VIC 20

QUICK REFERENCE HANDBOOK

EDITED BY JEFFREY DANIELS

Version 0.9

BLOCK		Unexpanded	with 3K RAM	with 8K+ RAM	
$0 0	0	BASIC RAM 1K	BASIC RAM 1K	BASIC RAM 1K	$0 0
$400 1024		unused 3K	BASIC storage 6.5K POKE 642, 4 POKE 644, 30	unused 3K	$400 1024
$1000 4096		BASIC storage 3.5K POKE 642, 16 POKE 644, 30		Screen RAM 0.5K	$1000 4096
$1200 4608				BASIC storage 3.5K POKE 642, 18 POKE 644, 32	$1200 4608
$1E00 7680		Screen RAM 0.5K	Screen RAM 0.5K		$1E00 7680
$2000 8192	1	unused 8K	unused 8K	BLOCK 1 RAM 8K POKE 644, 64	$2000 8192
$4000 16384	2	unused 8K	unused 8K	BLOCK 2 RAM 8K POKE 644, 96	$4000 16384
$6000 24576	3	unused 8K	unused 8K	BLOCK 3 RAM 8K POKE 644, 128	$6000 24576
$8000 32768	4	Character ROM 4K	Character ROM 4K	Character ROM 4K	$8000 32768
$9000 36864		VIC CHIP 6561 0.25K	VIC CHIP 6561 0.25K	VIC CHIP 6561 0.25K	$9000 36864
$9100 37120		VIA CHIPS 6522 0.75K	VIA CHIPS 6522 0.75K	VIA CHIPS 6522 0.75K	$9100 37120
$9400 37888		unused 0.5K	unused 0.5K	Screen Color RAM 0.5K	$9400 37888
$9600 38400		Screen Color RAM 0.5K	Screen Color RAM 0.5K	unused 0.5K	$9600 38400
$9800 38912		I/O 2K	I/O 2K	I/O 2K	$9800 38912
$A000 40960	5	unused 8K	unused 8K	BLOCK 5 RAM 8K	$A000 40960
$C000 49152	6	BASIC ROM 8K	BASIC ROM 8K	BASIC ROM 8K	$C000 49152
$E000 57344	7	KERNAL ROM 8K	KERNAL ROM 8K	KERNAL ROM 8K	$E000 57344
$FFFF 65535		Unexpanded	with 3K RAM	with 8K+ RAM	$FFFF 65535

detailed map on page 48

THIS BLANK PAGE WILL NOT APPEAR IN PRINT COPIES

VIC 20 QUICK REFERENCE HANDBOOK

EDITED BY JEFFREY DANIELS

TABLE OF CONTENTS

INTRODUCTION 2
SPECIFICATIONS..... 3
VALUE CHART 4
BASIC..... 12
BASIC LINE STRUCTURE / TOKENS .. 16
VARIABLES 18
OPTIMIZATION TIPS..... 19
MATH FUNCTIONS 20
LOGICAL OPERATORS 21
PETSCII 22
PRINTER 24
KEYBOARD CHARACTERS..... 25
SCREEN CHARACTERS..... 28
SCREEN MAP 30
MEMORY POINTERS 34
DISPLAY OPTIONS (VIA)..... 36
SCREEN AND BORDER COLOR 37
USER DEFINED GRAPHICS 38
BINARY NUMBER CHART 39
GRAPHIC LOCATIONS 44
SOUND AND MUSIC 46
MEMORY LOCATIONS 48
MEMORY LABELS..... 78
ASSEMBLY MNEMONICS..... 82
REGISTER AND CYCLE CHART 84
OPCODES.. 86
ASSEMBLY CHARTS 88
UNINTENDED OPCODES 96
KERNAL 99
RS-232 104
BLOCK DIAGRAMS 105
PORTS..... 106
KEYBOARD DETECTION 108
JOYSTICK AND PADDLES 109
VICMON. 110
SUPER EXPANDER..... 111
PROGRAMMER'S AID..... 112
UTILITY CARTRIDGES..... 113
DISK DRIVE 114
JIFFYDOS 118
MEMORY UTILITY PROGRAMS 119
TYPE-IN PROGRAMS..... 120
ERRORS 125
ABBREVIATIONS..... 126
PHOTOCOPY WORKSHEET..... 127
INDEX..... 128

QUICK START	
LOAD FROM TAPE:	LOAD then press play on tape
LOAD FROM DISK:	LOAD"* [or <filename>]",8,1
LOAD TWO PART IMAGE:	LOAD"<A000 file>",8,1 NEW LOAD"<2000 or 6000 file>",8,1 SYS64802
START BASIC PROGRAM:	RUN
STOP BASIC PROGRAM:	Hold RUN STOP and RESTORE keys
SOFT-RESET (START ROM):	SYS64802
START SCOTT ADAMS GAMES:	SYS32592
START IEEE-488 CARTRIDGE:	SYS45065
START VICMON ML MONITOR:	SYS64096 OR SYS24576

Welcome! This manual is intended to provide quick reference for the VIC 20 computer. It presumes a basic understanding of VIC 20 programming and is designed around my personal preferences as a user. This is version 0.9b released in May of 2023.

Tables and maps are informed by the foundational works of the following:

Jim Butterfield, Andy Finkel, Neil Harris, Paul Higginbottom,
Michael Tomczyk, Russ Davies, Bob Yannes, Dale Gilbert,
Brian Grainger, Karl Hildon, Raeto Colin West, Jim Wilcox,
Ward F. Shrake, Paul Le Brasse, Michael Krause, Andreas Dripke,
Sven Petersen, David Barron, Lee Davison,
and others including ongoing research by members of the Denial community.

Thanks to Denial contributors for suggestions, proofreading, and content:

chysn, wimoos, DarwinNE, Schlowski,
srowe, Vic-20-Ian, Noizer, joshuadenmark,
Kweepa, orion70, AndyH, Mike,
necronom, vicist, mathom, thegg,
doug_in_nc, pixel, and others

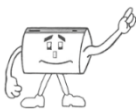
Special thanks for the generous support of the following:

Adrian Fox
Alessandro Ubiali
Anders Kator
Andrew Karlsson
Andrew Layden
Bindi Hawkey
Björg Stojalowski
Bo Goeran Kvamme
Brad Bell
Brent Santin
Bret Kellihan
Brian Lyons
Brian Thomson
Bryan Henry
Bryce Wilson
Christopher Prest
Clockwork Logic
Colton T Stephens
David T
Francios Leveille
Fredric Blåholtz
Gabriel Angelos
Geoff Nelson
Ghislain de Blois
Giovanni Giommarresi
Glen Richards
Heather Kent
Ian Buxton
James Happel
Jan-Erik Sundh
Jani Iltanen
jeff smaldon
Jess Ragan
Jesse Hagewood
Jonathan Brawn
Joonas Merienn
K Bergman
Kevin Potts
Larry Cameron
Laurent DIZY
Leif Bloomquist
Leo LaFlamme
Louis Mazzei
Marco Bergomi
mark gladson
Massimiliano Certelli
Mat Allen
Matt Dawson
Matthew Walworth
Michael Ingleby
Michael Kircher
mega-cart.com
Olli Savia
Paul Lambert
Paul Quirk
Pedro Bermejo
Pedro Lambrini
Peter Hurst
Peter Jensen
Renato Bugge
Robert Harbron
Roberto Bernardo
Robin Hurst
P1XL Games
Ryan Liston
Sampo Rintanen
Scott Walters
SignalDEV
Simon Henstock
Simon Rowe
Stefan Termén
Stephane Rogier
Steve McCrea
Tobias Andersson
VGA Foundation
Wayne Tomlinson
William M
and others

CC BY-NC-SA 2023 Attribution — NonCommercial — ShareAlike
This work is licensed under a Creative Commons Attribution-Noncommercial License (US/v3.0). The presentation of the information may be duplicated with attribution.

This publication is not-for-profit and for educational purposes. All trademarks (including VIC, VC20, VIC 20, Commodore, and all associated logos) are property of their respective trademark owners. Should this book contain copyrighted materials, all rights remain with the respective copyright holders. Permissions beyond the scope of this license are administered by Jeffrey Daniels. Information on how to request permission may be found at sleepingelephant.com/denial

Commodore VIC 20 VIC-1001 Japan 1980 VIC-20 USA May 1981 VC-20 Germany 1981 Discontinued: January 1985 CBM Basic V2 (8 KB) 20 KB ROM 5 KB RAM 3583 bytes free	1 game port: 9 pin male sub-D[1] cartridge port: 44 pin serial port: IEEE-488 6 pin DIN audio/Video: 5 pin DIN datassette: 12 pin PET style (edge) user port: RS-232 24 pin (edge) 66 keys QWERTY full stroke
KERNAL (8 KB) Charset (4 KB) CPU: 6502A 1.0227 MHz NTSC 1.108 MHz PAL Coprocessor: VIC6560 NTSC, 6561 PAL 6502A CPU, 6522 (2) VIA 6560 VIC (NTSC), 6561 VIC (PAL) 2114 (3) 1K x4 static RAM 2114 (8) 1K x4 static RAM (VIC-20) 6116 (2) 2K x8 static RAM (CR)	sound: VIC-1 6561 mono sound: 3 square, 1 noise NTSC, PAL, SECAM (by mod) 23 rows x 22 columns 176 x 184 3-bpp (88 x 184 mc) 8 screen colors 16 background/aux colors Power supply: 9V (most) 18 W consumption (most) 7 W consumption (CR model) Revision C: 9VAC and regulated 5VDC
Computer (most): 404 x 216 x 75mm, 1800g (15.9 x 8 x 2.9in., 4lbs.) Cartridge (most): 139.7 x 85.725 x 15.9mm (5.5 x 3.375 x 0.625in.) Game controller port: common 9-pin, male, sub-D connector (Atari VCS) [1] WARNING: other pin-compatible controllers such as Sega may damage	
Total disk storage capacity: 174,848 bytes per disk on 35 tracks Sequential files: 168,656 bytes per disk Relative files: 167,132 bytes per disk, 65,535 records per file Entries of a directory (track 18): 144 per disk Blocks: 683 (664 blocks free)	



PLEASE SUPPORT if this publication is of use to you.
A limited run of coil bound (“lay flat”) print books is available for purchase.
Shipped worldwide. Visit sleepingelephant.com/denial for details.

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
0	00	0		EoL	BRK	□□□□□□□□	dot mode
1	01	256			ORA(I,X)	□□□□□□□■	double width
2	02	512			JAM	□□□□□□■□	
3	03	768			SLO(I,X)	□□□□□□■■	stop
4	04	1024			NOP Z	□□□□□■□□	
5	05	1280		WHT	ORA Z	□□□□□■□■	WHT
6	06	1536			ASL Z	□□□□□■■□	
7	07	1792			SLO Z	□□□□□■■■	
8	08	2048			PHP	□□□□■□□□	lock
9	09	2304			ORA #	□□□□■□□■	unlock
10	0A	2560			ASL A	□□□□■□■□	line feed
11	0B	2816			ANC #	□□□□■□■■	
12	0C	3072			NOP	□□□□■■□□	
13	0D	3328			ORA	□□□□■■□■	car return
14	0E	3584			ASL	□□□□■■■□	lower case
15	0F	3840			SLO	□□□□■■■■	end double
16	10	4096			BPL	□□□■□□□□	tab
17	11	4352		down	ORA(I),Y	□□□■□□□■	down
18	12	4608		rvs on	JAM	□□□■□□■□	rvs on
19	13	4864		home	SLO(I),Y	□□□■□□■■	home
20	14	5120		insert	NOP Z,X	□□□■□■□□	delete
21	15	5376			ORA Z,X	□□□■□■□■	
22	16	5632			ASL Z,X	□□□■□■■□	
23	17	5888			SLO Z,X	□□□■□■■■	
24	18	6144			CLC	□□□■■□□□	
25	19	6400			ORA Y	□□□■■□□■	
26	1A	6656			NOP	□□□■■□■□	repeat
27	1B	6912			SLO, Y	□□□■■□■■	dot address
28	1C	7168		RED	NOP X	□□□■■■□□	RED
29	1D	7424		right	ORA X	□□□■■■□■	right
30	1E	7680		GRN	ASL X	□□□■■■■□	GRN
31	1F	7936		BLU	SLO X	□□□■■■■■	BLU

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
32	20	8192	space	space	JSR	□□■□□□□□	space
33	21	8448	!	!	AND(I,X)	□□■□□□□■	!
34	22	8704	"	"	JAM	□□■□□□■□	"
35	23	8960	#	#	RLA(I,X)	□□■□□□■■	#
36	24	9216	$	$	BIT Z	□□■□□■□□	$
37	25	9472	%	%	AND Z	□□■□□■□■	%
38	26	9728	&	&	ROL Z	□□■□□■■□	&
39	27	9984	'	'	RLA Z	□□■□□■■■	'
40	28	10240	(	(	PLP	□□■□■□□□	(
41	29	10496	)	)	AND #	□□■□■□□■	)
42	2A	10752	*	*	ROL A	□□■□■□■□	*
43	2B	11008	+	+	ANC #	□□■□■□■■	+
44	2C	11264	,	,	BIT	□□■□■■□□	,
45	2D	11520	-	-	AND	□□■□■■□■	-
46	2E	11776	.	.	ROL	□□■□■■■□	.
47	2F	12032	/	/	RLA	□□■□■■■■	/
48	30	12288	0	0	BMI	□□■■□□□□	0
49	31	12544	1	1	AND(I),Y	□□■■□□□■	1
50	32	12800	2	2	JAM	□□■■□□■□	2
51	33	13056	3	3	RLA(I),Y	□□■■□□■■	3
52	34	13312	4	4	NOP Z,X	□□■■□■□□	4
53	35	13568	5	5	AND Z,X	□□■■□■□■	5
54	36	13824	6	6	ROL Z,X	□□■■□■■□	6
55	37	14080	7	7	RLA Z,X	□□■■□■■■	7
56	38	14336	8	8	SEC	□□■■■□□□	8
57	39	14592	9	9	AND Y	□□■■■□□■	9
58	3A	14848	:	:	NOP	□□■■■□■□	:
59	3B	15104	;	;	RLA Y	□□■■■□■■	;
60	3C	15360	<	<	NOP	□□■■■■□□	<
61	3D	15616	=	=	AND X	□□■■■■□■	=
62	3E	15872	>	>	ROL X	□□■■■■■□	>
63	3F	16128	?	?	RLA X	□□■■■■■■	?

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
64	40	16384		@	RTI	□■□□□□□□	@
65	41	16640	A	A	EOR(I,X)	□■□□□□□■	A
66	42	16896	B	B	JAM	□■□□□□■□	B
67	43	17152	C	C	SRE(I,X)	□■□□□□■■	C
68	44	17408	D	D	NOP Z	□■□□□■□□	D
69	45	17664	E	E	EOR Z	□■□□□■□■	E
70	46	17920	F	F	LSR Z	□■□□□■■□	F
71	47	18176	G	G	SRE Z	□■□□□■■■	G
72	48	18432	H	H	PHA	□■□□■□□□	H
73	49	18688	I	I	EOR #	□■□□■□□■	I
74	4A	18944	J	J	LSR A	□■□□■□■□	J
75	4B	19200	K	K	ALR #	□■□□■□■■	K
76	4C	19456	L	L	JMP	□■□□■■□□	L
77	4D	19712	M	M	EOR	□■□□■■□■	M
78	4E	19968	N	N	LSR	□■□□■■■□	N
79	4F	20224	O	O	SRE	□■□□■■■■	O
80	50	20480	P	P	BVC	□■□■□□□□	P
81	51	20736	Q	Q	EOR(I),Y	□■□■□□□■	Q
82	52	20992	R	R	JAM	□■□■□□■□	R
83	53	21248	S	S	SRE(I),Y	□■□■□□■■	S
84	54	21504	T	T	NOP	□■□■□■□□	T
85	55	21760	U	U	EOR Z,X	□■□■□■□■	U
86	56	22016	V	V	LSR Z,X	□■□■□■■□	V
87	57	22272	W	W	SRE Z,X	□■□■□■■■	W
88	58	22528	X	X	CLI	□■□■■□□□	X
89	59	22784	Y	Y	EOR Y	□■□■■□□■	Y
90	5A	23040	Z	Z	NOP	□■□■■□■□	Z
91	5B	23296		[	SRE Y	□■□■■□■■	[
92	5C	23552		£	NOP X	□■□■■■□□	£
93	5D	23808		]	EOR X	□■□■■■□■	]
94	5E	24064		↑	LSR X	□■□■■■■□	↑
95	5F	24320		←	SRE X	□■□■■■■■	←

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
96	60	24576	space		RTS	□■■□□□□□	
97	61	24832			ADC(I,X)	□■■□□□□■	
98	62	25088			JAM	□■■□□□■□	
99	63	25344			RRA(I,X)	□■■□□□■■	
100	64	25600			NOP Z	□■■□□■□□	
101	65	25856			ADC Z	□■■□□■□■	
102	66	26112			ROR Z	□■■□□■■□	
103	67	26368			RRA Z	□■■□□■■■	
104	68	26624			PLA	□■■□■□□□	
105	69	26880			ADC #	□■■□■□□■	
106	6A	27136			ROR A	□■■□■□■□	
107	6B	27392			ARR #	□■■□■□■■	
108	6C	27648			JMP(I)	□■■□■■□□	
109	6D	27904			ADC	□■■□■■□■	
110	6E	28160			ROR	□■■□■■■□	
111	6F	28416			RRA	□■■□■■■■	
112	70	28672			BVS	□■■■□□□□	
113	71	28928			ADC(I),Y	□■■■□□□■	
114	72	29184			JAM	□■■■□□■□	
115	73	29440			RRA(I),Y	□■■■□□■■	
116	74	29696			NOP Z,X	□■■■□■□□	
117	75	29952			ADC Z,X	□■■■□■□■	
118	76	30208			ROR Z,X	□■■■□■■□	
119	77	30464			RRA Z,X	□■■■□■■■	
120	78	30720			SEI	□■■■■□□□	
121	79	30976			ADC Y	□■■■■□□■	
122	7A	31232			NOP	□■■■■□■□	
123	7B	31488			RRA Y	□■■■■□■■	
124	7C	31744			NOP X	□■■■■■□□	
125	7D	32000			ADC X	□■■■■■□■	
126	7E	32256			ROR X	□■■■■■■□	
127	7F	32512			RRA X	□■■■■■■■	

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
128	80	32768		END	NOP #	■□□□□□□□	
129	81	33024		FOR	STA(I,X)	■□□□□□□■	end double
130	82	33280		NEXT	NOP #	■□□□□□■□	
131	83	33536		DATA	SAX(I,X)	■□□□□□■■	
132	84	33792		INPUT#	STY Z	■□□□□■□□	
133	85	34048		INPUT	STA Z	■□□□□■□■	F1
134	86	34304		DIM	STX Z	■□□□□■■□	F3
135	87	34560		READ	SAX Z	■□□□□■■■	F5
136	88	34816		LET	DEY	■□□□■□□□	F7
137	89	35072		GOTO	NOP #	■□□□■□□■	F2
138	8A	35328		RUN	TXA	■□□□■□■□	F4
139	8B	35584		IF	ANE #	■□□□■□■■	F6
140	8C	35840		RESTORE	STY	■□□□■■□□	F8
141	8D	36096		GOSUB	STA	■□□□■■□■	shift return
142	8E	36352		RETURN	STX	■□□□■■■□	upper case
143	8F	36608		REM	SAX	■□□□■■■■	
144	90	36864		STOP	BCC	■□□■□□□□	BLK
145	91	37120		ON	STA(I),Y	■□□■□□□■	up
146	92	37376		WAIT	JAM	■□□■□□■□	rvs off
147	93	37632		LOAD	SHA(I),Y	■□□■□□■■	clear
148	94	37888		SAVE	STY Z,X	■□□■□■□□	insert
149	95	38144		VERIFY	STA Z,X	■□□■□■□■	
150	96	38400		DEF	STX Z,Y	■□□■□■■□	
151	97	38656		POKE	SAX Z, Y	■□□■□■■■	
152	98	38912		PRINT#	TYA	■□□■■□□□	
153	99	39168		PRINT	STA Y	■□□■■□□■	
154	9A	39424		CONT	TXS	■□□■■□■□	
155	9B	39680		LIST	TAS Y	■□□■■□■■	
156	9C	39936		CLR	SHY	■□□■■■□□	PUR
157	9D	40192		CMD	STA X	■□□■■■□■	left
158	9E	40448		SYS	SHX Y	■□□■■■■□	YEL
159	9F	40704		OPEN	SHA Y	■□□■■■■■	CYN

