clock, Reset to 0 after 24h Werte: $0000-$4F19FF, 0-518399 (PAL) $A0 Update INC: 65536 jiffies (18,2044 Min)	CLOSE		SHIFT SPACE	SHIFT SPACE
161 $A1	LDA imm	Jiffy Clock: TI, TI$, Softwareclock 1 jiffy = 0.1667 second (1/60) 24h clock, Reset to 0 after 24h Werte: $0000-$4F19FF, 0-518399 (PAL) $A1 Update INC: 256 jiffies (4,2267 Sek)	GET		▮	▮
162 $A2	LDX imm	Jiffy Clock: TI, TI$; Softwareclock 1 jiffy = 0.1667 second (1/60) 24h clock, Reset to 0 after 24h Werte: $0000-$4F19FF, 0-518399 (PAL) $A2 Update INC: 1 jiffy (0,1667 Sek)	NEW		▬	▬

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
163 $A3	[LAX] inx	Temp: Zwischenspeicher IEC, I/O, Buffer Bitzähler für serielle Ausgabe Bit #7: 0 = Send byte right after handshake 1 = Do EOI delay first	TAB(			
164 $A4	LDY zp	Temp: Zwischenspeicher IEC, I/O, Byte Buffer Fehlerkontrolle Parität Tape I/O	TO			
165 $A5	LDA zp	Bitzähler: Synchronbits, Kassetten-Synchronisation	FN			
166 $A6	LDX zp	Pointer: Bytezähler, Tape I/O Buffer Force output: POKE 166,191	SPC(			
167 $A7	[LAX] zp	Temp: Zwischenspeicher Tape I/O Buffer RS232 (Userport)	THEN			
168 $A8	TAY imp	Bitzähler Tape I/O Buffer RS232 (Userport, Tapeport)	NOT			
169 $A9	LDA imm	Flag: RS232 Startbit-Prüfung $00: Startbit nicht empfangen, $90: Startbit empfangen	STEP			
170 $AA	TAX imp	Temp: Zwischenspeicher Input Byte RS232/Tape I/O	+			
171 $AB	[LAX] imm	Fehlerkontrolle Input Parität RS232/Tape I/O; Zähler Tape-Header Quersummenbildung	-			
172 $AC	LDY abs	Pointer: Tape-Buffer; Scrolling; Startadresse SAVE	*			

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
173 $AD	LDA abs	Pointer: Tape-Buffer; Scrolling; Startadresse SAVE	/		▙	▙
174 $AE	LDX abs	Pointer: Endardresse SAVE; Endadresse nach LOAD/VERIFY	^		▜	▜
175 $AF	[LAX] abs	Pointer: Endardresse SAVE; Endadresse nach LOAD/VERIFY	AND		▬	▬
176 $B0	BCS rel	Konstante: Tape Timing; Tape I/O Read; Zeitkonstante einstellbar	OR		▛	▛
177 $B1	LDA iny	Konstante: Tape Timing; Tape I/O Read; Zeitkonstante einstellbar	>		▟	▟
178 $B2	[JAM]	Pointer: Start Tape Buffer Default: $033C, 828 Dieser Zeiger muss eine Adresse größer oder gleich 512 ($0200) enthalten, sonst wird ein Fehler „ILLEGAL DEVICE NUMBER“ gesendet, wenn Tape-I/O versucht wird.	=		▀	▀
179 $B3	[LAX] iny	Pointer: Start Tape Buffer Default: $033C, 828 Dieser Zeiger muss eine Adresse größer oder gleich 512 ($0200) enthalten, sonst wird ein Fehler „ILLEGAL DEVICE NUMBER“ gesendet, wenn Tape-I/O versucht wird.	<		▌	▌

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
180 $B4	LDY zpx	Temp: Zwischenspeicher für RS232/Tape I/O Bitzähler Bits #0-#6: Bit count. Bit #7: 0 = Data bit; 1 = Stop bit	SGN		▌	▌
181 $B5	LDA zpx	RS-232 Next Bit to Send/Tape EOT Flag	INT		▌	▌
182 $B6	LDX zpy	Temp: RS-232 Output Byte Buffer; Tape SYNO	ABS		▌	▌
183 $B7	[LAX] zpy	Länge des Filenamens, Disk Kommando; 1. Parameter bei LOAD, SAVE, VERIFY; 4. Parameter bei OPEN Werte: $00: No parameter $01-$FF: Parameter length	USR		▬	▬
184 $B8	CLV imp	Aktuelle logische Filenummer	FRE		▬	▬
185 $B9	LDA aby	Aktuelle File Sekundäradresse	POS		▬	▬
186 $BA	TSX imp	Aktuelle Gerätenummer 0, Tastatur 1, Datasette 2, RS232- (User-Port) Schnittstelle 3, Bildschirm 4, Drucker (normal) 5, Drucker (zusätzlich) 8, Disketten-Laufwerk Nr. 0 9, Disketten-Laufwerk Nr. 1 10, Disketten-Laufwerk Nr. 2 11, Disketten-Laufwerk Nr. 3	SQR		▟	↙

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
187 $BB	[LAS] aby	Pointer: Adresse des aktuellen Filenamen; 1. Parameter bei LOAD, SAVE, VERIFY; 4. Parameter bei OPEN	RND		■	■
188 $BC	LDY abx	Pointer: Adresse des aktuellen Filenamen; 1. Parameter bei LOAD, SAVE, VERIFY; 4. Parameter bei OPEN	LOG		■	■
189 $BD	LDA abx	RS-232 Output Parity Byte; Temp: Cassette Temporary Storage Tape I/O	EXP		▟	▟
190 $BE	LDX aby	Blockzähler für Tape I/O	COS		■	■
191 $BF	[LAX] aby	Temp: Tape Input Byte Buffer LOAD	SIN		▚	▚
192 $C0	CPY imm	Tape Motor Interlock; Motorsperre $00: No button was pressed, motor has been switched off. If a button is pressed on the datasette, must switch motor on. $01-$FF: Motor is on.	TAN		▬	▬
193 $C1	CMP inx	I/O Start Address SAVE, serial bus: I/O Start Address LOAD/VERIFY Tape; Pointer: Aktuelles Byte beim Speichertest	ATN		♠	A
194 $C2	[NOP] imm	I/O Start Address SAVE, serial bus: I/O Start Address LOAD/VERIFY Tape; Pointer: Aktuelles Byte beim Speichertest	PEEK		▌	B

