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			▀	▚

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