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
160	A0	40960		CLOSE	LDY #	■□■□□□□□	space
161	A1	41216		GET	LDA(I,X)	■□■□□□□■	゜
162	A2	41472		NEW	LDX #	■□■□□□■□	イ
163	A3	41728		TAB(	LAX(I,X)	■□■□□□■■	ウ
164	A4	41984		TO	LDY Z	■□■□□■□□	エ
165	A5	42240		FN	LDA Z	■□■□□■□■	オ
166	A6	42496		SPC(	LDX Z	■□■□□■■□	ヲ
167	A7	42752		THEN	LAX Z	■□■□□■■■	キ
168	A8	43008		NOT	TAY	■□■□■□□□	ク
169	A9	43264		STEP	LDA #	■□■□■□□■	ケ
170	AA	43520		+	TAX	■□■□■□■□	゛
171	AB	43776		-	LXA #	■□■□■□■■	ト
172	AC	44032		*	LDY	■□■□■■□□	ス
173	AD	44288		/	LDA	■□■□■■□■	L
174	AE	44544		↑	LDX	■□■□■■■□	┐
175	AF	44800		AND	LAX	■□■□■■■■	゜
176	B0	45056		OR	BCS	■□■■□□□□	┌
177	B1	45312		>	LDA(I),Y	■□■■□□□■	ア
178	B2	45568		=	JAM	■□■■□□■□	イ
179	B3	45824		<	LAX(I),Y	■□■■□□■■	ウ
180	B4	46080		SGN	LDY Z,X	■□■■□■□□	エ
181	B5	46336		INT	LDA Z,X	■□■■□■□■	オ
182	B6	46592		ABS	LDX Z,Y	■□■■□■■□	カ
183	B7	46848		USR	LAX Z,Y	■□■■□■■■	キ
184	B8	47104		FRE	CLV	■□■■■□□□	ク
185	B9	47360		POS	LDA Y	■□■■■□□■	ケ
186	BA	47616		SQR	TSX	■□■■■□■□	コ
187	BB	47872		RND	LAS Y	■□■■■□■■	サ
188	BC	48128		LOG	LDY X	■□■■■■□□	シ
189	BD	48384		EXP	LDA X	■□■■■■□■	ス
190	BE	48640		COS	LDX Y	■□■■■■■□	セ
191	BF	48896		SIN	LAX Y	■□■■■■■■	ソ

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
192	C0	49152		TAN	CPY #	■■□□□□□□	タ
193	C1	49408		ATN	CMP(I),X	■■□□□□□■	チ
194	C2	49664		PEEK	NOP #	■■□□□□■□	ツ
195	C3	49920		LEN	DCP(I,X)	■■□□□□■■	テ
196	C4	50176		STR$	CPY Z	■■□□□■□□	ト
197	C5	50432		VAL	CMP Z	■■□□□■□■	ナ
198	C6	50688		ASC	DEC Z	■■□□□■■□	ニ
199	C7	50944		CHR$	DCP Z	■■□□□■■■	ヌ
200	C8	51200		LEFT$	INY	■■□□■□□□	ネ
201	C9	51456		RIGHT$	CMP #	■■□□■□□■	ノ
202	CA	51712		MID$	DEX	■■□□■□■□	ハ
203	CB	51968		GO	SBX #	■■□□■□■■	ヒ
204	CC	52224		?SYNTAX	CPY	■■□□■■□□	フ
205	CD	52480		FOR	CMP	■■□□■■□■	ヘ
206	CE	52736		NEXT	DEC	■■□□■■■□	ホ
207	CF	52992		DATA	DCP	■■□□■■■■	マ
208	D0	53248		INPUT#	BNE	■■□■□□□□	ミ
209	D1	53504		INPUT	CMP(I),Y	■■□■□□□■	ム
210	D2	53760		DIM	JAM	■■□■□□■□	メ
211	D3	54016		READ	DCP(I),Y	■■□■□□■■	モ
212	D4	54272		LET	NOP Z,X	■■□■□■□□	ヤ
213	D5	54528		GOTO	CMP Z,X	■■□■□■□■	ユ
214	D6	54784		RUN	DEC Z,X	■■□■□■■□	ヨ
215	D7	55040		IF	DCP Z,X	■■□■□■■■	ラ
216	D8	55296		RESTORE	CLD	■■□■■□□□	リ
217	D9	55552		GOSUB	CMP Y	■■□■■□□■	ル
218	DA	55808		RETURN	NOP	■■□■■□■□	レ
219	DB	56064		REM	DCP Y	■■□■■□■■	ロ
220	DC	56320		STOP	NOP X	■■□■■■□□	ワ
221	DD	56576		ON	CMP X	■■□■■■□■	ン
222	DE	56832		WAIT	DEC X	■■□■■■■□	゛
223	DF	57088		LOAD	DCP X	■■□■■■■■	ヲ

DEC	HEX	MSB	SCRN	BASIC	6502	BIN	PETSCII
224	E0	57344		SAVE	CPX #	■■■□□□□□	Space π
225	E1	57600		VERIFY	SBC(I),X	■■■□□□□■	
226	E2	57856		DEF	NOP #	■■■□□□■□	
227	E3	58112		POKE	ISC(Z,X)	■■■□□□■■	
228	E4	58368		PRINT#	CPX Z	■■■□□■□□	
229	E5	58624		PRINT	SBC Z	■■■□□■□■	
230	E6	58880		CONT	INC Z	■■■□□■■□	
231	E7	59136		LIST	ISC Z	■■■□□■■■	
232	E8	59392		CLR	INX	■■■□■□□□	
233	E9	59648		CMD	SBC #	■■■□■□□■	
234	EA	59904		SYS	NOP	■■■□■□■□	
235	EB	60160		OPEN	USBC #	■■■□■□■■	
236	EC	60416		CLOSE	CPX	■■■□■■□□	
237	ED	60672		GET	SBC	■■■□■■□■	
238	EE	60928		NEW	INC	■■■□■■■□	
239	EF	61184		TAB(	ISC	■■■□■■■■	
240	F0	61440		TO	BEQ	■■■■□□□□	
241	F1	61696		FN	SBC(I),Y	■■■■□□□■	
242	F2	61952		SPC(	JAM	■■■■□□■□	
243	F3	62208		THEN	ISC(I),Y	■■■■□□■■	
244	F4	62464		NOT	NOP Z,X	■■■■□■□□	
245	F5	62720		STEP	SBC Z,X	■■■■□■□■	
246	F6	62976		+	INC Z,X	■■■■□■■□	
247	F7	63232		-	ISC Z,X	■■■■□■■■	
248	F8	63488		*	SED	■■■■■□□□	
249	F9	63744		/	SBC Y	■■■■■□□■	
250	FA	64000		↑	NOP	■■■■■□■□	
251	FB	64256		AND	ISC Y	■■■■■□■■	
252	FC	64512		OR	NOP X	■■■■■■□□	
253	FD	64768		>	SBC X	■■■■■■□■	
254	FE	65024		=	INC X	■■■■■■■□	
255	FF	65536		π	ISC X	■■■■■■■■	

ABS	A shift B	returns absolute value of X	DC58	182
		ABS(<number>)		
AND	A shift N	AND logical operand	CFE9	175
		<expression>AND<expression>		
ASC	A shift S	returns ASCII code	D78B	198
		ASC(<string>)		
ATN	A shift T	returns arctangent	E308	193
		ATN(<number>)		
CHR$	C shift H	returns ASCII string	D6E6	199
		CHR$(<number 0-255>)		
CLOSE	CL shift O	closes file engaged by OPEN	FFC3	160
		CLOSE<file number>		
CLR	C shift L	clears all defined variables	C65E	156
		CLR		
CMD	C shift M	changes output to device	CA86	157
		CMD<file number>[,<string>]		
CONT	C shift O	resume from STOP (direct mode)	C857	154
		CONT		
COS		returns cosine in radians	E261	190
		COS(<number>)		
DATA	D shift A	hold list of values for READ	C858	131
		DATA<value>[<,value…>]		
DEF	D shift E	defines a function FN	D3B3	150
		DEF FN<name>(<variable>)=<expression>		
DIM	D shift I	defines an array	D081	134
		DIM<variable>(<subscripts>)[,<var>(<sub>)…]		
END	E shift N	exits program	C831	128
		END		
EXP	E shift X	returns exponential (X>0 and X<88.03)	DFED	189
		EXP(<number>)		
FN		function defined by DEF	D3F4	165
		FN<name>(<expression>)		
FOR	F shift O	repeats section before NEXT *	C742	129
		FOR<variable>=<start>TO<limit>[STEP<increment>]		
FRE	F shift R	returns free memory in bytes	D37D	184
		FRE(<dummy number>)		

* FOR stack entry: Hi Lo pointer to statement in loop, Hi Lo line of first statement, TO value M4 M3 M2 M1 EXP, STEP sign value (M4 M3 M2 M1 EXP), Hi Lo variable, token ($81)

GET	G shift E	defines var from key pressed	FFE4	161
		GET<variable>		
GET#		returns variable from device		
		GET#<file number>,<variable>		
GO		GO TO variation of GOTO	C8A0	203
		GO TO<line number>		
GOSUB	GO shift S	goes to subroutine **	C883	141
		GOSUB<line number>		
GOTO	G shift O	goes to line number	C8A0	137
		GOTO<line number>		
IF		evaluates condition	C928	139
		IF<expression>THEN<line number/statement>		
INPUT		defines variable from user input	FFCF	133
		INPUT["<prompt>"];<variable>[,variable…]>		
INPUT#	I shift N	defines value from device	CBA5	132
		INPUT#<file number>,<variable [,variable …]>		
INT		returns integer	DCCC	181
		INT(<number>)		
LEFT$	LE shift F	returns leftmost chars of string	D700	200
		LEFT$(<string>,<length>)		
LEN		returns number of chars in string	D77C	195
		LEN(<string>)		
LET	L shift E	assigns value to variable	C9A5	136
		[LET]<variable>=<value>		
LIST	L shift I	lists the program in memory	C69C	155
		LIST[line number][-[line number]]		
LOAD	L shift O	transfers program from device	FFD5	147
		LOAD["<file>"[,<device>[,<command>]]]		
LOG		returns natural logarithm	D9EA	188
		LOG(<number>)		
MID$	M shift I	returns string from middle	D737	202
		MID$(<string>,<start>[,length])		
NEW		erases current program	C642	162
		NEW		
NEXT	N shift E	loops back to original FOR *	CD1E	130
		NEXT[<counter>[,<counter>…]]		

** GOSUB stack entry: Hi Lo pointer to statement, Hi Lo line number of statement, token ($8D)

NOT	N shift O	NOT logical operand	**CED4**	168
		NOT<expression>		
ON		sets branch point on condition	**C94B**	145
		ON<variable>GOTO/GOSUB<line>[,<line>…]		
OPEN	O shift P	prepares device for access	**FFC0**	159
		OPEN<file num>[,<device>[,<command>[,<name>]]]		
OR		OR logical operand	**CFE6**	176
		<expression>OR<expression>		
PEEK	P shift E	returns byte from memory location	**D80D**	194
		PEEK(<memory location>)		
POKE	P shift O	places value in RAM location	**D824**	151
		POKE<memory location>,<value>		
POS		returns current cursor position	**D39E**	185
		POS(<dummy number>)		
PRINT	?	displays on screen	**FFD2**	153
		PRINT[<variable> or <string>…]		
PRINT#	P shift R	sends data to device	**CA80**	152
		PRINT#<file num>,<variable>[,<variable>…]		
READ	R shift E	retrieve next DATA	**CC06**	135
		READ<variable>[,<variable>…]		
REM		holds programmer's remark	**C93B**	143
		REM<any text>		
RESTORE	RE shift S	resets DATA pointer	**C81D**	140
		RESTORE		
RETURN	RE shift T	completes GOSUB subroutine	**C8D2**	142
		RETURN		
RIGHT$	R shift I	returns rightmost chars of string	**D72C**	201
		RIGHT$(<string>,<length>)		
RND	R shift N	returns random number	**E094**	187
		RND(<number>)		
RUN	R shift U	executes current program	**C871**	138
		RUN[<line number>]		
SAVE	S shift A	saves program to device	**FFD8**	148
		SAVE[<"file">[,<device>[,<command>]]]		
SGN	S shift G	returns sign	**DC39**	180
		SGN(<number>)		

SIN	S shift I	returns sine in radians	**E268**	191
		SIN(<number>)		
SPC(	S shift P	prints blank spaces	**CAF8**	166
		SPC(<number 0-255>)		
SQR	S shift Q	returns square root	**DF71**	186
		SQR(<number>)		
STATUS	ST	returns status of I/O operation	**C795**	---
		STATUS		
STEP	ST shift E	sets increment of FOR loop *	**C82F**	169
		FOR<variable>=<start>TO<limit>[STEP<increment>]		
STOP	S shift T	breaks program	**D465**	144
		STOP		
STR$	ST shift R	returns string value of variable	**E127**	196
		STR$(<number>)		
SYS	S shift Y	begins machine code at location	**E127**	158
		SYS<memory location>		
TAB(	T shift A	positions print starting next line	**CAFB**	163
		TAB(<number 0-255>)		
TAN		returns tangent in radians	**E2B1**	192
		TAN(<number>)		
THEN	T shift H	decides outcome based on IF		167
		IF<expression>THEN<line number/statement>		
TIME	TI	returns timer as variable		---
		TI		
TIME$	TI$	returns timer string HH MM SS		---
		TI$="000000" (resets clock)		
TO		sets end point of FOR loop *		164
		FOR<var>=<start>TO<limit>[STEP<increment>]		
USR	U shift S	starts ML code with variable		183
		USR(<number>)		
VAL	V shift A	returns numeric value	**D7AD**	197
		VAL(<string>)		
VERIFY	V shift E	compares program to storage	**FFDB**	149
		VERIFY[<"file">[,<device>[,<command>]]]		
WAIT	W shift A	halts prog until mem condition	**D82D**	146
		WAIT<memory location>,<AND-mask>[,<XOR-mask>]		

condensed BASIC command and abbreviation list on page 126

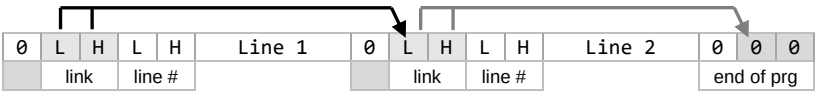
BASIC TOKENS

0	EoL	64	@	128	END	192	TAN
1	null	65	A	129	FOR	193	ATN
2	null	66	B	130	NEXT	194	PEEK
3	null	67	C	131	DATA	195	LEN
4	null	68	D	132	INPUT#	196	STR$
5	WHT	69	E	133	INPUT F1	197	VAL
6	null	70	F	134	DIM F3	198	ASC
7	null	71	G	135	READ F5	199	CHR$
8	case off	72	H	136	LET F7	200	LEFT$
9	case on	73	I	137	GOTO F2	201	RIGHT$
10	null	74	J	138	RUN F4	202	MID$
11	null	75	K	139	IF F6	203	GO
12	null	76	L	140	RESTORE F8	204	?SYNTAX
13	null	77	M	141	GOSUB shift ret	205	FOR
14	lower	78	N	142	RETURN upper	206	NEXT
15	null	79	O	143	REM	207	DATA
16	null	80	P	144	STOP BLK	208	INPUT#
17	down	81	Q	145	ON up	209	INPUT
18	rvs on	82	R	146	WAIT off	210	DIM
19	home	83	S	147	LOAD clear	211	READ
20	delete	84	T	148	SAVE insert	212	LET
21	null	85	U	149	VERIFY	213	GOTO
22	null	86	V	150	DEF	214	RUN
23	null	87	W	151	POKE	215	IF
24	null	88	X	152	PRINT#	216	RESTORE
25	null	89	Y	153	PRINT	217	GOSUB
26	null	90	Z	154	CONT	218	RETURN
27	null	91	[	155	LIST	219	REM
28	RED	92	£	156	CLR PUR	220	STOP
29	right	93	]	157	CMD left	221	ON
30	GRN	94	↑	158	SYS YEL	222	WAIT
31	BLU	95	←	159	OPEN CYN	223	LOAD
32	space	96		160	CLOSE	224	SAVE
33	!	97	♠	161	GET	225	VERIFY
34	"	98		162	NEW	226	DEF
35	#	99		163	TAB(	227	POKE
36	$	100		164	TO	228	PRINT#
37	%	101		165	FN	229	PRINT
38	&	102		166	SPC(	230	CONT
39	'	103		167	THEN	231	LIST
40	(	104		168	NOT	232	CLR
41	)	105		169	STEP	233	CMD
42	*	106		170	+	234	SYS
43	+	107		171	-	235	OPEN
44	,	108		172	*	236	CLOSE
45	-	109		173	/	237	GET
46	.	110		174	↑	238	NEW
47	/	111		175	AND	239	TAB(
48	0	112		176	OR	240	TO
49	1	113	●	177	>	241	FN
50	2	114		178	=	242	SPC(
51	3	115	♥	179	<	243	THEN
52	4	116		180	SGN	244	NOT
53	5	117		181	INT	245	STEP
54	6	118	╳	182	ABS	246	+
55	7	119	○	183	USR	247	-
56	8	120	♣	184	FRE	248	*
57	9	121		185	POS	249	/
58	:	122	◆	186	SQR	250	↑
59	;	123	+	187	RND	251	AND
60	<	124	▒	188	LOG	252	OR
61	=	125		188	EXP	253	>
62	>	126	π	190	COS	254	=
63	?	127	◥	191	SIN	255	π

EXAMPLE OF BASIC PROGRAM LINE STRUCTURE

```
 1 A=2:PRINT"■"A
 2 GOTO1
```

memory	BASIC	value	program explication
4096		0	Program start ↓ pointer 43-44 (2B-2C)
4097		15	Next line (low byte): 15
4098		16	Next line (high byte, *256 = 4096) = 4111
4099	1	1	Line Number (low byte)
4100		0	Line Number (high byte *256)
4101	A	65	Variable A (PETSCII A)
4102	=	178	Relational operator token
4103	2	50	Value of variable (PETSCII 2)
4104	:	58	Colon
4105	PRINT	153	BASIC keyword token
4106	"	34	Quotation mark (enter quote mode)
4107	■	28	BASIC keyword token for RED (quote more)
4108	"	34	Quotation mark (exit quote mode)
4109	A	65	Variable A (PETSCII A)
4110		0	End of line marker (EoL)
4111		22	Next line (low byte): 22
4112		16	Next line (high byte *256 = 4096) = 4118
4113	2	2	Line Number (low byte)
4114		0	Line Number (high byte *256)
4115	GOTO	137	BASIC keyword token
4116	1	49	Destination line number for GOTO
4117		0	End of line marker (EoL)
4118		0	0 (end of program indicator) 0
4119		0	0 (end of program indicator) 0
4120-	Variables start		pointer 45-46 (2D-2E)
	Arrays start (variables end)		pointer 47-48 (2F-30)
	Array end and start of empty space		pointer 49-50 (2B-2C)
	String start (direction ↑)		pointer 51-52 (33-34)
	Active string start		pointer 53-54 (35-36)
-7679	Program memory end		pointer 55-56 (37-38)



VARIABLES

Only the first two characters, plus the identifier symbol ($, #), are significant in identifying a variable.