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
195 $C3	[DCP] iny	Pointer: Anfang des Programms nach Tape-Header; I/O Vectors	LEN			C
196 $C4	CPY zp	Pointer: Anfang des Programms nach Tape-Header; I/O Vectors	STR$			D
197 $C5	CMP zp	Tasten-Code der zuletzt gedrückten Taste $00-$3F: Keyboard matrix code. $40: No key was pressed at the time of previous check	VAL			E
198 $C6	DEC zp	Anzahl der Zeichen im Tastaturpuffer $00, 0: Buffer is empty. $01-$0A, 1-10: Buffer length.	ASC			F
199 $C7	[DCP] zp	Flag: RVS-Modus $00: Normal mode. $12: Reverse mode	CHR$			G
200 $C8	INY imp	Pointer: Zeiger auf das Ende der eingegebenen logischen Zelle (0-79) Length of line minus 1 during screen input. Values: $27, 39; $4F, 79	LEFT$			H
201 $C9	CMP imm	Pointer: Cursor X,Y Position Cursor-Zeile Cursor row during screen input. Values: $00-$18, 0-24	RIGHT$			I
202 $CA	DEX imp	Pointer: Cursor X,Y Position Cursor-Spalte Cursor column during screen input. Values: $00-$27, 0-39	MID$			J

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
203 $CB	[AXS] imm	Tastencode der gerade gedrückten Taste $00-$3F: Keyboard matrix code. $40: No key is currently pressed	GO			K
204 $CC	CPY abs	Flag: Cursor $00: Cursor is on $01-$FF: Cursor is off				L
205 $CD	CMP abs	Zähler für Blinkfrequenz des Cursors $00, 0: Must change cursor phase $01-$14, 1-20: Delay				M
206 $CE	DEC abs	Bildschirmcode des Zeichens unter dem Cursor				N
207 $CF	[DCP] abs	Flag: Cursor Phase $00: Cursor off phase, original character visible $01: Cursor on phase, reverse character visible				O
208 $D0	BNE rel	Flag: Eingabe von Tastatur oder Bildschirm Screen = $03, or Keyboard = $00 $00: Return character reached, end of line. $01-$FF: Still reading characters from line				P
209 $D1	CMP iny	Pointer: Adresse Start der aktuellen Bildschirmzeile				Q
210 $D2	[JAM]	Pointer: Adresse Start der aktuellen Bildschirmzeile				R
211 $D3	[DCP] iny	Aktuelle physikalische Cursorspalte Werte: $00-$27, 0-39				S

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
212 $D4	[NOP] zpx	Flag für Hochkommamodus $00: Normal mode $01: Quotation mode				
213 $D5	CMP zpx	Länge der Bildschirmzeile Werte: $27, 39; $4F, 79 40/80 max positon				
214 $D6	DEC zpx	Aktuelle physikalische Cursorzeile Werte: $00-$18, 0-24				
215 $D7	[DCP] zpx	Zwischenspeicher: ASCII-Codewert letzten Taste; Bit Buffer Tape Input; Block Checksum Tape Output				
216 $D8	CLD imp	Flag: Insert Mode; Anzahl der Inserts $00: No insertions made, normal mode, control codes change screen layout or behavior $01-$FF: Number of insertions, when inputting this many character next, those must be turned into control codes, similarly to quotation mode				
217 $D9	CMP aby	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
218 $DA	[NOP] imp	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
219 $DB	[DCP] aby	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
220 $DC	[NOP] abx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
221 $DD	CMP abx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
222 $DE	DEC abx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
223 $DF	[DCP] abx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
224 $E0	CPX imm	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
225 $E1	SBC inx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
226 $E2	[NOP] imm	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			▬	▬
227 $E3	[ISC] inx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			▁	▁
228 $E4	CPX zp	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			▁	▁
229 $E5	SBC zp	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			▌	▌

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
230 $E6	INC zp	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
231 $E7	[ISC] zpx	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
232 $E8	INX imp	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
233 $E9	SBC imm	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
234 $EA	NOP imp	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
235 $EB	[SBC] imm	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
236 $EC	CPX abs	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
237 $ED	SBC abs	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
238 $EE	INC abs	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			┐	┐
239 $EF	[ISC] abs	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			▬	▬
240 $F0	BEQ rel	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			┌	┌
241 $F1	SBC iny	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen			┴	┴

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
242 $F2	[JAM]	Pointer: Screen Link-Tabelle der Bildschirm-Zeilen MSB der Bildschirmzeilenanfänge $00-$7F: Pointer high byte $80-$FF: No pointer, line is an extension of previous line on screen				
243 $F3	[ISC] iny	Pointer: Aktuelle Zeile im Farb-RAM				
244 $F4	[NOP] zpx	Pointer: Aktuelle Zeile im Farb-RAM				
245 $F5	SBC zpx	Pointer: Tastatur-Dekodiertabelle Matrixtabellen: $EB81 (60289) default uppercase/graphics characters (unshifted) $EBC2 (60354) shifted characters $EC03 (60419) Commodore logo key characters $EC78 (60536) CTRL characters				
246 $F6	INC zpx	Pointer: Tastatur-Dekodiertabelle Matrixtabellen: $EB81 (60289) default uppercase/graphics characters (unshifted) $EBC2 (60354) shifted characters $EC03 (60419) Commodore logo key characters $EC78 (60536) CTRL characters				

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
247 $F7	[ISC] zpx	Pointer: RS-232 Eingabepuffer $00-$FF: No buffer defined, a new buffer must be allocated upon RS232 output $0100-$FFFF: Buffer pointer			▬	▬
248 $F8	SED imp	Pointer: RS-232 Eingabepuffer $00-$FF: No buffer defined, a new buffer must be allocated upon RS232 output $0100-$FFFF: Buffer pointer			▬	▬
249 $F9	SBC aby	Pointer: RS-232 Ausgabepuffer $00-$FF: No buffer defined, a new buffer must be allocated upon RS232 output $0100-$FFFF: Buffer pointer			▬	▬
250 $FA	[NOP] imp	Pointer: RS-232 Ausgabepuffer $00-$FF: No buffer defined, a new buffer must be allocated upon RS232 output $0100-$FFFF: Buffer pointer			▟	↙
251 $FB	[ISC] aby	Freier Zero Page User Space 4 Bytes BASIC ändert die 4 Bytes nicht!			▪	▪
252 $FC	[NOP] abx	Freier Zero Page User Space			▪	▪