REAL NUMERIC (FLOATING POINT)						$2D $2E
RANGE	any decimal ranged ±2.93×10^{-38} to ±1.70×10^{38}					
EXAMPLES	A AB A1			reserved:	ST TI	
MEMORY	7 bytes: 1 byte for exponent, 4 bytes for mantissa					
name ascii character 1	name ascii character 2	binary exp +129	binary mantissa	binary mantissa	binary mantissa	binary mantissa
ARRAYS	5 bytes per element					
POINTERS	45-46 ($2D-$2E)					

ABS(X)	EXP(X)	PEEK(X)	SQR(X)	USR(X)
ATN(X)	FRE(X)	RND(X)	ST	VAL(A$)
COS(X)	INT(X)	SGN(X)	TAN(X)	
DEF FN	LOG(X)	SIN(X)	TI	

STRING						$33 $34
RANGE	0 to 255 PETSCII characters					
EXAMPLES	A$ AB$ A1$			reserved:	TI$	
MEMORY	7 bytes: 1 byte for length, 2 bytes pointer, rest is 0					
name ascii character 1	name char 2 +128	number of characters	LSB of address	MSB of address	0	0
ARRAYS	3 bytes plus length of string per element					
POINTERS	51-52 ($33-$34)					

ASC(A$)	LEN(A$)	RIGHT$(A$,X)	TAB(X)
CHR$(X)	MID$(A$,S,L)	SPC(X)	TI$
LEFT$(A$,X)	POS(X)	STR$(X)	VAL(A$)

INTEGER						$2D $2E
RANGE	any whole number ranged −32768 to 32767					
EXAMPLES	A% AB% A1%					
MEMORY	7 bytes: 2 bytes for a 16-bit value, rest is 0					
name ascii character 1	name char +128	LSB of value	MSB of value	0	0	0
ARRAYS	2 bytes per element					
POINTERS	45-46 ($2D-$2E)					

0 = □□□□□□□□ □□□□□□□□ -1 = ■■■■■■■■ ■■■■■■■■
1 = □□□□□□□□ □□□□□□□■ -2 = ■■■■■■■■ ■■■■■■■□
2 = □□□□□□□□ □□□□□□■□ -3 = ■■■■■■■■ ■■■■■■□■
3 = □□□□□□□□ □□□□□□■■ -4 = ■■■■■■■■ ■■■■■■□□

VARIABLE DUMP

This program stores a machine language routine in the tape buffer. When called, it shows all variables currently in memory. The code is based on the work of Michael Krause, Andreas Dripke, wimoos, and Mike Kircher.
Type NEW after RUNning program. Activate with SYS828.

```
1 FORT=828TO978:READA:POKET,A:S=S+A:NEXT:IFS<>16561
  THENPRINT"CHECKSUM ERROR"
2 DATA234,234,165,45,164,46,196,48,208,2,197,47,176,
  86,133,20,132,21,105,2,144,1,200
3 DATA133,34,132,35,32,126,3,48,9,32,199,3,32,113,3,
  32,203,3,165,20,164,21,24,105,7
4 DATA144,212,200,208,209,138,48,45,152,48,57,32,166,
  219,200,76,215,221,160,0,177,20
5 DATA200,170,16,7,81,20,48,22,138,41,127,32,210,255,
  177,20,168,41,127,240,3,32,210
6 DATA255,138,48,31,152,48,31,96,160,0,177,34,133,98,
  200,177,34,168,32,151,211,80,202
7 DATA32,185,3,32,177,214,32,36,203,169,34,44,169,37,
  44,169,36,76,210,255,32,210,255
8 DATA169,61,208,246,72,169,13,32,210,255,104,96
```

BASIC MEMORY OPTIMIZATION

- ☐ Remove unnecessary REM and LET statements
- ☐ Remove extra spaces in lines
- ☐ Shorten line numbers (1 instead of 100)
- ☐ Combine multiple statements in each line
- ☐ Use abbreviations to exceed line limit
- ☐ Compare TAB and SPC to cursor positioning
- ☐ Use GOSUB routines for repeated task
- ☐ Streamline math operations
- ☐ Employ variables and strings for repeat use
- ☐ Use single character variables
- ☐ Recycle variables after use
- ☐ Use integers when possible
- ☐ Use arrays for management of larger data
- ☐ Use integer arrays to save memory
- ☐ Use READ and DATA to manage larger data
- ☐ DATA does not require a separate line
- ☐ Use a line zero
- ☐ Remove semicolons in PRINT when possible
- ☐ Use ON to combine GOTO statements
- ☐ Remove GOTO after THEN statements

BASIC SPEED OPTIMIZATION

- ☐ Follow nonconflicting memory reduction tips
- ☐ Put frequent routines early in program
- ☐ False IF statements are faster
- ☐ Direct PRINT is faster than string variables
- ☐ FOR loops are faster than GOTO
- ☐ Predefined variables are faster than numbers
- ☐ Replace isolated zeros with periods
- ☐ Streamline math operations
- ☐ IF A THEN IF B is faster than IF A AND B
- ☐ Numeric variables are faster than integers

ORDER OF OPERATIONS ►►►								performed left to right
( )	↑	NEG	* /	+ −	<=>	NOT	AND	OR

0 AND 0 = 0	0 OR 0 = 0	0 XOR 0 = 0
0 AND 1 = 0	0 OR 1 = 1	0 XOR 1 = 1
1 AND 0 = 0	1 OR 0 = 1	1 XOR 0 = 1
1 AND 1 = 1	1 OR 1 = 1	1 XOR 1 = 0

TRUE = -1
(1=1)=-1
FALSE = 0
(1=2)=0

XOR	X XOR Y = (X AND NOT Y) OR (NOT X AND Y)
NOR	A NOR B = NOT (A OR B)
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))+∏/2
inverse secant	ARCSEC(X) = ATN(X/SQR(X*X-1))
inverse cosecant	ARCCSC(X) = ATN(X/SQR(X*X-1))+(SGN(X)-1*∏/2
inverse cotangent	ARCOT(X) = ATN(X)+∏/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

NOT	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
=	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8

AND	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
2	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
3	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
4	0	0	0	4	4	4	4	0	0	0	0	4	4	4	4
5	1	0	1	4	5	4	5	0	1	0	1	4	5	4	5
6	0	2	2	4	4	6	6	0	0	2	2	4	4	6	6
7	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
8	0	0	0	0	0	0	0	8	8	8	8	8	8	8	8
9	1	0	1	0	1	0	1	8	9	8	9	8	9	8	9
10	0	2	2	0	0	2	2	8	8	10	10	8	8	10	10
11	1	2	3	0	1	2	3	8	9	10	11	8	9	10	11
12	0	0	0	4	4	4	4	8	8	8	8	12	12	12	12
13	1	0	1	4	5	4	5	8	9	8	9	12	13	12	13
14	0	2	2	4	4	6	6	8	8	10	10	12	12	14	14
15	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

OR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	3	3	5	5	7	7	9	9	11	11	13	13	15	15
2	3	2	3	6	7	6	7	10	11	10	11	14	15	14	15
3	3	3	3	7	7	7	7	11	11	11	11	15	15	15	15
4	5	6	7	4	5	6	7	12	13	14	15	12	13	14	15
5	5	7	7	5	5	7	7	13	13	15	15	13	13	15	15
6	7	6	7	6	7	6	7	14	15	14	15	14	15	14	15
7	7	7	7	7	7	7	7	15	15	15	15	15	15	15	15
8	9	10	11	12	13	14	15	8	9	10	11	12	13	14	15
9	9	11	11	13	13	15	15	9	9	11	11	13	13	15	15
10	11	10	11	14	15	14	15	10	11	10	11	14	15	14	15
11	11	11	11	15	15	15	15	11	11	11	11	15	15	15	15
12	13	14	15	12	13	14	15	12	13	14	15	12	13	14	15
13	13	15	15	13	13	15	15	13	13	15	15	13	13	15	15
14	15	14	15	14	15	14	15	14	15	14	15	14	15	14	15
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

PETSCII												
0	dot mode	32			64	@	@	96	─	─
1	double width	33	!	!	65	A	a	97	♠	A
2		34	"	"	66	B	b	98	│	B
3		35	#	#	67	C	c	99	─	C
4		36	$	$	68	D	d	100	─	D
5	WHT	37	%	%	69	E	e	101	─	E
6		38	&	&	70	F	f	102	─	F
7		39	'	'	71	G	g	103	│	G
8	enable case	40	(	(	72	H	h	104	│	H
9	disable case	41	)	)	73	I	i	105	╮	I
10	line feed	42	*	*	74	J	j	106	╰	J
11		43	+	+	75	K	k	107	╯	K
12		44	,	,	76	L	l	108	└	L
13	car return	45	-	-	77	M	m	109	╲	M
14	lower case	46	.	.	78	N	n	110	╱	N
15	end double	47	/	/	79	O	o	111	┌	O
16	tab	48	0	0	80	P	p	112	┐	P
17	down (low)	49	1	1	81	Q	q	113	●	Q
18	rvs on	50	2	2	82	R	r	114	▁	R
19	home	51	3	3	83	S	s	115	♥	S
20	insert	52	4	4	84	T	t	116	▕	T
21		53	5	5	85	U	u	117	▗	U
22		54	6	6	86	V	v	118	╳	V
23		55	7	7	87	W	w	119	○	W
24		56	8	8	88	X	x	120	♣	X
25		57	9	9	89	Y	y	121	▕	Y
26	repeat	58	:	:	90	Z	z	122	♦	Z
27	dot address	59	;	;	91	[	Ä Æ	123	┼	+
28	RED	60	<	<	92	£	Ö Ä Ø	124	▒	▒
29	right	61	=	=	93	]	Å Ü	125	│	|
30	GRN	62	>	>	94	↑	↑	126	π	▚
31	BLU	63	?	?	95	←	←	127	◥	▨

printer codes on page 24

PETSCII							
128		160	shifted space	192	—	224	
129	end double	161		193	A	225	
130		162		194	B	226	
131		163		195	C	227	
132		164		196	D	228	
133	F1	165		197	E	229	
134	F3	166		198	F	230	
135	F5	167		199	G	231	
136	F7	168		200	H	232	
137	F2	169		201	I	233	
138	F4	170		202	J	234	
139	F6	171		203	K	235	
140	F8	172		204	L	236	
141	shift return	173		205	M	237	
142	upper case	174		206	N	238	
143		175		207	O	239	
144	BLK	176		208	P	240	
145	up (up)	177		209	Q	241	
146	rvs off	178		210	R	242	
147	clear	179		211	S	243	
148	insert	180		212	T	244	
149		181		213	U	245	
150		182		214	V	246	
151		183		215	W	247	
152		184		216	X	248	
153		185		217	Y	249	
154		186		218	Z	250	
155		187		219	+	251	
156	PUR	188		220		252	
157	left	189		221		253	
158	YEL	190		222	π	254	
159	CYN	191		223		255	π

QUOTATION MODE

	KEY	PETSCII
0 BLK	CTRL 1	144
1 WHT	CTRL 2	5
2 RED	CTRL 3	28
3 CYN	CTRL 4	159
4 PUR	CTRL 5	156
5 GRN	CTRL 6	30
6 BLU	CTRL 7	31
7 YEL	CTRL 8	158

	KEY	PETSCII
F1	F1	133
F2	SHIFT F1	137
F3	F3	134
F4	SHIFT F3	138
F5	F5	135
F6	SHIFT F5	139
F7	F7	136
F8	SHIFT F7	140

	KEY	PETSCII
R ON	CTRL 9	18
R OFF	CTRL 0	146
home	HOME	19
clear	SHIFT HOME	147
up	SHIFT DWN	145
down	CRSR DWN	17
left	SHIFT RIGHT	157
right	CRSR RIGHT	29
insert	SHIFT DEL	148

Enter these manually in RVS mode		
disable case	H	9
enable case	I	8
shift return	SHIFT M	141
up case	SHIFT N	142
lower case	N	14
syntax error	SHIFT L	204
delete	DEL	20

Example: leave space between quotes, press SHIFT Return, use cursor keys to return to the space, press CTRL 9 then character key

PRINTER COMMANDS

PETSCII printer codes on page 22

List to Printer

```
OPEN4,4:CMD4:LIST
PRINT#4:CLOSE4
```

Wide Mode

```
PRINT#4,CHR$(14)          STANDARD: PRINT#1,CHR$(15)
```

Reverse Mode

```
PRINT#4,CHR$(18)          OFF: PRINT#1,CHR$(146)
```

Print Position

```
PRINT#4,CHR$(16)"<2 digit column><text>"
```

Graphic Mode — Each line binary AND 128

```
PRINT#4,CHR$(8)CHR$(<line 1 binary>)CHR$(<line 2 binary>). . .
```

each line binary is limited to 7 bits (127 to 0), follow with CHR$(15) for normal text

SCREEN CHARACTER CODES

←	1	2	3	4	5	6	7	8	9	0	+	-	£
31	49	50	51	52	53	54	55	56	57	48	43	45	28
128	177	178	179	180	181	182	183	184	185	176	171	173	156
	Q	W	E	R	T	Y	U	I	O	P	@	*	↑
	17	23	5	18	20	25	21	9	15	16	0	42	30
	145	151	133	146	148	153	149	137	143	144	128	170	158
RUN STOP	A	S	D	F	G	H	J	K	L	:	;	=	
	1	19	4	6	7	8	10	11	12	58	59	61	
	129	147	132	134	135	136	138	139	140	186	187	189	
	Z	X	C	V	B	N	M	,	.	/			
	26	24	3	22	2	14	13	44	46	47			
	154	152	131	150	130	142	141	172	174	175			
	space 32 160												

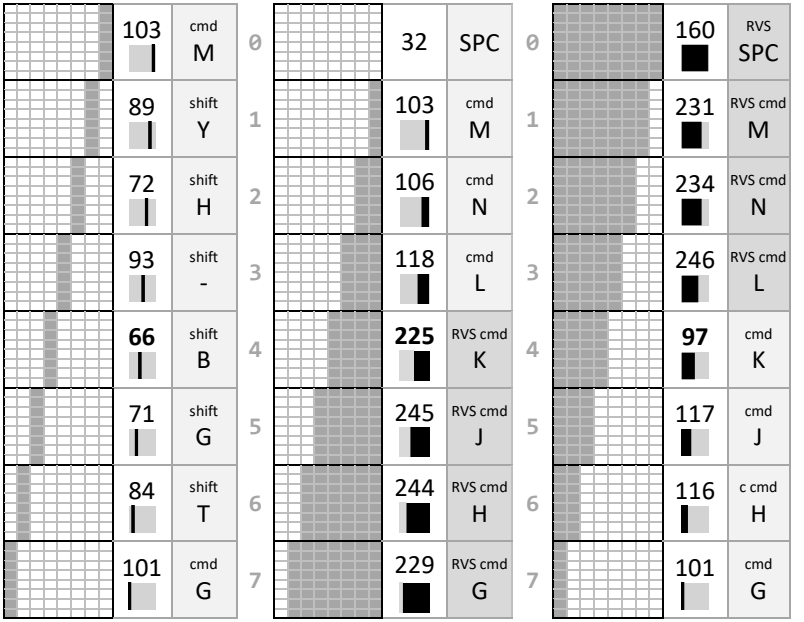
	!	"	#	$	%	&	'	(	)		[	]	↑
	33	34	35	36	37	38	39	40	41		91	93	105
	161	162	163	164	165	166	167	168	169		219	221	233
	81	87	69	82	84	89	85	73	79	80	122	64	94
	209	215	197	210	212	217	213	201	207	208	250	192	222
RUN STOP													
	65	83	68	70	71	72	74	75	76	27	29		
	202	203	204	198	199	200	202	203	204	155	157		
	90	88	67	86	66	78	77	60	62	63			
	218	216	195	214	194	206	205	188	190	191			

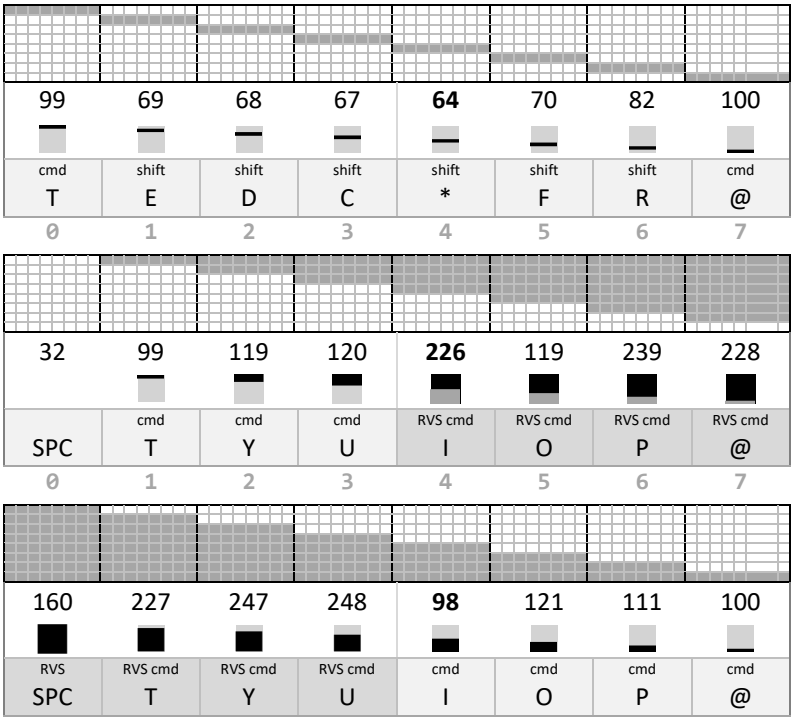
SHIFT

	1	2	3	4	5	6	7	8	9	0			
	BLK	WHT	RED	CYN	PUR	GRN	BLU	YEL	RVS	OFF	102	92	104
											230	220	232
CTRL													
	107	115	113	114	99	119	120	98	121	111	100	95	
	235	243	241	242	227	247	248	226	249	239	228	223	
RUN STOP													
	112	110	108	123	101	116	117	97	118				
	240	238	236	251	229	244	245	225	246				
	109	125	124	126	127	106	103						
	237	253	252	254	255	234	231						

complete character code chart on page 28

keyboard codes on page 108





complete character code chart on page 28

85	73
shift U	shift I
74	75
shift J	shift K

112	110
cmd A	cmd S
109	125
cmd Z	cmd X

107	115
cmd Q	cmd W
113	114
cmd E	cmd R

64	66
shift *	shift B
91	86
shift +	shift V

79	80
shift O	shift P
76	122
shift L	shift @

101	103
cmd M	cmd G
99	100
cmd T	cmd @

97	225
shift K	R shift K
226	98
R shift I	shift I

78	77
shift N	shift M
255	127
R cmd B	cmd B

233	223
R shift £	R cmd *
95	105
cmd *	shift £

108	123
cmd D	cmd F
124	126
cmd C	cmd V

236	251
R cmd D	R cmd F
252	254
R cmd C	R cmd V

65	83
shift A	shift S
88	90
shift X	shift Z

160	32
R space	space

81	87
shift Q	shift W

102	230
cmd +	R cmd +

complete character code chart on page 28

SCREEN CHARACTER CODES

00	0	@		20	32		40	64		60	96	
01	1	A	a	21	33	!	41	65	A	61	97	
02	2	B	b	22	34	"	42	66	B	62	98	
03	3	C	c	23	35	#	43	67	C	63	99	
04	4	D	d	24	36	$	44	68	D	64	100	
05	5	E	e	25	37	%	45	69	E	65	101	
06	6	F	f	26	38	&	46	70	F	66	102	
07	7	G	g	27	39	'	47	71	G	67	103	
08	8	H	h	28	40	(	48	72	H	68	104	
09	9	I	i	29	41	)	49	73	I	69	105	
0A	10	J	j	2A	42	*	4A	74	J	6A	106	
0B	11	K	k	2B	43	+	4B	75	K	6B	107	
0C	12	L	l	2C	44	,	4C	76	L	6C	108	
0D	13	M	m	2D	45	-	4D	77	M	6D	109	
0E	14	N	n	2E	46	.	4E	78	N	6E	110	
0F	15	O	o	2F	47	/	4F	79	O	6F	111	
10	16	P	p	30	48	0	50	80	P	70	112	
11	17	Q	q	31	49	1	51	81	Q	71	113	
12	18	R	r	32	50	2	52	82	R	72	114	
13	19	S	s	33	51	3	53	83	S	73	115	
14	20	T	t	34	52	4	54	84	T	74	116	
15	21	U	u	35	53	5	55	85	U	75	117	
16	22	V	v	36	54	6	56	86	V	76	118	
17	23	W	w	37	55	7	57	87	W	77	119	
18	24	X	x	38	56	8	58	88	X	78	120	
19	25	Y	y	39	57	9	59	89	Y	79	121	
1A	26	Z	z	3A	58	:	5A	90	Z	7A	122	
1B	27	[		3B	59	;	5B	91		7B	123	
1C	28	£		3C	60	<	5C	92		7C	124	
1D	29	]		3D	61	=	5D	93		7D	125	
1E	30	↑		3E	62	>	5E	94		7E	126	
1F	31	←		3F	63	?	5F	95		7F	127	

screen map on page 30

condensed map on page 35

SCREEN CHARACTER CODES

80	128		A0	160		C0	192		E0	224	
81	129		A1	161		C1	193		E1	225	
82	130		A2	162		C2	194		E2	226	
83	131		A3	163		C3	195		E3	227	
84	132		A4	164		C4	196		E4	228	
85	133		A5	165		C5	197		E5	229	
86	134		A6	166		C6	198		E6	230	
87	135		A7	167		C7	199		E7	231	
88	136		A8	168		C8	200		E8	232	
89	137		A9	169		C9	201		E9	233	
8A	138		AA	170		CA	202		EA	234	
8B	139		AB	171		CB	203		EB	235	
8C	140		AC	172		CC	204		EC	236	
8D	141		AD	173		CD	205		ED	237	
8E	142		AE	174		CE	206		EE	238	
8F	143		AF	175		CF	207		EF	239	
90	144		B0	176		D0	208		F0	240	
91	145		B1	177		D1	209		F1	241	
92	146		B2	178		D2	210		F2	242	
93	147		B3	179		D3	211		F3	243	
94	148		B4	180		D4	212		F4	244	
95	149		B5	181		D5	213		F5	245	
96	150		B6	182		D6	214		F6	246	
97	151		B7	183		D7	215		F7	247	
98	152		B8	184		D8	216		F8	248	
99	153		B9	185		D9	217		F9	249	
9A	154		BA	186		DA	218		FA	250	
9B	155		BB	187		DB	219		FB	251	
9C	156		BC	188		DC	220		FC	252	
9D	157		BD	189		DD	221		FD	253	
9E	158		BE	190		DE	222		FE	254	
9F	159		BF	191		DF	223		FF	255	

	0	1	2	3	4	5	6	7	8	9	10
1	7680 38400	7681 38401	7682 38402	7683 38403	7684 38404	7685 38405	7686 38406	7687 38407	7688 38408	7689 38409	7690 38410
2	7702 38422	7703 38423	7704 38424	7705 38425	7706 38426	7707 38427	7708 38428	7709 38429	7710 38430	7711 38431	7712 38432
3	7724 38444	7725 38445	7726 38446	7727 38447	7728 38448	7729 38449	7730 38450	7731 38451	7732 38452	7733 38453	7734 38454
4	7746 38466	7747 38467	7748 38468	7749 38469	7750 38470	7751 38471	7752 38472	7753 38473	7754 38474	7755 38475	7756 38476
5	7768 38488	7769 38489	7770 38490	7771 38491	7772 38492	7773 38493	7774 38494	7775 38495	7776 38496	7777 38497	7778 38498
6	7790 38510	7791 38511	7792 38512	7793 38513	7794 38514	7795 38515	7796 38516	7797 38517	7798 38518	7799 38519	7800 38520
7	7812 38532	7813 38533	7814 38534	7815 38535	7816 38536	7817 38537	7818 38538	7819 38539	7820 38540	7821 38541	7822 38542
8	7834 38554	7835 38555	7836 38556	7837 38557	7838 38558	7839 38559	7840 38560	7841 38561	7842 38562	7843 38563	7844 38564
9	7856 38576	7857 38577	7858 38578	7859 38579	7860 38580	7861 38581	7862 38582	7863 38583	7864 38584	7865 38585	7866 38586
10	7878 38598	7879 38599	7880 38600	7881 38601	7882 38602	7883 38603	7884 38604	7885 38605	7886 38606	7887 38607	7888 38608
11	7900 38620	7901 38621	7902 38622	7903 38623	7904 38624	7905 38625	7906 38626	7907 38627	7908 38628	7909 38629	7910 38630
12	7922 38642	7923 38643	7924 38644	7925 38645	7926 38646	7927 38647	7928 38648	7929 38649	7930 38650	7931 38651	7932 38652
13	7944 38664	7945 38665	7946 38666	7947 38667	7948 38668	7949 38669	7950 38670	7951 38671	7952 38672	7953 38673	7954 38674
14	7966 38686	7967 38687	7968 38688	7969 38689	7970 38690	7971 38691	7972 38692	7973 38693	7974 38694	7975 38695	7976 38696
15	7988 38708	7989 38709	7990 38710	7991 38711	7992 38712	7993 38713	7994 38714	7995 38715	7996 38716	7997 38717	7998 38718
16	8010 38730	8011 38731	8012 38732	8013 38733	8014 38734	8015 38735	8016 38736	8017 38737	8018 38738	8019 38739	8020 38740
17	8032 38752	8033 38753	8034 38754	8035 38755	8036 38756	8037 38757	8038 38758	8039 38759	8040 38760	8041 38761	8042 38762
18	8054 38774	8055 38775	8056 38776	8057 38777	8058 38778	8059 38779	8060 38780	8061 38781	8062 38782	8063 38783	8064 38784
19	8076 38796	8077 38797	8078 38798	8079 38799	8080 38800	8081 38801	8082 38802	8083 38803	8084 38804	8085 38805	8086 38806
20	8098 38818	8099 38819	8100 38820	8101 38821	8102 38822	8103 38823	8104 38824	8105 38825	8106 38826	8107 38827	8108 38828
21	8120 38840	8121 38841	8122 38842	8123 38843	8124 38844	8125 38845	8126 38846	8127 38847	8128 38848	8129 38849	8130 38850
22	8142 38862	8143 38863	8144 38864	8145 38865	8146 38866	8147 38867	8148 38868	8149 38869	8150 38870	8151 38871	8152 38872
23	8164 38884	8165 38885	8166 38888	8167 38887	8168 38888	8169 38889	8170 38890	8171 38891	8172 38892	8173 38893	8174 38894
	0	1	2	3	4	5	6	7	8	9	10

character codes on page 28 condensed map on page 35

11	12	13	14	15	16	17	18	19	20	21	
7691 38411	7692 38412	7693 38413	7694 38414	7695 38415	7696 38416	7697 38417	7698 38418	7699 38419	7700 38420	7701 38421	22
7713 38433	7714 38434	7715 38435	7716 38436	7717 38437	7718 38438	7719 38439	7720 38440	7721 38441	7722 38442	7723 38443	44
7735 38455	7736 38456	7737 38457	7738 38458	7739 38459	7740 38460	7741 38461	7742 38462	7743 38463	7744 38464	7745 38465	66
7757 38477	7758 38478	7759 38479	7760 38480	7761 38481	7762 38482	7763 38483	7764 38484	7765 38485	7766 38486	7767 38487	88
7779 38499	7780 38500	7781 38501	7782 38502	7783 38503	7784 38504	7785 38505	7786 38506	7787 38507	7788 38508	7789 38509	110
7801 38521	7802 38522	7803 38523	7804 38524	7805 38525	7806 38526	7807 38527	7808 38528	7809 38529	7810 38530	7811 38531	132
7823 38543	7824 38544	7825 38545	7826 38546	7827 38547	7828 38548	7829 38549	7830 38550	7831 38551	7832 38552	7833 38553	154
7845 38565	7846 38566	7847 38567	7848 38568	7849 38569	7850 38570	7851 38571	7852 38572	7853 38573	7854 38574	7855 38575	176
7867 38587	7868 38588	7869 38589	7870 38590	7871 38591	7872 38592	7873 38593	7874 38594	7875 38595	7876 38596	7877 38597	198
7889 38609	7890 38610	7891 38611	7892 38612	7893 38613	7894 38614	7895 38615	7896 38616	7897 38617	7898 38618	7899 38619	220
7911 38631	7912 38632	7913 38633	7914 38634	7915 38635	7916 38636	7917 38637	7918 38638	7919 38639	7920 38640	7921 38641	242
7933 38653	7934 38654	7935 38655	7936 38656	7937 38657	7938 38658	7939 38659	7940 38660	7941 38661	7942 38662	7943 38663	264
7955 38675	7956 38676	7957 38677	7958 38678	7959 38679	7960 38680	7961 38691	7962 38682	7963 38683	7964 38684	7965 38685	286
7977 38697	7978 38698	7979 38699	7980 38700	7981 38701	7982 38702	7983 38703	7984 38704	7985 38705	7986 38706	7987 38707	308
7999 38719	8000 38720	8001 38721	8002 38722	8003 38723	8004 38724	8005 38725	8006 38726	8007 38727	8008 38728	8009 38729	330
8021 38741	8022 38742	8023 38743	8024 38744	8025 38745	8026 38746	8027 38747	8028 38748	8029 38749	8030 38750	8031 38751	352
8043 38763	8044 38764	8045 38765	8046 38766	8047 38767	8048 38768	8049 38769	8050 38770	8051 38771	8052 38772	8053 38773	374
8065 38785	8066 38786	8067 38787	8068 38788	8069 38789	8070 38790	8071 38791	8072 38792	8073 38793	8074 38794	8075 38795	396
8087 38807	8088 38808	8089 38809	8090 38810	8091 38811	8092 38812	8093 38813	8094 38814	8095 38815	8096 38816	8097 38817	418
8109 38829	8110 38830	8111 38831	8112 38832	8113 38833	8114 38834	8115 38835	8116 38836	8117 38837	8118 38838	8119 38839	440
8131 38851	8132 38852	8133 38853	8134 38854	8135 38855	8136 38856	8137 38857	8138 38858	8139 38859	8140 38860	8141 38861	462
8153 38873	8154 38874	8155 38875	8156 38876	8157 38877	8158 38878	8159 38879	8160 38880	8161 38881	8162 38882	8163 38883	484
8175 38895	8176 38896	8177 38897	8178 38898	8179 38899	8180 38900	8181 38901	8182 38902	8183 38903	8184 38904	8185 38905	506
11	12	13	14	15	16	17	18	19	20	21	

new memory page at 7936

SCREEN	COLOR	8K+ SCREEN	8K+ COLOR
1Exx	96xx	10xx	94xx
1Fxx	97xx	11xx	95xx

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
2	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B
3	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41
4	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57
5	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D
6	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80	81	82	83
7	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99
8	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
9	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5
10	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB
11	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1
12	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	00	01	02	03	04	05	06	07
13	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
14	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33
15	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49
16	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
17	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75
18	76	77	78	79	7A	7B	7C	7D	7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B
19	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1
20	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7
21	B8	B9	BA	BB	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD
22	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3
23	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9

0	1	7680	1E00	38400	4096	1000	37888
22	2	7702	1E16	38422	4118	1016	37910
44	3	7724	1E2C	38444	4140	102C	37932
66	4	7746	1E42	38466	4162	1042	37954
88	5	7768	1E58	38488	4184	1058	37976
110	6	7790	1E6E	38510	4206	106E	37998
132	7	7812	1E84	38532	4228	1084	38020
154	8	7834	1E9A	38554	4250	109A	38042
176	9	7856	1EB0	38576	4272	10B0	38064
198	10	7878	1EC6	38598	4294	10C6	38086
220	11	7900	1EDC	38620	4316	10DC	38108
242	12	7922	1EF2	38642	4338	10F2	38130
264	13	7944	1F08	38664	4360	1108	38152
286	14	7966	1F1E	38686	4382	111E	38174
308	15	7988	1F34	38708	4404	1134	38196
330	16	8010	1F4A	38730	4426	114A	38218
352	17	8032	1F60	38752	4448	1160	38240
374	18	8054	1F76	38774	4470	1176	38262
396	19	8076	1F8C	38796	4492	118C	38284
418	20	8098	1FA2	38818	4514	11A2	38306
440	21	8120	1FB8	38840	4536	11B8	38328
462	22	8142	1FCE	38862	4558	11CE	38350
484	23	8164	1FE4	38884	4580	11E4	38372

SCREEN	COLOR	8K+ SCREEN	8K+ COLOR
7680 1E00	38400 9600	4096 1100	37888 9400

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
3	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65
4	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
5	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109
6	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131
7	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
8	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
9	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197
10	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219
11	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241
12	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263
13	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285
14	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307
15	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329
16	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351
17	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373
18	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395
19	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417
20	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439
21	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461
22	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483
23	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505

character code chart on page 28

color map
+/- 30720

8K color map
+/- 30208

3K to 8K
+/- 3584

3K to 8K color
+/- 512

-115	-114	-113	-112	-111	-110	-109	-108	-107	-106	-105
-93	-92	-91	-90	-89	-88	-87	-86	-85	-84	-85
-71	-70	-69	-68	-67	-66	-65	-64	-63	-62	-61
-49	-48	-47	-46	-45	-44	-43	-42	-41	-40	-39
-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17
-5	-4	-3	-2	-1		+1	+2	+3	+4	+5
+17	+18	+19	+20	+21	+22	+23	+24	+25	+26	+27
+39	+40	+41	+42	+43	+44	+45	+46	+47	+48	+49
+61	+62	+63	+64	+65	+66	+67	+68	+69	+70	+71
+83	+84	+85	+86	+87	+88	+89	+90	+91	+92	+93
+115	+114	+113	+112	+111	+110	+109	+108	+107	+106	105

SET SCREEN LOCATION $0288 $9002 $9005

```
POKE 36869, (PEEK(36869) AND 15) OR X : POKE 648, Z
POKE 36866, (PEEK(36866) AND 127) OR Y : SYS 58648
```

DEC	HEX	X	Y	Z
1024	0400	129	0	4
3072	0C00	131	0	12
3584	0E00	131	128	14
4096	1000	132	0	16
4608	1200	132	128	18
5120	1400	133	0	20
5632	1600	133	128	22
6144	1800	134	0	24
6656	1A00	134	128	26
7168	1C00	135	0	28
7680	1E00	135	128	30
8192	2000	136	0	32
15872	3E00	143	128	62

RETURN SCREEN LOCATION $9002 $9005

```
4* (PEEK(36866) AND 128) + 64*(PEEK(36869) AND 112)
```

RETURN COLOR MAP LOCATION $9002 $9400

```
37888 + 4 * (PEEK(36866) AND 128)
```

SET TOP LIMIT OF MEMORY POINTER $0034 $0038

```
POKE 51, 0 : POKE 52, X : POKE 55, 0 : POKE 56, X : CLR
```

DEC	HEX	X
5120	1400	20
6144	1800	24
7168	1C00	28
7680	1E00	30

SET BASIC PROGRAM LOCATION $0282 $0284 $E378

```
POKE 642, X : POKE 644, Y : SYS 58232
```

start location MSB = X * 256 end location MSB = Y * 256

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
7680	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	00	01	22
7702	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	44
7724	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	66
7746	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	88
7768	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	110
7790	90	91	92	93	94	95	96	97	98	99	00	01	02	03	04	05	06	07	08	09	10	11	132
7812	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	154
7834	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	176
7856	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	198
7878	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	220
7900	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	242
7922	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	264
7944	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	286
7966	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	308
7988	88	89	90	91	92	93	94	95	96	97	98	99	00	01	02	03	04	05	06	07	08	09	330
8010	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	352
8032	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	374
8054	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	396
8076	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	418
8098	98	99	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	440
8120	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	462
8142	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	484
8164	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	506

@	0	$00	128		32	$20	160	—	64	$40	192		96	$60	224
A	1	$01	129	!	33	$21	161	♠	65	$41	193	▌	97	$61	225
B	2	$02	130	"	34	$22	162	▏	66	$42	194	▂	98	$62	226
C	3	$03	131	#	35	$23	163	─	67	$43	195	▔	99	$63	227
D	4	$04	132	$	36	$24	164	┬	68	$44	196	_	100	$64	228
E	5	$05	133	%	37	$25	165	▔	69	$45	197	▏	101	$65	229
F	6	$06	134	&	38	$26	166	▁	70	$46	198	▒	102	$66	230
G	7	$07	135	´	39	$27	167	▏	71	$47	199	▕	103	$67	231
H	8	$08	136	(	40	$28	168	▏	72	$48	200	▒	104	$68	232
I	9	$09	137	)	41	$29	169	┐	73	$49	201	◤	105	$69	233
J	10	$0A	138	*	42	$2A	170	└	74	$4A	202	▌	106	$6A	234
K	11	$0B	139	+	43	$2B	171	┘	75	$4B	203	├	107	$6B	235
L	12	$0C	140	,	44	$2C	172	∟	76	$4C	204	▪	108	$6C	236
M	13	$0D	141	-	45	$2D	173	╲	77	$4D	205	└	109	$6D	237
N	14	$0E	142	.	46	$2E	174	╱	78	$4E	206	┐	110	$6E	238
O	15	$0F	143	/	47	$2F	175	┌	79	$4F	207	▁	111	$6F	239
P	16	$10	144	0	48	$30	176	┐	80	$50	208	┌	112	$70	240
Q	17	$11	145	1	49	$31	177	●	81	$51	209	┴	113	$71	241
R	18	$12	146	2	50	$32	178	_	82	$52	210	┬	114	$72	242
S	19	$13	147	3	51	$33	179	♥	83	$53	211	┤	115	$73	243
T	20	$14	148	4	52	$34	180	▏	84	$54	212	▌	116	$74	244
U	21	$15	149	5	53	$35	181	▗	85	$55	213	▌	117	$75	245
V	22	$16	150	6	54	$36	182	╳	86	$56	214	▌	118	$76	246
W	23	$17	151	7	55	$37	183	○	87	$57	215	▔	119	$77	247
X	24	$18	152	8	56	$38	184	♣	88	$58	216	▀	120	$78	248
Y	25	$19	153	9	57	$39	185	▏	89	$59	217	▂	121	$79	249
Z	26	$1A	154	:	58	$3A	186	♦	90	$5A	218	▁	122	$7A	250
[	27	$1B	155	;	59	$3B	187	┼	91	$5B	219	▪	123	$7B	251
£	28	$1C	156	<	60	$3C	188	▒	92	$5C	220	▪	124	$7C	252
]	29	$1D	157	=	61	$3D	189	▏	93	$5D	221	┘	125	$7D	253
↑	30	$1E	158	>	62	$3E	190	π	94	$5E	222	▪	126	$7E	254
←	31	$1F	159	?	63	$3F	191	◥	95	$5F	223	▚	127	$7F	255

DISPLAY OPTIONS (VIA CHIP)

INTERLACE MODE (NTSC only)	A-------	$9000
POKE 36864, PEEK(36864) OR 128 [ON]		
POKE 36864, PEEK(36864) AND 127 [OFF]		

HORIZONTAL SCREEN ORIGIN	-BBBBBBB	$9000
POKE 36864, PEEK(36864) AND 128 OR X		

VERTICAL SCREEN ORIGIN	CCCCCCCC	$9001
POKE 36865, X		

NUMBER OF COLUMNS	-DDDDDDD	$9002
POKE 36867, PEEK(36867) AND 128 OR X		

NUMBER OF ROWS	-EEEEEE-	$9003
POKE 36867, PEEK(36867) AND 129 OR (X*2)		

8 x 16 CHARACTER MODE	-------F	$9003
POKE 36867, PEEK(36867) OR 1 [ON]		
POKE 36867, PEEK(36867) AND 254 [OFF]		

CHARACTER BASE LOCATION	----BBBB	$9005
POKE 36869, PEEK(36869) AND 240 OR X		

AUXILIARY COLOR	AAAA----	$900E
POKE 36878, PEEK (36878) AND 15 OR X*16		

additional info on page 40

SCREEN REVERSE MODE	----B---	$900F
POKE 36879, PEEK (36879) AND 247 [ON]		
POKE 36879, PEEK (36879) OR 8 [OFF]		

memory list on page 60

	BORDER							
SCREEN	8 BLK	9 WHT	A RED	B CYN	C PUR	D GRN	E BLU	F YEL
0 BLK	8 08	9 09	10 0A	11 0B	12 0C	13 0D	14 0E	15 0F
1 WHT	24 18	25 19	26 1A	27 1B	28 1C	29 1D	30 1E	31 1F
2 RED	40 28	41 29	42 2A	43 2B	44 2C	45 2D	46 2E	47 2F
3 CYN	56 38	57 39	58 3A	59 3B	60 3C	61 3D	62 3E	63 3F
4 PUR	72 48	73 49	74 4A	75 4B	76 4C	77 4D	78 4E	79 4F
5 GRN	88 58	89 59	90 5A	91 5B	92 5C	93 5D	94 5E	95 5F
6 BLU	104 68	105 69	106 6A	107 6B	108 6C	109 6D	110 6E	111 6F
7 YEL	120 78	121 79	122 7A	123 7B	124 7C	125 7D	126 7E	127 7F
8 ORG	136 88	137 89	138 8A	139 8B	140 8C	141 8D	142 8E	143 8F
9 ORG2	152 98	153 99	154 9A	155 9B	156 9C	157 9D	158 9E	159 9F
A PNK	168 A8	169 A9	170 AA	171 AB	172 AC	173 AD	174 AE	175 AF
B CYN2	184 B8	185 B9	186 BA	187 BB	188 BC	189 BD	190 BE	191 BF
C PUR2	200 C8	201 C9	202 CA	203 CB	204 CC	205 CD	206 CE	207 CF
D GRN2	216 D8	217 D9	218 DA	219 DB	220 DC	221 DD	222 DE	223 DF
E BLU2	232 E8	233 E9	234 EA	235 EB	236 EC	237 ED	238 EE	239 EF
F YEL2	248 F8	249 F9	250 FA	251 FB	252 FC	253 FD	254 FE	255 FF
	BLK	WHT	RED	CYN	PUR	GRN	BLU	YEL