Code dez hex	CPU 6510	Zeropage	BASIC	ASCII	Font up	Font lo
253 $FD	SBC abx	Freier Zero Page User Space			▟	▟
254 $FE	INC abx	Freier Zero Page User Space			■	■
255 $FF	[ISC] abx	BASIC Zwischenspeicher			▀	▚

CBM DOS: Floppy Fehlermeldungen

geschrieben von Andreas Potthoff | 10. Juni 2023
Die Fehlermeldungen der CBM Laufwerke für:

- **CBM 1540, CBM 1541, CBM 1541-II, CBM 1541 C, CBM 1551, CBM 1570, CBM 1571, CBM 1581**
- **CBM 2020, CBM 2031, CBM 2040, CBM 3040, CBM 4031, CBM 4040**
- **CBM 8050, CBM 8060, CBM 8061, CBM 8062, CBM 8250, CBM 8280**
- **Enhancer 2000, FD-200, OC-118N, SFD-1001**

Fehlererkennung

Am Blinken der Kontroll-LED (z.B. bei einem 1541-Laufwerk) kann man erkennen, dass bei einer Diskettenoperation eine Fehlersituation aufgetreten ist. Das CBM-DOS gibt eine entsprechende Statusmeldung über den Fehlerkanal aus. Beim BASIC werden Fehlermeldungen direkt auf dem Bildschirm

ausgegeben. Leider ist die Anzeige von Laufwerksfehlern nur über das Auslesen des Fehlerkanals möglich.

Auslesen des Fehlerkanals

BASIC-Programm

Da der INPUT#-Befehl nur innerhalb eines Programms funktioniert ist ein kleines Programm zum Auslesen des Fehlerkanals nötig.

```
10 OPEN 1,8,15
20 INPUT# 1,A,B$,C,D
30 PRINT A,B$,C,D
40 CLOSE 1
```

Legende:

A : Nummer des Fehlers
B$: Fehlerbezeichnung im Klartext
C : Track (Spur)
D : Sektor (Block)

Direktmodus

Man kann den Fehlerkanal auch ohne ein BASIC-Programm im Direktmodus auslesen.

Option A

```
OPEN 1,8,15:FOR I=0 TO 2:POKE 58,1:GET#1,F$:PRINT F$;:I=255 AND ST:NEXT:CLOSE 1
```

Option B

```
OPEN 1,8,15:FOR I=1 TO 40:POKE 781,1:SYS 65478:SYS 65487:SYS 65490:SYS 65484:IF ST=0 THEN NEXT:CLOSE 1
```

Tabelle CBM Floppy-Fehlermeldungen (deutsch)

Fehlermeldung	Beschreibung
00,OK,00,00	OK, es liegt kein Fehler vor.
01,FILES SCRATCHED,XX,00	Kein Fehler. Rückmeldung nach SCRATCH-Befehl, wobei XX die Anzahl der gelöschten Dateien darstellt.
20,READ ERROR,TT,SS	LESE-FEHLER: Blockheader nicht gefunden - Der Festplattencontroller kann den Header des angeforderten Datenblocks nicht finden. Verursacht durch eine unzulässige Blocknummer oder der Header wurde zerstört.
21,READ ERROR,TT,SS	LESE-FEHLER: Keine SYNC-Markierung - Der Laufwerkscontroller kann keine Sync-Markierung auf dem gewünschten Track erkennen. Verursacht durch Fehlausrichtung des Lese-/Schreibkopfes, keine Diskette vorhanden oder unformatiert oder falsch sitzende Diskette. Kann auch auf einen Hardwarefehler hinweisen.
22,READ ERROR,TT,SS	LESE-FEHLER: Datenblock nicht vorhanden - Der Laufwerkscontroller wurde aufgefordert, einen Datenblock zu lesen oder zu überprüfen der nicht ordnungsgemäß geschrieben wurde. Diese Fehlermeldung tritt in Verbindung mit den BLOCK-Befehlen auf und zeigt eine unzulässige Spur- und/oder Blockanforderung an.

Fehlermeldung	Beschreibung
23,READ ERROR,TT,SS	LESE-FEHLER: Prüfsummenfehler im Datenblock - Diese Fehlermeldung zeigt an, dass in einem oder mehreren der Datenbytes ein Fehler vorliegt. Die Daten wurden in den DOS-Speicher eingelesen, aber die Prüfsumme der Daten ist falsch. Diese Meldung kann auch auf Erdungsprobleme hinweisen.
24,READ ERROR,TT,SS	LESE-FEHLER: Byte-Dekodierungsfehler - Die Daten oder der Header wurden in den DOS-Speicher gelesen, aber es wurde ein Hardwarefehler wegen einen ungültigen Bitmuster im Datenbyte erzeugt. Diese Meldung kann auch auf Erdungsprobleme hinweisen.
25,WRITE ERROR,TT,SS	SCHREIB-FEHLER: Schreib-Verifizierungsfehler - Diese Nachricht wird generiert, wenn der Laufwerkscontroller eine Diskrepanz zwischen den geschriebenen Daten und den Daten im DOS-Speicher erkennt.
26,WRITE PROTECT ON,TT,SS	SCHREIBSCHUTZ EIN - Diese Meldung wird generiert, wenn der Laufwerkscontroller aufgefordert wurde einen Datenblock zu Schreiben während der Schreibschutz aktiviert ist. Normalerweise wird dies durch die Verwendung einer Diskette verursacht die mit einer Schreibschutzlasche über der Kerbe versehen ist.
27,READ ERROR,TT,SS	LESE-FEHLER: Prüfsummenfehler im Header - Der Laufwerkscontroller hat einen Fehler im Header des angeforderten Datenblocks festgestellt. Der Block wurde nicht in den DOS-Speicher eingelesen. Diese Meldung kann auch auf Erdungsprobleme hinweisen.