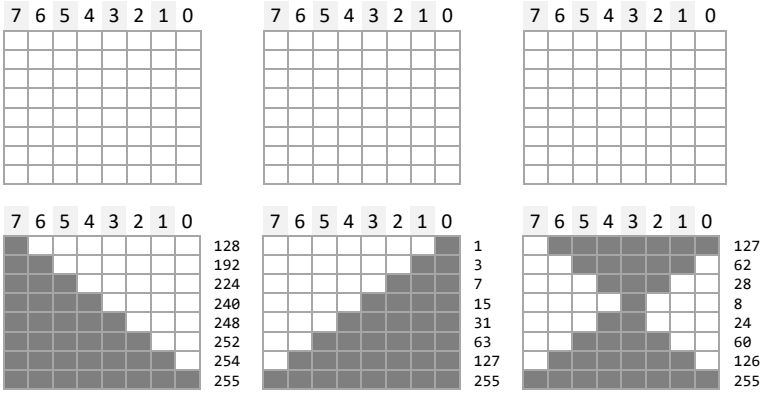
SCREEN AND BORDER COLOR	AAAABCCC	$900F
POKE 36879, X		

BORDER COLOR	-----CCC	$900F
POKE 36879, PEEK (36879) AND 248 OR X		

SCREEN COLOR	AAAA----	$900F
POKE 36879, PEEK (36879) AND 15 OR X*16		

full color chart on inside back cover

BIT	7	6	5	4	3	2	1	0
VALUE	128	64	32	16	8	4	2	1

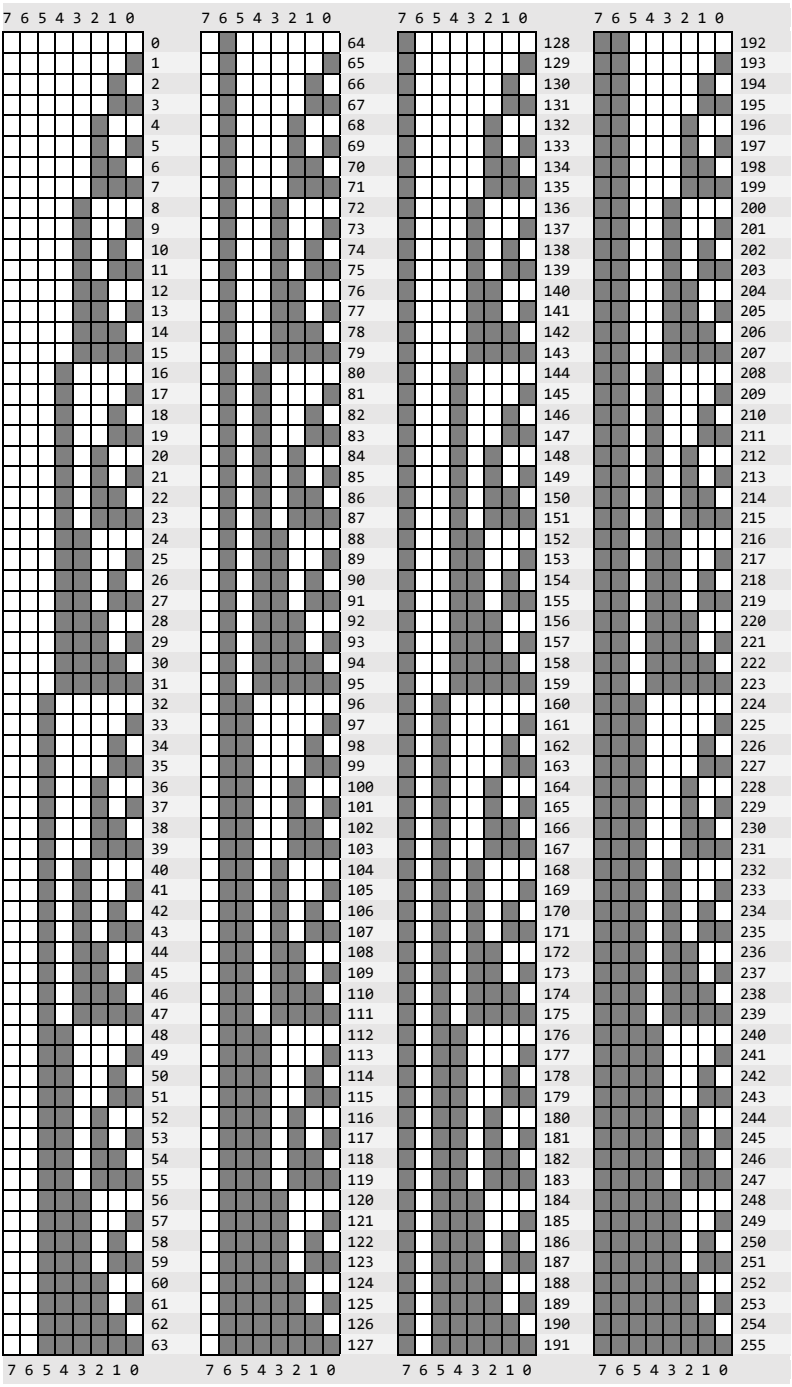


POKE 36869	Location	Contents	POKE 52 & 56
240	32768	upper case	sets free memory pointers to protect user character set
241	33792	upper case rvs	
242	34816	lower case	
243	35840	lower case rvs	
249 - 251	1024	EXPANSION RAM	
252	4096	BASIC RAM	
253	5120	BASIC RAM	20
254	6144	BASIC RAM	24
255	7168	BASIC RAM	28

CHARACTER EDITOR

Use on-screen editor (cursor). Change @ to the character you would like to edit. Empty spaces are OFF; all other characters are ON. Execute RUN2 on screen. View new character while holding return key. Release key to display DATA.

```
1 PRINT"{CLR}{RVS ON}76543210{RVS OFF}"SPC(222)"@",
  "RUN2
2 PRINT"{HOME}":FORL=0TO7:R=0:I=128:FORT=0TO7:R=R-I*
  (PEEK(7702+L*22+T)><32):I=I/2:NEXT
3 POKEPEEK(7910)*8+7168+L,R:PRINT,R"{LEFT}{2 SPACES}"
  :NEXT:POKE36869,255:WAIT197,64:POKE36869,240
```



HEX conversions on page at 43

BIT	7	6	5	4	3	2	1	0
VALUE	128	64	32	16	8	4	2	1

SCREEN □□	0	0	0	0
BORDER □■	64	16	4	1
CHARACTER ■□	128	32	8	2
AUX ■■	192	48	12	3

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

SET SCREEN / BORDER COLOR $900F

```
POKE 36879, <screen color>* 16 + <border color> OR 8
```

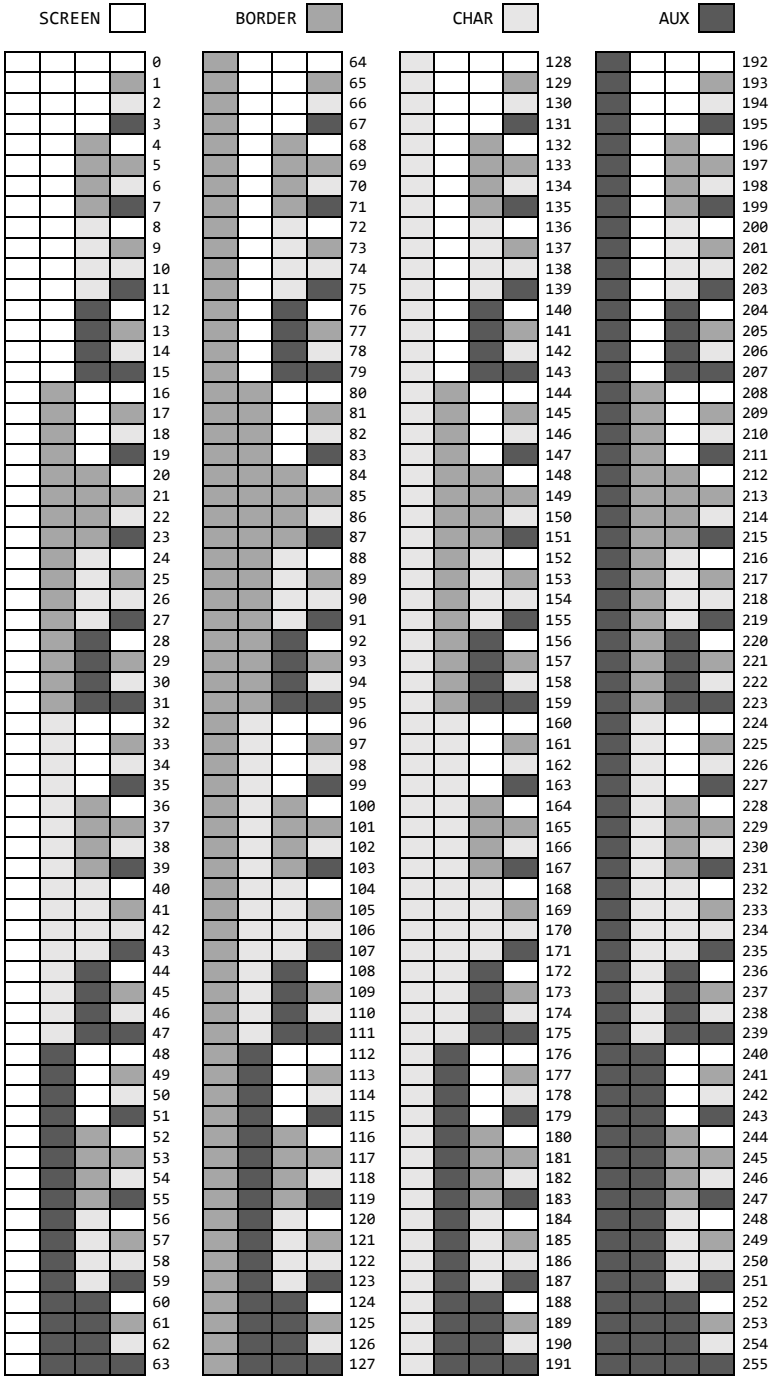
SET CURSOR COLOR $0286

```
POKE 646, X
```

SET AUXILARY COLOR $900E

```
POKE 36878, <aux color> + <volume 0-15>
```

	COLOR CODE	MULTI COLOR	AUX	RGB dither NTSC	RGB dither PAL
BLK	0	8	0	00 00 00 0	00 00 00 0
WHT	1	9	16	FF FF FF E	FF FF FF E
RED	2	10	32	F9 11 37 4	B6 1F 21 4
CYN	3	11	44	35 F9 F6 C	4D F0 FF C
PUR	4	12	64	FF 3C C6 8	B4 3F FF 8
GRN	5	13	80	3C ED A9 4	44 E2 37 4
BLU	6	14	96	0F 57 F7 4	1A 34 FF 4
YEL	7	15	112	FE E9 63 C	DC D7 1B C
ORG	--	--	128	FB 62 44 4	CA 54 00 4
Lt. ORG	--	--	144	FB BF DE 4	E9 B0 72 4
PNK	--	--	160	F3 AC E5 8	E7 92 93 8
Lt. CYN	--	--	176	A8 EA DD 4	9A F7 FD 4
Lt. PUR	--	--	192	E6 B8 F7 8	E0 9F FF 8
Lt. GRN	--	--	208	AB DD A4 8	8F E4 93 8
Lt. BLU	--	--	224	6A B3 E7 8	82 90 FF 8
Lt. YEL	--	--	240	F7 DA A5 C	E5 DE 85 C

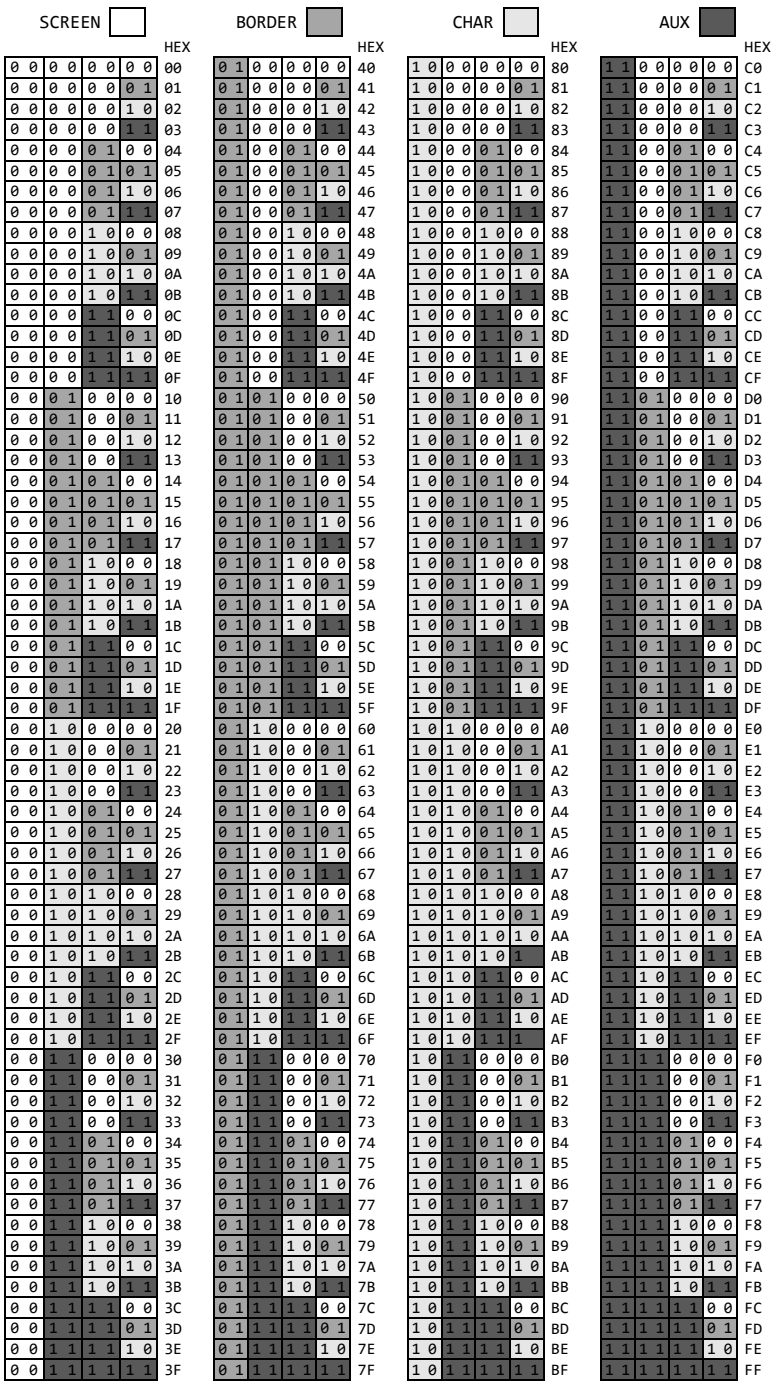


HEX conversions on page at 43

USER DEFINED CHARACTER EXAMPLES

```
1 FORT=0TO7: READA: POKE7168+T,A: NEXT: POKE36869,255
2 DATA 001, 002, 003, 004, 005, 006, 007, 008
```

007, 024, 032, 064, 064, 128, 128, 128	060, 090, 219, 255, 255, 129, 066, 060
224, 024, 004, 002, 002, 001, 001, 001	060, 066, 165, 129, 165, 153, 066, 060
007, 031, 063, 127, 127, 255, 255, 255	028, 028, 009, 062, 072, 028, 020, 054
224, 248, 252, 254, 254, 255, 255, 255	048, 048, 156, 122, 024, 037, 254, 068
008, 008, 028, 028, 062, 062, 127, 127	024, 024, 060, 126, 219, 024, 024, 060
016, 048, 126, 254, 126, 048, 016, 000	016, 024, 140, 255, 255, 140, 024, 016
008, 028, 062, 127, 028, 028, 028, 000	252, 135, 133, 133, 253, 143, 128, 128
008, 012, 126, 127, 126, 012, 008, 000	102, 153, 126, 137, 008, 016, 016, 056
248, 240, 240, 248, 156, 014, 004, 000	048, 008, 060, 106, 235, 255, 227, 126
031, 015, 015, 031, 057, 112, 032, 000	061, 127, 255, 189, 024, 024, 024, 024
066, 068, 072, 022, 033, 066, 135, 000	062, 020, 020, 034, 065, 127, 095, 062
066, 068, 072, 018, 038, 079, 130, 000	024, 090, 126, 090, 024, 219, 255, 195
031, 016, 016, 016, 144, 080, 048, 016	028, 058, 125, 127, 127, 062, 028, 008
028, 034, 065, 065, 034, 020, 119, 000	127, 008, 030, 185, 255, 127, 036, 126
062, 008, 028, 042, 042, 028, 008, 062	126, 255, 255, 103, 103, 190, 248, 168
126, 098, 016, 008, 016, 098, 126, 000	062, 127, 093, 073, 127, 062, 042, 000
028, 034, 065, 065, 065, 034, 028, 000	024, 060, 126, 219, 255, 036 090, 129
000, 024, 036, 036, 024, 000, 000, 000	060, 114, 255, 189, 189, 189, 036, 102
034, 020, 008, 062, 008, 062, 008, 000	007, 022, 008, 062, 095, 127, 127, 062
008, 028, 042, 040, 042, 028, 008, 000	014, 026, 050, 098, 182, 251, 111, 006
060, 062, 153, 161, 161, 153, 066, 060	000, 028, 056, 191, 188, 188, 024, 000
028, 046, 095, 127, 008, 008, 040, 016	065, 162, 116, 008, 116, 162, 065, 000
048, 096, 189, 254, 062, 028, 034, 000	003, 007, 010, 084, 104, 048, 088, 128
024, 024, 016, 092, 158, 130, 139, 112	028, 054, 125, 097, 073, 034, 028, 000
028, 034, 073, 077, 065, 034, 028, 000	252, 221, 141, 221, 221, 253, 253, 127
028, 034, 034, 127, 119, 119, 127, 062	003, 011, 043, 168, 163, 143, 063, 255
000, 096, 191, 165, 097, 000, 000, 000	136, 248, 168, 113, 034, 114, 122, 254
252, 191, 189, 191, 056, 120, 048, 000	051, 030, 020, 094, 141, 124, 124, 084
000, 255, 195, 165, 153, 129, 255, 000	008, 012, 010, 010, 056, 120, 048, 000
032, 019, 011, 003, 127, 062, 028, 062	223, 223, 000, 253, 253, 000, 223, 223
073, 042, 000, 099, 000, 042, 073, 000	221, 193, 056, 187, 187, 131, 028, 221



DECIMAL conversions on pages 39 and 41

		255	254	253	240			255	254	253	240
0	@	7168	6144	5120	32768	32		7424	6400	5376	33024
1	A	7176	6152	5128	32776	33	!	7432	6408	5384	33032
2	B	7184	6160	5136	32784	34	"	7440	6416	5392	33040
3	C	7192	6168	5144	32792	35	#	7448	6424	5400	33048
4	D	7200	6176	5152	32800	36	$	7456	6432	5408	33056
5	E	7208	6184	5160	32808	37	%	7464	6440	5416	33064
6	F	7216	6192	5168	32816	38	&	7472	6448	5424	33072
7	G	7224	6200	5176	32824	39	'	7480	6456	5432	33080
8	H	7232	6208	5184	32832	40	(	7488	6464	5440	33088
9	I	7240	6216	5192	32840	41	)	7496	6472	5448	33096
10	J	7248	6224	5200	32848	42	*	7504	6480	5456	33104
11	K	7256	6232	5208	32856	43	+	7512	6488	5464	33112
12	L	7264	6240	5216	32864	44	,	7520	6496	5472	33120
13	M	7272	6248	5224	32872	45	-	7528	6504	5480	33128
14	N	7280	6256	5232	32880	46	.	7536	6512	5488	33136
15	O	7288	6264	5240	32888	47	/	7544	6520	5496	33144
16	P	7296	6272	5248	32896	48	0	7552	6528	5504	33152
17	Q	7304	6280	5256	32904	49	1	7560	6536	5512	33160
18	R	7312	6288	5264	32912	50	2	7568	6544	5520	33168
19	S	7320	6296	5272	32920	51	3	7576	6552	5528	33176
20	T	7328	6304	5280	32928	52	4	7584	6560	5536	33184
21	U	7336	6312	5288	32936	53	5	7592	6568	5544	33192
22	V	7344	6320	5296	32944	54	6	7600	6576	5552	33200
23	W	7352	6328	5304	32952	55	7	7608	6584	5560	33208
24	X	7360	6336	5312	32960	56	8	7616	6592	5568	33216
25	Y	7368	6344	5320	32968	57	9	7624	6600	5576	33224
26	Z	7376	6352	5328	32976	58	:	7632	6608	5584	33232
27	[	7384	6360	5336	32984	59	;	7640	6616	5592	33240
28	£	7392	6368	5344	32992	60	<	7648	6624	5600	33248
29	]	7400	6376	5352	33000	61	=	7656	6632	5608	33256
30	↑	7408	6384	5360	33008	62	>	7664	6640	5616	33264
31	←	7416	6392	5368	33016	63	?	7672	6648	5624	33272

Relocate 0K set to 3K: move 4096 to 7168 (512 bytes) after opening in 3K

	255	254	253	240		255	254	253	240
64	7680	6656	5632	33280	96	7936	6696	5888	33536
65	7688	6664	5640	33288	97	7944	6704	5896	33544
66	7696	6672	5648	33296	98	7952	6712	5904	33552
67	7704	6680	5656	33304	99	7960	6720	5912	33560
68	7712	6688	5664	33312	100	7968	6728	5920	33568
69	7720	6696	5672	33320	101	7976	6736	5928	33576
70	7728	6704	5680	33328	102	7984	6744	5936	33584
71	7736	6712	5688	33336	103	7992	6752	5944	33592
72	7744	6720	5696	33344	104	8000	6760	5952	33600
73	7752	6728	5704	33352	105	8008	6768	5960	33608
74	7760	6736	5712	33360	106	8016	6776	5968	33616
75	7768	6744	5720	33368	107	8024	6784	5976	33624
76	7776	6752	5728	33376	108	8032	6792	5984	33632
77	7784	6760	5736	33384	109	8040	6800	5992	33640
78	7792	6768	5744	33392	110	8048	6808	6000	33648
79	7800	6776	5752	33400	111	8056	6816	6008	33656
80	7808	6784	5760	33408	112	8064	6824	6016	33664
81	7816	6792	5768	33416	113	8072	6832	6024	33672
82	7824	6800	5776	33424	114	8080	6840	6032	33680
83	7832	6808	5784	33432	115	8088	6848	6040	33688
84	7840	6816	5792	33440	116	8096	6856	6048	33696
85	7848	6824	5800	33448	117	8104	6864	6056	33704
86	7856	6832	5808	33456	118	8112	6872	6064	33712
87	7864	6840	5816	33464	119	8120	6880	6072	33720
88	7872	6848	5824	33472	120	8128	6888	6080	33728
89	7880	6856	5832	33480	121	8136	6896	6088	33736
90	7888	6864	5840	33488	122	8144	6904	6096	33744
91	7896	6656	5848	33496	123	8152	6912	6104	33752
92	7904	6664	5856	33504	124	8160	6920	6112	33760
93	7912	6672	5864	33512	125	8168	6928	6120	33768
94	7920	6680	5872	33520	126	8176	6936	6128	33776
95	7928	6688	5880	33528	127	8184	6944	6136	33784

Relocate 3K set to 0K: move 7168 to 4096 (512 bytes) before opening in 0K

36874 (alto)
128-255

36875 (tenor)
128-255

36876 (soprano)
128-255

36877 (noise)
128-255

36878 (volume)
0-15

255	C	*do*
134	C#	
141	D	*re*
147	D#	
153	E	*mi*
159	F	
164	F#	*fa*
170	G	
174	G#	*sol*
179	A	
183	A#	*la*
187	B	*ti*
191	C	*do*
195	C#	

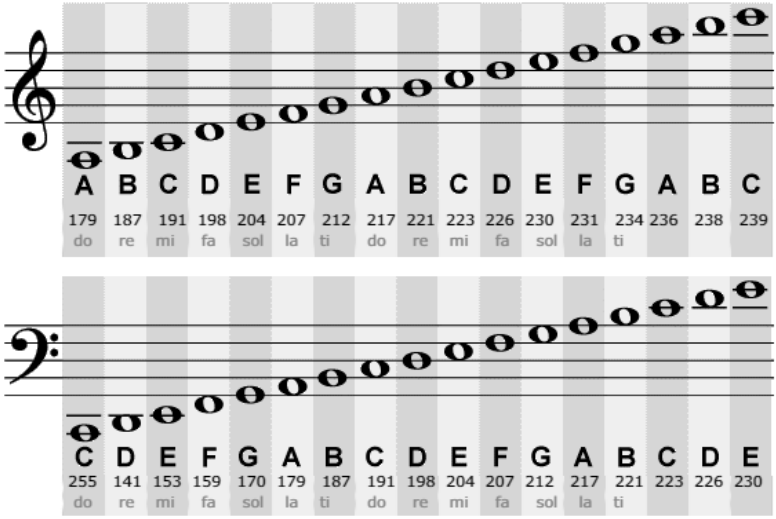
191	C	*do*
195	C#	
198	D	*re*
201	D#	
204	E	*mi*
207	F	
210	F#	*fa*
212	G	
215	G#	*sol*
217	A	
219	A#	*la*
221	B	*ti*
223	C	*do*
225	C#	

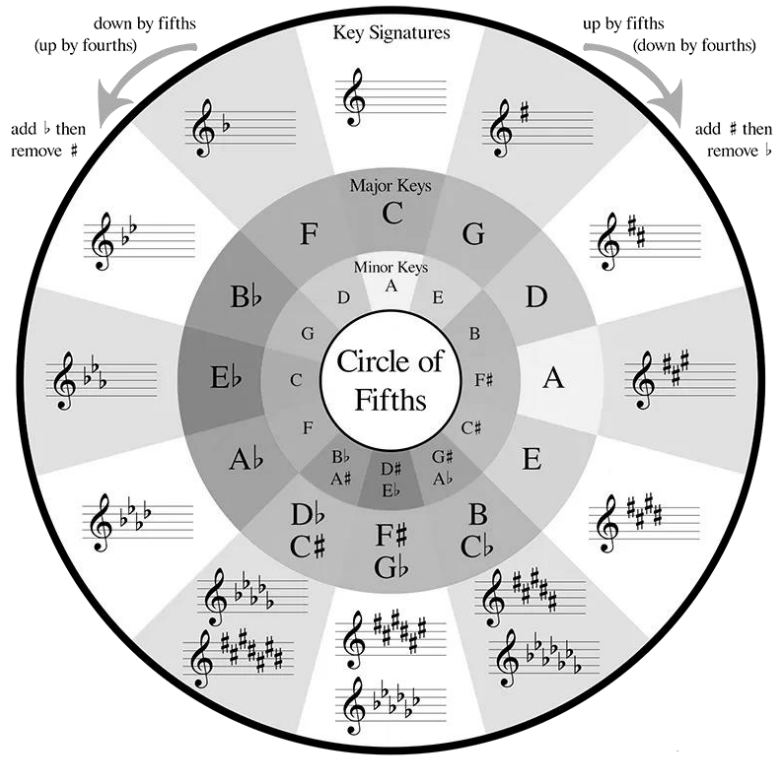
223	C	*do*
225	C#	
226	D	*re*
228	D#	
230	E	*mi*
231	F	
232	F#	*fa*
234	G	
235	G#	*sol*
236	A	
237	A#	*la*
238	B	*ti*
239	C	*do*
0		

	134 C#	**147 D#**		**164 F#**	**174 G#**	**183 A#**		**195 C#**		
255 C	141 D	153 E	159 F	170 G	179 A	187 B	191 C	198 D		

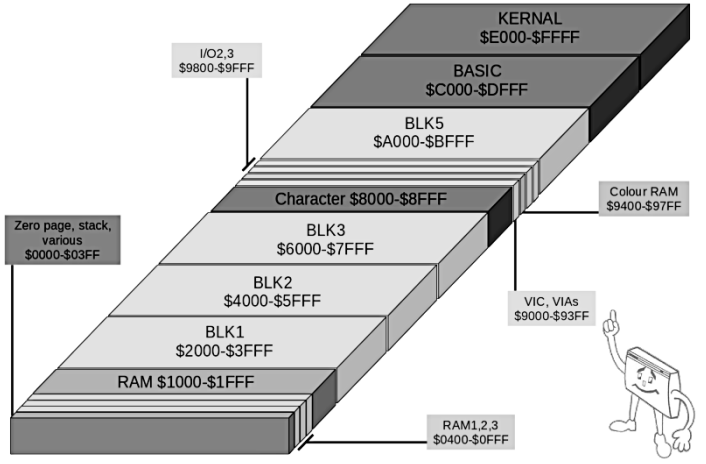
	195 C#	**201 D#**		**210 F#**	**215 G#**	**219 A#**		**225 C#**		
187 B	191 C	198 D	204 E	207 F	212 G	217 A	221 B	223 C	226 D	

	225 C#	**228 D#**		**232 F#**	**235 G#**	**237 A#**				
221 B	223 C	226 D	230 E	231 F	234 G	236 A	238 B	239 C	D	


A B C D E F G A B C D E F G A B C
179 187 191 198 204 207 212 217 221 223 226 230 231 234 236 238 239
do re mi fa sol la ti do re mi fa sol la ti
C D E F G A B C D E F G A B C D E
255 141 153 159 170 179 187 191 198 204 207 212 217 221 223 226 230
do re mi fa sol la ti do re mi fa sol la ti


down by fifths
(up by fourths)
Key Signatures
up by fifths
(down by fourths)
add ♭ then
remove ♯
add ♯ then
remove ♭
Major Keys
C
F
G
B♭
D
E♭
A
A♭
E
D♭
C♯
F♯
G♭
B
C♭
Minor Keys
A
D
E
G
B
C
F♯
F
C♯
B♭
A♯
D♯
E♭
G♯
A♭
Circle of
Fifths

MEMORY MAP



courtesy of srowe

VIC 1110 MEMORY EXPANSION SETTINGS						
	HEX	DEC	DIP SWITCH			
BLK 1	$2000-$3FFF	8192-16383	OFF	OFF	OFF	ON
BLK 2	$4000-$5FFF	16384-24575	OFF	OFF	ON	OFF
BLK 3	$6000-$7FFF	24576-32767	OFF	ON	OFF	OFF
BLK 5	$A000-$BFFF	40960-49151	ON	OFF	OFF	OFF

WARNING: Never have more than one switch on at the same time!

USER MEMORY ALLOCATION AND POINTERS

0	PROGRAM →	000	VARIABLES →	ARRAYS →	EMPTY	← STRINGS	TOP
↑ $2B,2C 43,44			↑ $2D,2E 45,46	↑ $2F,30 47,48	↑ $31,32 49,50	↑ $33,34 51,52	↑ $37,38 55,56
start of BASIC program, typically, 4097 43 = 1 44 = 17		end of program indictor	end of BASIC program, start of variable data	end of variables, start of array data	end of arrays, start of empty space	end of strings: which starts at top and stored downward	limit of free memory, typically, 7680

COMMON POINTER VALUES (MSB)

see page 124 for more locations

1024	4	$04	start of user ram with 3k expansion
4096	17	$11	start of user program area (default); screen RAM with 8K+ expansion
4608	18	$12	start of user program area with 8K+ expansion
5120	20	$14	start of user character set when 36869 is 253
6144	24	$18	start of user character set when 36869 is 254
7168	28	$1C	start of user character set when 36869 is 255
7680	30	$1E	Top of BASIC program space (default); screen RAM (default)

HEX	DEC		DESCRIPTION
	0-143		BASIC Working Storage
0000	0	USRPOK	JMP instruction for USR function default value 76
0001	1-	ADDPRC	Vector for USR function LSB (low byte) default value 72
0002	2		Vector for USR function MSB (high byte) default value 210
0003	3-	ADRAY1	Float-Fixed vector, Convert float to integer (LSB) default value 170
0004	4		Float-Fixed vector, Convert float to integer (MSB) default value 209
0005	5-	ADRAY2	Fixed-Float vector, Convert integer to float (LSB) default value 145
0006	6		Fixed-Float vector, Convert integer to float (MSB) default value 211
0007	7	CHARAC	General counter for Basic. Search colon or end line 111 temporary value for string pointer
0008	8	ENDCHR	Scan-quotes flag 0 delimiter
0009	9	TRMPOS	TAB col save; Cursor col before last TAB or SPC 0 to 87 range
000A	10	VERCHK	Load / Verify flag 0 LOAD 1 VERIFY
000B	11	COUNT	Input buffer pointer/# subscript, array dimensions Temp byte; line crunch / array; access / logic ops AND sets location to 0 OR sets location to 255 DIM: number of dimensions in array
000C	12	DIMFLG	Default DIM flag, First character of array name
000D	13	VALTYP	Variable type, (FRMEVL routines set) 0 numeric 255 ($FF) string
000E	14	INTFLG	Integer flag (type) when location 13 is 0. 0 floating point 128 ($80) integer
000F	15	GARBFL	DATA scan / LIST quote/memory flag LIST and DATA: prevents detokenization
0010	16	SUBFLG	Subscript flag / FNx flag 128 when FN is defined
0011	17	INPFLG	Input flags; line number for input error message 0 INPUT 64 ($40) GET 152 ($98) READ ERROR message pointer

HEX	DEC		DESCRIPTION
0012	18	TANSGN	ATN, TAN, SIN, and Comparison eval flag
			bit 0: > (greater than)
			bit 1: = (equal)
			bit 2: < (less than)
0013	19	CHANNL	Current I/O channel / device prompt flag
			0 keyboard (default) 1 tape
			2 RS-232 / user port 3 screen
			4-5 printer 6-7 plotter
			8-11 disk 4-31 any serial device
0014	20-	LINNUM	Line number integer in (LSB); temp integer
0015	21		Line number integer in (MSB) ; temp integer
			BASIC integer address (for SYS, GOTO, LIST, etc.)
0016	22	TEMPPT	Temp string descriptor stack pointer (default 25)
			Next available slot in temp string descriptor
			25 empty (default), zero descriptors used
			34 results in ?FORMULA TOO COMPLEX
			35 disable line numbers in LIST
0017	23-24 L/M	LASTPT	Last temporary string vector (default 22,0)
			25, 0 one string descriptor being used
			28, 0 two string descriptors being used
			31, 0 all three string descriptors being used
0019	25-33	TEMPST	Stack of descriptors for three temporary strings
			Length of string, LSB address, MSB address
0022	34-35	INDEX1	First utility pointer: Pointer for number transfer
0024	36-37	INDEX2	Second utility pointer: Pointer for number transfer
0026	38-42	RESHO	Product area for BASIC multiplication
002B	43-	TXTTAB	Pointer: Start of Basic program (LSB)
002C	44		Pointer: Start of Basic program (MSB)
			default values 1, 16 (4097)
			PEEK(43)+256*PEEK(44) detects start of BASIC
002D	45-	VARTAB	Pointer: Start of Variables (end of prog) (LSB)
002E	46		Pointer: Start of Variables (MSB)
			PEEK(45)+256*PEEK(46) detects end of BASIC
002F	47-	ARYTAB	Pointer: Start of Arrays (end of vars) (LSB)
0030	48		Pointer: Start of Arrays (end of vars) (MSB)
			One byte past the end of the BASIC program
0031	49-	STREND	Pointer: End of Arrays (LSB)
0032	50		Pointer: End of Arrays (MSB)
0033	51-	FRETOP	Pointer: String storage (end of strings) (LSB)
0034	52		Pointer: String storage (end of strings) (MSB)
0035	53-	FRESPC	Utility pointer to active string (LSB)
0036	54		Utility pointer to active string (MSB)
0037	55-	MEMSIZ	Pointer: End of program memory (LSB) default 0
0038	56		Pointer: End of program memory (MSB) default 30
			POKE56,28 protects user characters at 7168

HEX	DEC		DESCRIPTION
0039	57	CURLIN	Current Basic line number (LSB)
003A	58		Current Basic line number (MSB)
			FF (255) direct mode statement
003B	59-60 L/M	OLDLIN	Previous Basic line number (break line number)
003D	61-62 L/M	OLDTXT	Pointer: Basic statement for CONT (line loc)
003F	63-64 L/M	DATLIN	Current DATA line number
0041	65-66 L/M	DATPTR	Current DATA address
0043	67-68 L/M	INPPTR	Input vector; pointer to source of INPUT GET READ
0045	69-70 L/M	VARNAM	Current variable name (2 characters with type flags)
			If one character, 70=zero
			Floating point: no high bits on (bit 7 OFF OFF)
			Integer: both have high bits on (bit 7 ON ON)
			String: (bit 7 OFF ON)
			Function (bit 7 ON OFF)
0047	71-72 L/M	VARPNT	Current variable address (1 byte after name)
0049	73-74 L/M	FORPNT	Variable pointer for FOR / NEXT
			LIST uses 73 as temp save area
			WAIT uses 73 as second parameter, 74 third
			CLOSE uses 73 to save file number
			LOAD and SAVE uses 73 to save device number
004A	74		WAIT uses 74 as third parameter
			RETURN uses 74 as a flag set to 255 for GOSUB
004B	75	OPPTR	Y save reg-new oper save / LSB precedence flag
004C	76		BASIC execute pointer temp MSB
004D	77	OPMASK	Current operator, comparison symbol accumulator
			bit 0: > (greater than)
			bit 1: = (equal)
			bit 2: < (less than)
004E	78-79 L/M	DEFPNT	Misc. work area; FN definition pointer high-low
0050	80-81 L/M	DSCPTN	Work area; pointer to current string description
0052	82	SIZE	Length of above string (location 80 $50)
0053	83	FOUR6	Constant, 3 or 7 for garbage collection step size
0054	84-86	JMPER	Jump vector for functions
	84		76 jump opcode
0055	85		Function address LSB / garbage collection
0056	86		Function address MSB / exp, addition work area
0057	87-96	TEMPF3	Misc. numeric work area
	95		Pointer to variable 7 byte descriptor
	95		Decimal point flag when converting a string
	95-96		LIST active pointer
0061	97-102	FAC	BASIC floating point accumulator one
	97		Accum#1: Exponent
	98-99		Integer stored during floating conversion
	98-101		Accum#1: Normalized mantissa
	100-101		String descriptor during string processing routines

HEX	DEC		DESCRIPTION
0066	102		Accum#1: Sign
			0 positive
			128-255 negative
0067	103	SGNFLG	Series evaluation constant pointer
0068	104	BITS	Accum#1 hi-order (overflow)
0069	105-110	FAC2	BASIC floating point Accum#2: Exponent, etc (Alt
	105	ARGEXP	Exponent +128
006A	106-110		Accum#2: Mantissa
	106-109	ARGHO	Normalized mantissa of the value
	110	ARGSGN	0 positive, 128-255 negative
006F	111	ARISGN	Sign comparison, Acc#1 vs #2
			0 FAC and FAC2 have same sign
			225 FAC and FAC2 have different signs
0070	112	FACOV	Accum#1 lo-order (rounding)
0071	113-114 L/M	FBUFPT	Cassette buffer length/Series pointer
0073	115-138	CHRGET	Subroutine increment and scan (get BASIC char)
0079	121		Scan memory, BASIC byte get
007A	122-123 L/M		Basic pointer (within subroutine)
008B	139-143	RNDX	RND seed value, work area
0090	144-255		KERNAL working storage
0090	144	STATUS	Status word ST, status of I/O completion
0091	145	STKEY	Key switch PIA: STOP and RVS flags (bottom row)
			127 ($7F) cursor down/up key
			223 ($DF) comma key pressed
			239 ($EF) N key pressed
			247 ($F7) V key pressed
			251 ($FB) X key pressed
			253 ($FD) left shift key pressed
			254 ($FE) STOP key pressed
			255 ($FF) no key pressed
0092	146	SVXT	Timing constant for tape read
			resets to 0 after each bit has been read
0093	147	VERCK	Verify check flag
			0 LOAD
			1 VERIFY
0094	148	C3PO	Serial output: deferred char flag
			0 no character waiting, >0 character waiting
0095	149	BSOUR	Serial deferred character
			255 ($FF) no character, <255 waiting character
0096	150	SYNO	Tape EOT received
			0 no block recognized OR data being read
			16-126 has read at least 16 leader bits
0097	151	XSAV	Register save
0098	152	LDTND	Number of open files (not to exceed 10)
			default value 0