Fehlermeldung	Beschreibung
28,WRITE ERROR,TT,SS	SCHREIB-FEHLER: Langer Datenblock - Der Laufwerkscontroller versucht, die SYNC-Markierung des nächsten Headers nach dem Schreiben eines Datenblocks zu lesen. Wenn die SYNC-Markierung nicht innerhalb einer vorgegebenen Zeit erscheint, wird diese Fehlermeldung generiert. Der Fehler wird durch ein falsches Diskettenformat verursacht (die Daten erstrecken sich in den nächsten Block) oder durch Hardwarefehler.
29,DISK ID MISMATCH,TT,SS	DISK ID NICHTÜBEREINSTIMMUNG: - Diese Nachricht wird generiert, wenn der Laufwerkscontroller aufgefordert wurde, auf eine nicht initialisierte Diskette zuzugreifen. Die Meldung kann auch auftreten, wenn eine Diskette einen fehlerhaften Header hat.
30,SYNTAX ERROR,00,00	SYNTAX-FEHLER: Allgemeine Syntax - Das DOS kann den Befehl der an den Befehlskanal gesendet wurde nicht interpretieren. Typischerweise wird dies durch eine unzulässige Anzahl von Dateinamen oder Mustern die illegal verwendet werden ausgelöst. Beispiel: Beim COPY-Befehl werden zwei Dateinamen auf der linken Seite verwendet.
31,SYNTAX ERROR,00,00	SYNTAX-FEHLER: Ungültiger Befehl - Das DOS erkennt den Befehl nicht. Der Befehl muss an erster Stelle stehen.
32,SYNTAX ERROR,00,00	SYNTAX-FEHLER: Ungültiger Befehl - Der gesendete Befehl ist länger als 40 Zeichen.
33,SYNTAX ERROR,00,00	SYNTAX-FEHLER: Ungültiger Dateiname - Musterabgleich der beim Befehl OPEN-oder SAVE-Befehl verwendet wird ist ungültig.

Fehlermeldung	Beschreibung
34,SYNTAX ERROR,00,00	SYNTAX-FEHLER: Keine Datei angegeben - Der Dateiname wurde bei einem Befehl weggelassen oder das DOS erkennt es nicht als solches. Typischerweise wurde ein Doppelpunkt (:) bei dem Befehl weggelassen.
39,SYNTAX ERROR,00,00	SYNTAX-FEHLER: - Ungültiger Befehl - Dieser Fehler kann auftreten, wenn der Befehl an den Befehlskanal (Sekundäradresse 15) vom DOS nicht erkannt wird.
50,RECORD NOT PRESENT,00,00	EINTRAG NICHT VORHANDEN: - Ergebnis des Lesens der Diskette über den letzten Datensatz hinaus durch INPUT#- oder GET#-Befehle. Diese Meldung wird auch dann erscheinen wenn die Positionierung auf einen Datensatz bei einer relativen Datei über das Dateiende hinausgeht. Wenn die Absicht darin besteht, die Datei zu erweitern, indem der neue Datensatz hinzugefügt wird (mit einem PRINT# Befehl) kann die Fehlermeldung ignoriert werden. INPUT oder GET sollten nicht versucht werden, nachdem dieser Fehler ohne vorherige Neupositionierung erkannt wurde.
51,OVERFLOW IN RECORD,00,00	ÜBERLAUF IM EINTRAG: - Die PRINT#-Anweisung überschreitet die Datensatzgrenze. Informationen sind abgeschnitten. Da der Wagenrücklauf (RETURN) als Datensatz gesendet wird und der Eintragsabschluss in der Datensatzgröße mitgezählt wird. Diese Meldung wird angezeigt, wenn die Gesamtzahl der Zeichen im Datensatz (einschließlich des letzten RETUNRs) die definierte Größe überschreitet.

Fehlermeldung	Beschreibung
52,FILE TOO LARGE,00,00	DATEI ZU GROSS: - Die Aufnahmeposition innerhalb einer relativen Datei zeigt an, dass ein Disküberlauf zur Folge hat.
60,WRITE FILE OPEN,00,00	SCHREIBDATEI OFFEN: - Diese Nachricht wird generiert, wenn eine Schreibdatei die nicht geschlossen wurde, zum Lesen geöffnet wird.
61,FILE NOT OPEN,00,00	DATEI NICHT GEÖFFNET: - Diese Meldung wird generiert, wenn auf eine Datei ein Zugriff erfolgt, die nicht im DOS geöffnet wurde. Manchmal, in diesem Fall, wird eine Nachricht nicht generiert; die Anfrage wird einfach ignoriert.
62,FILE NOT FOUND,00,00	DATEI NICHT GEFUNDEN: - Die angeforderte Datei existiert nicht auf dem angegebenen Laufwerk.
63,FILE EXISTS,00,00	DATEI EXISTIERT: - Der Dateiname der Datei, die erstellt erstellt werden soll, ist auf der Diskette bereits vorhanden.
64,FILE TYPE MISMATCH,00,00	DATEITYP NICHTÜBEREINSTIMMUNG: - Der Dateityp stimmt nicht mit dem Dateityp im Verzeichniseintrag für die angeforderte Datei überein.
65,NO BLOCK,TT,SS	KEIN BLOCK: - Diese Meldung tritt in Verbindung mit dem B-A Befehl auf. Es zeigt an, dass der zuzuweisende Block zuvor vergeben wurde. Die Parameter geben die Spur und den Sektor an mit der nächsthöheren Nummer der verfügbar ist. Wenn die Parameter null sind (0), dann werden alle Blöcke mit höherer Nummer verwendet.

Fehlermeldung	Beschreibung
66,ILLEGAL TRACK OR SECTOR,TT,SS	ILLEGALER TRACK ODER SEKTOR: - Das DOS hat versucht, auf eine Spur oder Block zuzugreifen, der im verwendeten Format nicht vorhanden ist. Dies kann auf ein Problem beim Lesen des Zeigers auf den nächsten Block hinweisen.
67,ILLEGAL TRACK OR SECTOR,TT,SS	ILLEGALER SYSTEM T ODER S: - Diese spezielle Fehlermeldung weist auf einen illegalen Track (Spur) oder Sector (Block) hin.
70,NO CHANNEL,00,00	KEIN KANAL: - Der angeforderte Kanal ist nicht verfügbar, oder alle Kanäle sind belegt. Es dürfen maximal fünf sequentielle Dateien auf einmal für das DOS geöffnet sein. Direktzugriffskanäle können sechs geöffnete Dateien haben.
71,DIR ERROR,TT,SS	VERZEICHNIS-FEHLER: - Die BAM stimmt nicht mit dem internen Zähler überein. Es liegt ein Problem bei der BAM-Zuordnung vor oder die BAM wurde im DOS-Speicher überschrieben. Um dieses Problem zu beheben, initialisieren Sie die Diskette, um die BAM im Speicher wiederherzustellen. Einige aktive Dateien werden möglicherweise durch die Korrekturmaßnahme beendet. HINWEIS: BAM = Blockverfügbarkeit Karte
72,DISK FULL,00,00	DISKETTE VOLL: - Entweder werden die Blöcke auf der Diskette alle verwendet oder das Verzeichnis ist an der Eintragsgrenze angekommen. DISK FULL wird gesendet, wenn nur noch zwei Blöcke auf dem 1541 Laufwerk verfügbar sind, damit die aktuelle Datei geschlossen werden kann.

Fehlermeldung	Beschreibung
73,CBM DOS V2.6 1541,00,00	DOS MISMATCH (73, CBM DOS V2.6 1541): - DOS 1 und DOS 2 werden kompatibel gelesen, sind aber nicht schreibkompatibel. Disketten können zwischen den DOS-Versionen austauschbar sein und mit DOS gelesen werden, aber eine Diskette, die auf einer Version formatiert ist, kann nicht mit der anderen Version geschrieben, weil das Format anders ist. Dieser Fehler wird immer angezeigt, wenn versucht wird, auf einen Datenträger zu schreiben, der in einem nicht kompatiblen Format formatiert wurde. (Ein Dienstprogramm ist verfügbar, um bei der Konvertierung von einem Format in das andere zu helfen.) Diese Meldung kann auch nach dem Einschalten des Laufwerks erscheinen.
74,DRIVE NOT READY,00,00	LAUFWERK NICHT BEREIT: - Es wurde versucht, auf ein 1541 Laufwerk zuzugreifen ohne Diskette in einem Laufwerk.

Tabelle CBM Floppy-Fehlermeldungen (englisch)

CBM FLOPPY ERROR MESSAGES

```
 0         OK, no error exists.
 1         Files scratched response. Not an error condition.
 2-19      Unused error messages: should be ignored.
 20        Block header not found on disk.
 21        Sync character not found.
 22        Data block not present.
 23        Checksum error in data.
 24        Byte decoding error.
 25        Write-verify error.
 26        Attempt to write with write protect on.
```

```
 27        Checksum error in header.
 28        Data extends into next block.
 29        Disk id mismatch.
 30        General syntax error
 31        Invalid command.
 32        Long line.
 33        Invalid filename.
 34        No file given.
 39        Command file not found.
 50        Record not present.
 51        Overflow in record.
 52        File too large.
 60        File open for write.
 61        File not open.
 62        File not found.
 63        File exists.
 64        File type mismatch.
 65        No block.
 66        Illegal track or sector.
 67        Illegal system track or sector.
 70        No channels available.
 71        Directory error.
 72        Disk full or directory full.
 73          Power up message, or write attempt with DOS
Mismatch.
 74        Drive not ready.
```

DESCRIPTION OF DOS ERROR MESSAGES

NOTE: Error message numbers less than 20 should be ignored with the exception of 01 which gives information about the number of files scratched with the SCRATCH command.

20: READ ERROR (block header not found) -- The disk controller is unable to locate the header of the requested data block. Caused by an illegal block number, or the header has been destroyed.

21: READ ERROR (no sync character) -- The disk controller is unable to detect a sync mark on the desired track. Caused by misalignment of the read/writer head, no diskette is present,

or unformatted or improperly seated diskette. Can also indicate a hardware failure.

22: READ ERROR (data block not present) -- The disk controller has been requested to read or verify a data block that was not properly written. This error message occurs in conjunction with the BLOCK commands and indicates an illegal track and/or block request.

23: READ ERROR (checksum error in data block) -- This error message indicates that there is an error in one or more of the data bytes. The data has been read into the DOS memory, but the checksum over the data is in error. This message may also indicate grounding problems.

24: READ ERROR (byte decoding error) -- The data or header as been read into the DOS memory, but a hardware error has been created due to an invalid bit pattern in the data byte. This message may also indicate grounding problems.

25: WRITE ERROR (write-verify error) -- This message is generated if the controller detects a mismatch between the written data and the data in the DOS memory.

26: WRITE PROTECT ON -- This message is generated when the controller has been requested to write a data block while the write protect switch is depressed. Typically, this is caused by using a diskette with a write a protect tab over the notch.

27: READ ERROR (checksum error in header) -- The controller has detected an error in the header of the requested data block. The block has not been read into the DOS memory. This message may also indicate grounding problems.

28: WRITE ERROR (long data block) -- The controller attempts to detect the sync mark of the next header after writing a data block. If the sync mark does not appear within a predetermined time, the error message is generated. The error is caused by a bad diskette format (the data extends into the next block), or by hardware failure.

29: DISK ID MISMATCH -- This message is generated when the controller has been requested to access a diskette which has not been initialized. The message can also occur if a diskette has a bad header.

30: SYNTAX ERROR (general syntax) -- The DOS cannot interpret the command sent to the command channel. Typically, this is caused by an illegal number of file names, or pattems are illegally used. For example, two file names may appear on the left side of the COPY command.

31: SYNTAX ERROR (invalid command) -- The DOS does not recognize the command. The command must start in the first position.

32: SYNTAX ERROR (invalid command) -- The command sent is longer than 58 characters.

33: SYNTAX ERROR (invalid file name) -- Pattem matching is invalidly used in the OPEN or SAVE command.

34: SYNTAX ERROR (no file given) -- the file name was left out of a command or the DOS does not recognize it as such. Typically, a colon (:) has been left out of the command,

39: SYNTAX ERROR (invalid command) -- This error may result if the command sent to command channel (secondary address 15) is unrecognized by the DOS.

50: RECORD NOT PRESENT -- Result of disk reading past the last record through INPUT#, or GET# commands. This message will also occur after positioning to a record beyond end of file in a relative file. If the
intent is to expand the file by adding the new record (with a PRINT# command), the error message may be ignored. INPUT or GET should not be attempted after this error is detected without first repositioning.

51: OVERFLOW IN RECORD -- PRINT# statement exceeds record boundary. Information is cut off. Since the carriage return is sent as a record terminator is counted in the record size.

This message will occur if the total characters in the record (including the final carriage return) exceeds the defined size.

52: FILE TOO LARGE -- Record position within a relative file indicates that disk overflow will result.

60: WRITE FILE OPEN -- This message is generated when a write file that has not been closed is being opened for reading.