HEX	DEC		DESCRIPTION
0099	153	DFLTN	Input device (normally 0)
			see value chart in location 19
009A	154	DFLTO	Output (CMD) device, (normally 3: screen)
			see value chart in location 19
009B	155	PRTY	Tape character parity
			bit 0 used to calculate parity
009C	156	DPSW	Tape dipole switch; Byte-received flag
			0 waiting or in process
			1 received
009D	157	MSGFLG	KERNAL message control flag
			0 output control
			64 ($40) error messages wanted
			128 ($80) control messages wanted
			192 ($C0) error and control messages wanted
009E	158	PTRI	Tape 1 error log/char buffer, header ID out byte
			Tape SAVE: 1 relocatable
			Tape SAVE: 2 user data record
			Tape SAVE: 3 nonrelocatable
			Tape SAVE: 4 user data header
			Tape SAVE: 5 end of tape
			Tape LOAD: pass 1 error equals 2 times # of errors
			Tape header LOAD: index for filename compare
			Tape write from BASIC: character for CHROUT
009F	159	PTR2	Tape LOAD: pass 2 error log corrected
			Tape header LOAD: index into filename in buffer
00A0	160-162	TIME	Jiffy Clock (HML)
	160		High: Incremented every 18.2044 minutes
	161		Mid: Incremented every 4.26667 seconds
	162		Low: Incremented every 0.01667 second (1 jiffy)
00A3	163	PCNTR	Serial bit count/Tape input/EOI flag
			SERIAL: flag indicates last byte sent to device
			TAPE: Set to 8 and decremented each read / write
00A4	164	FIRT	Cycle count, tape dipole number, serial input byte
			SERIAL: the input byte read in LOAD or SAVE
			TAPE: 0 second half of dipole processed
			TAPE: 1 first half of dipole processed
00A5	165	CNTDN	Serial countdown, tape write/bit count
			TAPE: initializes to 9 each block, counts down
			SERIAL: initializes to 8, bits to be sent, counts down
00A6	166	BUFPNT	Pointer: tape buffer
00A7	167	INBIT	Tape Write leader / Read pass / input bit
			Tape write leader length counter
			Set to 0 before writing tape leader for header
			Set to 128 before writing leader between blocks
			Tape LOAD: 0 all blocks read

HEX	DEC		DESCRIPTION
			Tape LOAD: 1 second block loading
			Tape LOAD: 2 first block loading
			RS-232: bit 0 is temp storage for input bit
00A8	168	BITCI	Tape Write new byte/Read error / in bit
			TAPE: 0 no errors
			RS-232: input byte bit count
00A9	169	RINONE	Tape dipole balance counter, RS input flag start bit
			TAPE: 126 maximum value
			RS-232: 144 no start bit received
00AA	170	RIDATA	Tape input status flag, sync count, RS byte
			Tape LOAD: 0 waiting for first character to arrive
			Tape LOAD: 1-15 block countdown being read
			Tape LOAD: 64 valid character has arrived
			Tape LOAD: 128 first block loaded, search for next
			Tape SAVE: 128 start address is greater than end
			RS-232:byte assembly area
00AB	171	RIPRTY	Tape write lead length / Rd checksum / parity
			Set to 0 for leader between blocks
			Set to 20 before writing leader first block
			Set to 105 before writing the header leader
00AC	172-173 L/M	SAL	Pointer: tape buffer, scrolling
00AE	174-175 L/M	EAL	Tape end addresses/End of program
00B0	176	CMPO	Tape timing constant, dipole timing adjustment
00B1	177	TEMPI	Tape timing constant 2, dipole timer 2 difference
00B2	178-179 L/M	TAPEI	Pointer: start of tape buffer (LSB/MSB)
00B4	180	BITTS	Tape timer (1=enable); bit count
			TAPE: 0 rest between blocks
			TAPE: >0 ready to receive data
			RS-232: transmit bit count out, timer flag, parity
00B5	181	NXTBIT	Tape EOT/RS-232 next bit to send (>0 read)
00B6	182	RODATA	Read character error / out byte buffer
			TAPE: >0 byte read is in error
00B7	183	FNLEN	Length of current file name string
00B8	184	LA	Current logical file number
00B9	185	SA	Current secondary address, or R / W command
			0-31 valid serial devices
			0-127 non-serial devices
00BA	186	FA	Current device number
			see value chart in location 19
00BB	187-188 L/M	FNADR	Address of current file name string
00BD	189	ROPRTY	Write shift word / Read input char
00BE	190	FSBLK	Number of blocks remaining to Write / Read
			0 both copies of block are done
			1 last copy of block remains to save/load
			2 both copies of block remain to save/load

HEX	DEC		DESCRIPTION
00BF	191	MYCH	Serial word buffer
00C0	192	CASI	Tape motor interlock, manual / contr switch 0,2,4,6,10 stops the motor 8 no change 12,14 starts the motor
00C1	193-194 L/M	STAL	I/O start addresses
00C3	195-196 L/M	MEMUSS	KERNAL setup pointer
00C5	197	LSTX	Current key pressed (default 64)

0	1	12	I	26	X	39	F1	52	O
1	3	13	P	27	V	41	2	53	@
2	5	14	*	28	N	42	F	54	↑
3	7	15	ret	29	,	43	H	55	F5
4	9	17	A	30	/	44	K	56	2
5	+	18	D	31	↕	45	:	57	4
6	£	19	G	32	spc	46	=	58	6
7	del	20	J	33	Z	47	F3	59	8
8	←	21	L	34	C	48	Q	60	0
9	W	22	;	35	B	49	E	61	-
10	R	23	↔	36	M	50	T	62	ho
11	Y	24	stp	37	.	51	U	63	F7

64 no key pressed (default) see page 108

00C6	198	NDX	Number of chars in keyboard buffer 0 clear, no keys (default) 10 maximum value
00C7	199	RVS	Screen reverse flag 0 disable reverse mode (default) 18 ($12) enable reverse mode
00C8	200	INDX	Pointer: End-of-line for input
00C9	201	LXSP	Input cursor log (row) range 0 to 22
00CA	202	LXSP	Input cursor log (column) range 0 to 87
00CB	203	SFDX	Shift mode on print flag; current key press see location 197; see page 108
00CC	204	BLNSW	Cursor blink switch 0 flash >0 disable, cursor will not flash
00CD	205	BLNCT	Cursor timing countdown (normal 20 jiffies)
00CE	206	GDBLN	Character under cursor in PETSCII
00CF	207	BLNON	Cursor in blink phase 0 not reversed 1 reversed character
00D0	208	CRSW	Input from screen/from keyboard 0 input from keyboard possible values: 21, 43, 65, 87

HEX	DEC			DESCRIPTION
00D1	209-210	L/M	PNT	Pointer to screen line address
00D3	211		PNTR	Cursor column (position of cursor on current line) 0 to 87 possible range
00D4	212		QTSW	Quote mode flag 0 not within quotes (off) 1 within quotes (on)
00D5	213		LNMX	Current screen line length possible lengths: 21, 43, 65, 87
00D6	214		TBLX	Current screen row number
00D7	215		ASCII	ASCII value of last key press / checksum/buffer
00D8	216		INSRT	Insert mode flag (outstanding inserts remaining) 0 turn off insert more
00D9	217-241		LDTB1	Screen line link table
00F1	241			Save byte for screen link table
00F2	242			Screen row marker
00F3	243-244	L/M	USER	Pointer to current screen line color map area
00F5	245-246	L/M	KEYTAB	Pointer to current keyboard table
00F7	247-248	L/M	RIBUF	RS-232 pointer to start of receiving buffer
00F9	249-250	L/M	ROBUF	RS-232 Tx pointer to start of transmitting buffer
00FB	251-254		FREEKZP	Operating system free zero page space
00FF	255		BASZPT	Basic storage for floating point to ASCII conversion
0100	256-511		STACK	Stack area; 256 is bottom of stack
0100	256-266			Floating to ASCII work area (62 bytes)
0100	256-318		BAD	Tape error log
0140	320-511		BASTACK	BASIC Proc stack area ?OUT OF MEMORY error if exceeded
01FC	508-509	L/M		Chain link pointer
01FE	510-511	L/M		Line number before crunched line
0200	512-600		BUF	Basic input buffer (89 bytes) Allows 88 characters in input line, 0 at end of line
0259	601-610		LAT	Logical file table
0263	611-620		FAT	Open device number table
026D	621-630		SAT	Open secondary Address table
0277	631-640		KEYD	Keyboard buffer (10 characters)
0281	641-		MEMSTR	Start of memory pointer (LSB)
0282	642			Start of memory pointer (MSB) 16: unexpanded default value (4096 $1000) 4: +3K expanded default value (1024 $0400) 18: +8K+ expanded default value (4608 $1200) POKE642,PEEK(642)+{pages up}: POKE644,PEEK(644)-{pages down}:SYS58232
0283	643-		MEMHIGH	Top of memory, pointer to end of user RAM +1
	644			Start of memory pointer (MSB) unexpanded and 3K default values 0, 30 (7680) 64 block 1 end value (8192 $2000)

HEX	DEC		DESCRIPTION
			96 block 2 end value (16384 $4000)
			128 block 3 end value (24576 $6000)
0285	645	TIMOUT	Serial bus timeout enable flag
0286	646	COLOR	Current color code (default 6)
			bit 2–0 standard color
			0 BLK 1 WHT 2 RED 3 CYN
			4 PUR 5 GRN 6 BLU 7 YEL
			bit 3 multi-color mode flag
0287	647	GDCOL	Color under cursor
0288	648	HIBASE	Screen memory page (current location MSB)
			unexpanded and 3K default value 30 (7680)
			8K+ expanded default value 16 (4096)
0289	649	XMAX	Max size of keyboard buffer (default 10)
			0 completely disables keyboard
			15 maximum range for normal buffer (631-645)
028A	650	RPTFLG	Key repeat flag (default 0)
			60 repeat no keys
			128 repeats all keys
028B	651	KOUNT	Repeat delay speed counter
			Initialized to 6 after first delay, decremented to 0
028C	652	DELAY	Repeat delay first timer counter
			Initialized to 16, decremented to 0
			key hold puts 4 in 651 for faster repeat
028D	653	SHFLAG	Shift / C= / Control key flag (default 0)
			bit 0 shift key (1 key down)
			bit 1 Commodore key (2 key down)
			bit 2 control key (4 key down)
028E	654	LSTSHF	Last keyboard shift pattern
028F	655-656 L/M	KEYLOG	Pointer: decode logic (table address stored in 245)
0291	657	MODE	Shift mode switch
			0 enabled (default) or CHR$(9)
			128 disabled or CHR$(8)
0292	658	AUTODN	Auto scroll down flag
			0 enable (default)
			1-255 disable
0293	659	M51CTR	Pseudo 6551 / RS-232 control register
			A bit 7 stop bits: 0=1 stop bit, 1=2 stop bits
		ABBCDDDD	B bit 6-5 word length:
			00=8 bits 01=7 bits 10=6 bits 00=5 bits
			C bit 4 unused
			D bit 3-0 baud rate:
			0001=50 0010=75 0011=110 0100=134.5
			0100=150 0110=300 0111=600 1000=1200
			1001=1800 1010=2400 1011=3600 1100=4800
			rates above 3600 unavailable on VIC

HEX	DEC			DESCRIPTION		
0294	660		M51CDR	Pseudo 6551 / RS-232 command register		
				A bit 7-5 parity:		
			AAABCCCD	xx0=disabled	001=odd	011=even
				101=mark	111=space	
				B bit 4 duplex: 0=full, 1=half		
				C bit 3-1 unused		
				D bit 0 handshaking: 0=3 line, 1=X line		
0295	661-662		M51AJB	Nonstandard (Bit time / 2-100)		
0297	663		RSSTAT	RS-232 status register		
				bit 7 BREAK detected; interrupt has occurred		
				bit 6 data set not ready; modem not free		
				bit 5 unused		
				bit 4 CTS signal missing		
				bit 3 unused		
				bit 2 receiver buffer overrun		
				bit 1 framing error		
				bit 0 parity error		
0298	664		BITNUM	Number of bits to send or receive		
0299	665-666	L/M	BAUDOF	Baud rate (full) bit time divided by baud rate		
029B	667		RIDBE	RS-232 receive end pointer		
029C	668		RIDBS	RS-232 input start pointer		
029D	669		RODBS	RS-232 transmit start pointer		
029E	670		RODBE	RS-232 output end pointer		
029F	671-672		IRQTMP	Holds IRQ during tape operations		
02A1	673-767		USRVCTR	User Program indirect vectors		
0300	768-778		BVECTOR	BASIC vectors table		
0300	768-769	L/M	IERROR	Error message link		
0302	770-771	L/M	IMAIN	Basic warm start link		
0304	772-773	L/M	ICRNCH	Crunch Basic tokens link		
0306	774-775		IQPLOP	Print tokens, vectors to BASIC routines		
				POKE 774, 200 disables RUN STOP key		
				POKE 775, 171 crashes if LIST attempted		
0308	776-777		IGONE	Start new Basic code link		
030A	778-779		IEVAL	Get arithmetic element link (evaluate variables)		
030C	780-783			Register Save Area		
030C	780		SAREG	Storage for 6502 .A register		
030D	781		SXREG	Storage for 6502 .X register		
030E	782		SYREG	Storage for 6502 .Y register		
030F	783		SPREG	Storage for 6502 .P register		
0310	784-787		PG3FREE	Unused space		
0314	788-819		KVECTOR	Table of 16 KERNAL indirect vectors		
0314	788-789		CINV	Hardware (IRQ) interrupt vector (EABF)		
				POKE 788,194 disable timer and STOP key		
				POKE 788,181 enable timer and STOP key		
0316	790-791		CBINV	Break interrupt vector		

HEX	DEC		DESCRIPTION
0318	792-793	NMINV	NMI interrupt vector
031A	794-795	IOPEN	OPEN vector
031C	796-797	ICLOSE	CLOSE vector
031E	798-799	ICHKIN	Set-input vector CHKIN
0320	800-801	ICHKOUT	Set-output vector CHKOUT
0322	802-803	ICLRCH	Restore I/O vector CLRCHN
0324	804-805	IBASIN	INPUT vector CHRIN
0326	806-807	IBSOUT	Output vector CHROUT
0328	808	ISTOP	Test-STOP key vector 1
			100 disables RUN/STOP and RESTORE keys
			112 enables RUN/STOP and RESTORE keys
0329	809	ISTOP	Test-STOP key vector 2
032A	810-811	IGETIN	GET vector
032C	812-813	ICLALL	Abort I/O vector, close all files
032E	814-815	USRCMD	User vector (default BRK)
0330	816-817	ILOAD	Link to load RAM
0332	818-819	ISAVE	Link to save RAM
0334	820-827	USRCMDS	Unknown, likely unused (eight user bytes?)
033C	828-1019	TBUFFR	Cassette buffer (192 bytes)
033C	828	TPHDRID	Tape date block identifier
			1 relocatable BASIC program
			2 data block
			3 fixed address program (not relocatable)
			4 data file header
			5 logical end of tape
033D	829-830	TPHBGN	Starting address for tape LOAD
033D	829-1019	TPBLOCK	Block of 191 bytes for user data
033F	831-832	TPHNAME	Ending address of tape LOAD +1
03FC	1020-1023		Unused 4 bytes (possible user storage area)
0400	1024-4095	RAMBLK0	3K expansion RAM area (3072 bytes)
	3072-		User-defined character set when 36869 is 251
1000	4096-7679	USRPGM3	User Basic area; start of screen memory (8K+)
1000	4096-4607	SCREENX	Screen map visible when pointer is 18 (8K+)
	4096-		User-defined character set when 36869 is 252
1200	4608-8191	USRPGM8	First 3583 bytes of BASIC area on 8K+
	5120-		User-defined character set when 36869 is 253
	6144-		User-defined character set when 36869 is 254
	7168-		User-defined character set when 36869 is 255
1E00	7680-8185	SCREEN	Screen map visible for unexpanded and +3K
1FFA	8186-8191		Screen memory normally outside view by default
2000	8192-	RAMBLK1	8K expansion RAM/ROM block 1
4000	16384-	RAMBLK2	8K expansion RAM/ROM block 2
6000	24576-	RAMBLK3	8K expansion RAM/ROM block 3
8000	32768-36863		4K ROM Character Maps
8000	32768-	CASEU	Character ROM: Upper case and graphics

HEX	DEC		DESCRIPTION
			32768 start of alphabet (A-Z)
			33024 space
			33280 start of symbol graphics
8400	33792-	CASEURV	Character ROM: Reversed upper case and graphics
			33792 start of alphabet (A-Z)
			34048 space
			34304 start of symbol graphics
8800	34816-	CASEL	Character ROM: Upper and lower case
			34816 start of alphabet (A-Z)
			35090 space
			35328 start of symbol graphics
8C00	35840-	CASELRV	Character ROM: Reversed upper and lower case
			35840 start of alphabet (A-Z)
			36096 space
			36352 start of symbol graphics
9000	36864-37125		6560 Video Interface Chip
			A interlace mode (0 off, 1 on)
			B screen origin horizontal
9000	36864	ABBBBBBB	C screen origin vertical
9001	36865	CCCCCCCC	D number of video columns
9002	36866	HDDDDDDD	E number of video rows
9003	36867	GEEEEEEF	F character size (0 8x8, 1 8x16)
9004	36868	GGGGGGGG	G raster value
9005	36869	HHHHIIII	H screen memory location
9006	36870	JJJJJJJJ	I character memory location
9007	36871	KKKKKKKK	J light pen/gun horizontal
9008	36872	LLLLLLLL	K light pen/gun vertical
9009	36873	MMMMMMMM	L paddle 1
900A	36874	NRRRRRRR	M paddle 2
900B	36875	OSSSSSSS	N sound switch bass
900C	36876	PTTTTTTT	O sound switch alto
900D	36877	QUUUUUUU	P sound switch soprano
900E	36878	WWWWVVVV	Q sound switch noise
900F	36879	XXXXYZZZ	R bass frequency
see page 36			S alto frequency
			T soprano frequency
			U noise frequency
			V loudness
			W auxiliary color
			X screen color
			Y reverse screen mode
			Z border color
9000	36864	VICCR0	Horizontal centering and interlace scan
			bit 7 enable interlace scan bit (default 0)
			bit 6–0 horizontal screen position (default 5)

HEX	DEC		DESCRIPTION
9001	36865	VICCR1	Vertical picture origin
			default value 25 NTSC, 38 PAL
9002	36866	VICCR2	Number of columns
			bit 7 is bit 9 of screen memory location (see 36869)
			1 color map begins at 38400
			0 color map begins at 37888
			bit 6–0 number of screen columns
			default value 150
9003	36867	VICCR3	Number of rows and 8x16 mode
			bit 7 raster value (light pen sync with TV, 36868)
			bit 6-1 number of visible rows on screen x2 (0 to 23)
			bit 0 character size switch
			0 8x8 mode (default)
			1, 8x16 mode
			(default 46 or 176)
9004	36868	VICCR4	TV raster beam line
			low order bit combined with high order in 36867
9005	36869	VICCR5	bits 7 must be 1, video matrix
			A bits 7-4 are bits 10-12 of screen location
		AAAABBBB	1100 (192) begin at 4096 when bit 7 of 36866=0
			1100 (192) begin at 4608 when bit 7 of 36866=1
			1101 (208) begin at 5120 when bit 7 of 36866=0
			1101 (208) begin at 5632 when bit 7 of 36866=1
			1110 (224) begin at 6144 when bit 7 of 36866=0
			1110 (224) begin at 6656 when bit 7 of 36866=1
			1111 (240) begin at 7168 when bit 7 of 36866=0
			1111 (240) begin at 7680 when bit 7 of 36866=1
			B bits 0-3 start of character location (bit 13-10)
			0000 (240) begin at 32768 ($8000) Upper (default)
			0001 (241) begin at 33792 ($8400) Upper rev
			0010 (242) begin at 34816 ($8800) Lower case
			0011 (243) begin at 35840 ($8C00) Lower rev
			0100 – 1011 outside of normal use
			1100 (252) begin at 4096 ($1000) RAM
			1101 (253) begin at 5120 ($1400) RAM
			1110 (254) begin at 6144 ($1800) RAM
			1111 (255) begin at 7168 ($1C00) RAM
9006	36870	VICCR6	Horizontal position of light pen (default value 0)
9007	36871	VICCR7	Vertical position of light pen (default value 0)
9008	36872	VICCR8	Digitized value of paddle X
			0 maximum RIGHT position
			255 maximum LEFT position (default)
9009	36873	VICCR9	Digitized value of paddle Y
			0 maximum RIGHT position
			255 maximum LEFT position (default)