61: FILE NOT OPEN -- This message is generated when a file is being accessed that has not been opened in the DOS. Sometimes, in this case, a message is not generated; the request is simply ignored.

62: FILE NOT FOUND -- The requested file does not exist on the indicated drive.

63: FILE EXISTS -- The file name of the file being created already exists on the diskette.

64: FILE TYPE MISMATCH -- The file type does not match the file type in the directory entry for the requested file.

65: NO BLOCK -- This message occurs in conjunction with the B-A command. It indicates that the block to be allocated has been previously allocated. The parameters indicate the track and sector available with the next highest number. If the parameters are zero (0), then all blocks higher in number are in use.

66: ILLEGAL TRACK AND SECTOR -- The DOS has attempted to access a track or block which does not exist in the format being used. This may indicate a problem reading the pointer to the next block.

67: ILLEGAL SYSTEM T OR S -- This special error message indicates an illegal system track or block.

70: NO CHANNEL (available) -- The requested channel is not available, or all channels are in use. A maximum of five

sequential files may be opened at one time to the DOS. Direct access channels may have six opened files.

71: DIRECTORY ERROR -- The BAM does not match the internal count. There is a problem in the BAM allocation or the BAM has been overwritten in DOS memory. To correct this problem, reinitialize the diskette to restore the BAM in memory. Some active files may be terminated by the corrective action. NOTE: BAM = Block Availability Map

72: DISK FULL -- Either the blocks on the diskette are used or the directory is at its entry limit. DISK FULL is sent when two blocks are available on the 1541 to allow the current file to be closed.

73: DOS MISMATCH (73, CBM DOS V2.6 1541) -- DOS 1 and 2 are read compatible but not write compatible. Disks may be interchangeably read with either DOS, but a disk formatted on one version cannot be written upon with the other version because the format is different. This error is displayed whenever an attempt is made to write upon a disk which has been formatted in a non-compatible format. (A utility routine is available to assist in converting from one format to another.) This message may also appear after power up.

74: DRIVE NOT READY -- An attempt has been made to access the 1541 Single Drive Floppy Disk without any diskettes present in either drive.

C64 BASIC V2.0: Befehle, Schlüsselwörter und Token

geschrieben von Andreas Potthoff | 10. Juni 2023
Beim Commodore 64 (auch VC-20) werden bei der internen Verarbeitung der **BASIC-Befehle (Schlüsselwörter)** eines BASIC-

Programms *für eine schnellere Verarbeitung und Speicherplatzersparnis* sogenannte **Token (Schlüsselzeichen)** statt der Schlüsselwörter verwendet.

Die folgende Tabelle zeigt die BASIC-Schlüsselwörter, den dazugehörigen Token (dezimal / hexadezimal), die ROM-Einsprungadresse (hexadezimal), die Abkürzung und den Typ des Befehls.

Nach der Tabelle erhalten Sie weiterführende Informationen zur Tokenisation der Befehle.

Tabelle C64 Token

BASIC Schlüsselwort	Token dez	Token hex	ROM Exec dez	ROM Exec hex	Abk.	Typ
END	128	$80	43057	$A831	eN	Anweisung/Befehl
FOR	129	$81	42818	$A742	fO	Anweisung/Befehl
NEXT	130	$82	44318	$AD1E	nE	Anweisung/Befehl
DATA	131	$83	43256	$A8F8	dA	Anweisung/Befehl
INPUT#	132	$84	43941	$ABA5	iN	Anweisung/Befehl
INPUT	133	$85	43967	$ABBF	-	Anweisung/Befehl
DIM	134	$86	45185	$B081	dI	Anweisung/Befehl
READ	135	$87	44038	$AC06	rE	Anweisung/Befehl
LET	136	$88	43429	$A9A5	lE	Anweisung/Befehl
GOTO	137	$89	43168	$A8A0	gO	Anweisung/Befehl
RUN	138	$8A	43121	$A871	rU	Anweisung/Befehl
IF	139	$8B	43304	$A928	-	Anweisung/Befehl
RESTORE	140	$8C	43037	$A81D	reS	Anweisung/Befehl
GOSUB	141	$8D	43139	$A883	goS	Anweisung/Befehl
RETURN	142	$8E	43218	$A8D2	reT	Anweisung/Befehl
REM	143	$8F	43323	$A93B	-	Anweisung/Befehl

BASIC Schlüsselwort	Token dez	Token hex	ROM Exec dez	ROM Exec hex	Abk.	Typ
STOP	144	$90	43055	$A82F	sT	Anweisung/Befehl
ON	145	$91	43339	$A94B	-	Anweisung/Befehl
WAIT	146	$92	47149	$B82D	wA	Anweisung/Befehl
LOAD	147	$93	57704	$E168	lO	Anweisung/Befehl
SAVE	148	$94	57686	$E156	sA	Anweisung/Befehl
VERIFY	149	$95	57701	$E165	vE	Anweisung/Befehl
DEF	150	$96	46003	$B3B3	dE	Anweisung/Befehl
POKE	151	$97	47140	$B824	pO	Anweisung/Befehl
PRINT#	152	$98	43648	$AA80	pR	Anweisung/Befehl
PRINT	153	$99	43680	$AAA0	?	Anweisung/Befehl
CONT	154	$9A	43095	$A857	cO	Anweisung/Befehl
LIST	155	$9B	42652	$A69C	lI	Anweisung/Befehl
CLR	156	$9C	42590	$A65E	cL	Anweisung/Befehl
CMD	157	$9D	43654	$AA86	cM	Anweisung/Befehl
SYS	158	$9E	57642	$E12A	sY	Anweisung/Befehl
OPEN	159	$9F	57790	$E1BE	oP	Anweisung/Befehl
CLOSE	160	$A0	57799	$E1C7	clO	Anweisung/Befehl
GET	161	$A1	43899	$AB7B	gE	Anweisung/Befehl
NEW	162	$A2	42562	$A642	-	Anweisung/Befehl
TAB(	163	$A3	43752	$AAE8	tA	Anweisung/Befehl, Spezial
TO	164	$A4	42861	$A76D	-	Anweisung/Befehl, Spezial
FN	165	$A5	46068	$B3F4	-	Anweisung/Befehl, Spezial
SPC(	166	$A6	43769	$AAF9	sP	Anweisung/Befehl, Spezial

BASIC Schlüsselwort	Token dez	Token hex	ROM Exec dez	ROM Exec hex	Abk.	Typ
THEN	167	$A7	43314	$A932	tH	Anweisung/Befehl, Spezial
NOT	168	$A8	44756	$AED4	nO	Anweisung/Befehl, Spezial
STEP	169	$A9	42905	$A799	stE	Anweisung/Befehl, Spezial
+	170	$AA	47210	$B86A	-	Operator, numerisch/string
-	171	$AB	47187	$B853	-	Operator, numerisch
*	172	$AC	47659	$BA2B	-	Operator, numerisch
/	173	$AD	47890	$BB12	-	Operator, numerisch
^	174	$AE	49019	$BF7B	-	Operator, numerisch
AND	175	$AF	45033	$AFE9	aN	Operator, logisch
OR	176	$B0	45030	$AFE6	-	Operator, logisch
>	177	$B1	49076	$BFB4	-	Operator, logisch
=	178	$B2	44756	$AED4	-	Operator, logisch
<	179	$B3	45078	$B016	-	Operator, logisch
SGN	180	$B4	48185	$BC39	sG	Funktion, numerisch
INT	181	$B5	48332	$BCCC	-	Funktion, numerisch
ABS	182	$B6	48216	$BC58	abS	Funktion, numerisch
USR	183	$B7	784	$0310	uS	Funktion, numerisch/string

BASIC Schlüsselwort	Token dez	Token hex	ROM Exec dez	ROM Exec hex	Abk.	Typ
FRE	184	$B8	45949	$B37D	fR	Funktion, numerisch, Spezial
POS	185	$B9	45982	$B39E	-	Funktion, numerisch, Spezial
SQR	186	$BA	49009	$BF71	sQ	Funktion, numerisch
RND	187	$BB	57495	$E097	rN	Funktion, numerisch
LOG	188	$BC	45794	$B9EA	-	Funktion, numerisch
EXP	189	$BD	49133	$BFED	eX	Funktion, numerisch
COS	190	$BE	57956	$E264	-	Funktion, numerisch
SIN	191	$BF	57963	$E26B	sI	Funktion, numerisch
TAN	192	$C0	58036	$E2B4	-	Funktion, numerisch
ATN	193	$C1	58128	$E30E	aT	Funktion, numerisch
PEEK	194	$C2	47117	$B80E	pE	Funktion, numerisch
LEN	195	$C3	46972	$B77C	-	Funktion, numerisch
STR$	196	$C4	46181	$B465	stR	Funktion, string
VAL	197	$C5	47021	$B7AD	vA	Funktion, numerisch

BASIC Schlüsselwort	Token dez	Token hex	ROM Exec dez	ROM Exec hex	Abk.	Typ
ASC	198	$C6	46987	$B78B	aS	Funktion, numerisch
CHR$	199	$C7	46828	$B6EC	cH	Funktion, string
LEFT$	200	$C8	46848	$B700	leF	Funktion, string
RIGHT$	201	$C9	46892	$B72C	rI	Funktion, string
MID$	202	$CA	46903	$B737	mI	Funktion, string
GO	203	$CB	43026	$A812	-	Anweisung/Befehl, Spezial
π	255	$FF	44702	$AE9E	-	Funktion, numerisch, Konstante
ST (STATUS)	-	-	65463	$FFB7	-	Systemvariable
TI (TIME)	-	-	?	?	-	Systemvariable
TI$ (TIME$)	-	-	43488	$A9E0	-	Systemvariable

Tokenisation

Hier werden die Schlüsselwörter (Befehle) in ein Single-Byte-Wert (Token) umgewandelt, wenn sie in einem Programm gespeichert sind. Das geschieht entweder durch einen Programmstart mit RUN oder im Direktmodus (Konsole) durch das Drücken der Taste RETURN, wenn dort Kommandos eingegeben worden sind. Der BASIC-Interpreter arbeitet die Token der Reihenfolge nach ab.

De-Tokenisation

Hier werden die Token (Bytes) in lesbare BASIC-Befehle

(Schlüsselwörter) umgewandelt, was eigentlich *nur für das LIST-Kommando* als Ausgabe und für gute menschliche Lesbarkeit zutrifft. Ansonsten wird die De-Tokenisation nicht angewandt.

Single-Byte-Token

Ein Token kann beim Commodore BASIC V2 (ab 4.0) einen Single-Byte-Wert zwischen 128-255 ($80-$FF) haben und belegt nur 1 Byte Arbeitsspeicher. Token-Codes sind immer größer oder gleich 128 ($80); d.h. das höchstwertige Bit in einem Byte, das einen Token repräsentiert, ist immer gesetzt und somit werden keine PETSCII-Zeichen (<128/<$80) als Token-Code benutzt.

Two-Bytes-Token

Ab dem Commodore BASIC 7.0 (C 128) werden wegen des umfangreichen BASIC-Befehlssatzes zwei Bytes für ein Token benötigt.

Ausführen von Token

Jedes Token hat eine sogenannte Ausführungsadresse (EXEC) im ROM, wo dann der entsprechende Code für den jeweiligen Token bzw. BASIC-Befehl ausgeführt wird.

Schlüsselwörter

BASIC V2 enthält 76 Schlüsselwörter, 8 Operatoren, 1 Konstante und 3 Systemvariablen, die in verschiedenen Gruppen gegliedert sind:

- 128-162 ($80-$A2): *Befehle*
- 163-169 ($A3-$A9): “*Bywords*“, die Teil der Syntax der vorherigen Befehle sind
- 170-179 ($AA-$B3): *Arithmetische und logische Operatoren*
- 180-202 ($B4-$CA): *Funktionen*
- 203 ($CB): *Befehl* – GO (der hier als Ausnahme hinter den

Funktionen liegt)
- 204-254 ($CC-$FE): Ein freier Bereich für 51 zusätzliche Token, z.B. der für BASIC-Erweiterungen von Drittanbietern genutzt wird
- 255 ($FF): *Konstante* – Pi
- *Systemvariablen*: ST (STATUS), TI (TIME), TI$ (TIME$). Im BASIC V2 ROM werden die Systemvariablen als Ausnahmen in den Routinen zur Behandlung *normaler* Variablen behandelt.

Abkürzungen

Die meisten BASIC-Schlüsselwörter kann man bei der Eingabe abkürzen. Abgekürzte Schlüsselwörter werden i.d.R. gebildet, indem man die ersten (manchmal bis zu drei) Zeichen eintippt und das nächste Zeichen mit SHIFT eingibt.

Auch hier gibt es wieder einige Ausnahmen. Einige Schlüsselwörter (CLOSE, GOSUB, LEFT$, RESTORE, RETURN, STEP, STR$) benötigen gekürzt 3 statt 2 Zeichen. Der BASIC-Befehl PRINT wird nur mit einem Zeichen, dem ? abgekürzt. Es gibt auch einige BASIC-Befehle (INPUT, COS, FN, TO, IF, INT, LEN, LOG, NEW, ON, OR, POS, REM, TAN) die nicht abkürzbar sind.

Speichern

Es kann schon mal vorkommen, dass *eine Programmzeile* 80 Zeichen überschreitet, also dafür mehr als 2 Zeilen auf dem Bildschirm angezeigt werden. Dies geschieht dadurch, dass die Ausgabe der Token beim LIST-Befehl eben ungekürzt passiert und somit Programmzeilen mit mehr als 80 Zeichen auftreten können.

Wenn sie eine solche Programmzeile ändern wollen, müssen sie die Abkürzungen erneut eingeben bevor sie das Programm speichern. Achten sie dann darauf, dass sie nicht mehr als 80 Zeichen für eine Programmzeile insgesamt verwenden. Alle zusätzlichen Zeichen danach werden nach dem Drücken von RETURN

automatisch abgeschnitten. Beim Speichern eines Programms auf einen Datenträger werden die Token und nicht die Schlüsselwörter benutzt.

Befehle	CLOSE, CLR, CMD, CONT, DATA, DEF, DIM, END, FOR, GET, GET#, GOSUB, GOTO, IF, INPUT, INPUT#, LET, LIST, LOAD, NEW, NEXT, ON, OPEN, POKE, PRINT, PRINT#, READ, REM, RESTORE, RETURN, RUN, SAVE, STOP, SYS, VERIFY, WAIT
Befehle Spezial (Bywords)	FN, GO, NOT, SPC(, TAB(, THEN, TO, STEP
Arithmetische und logische Operatoren	+, –, *, /, ^, >, =, <, AND, OR
Funktionen	ABS, ASC, ATN, CHR$, COS, EXP, FRE, INT, LEFT$, LEN, LOG, MID$, PEEK, POS, RIGHT$, RND, SGN, SIN, SQR, STR$, TAN, USR, VAL
Konstanten und Systemvariablen	Pi, ST, TI, TI$

C64 BASIC V2.0: Ableitung mathematischer Funktionen

geschrieben von Andreas Potthoff | 10. Juni 2023

Mathematische Funktionen, die nicht Commodore 64 BASIC eigen sind, können wie folgt berechnet werden:

Mathematische Funktion	BASIC Äquivalent
SECANT	SEC(X) = 1/COS(X)
COSECANT	CSC(X) = 1/SIN(X)
COTANGENT	COT(X) = 1/TAN(X)
INVERSE SINE	ARCSIN(X) = ATN(X/SQR(-X*X+1))
INVERSE COSINE	ARCCOS(X) = -ATN(X/SQR(-X*X+1))+{pi}/2
INVERSE SECANT	ARCSEC(X) = ATN(X/SQR(X*X-1))
INVERSE COSECANT	ARCCSC(X) = ATN(X/SQR(X*X-1))+(SGN(X)-1*{pi}/2
INVERSE COTANGENT	ARCOT(X) = ATN(X)+{pi}/2
HYPERBOLIC SINE	SINH(X) = (EXP(X)-EXP(-X))/2
HYPERBOLIC COSINE	COSH(X) = (EXP(X)+EXP(-X))/2
HYPERBOLIC TANGENT	TANH(X) = EXP(-X)/(EXP(X)+EXP(-X))*2+1
HYPERBOLIC SECANT	SECH(X) = 2/(EXP(X)+EXP(-X))
HYPERBOLIC COSECANT	CSCH(X) = 2/(EXP(X)-EXP(-X))
HYPERBOLIC COTANGENT	COTH(X) = EXP(-X)/(EXP(X)-EXP(-X))*2+1
INVERSE HYPERBOLIC SINE	ARCSINH(X) = LOG(X+SQR(X*X+1))
INVERSE HYPERBOLIC COSINE	ARCCOSH(X) = LOG(X+SQR(X*X-1))
INVERSE HYPERBOLIC TANGENT	ARCTANH(X) = LOG((1+X)/(1-X))/2
INVERSE HYPERBOLIC SECANT	ARCSECH(X) = LOG((SQR(-X*X+1)+1/X)
INVERSE HYPERBOLIC COSECANT	ARCCSCH(X) = LOG((SGN(X)*SQR(X*X+1/X)
INVERSE HYPERBOLIC COTANGENT	ARCCOTH(X) = LOG((X+1)/(X-1))/2

C64 BASIC V2: Video: 10 selten benutzte BASIC-Features

geschrieben von Andreas Potthoff | 10. Juni 2023
In diesem Video (00:31:19, engl.) erklärt Robin Harbron einige Features von CBM BASIC v2.0 die eigentlich selten benutzt werden. Robin zeigt Tipps und Beispiele zu jedem BASIC-Feature live am C-64.

Folgende Themen werden dabei angesprochen:

1. DEF und FN
2. ON GOSUB / ON GOTO
3. Wissenschaftliche Notation
4. INPUT Möglichkeiten
5. LIST Parameter
6. STOP und END
7. CONT
8. RND()
9. USR()
10. WAIT

Commodore Geschichte: Video:

Commodore History – The 8-bit Guy (Teil 1-7, 2018, engl.)

geschrieben von Andreas Potthoff | 10. Juni 2023

Ein sehr gute Serie zum technischen Überblick der Computer und Peripherie in der Geschichte von Commodore, gezeigt von The 8-bit Guy.

- Teil 1: PET (00:20:10)
- Teil 2: VIC-20/VC-20 (00:27:49)
- Teil 3: C64 (00:34:59)
- Teil 4: Plus4, C16, C116 (00:28:59)
- Teil 5: C128 (00:31:45)
- Teil 6: PC kompatibel (00:23:22)
- Teil 7: Diskettenlaufwerke (00:28:06)

Teil 1

Teil 2

Teil 3

Teil 4

Teil 5

Teil 6

Teil 7