900A	36874	VICCRA	Frequency for oscillator 1 (low) bit 7 enable switch BASS bit 6–0 oscillator range
900B	36875	VICCRB	Frequency for oscillator 2 (medium) bit 7 enable switch ALTO bit 6–0 oscillator range
900C	36876	VICCRC	Frequency for oscillator 3 (high) bit 7 enable switch SOPRANO bit 6–0 oscillator range
900D	36877	VICCRD	Frequency of noise source bit 7 enable switch NOISE bit 6–0 oscillator range
900E	36878	VICCRE	Volume and auxiliary color bit 7–4 auxiliary color bit 3–0 volume
900F	36879	VICCRF AAAABCCC	Screen and border color register; see page 37 A bit 7–4 background color B bit 3 inverse switch C bit 2–0 border color 000 black 001 white 010 red 011 cyan 100 purple 101 green 110 blue 111 yellow 1000 org 1001 lt. org 1010 pink 1011 lt. cyn 1100 lt. pur 1101 lt. grn 1110 lt. blu 1111 lt. yel default value 27 (white screen, cyan border)
9010	36880-37125		Unused; Unreliable reflections of 36864-36868 (-16)
9110	37136-37151		6522 Versatile Interface Adapter 1
9110	37136	VIA1PB	Port B input / output register → bit 7 Data Set Ready (DSR) IN X-line → bit 6 Clear to Send (CTS) IN X-line bit 5 unused → bit 4 Data Carrier Detect (DCD) IN X-line → bit 3 Ring Indicator (RI) IN ← bit 2 Data Terminal Ready (DTR) OUT X-line ← bit 1 Request To Send (RTS) OUT X-line → bit 0 Received Data Signal (Sin) IN X-line, 3 line
9111	37137	VIA1PA1	Port A input / output register ← bit 7 serial attention out → bit 6 tape button down flag (cassette switch) → bit 5 light pen / joystick FIRE → bit 4 joystick LEFT off (JOY 2) → bit 3 joystick DOWN off (JOY 1) → bit 2 joystick UP off (JOY 0) → bit 1 serial data IN → bit 0 serial clock IN
9112	37138	VIA1DDRB	Data direction register B (also see bit 1 of 37147) 1 in bit indicates OUTPUT in port B (37136)

9113	37139	VIAIDDRA	Data direction register A (also see bit 0 of 37147)
			1 in bit indicates OUTPUT in port A (37137)
9114	37140	VIA1T1CL	Timer 1 low byte (LSB)
9115	37141	VIA1TICH	Timer 1 high byte & counter (MSB)
			setting this byte for timer 1 starts timer
9116	37142	VIA1T1LL	Timer 1 low byte (LSB)
9117	37143	VIA1T1LH	Timer 1 high byte (MSB)
9118	37144	VIA1T2CL	Timer 2 low byte (LSB)
9119	37145	VIA1T2CH	Timer 2 high byte (MSB)
911A	37146	VIA1SR	Shift register for parallel and serial conversion
			NOTE: the KERNAL does not use this shift register
			bit 7 shift out onto CB2 line
			bit 4-2 aux control register at 37147 select shift reg
			bit 2 interrupt enable and interrupt flag corresponds
911B	37147	VIA1ACR	Auxiliary control register
			A bit 7-6 timer 1 options: PB7 output (neg, square)
		AABCCCDE	00 single interval more, no PB7 output pulse
			01 free run more, no PB7 output pulse
			10 single interval more, PB7 negative pulsed
			11 single interval more, PB7 negative pulsed
			B bit 5 timer 2 options
			0 single interval timing
			1 countdown incoming PB6 pulses
			C bit 4-2 shift register options
			000 shift register disabled
			001 input CB2 shift bit 0, timer 2, output clock CB1
			010 input CB2 shift bit 0, system, output clock CB1
			011 output CB2 bit 7, timer 2, output clock CB1
			100 output CB2 bit 7, timer 2 as delay clock
			110 output CB2 bit 7, system, output clock CB1
			111 output CB2 bit 7, output external clock, in CB1
			D bit 1 Port B (location 37136 $9110)
			0 port B reflects changing values on pins
			1 port in latch mode (input CB1 interrupt occurred)
			E bit 0 Port A (location 37137 $9111)
			0 port A reflects changing values on pins
			1 port in latch mode (input CA1 interrupt occurred)
911C	37148	VIA1PCR	Peripheral control register (handshake)
			A bit 7 CB2 line control input output mode
		ABBCDDDE	0 input mode
			1 output mode
			B bit 6-5 CB2 line control
			00 IFR bit 3 high to low CB2, clears IFR
			01 IFR bit 3 high to low CB2, does not clear IFR
			10 IFR bit 3 low to high CB2, clears IFR
			11 IFR bit 3 low to high CB2, does not clear IFR

			C bit 4 CB1 line control low (0) or high (1)
			0 IFR bit 4 high to low transition of CB1 (default)
			1 IFR bit 4 low to high transition of CB1
			D bit 3-1 CA2 line control
			000 IFR bit 0 high to low CA2, clears CA2
			001 IFR bit 0 high to low CA2, does not clear CA2
			010 IFR bit 0 low to high CA2, clears CA2
			011 IFR bit 0 low to high CA2, does not clear CA2
			100 output (handshake) CA2 low
			101 output mode (pulse)
			110 output mode (manual), CA2 low
			111 output mode (manual), CA2 high
			E bit 0 CA1 line control low (0) or high (1)
			0 IFR bit 4 high to low transition of CA1
			1 IFR bit 4 low to high transition of CA1
911D	37149	VIA1IFR	Interrupt flag register (IFR) (service order 1,6,5,4)
			bit 7 IRQ status (NMI)
	NMI Interrupts		bit 6 timer 1 interrupt flag, RS-232 send
	service order		bit 5 timer 2 interrupt flag, RS-232 receive
	bit 1		bit 4 CB1 interrupt flag, RS-232 receive
	bit 6		bit 3 CB2 interrupt flag
	bit 5		bit 2 shift register flag
	bit 4		bit 1 CA1 interrupt flag
	*not by kernal routine		bit 0 CA2 interrupt flag, tape motor, if STOP, BRK
911E	37150	VIA1IER	Interrupt enable register (IER)
			bit 7 IER set / clear control (1 enable IFR)
			bit 6 timer 1 interrupt enable
			bit 5 timer 2 interrupt enable
			bit 4 CB1 interrupt enable
			bit 3 CB2 interrupt enable
			bit 2 shift register enable
			bit 1 CA1 interrupt enable
			bit 0 CA2 interrupt enable
911F	37151	VIA1PA2	Port A; mirror of 37137
9120	37152-37167		6522 Versatile Interface Adapter 2
9120	37152	VIA2PB	Port B input / output register; Key column scan
			default value 247 except when SCNKEY or UDTIM
		↔	bit 7 (127) key column 7; Joy 3, pin 4 game port
		←	bit 6 (191) key column 6
		←	bit 5 (223) key column 5
		←	bit 4 (239) key column 4
		←	bit 3 (247) key column 3; tape write, port E-5
		←	bit 2 (251) key column 2
		←	bit 1 (253) key column 1
		←	bit 0 (254) key column 0
NAIL			see value chart in 37153

9121	37153	VIA2PA1	Port A input / output register; Keyboard row scan bit 7-0 keyboard row 7-0, respectively

	7	6	5	4	3	2	1	0
← 7	F7	F5	F3	F1	↕	↔	Ret	De
← 6	HO	↑	=	R SHIFT	/	;	*	£
← 5	-	@	:	.	,	L	P	+
← 4	0	O	K	M	N	J	I	9
← 3	8	U	H	B	V	G	Y	7
← 2	6	T	F	C	X	D	R	5
← 1	4	E	S	Z	L SHIFT	A	W	3
← 0	2	Q	C=	SPC	St	Ct	←	1

column stored in 37152 ($9120)

9122	37154	VIA2DDRB	Data direction register B when a bit is set to 0, port B but is used for input 127 set direction for controller, some keys ignored 255 all lines as output (default and restore value)
9123	37155	VIA2DDRA	Data direction reg. A (default and restore value: 0) when a bit is set to 0, port A but is used for input Power-on/reset, RUN/STOP-RESTORE sets to 0
9124	37156	VIA2T1CL	Timer 1, low byte latch (LSB) used to generate the IRQ 60 times per second
9125	37157	VIA2T1CH	Timer 1, high byte latch (MSB) setting this byte of timer to 1 starts timer running
9126	37158	VIA2T1LL	Timer 1, low byte counter (LSB)
9127	37159	VIA2T1HL	Timer 1, high byte counter (MSB)
9128	37160	VIA2T2CL	Timer 2, latch (LSB):KERNAL (detect timeouts) timer used by the KERNAL for detecting timeouts
9129	37161	VIA2T2CH	Timer 2, high byte latch (MSB)
912A	37162	VIA2SR	Shift register for parallel and serial conversion
912B	37163	VIA2ACR	Auxiliary control register A bit 7-6 timer 1 options (single/free, PB7)
		AABCCCDE	B bit 5 timer 2 (0 single interval, 1 count PB6 pulse) C bit 4-2 shift register options 000 shift disabled 001 input on CB2 using timer 2 010 input on CB2 using system clock 011 output on CBs using timer 2 100 output on CB2 using timer 2 delay 110 output on CB2 using system clock 111 output on CB2 using external clock D bit 1 Port B latch enable options E bit 0 Port A latch enable options
912C	37164	VIA2PCR	Peripheral control register (handshake) A bit 7-5 CB2 line control, data out
		AAABCCCD	000 input mode IFR bit 3 hi and clear CB2 001 input mode IFR bit 3 hi and do not clear

			010 input mode IFR bit 3 lo and clear CB2
			011 input mode IFR bit 3 lo and do not clear
			100 output mode (handshake) CB2 low
			110 output mode (pulse) CB2 low
			111 output mode (manual) CB2 high
			000 shift disabled
			B bit 4 CB1 SRQ IN (0 high to low, 1 low to high)
			C bit 3-1 CA2 line control, serial clock out
			000 input mode IFR bit 0 hi and clear CA2
			001 input mode IFR bit 3 hi and do not clear
			010 input mode IFR bit 3 lo and clear CA2
			011 input mode IFR bit 3 lo and do not clear
			100 output mode (handshake) CA2 low
			110 output mode (pulse) CA2 low
			111 output mode (manual) CA2 high
			000 shift disabled
			D bit 0 CA1 line control, TAPE read line
912D	37165	VIA2IFR	Interrupt flag register (IFR)
			bit 7 IRQ occurred (0 clears all interrupts)
			bit 6 timer 1 interrupt, Normal IRQ handler (1/60s)
			bit 5 timer 2 interrupt, Tape IRQ handler, Tape write
			bit 4 CB1 transition interrupt *
			bit 3 CB2 transition interrupt *
			bit 2 shift register interrupt *
			bit 1 CA1 transition interrupt, Tape IRQ, Tape read
	*not by kernal routine		bit 0 CA2 transition interrupt
912E	37166	VIA2IER	Interrupt enable register (IER)
			bit 7 IFR enable (1) disable (1)
			bit 6 timer 1 interrupt enable (1)
			bit 5 timer 2 interrupt
			bit 4 CB1 interrupt
			bit 3 CB2 interrupt
			bit 2 shift register interrupt
			bit 1 CA1 (TAPE I/O) interrupt
			bit 0 CA2 interrupt
912F	37167	VIA2PA2	Port A output register; mirror 37153
9130	37168-		Unused I/O for future expansion block1
9400	37888-38399	COLORMA	Color RAM map for 8k+ expansion
			A bit 7-4 not used
		AAAABCCC	B bit 3 multicolor enable (1) disable default (0)
			C bit 2-0 foreground color (default value 1)
9600	38400-38911	COLORMA	Color RAM map for unexp. and +3K expansion
			bit 7-4 not used
			bit 3 multicolor enable (1) disable default (0)
			bit 2-0 foreground color (default value 1)
9800	38912-39935		I/O block 2

9C00	39936-40959		I/O block 3
A000	40960-	RAMBLK5	8K decoded for exp ROM block 4; auto start carts
A000	40960-40961		ROM cart: Vector for power-on/reset routines Auto start “hard start” location (lo-hi)
A002	40962-40963		ROM cart: Vector for NMI (RESTORE key) routines Auto start “warm start” location (lo-hi)
A004	40964-40968		Auto start code: 65,48,195,194,205 : A0CBM initialization routines to JSR when auto-start made $FD8D RAM test $FD52 set vectors $FDF9 initialize I/O, CLI $E518 initialize screen
C000	49152	COLDST	Vector to routine for cold start of BASIC (58232)
C002	49154	WARMST	Vector to routine for warm start of BASIC (58471)
CC04	49156-	CBMBASI	CBMBASIC characters in ASCII
C00C	49164-	STMDSP	Keyword vector table in token order (LSB/MSB)+1 END 51249 FOR 51010 NEXT 52510 DATA 51448 INPUT# 52133 INPUT 52519 DIM 53377 READ 52230 …and so forth
C052	49234-	FUNDSP	Function vector table in token order SGN 56377 INT 56524 ABS 56408 USR 0 FRE 54141 …and so forth
C080	49280-	OPTAB	Math operation pointer directory in token order
C080	49280		addition (ROUTINE: 55402)
C083	49283		subtraction (ROUTINE: 55379)
C086	49286		multiplication (ROUTINE: 55851)
C089	49289		division (ROUTINE: 56082)
C08C	49292		up arrow (ROUTINE: 57211)
C08F	49295		AND (ROUTINE: 53225)
C092	49298		OR (ROUTINE: 53222)
C095	49301		mondatic (ROUTINE: 57268)
C098	49304		NOT (ROUTINE: 52948)
C09B	49307		less than/ equal to / greater than (ROUTINE: 53270)
C09E	49310-	RESLST	BASIC keywords in token order and ASCII
C19D	49565-	ERRTAB	BASIC error messages in ASCII
C328	49960-	BMSGS	Pointers for 30 error messages
C364	50020-	MISCMSG	Miscellaneous error messages in ASCII
C38A	50058	SCNSTK	Routine: Find FOR and GOSUB stack entries
C3B8	50104*	MAKSPC	Open space in mem for new BASIC line / variable
C3BF	50111	MOVEBL	Routine: Move a block of memory
C3FB	50171	STKSPC	Routine: Check stack requested space available
C408	50184*	RAMSPC	Routine: Check space available for dynamic request
C435	50229	MEMERR	Set OUT OF MEMORY error message code
C437	50231*	ERROR	Routine: Look up BASIC error message
C469	50281	PRDY	Display ERROR or BREAK message
C474	50292	READY	Routine: Display READY message
C480	50304	MAIN	Routine: Receive and execute or store BASIC line
C48A	50314*		Main CHRGET entry

C49C	50332*	NEWLIN	Routine: Store or replace a BASIC program line
C52A	50474		Fix chaining, CLR and READY
C533	50483	LNKPRG	Routine: Rechain BASIC program lines
C560	50528*	GETLIN	Receive input from device
C579	50553*	CRNCH	Tokenize the BASIC line in BASIC text buffer
C613	50707*	FINLIN	Routine: Find BASIC program line from line number
C642	50754	NEW	Routine: BASIC NEW
C659	50777		Reset exec to start of program and CLR
C65E	50782	CLR	Routine: BASIC CLR (C659?)
C68E	50830	STXTPT	Routine (SYS will branch to first program line)
C69C	50844	LIST	Routine: BASIC LIST
C71A	50970	QPLOP	Routine: List detokenized BASIC keywords
C742	51010	FOR	Routine: BASIC FOR
C7AE	51118	NEWSTT	Routine: Find next BASIC statement for execution
C7E4	51172	GONE	Routine: Execute current BASIC statement
C81D	51229	RESTORE	Routine: BASIC RESTORE
C82C	51244	TSTSTOP	Test for STOP key
C82F	51247	BSTOP	Routine: BASIC STOP
C831	51249	END	Routine: BASIC END
C857	51287*	CONT	Routine: BASIC CONT
C871	51313	RUN	Routine: BASIC RUN
C883	51331	GOSUB	Routine: BASIC GOSUB
C8A0	51360	GOTO	Routine: BASIC GOTO
C8D2	51410	RETURN	Routine: BASIC RETURN
C8F8	51448	SKIPST	Routine: BASIC DATA
C8FB	51451	BUMPTP	Increment TXTPTR by amount in .Y
C906	51462	FIND2	Scan for the next BASIC statement
C928	51496	IF	Routine: BASIC IF
C93B	51515	REM	Routine: BASIC REM
C94B	51531	ON	Routine: BASIC ON
C96B	51563*	DECBIN	Convert decimal line no. into LSB-HSB format
C9A5	51621	LET	Routine: BASIC LET
C9C2	51650	LET2	Routine: Assign integer variable (LET)
C9DA	51674	LET5	Routine: Assign TI$ (LET)
C9E3	51683		Set TI$ from string
CA2C	51756	LET9	Routine: Assign string variable (LET)
CA80	51840	PRINTN	Routine: BASIC PRINT#
CA86	51846	CMD	Routine: BASIC CMD
CA9A	51866-51871	PRT1	Instructions for PRINT routine
CAA0	51872	PRINT	Routine: BASIC PRINT
CAD3	51923*		Send RETURN, linefeed if in screen mode
CAD7	51927*		Send RETURN, linefeed
CAE8	51944	PRT6	Tabs to column when comma is used
CAF8	51960	PRT7	Routine: BASIC TAB and SPC
CB1E	51998*	PRTSTR	Print string pointed to by .A .Y with cR return
CB21	52001		Print string pointed to by (FAC +3) until zero byte

CB24	52004*		Print string (point $22,23) of length .A
CB3B	52027	PRTOS	PRINT routine for special chars (space, right, ?)
CB45	52037		PRINT “?”
CB47	52039		PRINT char (output .A to device)
CB4D	52045	IGRERR	Error msg format routine for GET, INPUT, READ
CB7B	52091	GET	Routine: BASIC GET
CBA5	52133	INPUTN	Routine: BASIC INPUT#
CBBF	52159	INPUT	Routine: BASIC INPUT
CC06	52230	READ	Routine: Scan for next DATA item for READ
CCFC	52476-52509	EXTRA	INPUT error messages in ASCII
CD1E	52510	NEXT	Routine: BASIC NEXT
CD8A	52618-	TYPCHK	Four routines to check variable type
CD9E	52638*	FRMEVL	Master routine for formula and expression
CE83	52867	EVAL	Evaluate a single term of an expression
CEA8	52904-52908	PIVAL	Value of PI
CED4	52948		Routine: BASIC NOT
CEF1	52977	PAREXP	Evaluation within parenthesis
CEF7	52983*	RPACHK	Check for close parenthesis
CEFA	52986*	LPACHK	Check for open parenthesis
CEFD	52989	COMCHK	Check for comma in syntax
CEFF	52991*	SYNCHR	Syntax check for specific character from CHRGET
CF08	53000	SYNERR	Cause SYNTAX ERROR message
CF0D	53005	FACT10	Set index for nomadic minus, FACT10
CF14	53012	VARRANG	Check range of variable
CF28	53032	FACT12	Get variable name and type, check for null string
CF48	53064		Convert TI to ASCII string and set (FAC +3) to string
CFA7	53159	FACT17	Invoke function FACT17
CFE6	53222	ORR	Routine: BASIC OR
CFE9	53225	ANDD	BASIC AND if .Y=0 ($00); OR if .Y=255($FF)
D016	53270	COMPAR	Routine: Compare numbers or strings: < = >
D02E	53294	CMPST	Compare strings
D081	53377	DIM	Routine: BASIC DIM
D08B	53387	EVLVAR	Evaluate / create a variable, return address
D0E7	53479*	FNDVAR	Locate variable, jump to make var if not found
D113	53523	CHRTST	Check if ASCII character is alphabetic
D11D	53533	MAKVAR	Routine: Create a new variable
D185	53637*	RETVP	Return the address of found or created variable
D194	53652	ARYHED	Calculate the length of an array
D1A5	53669	MAXINT	Max length of integer value in floating point: 32768
D1AA	53674	INTIDX	Convert float point to 2-byte integer in A and Y
D1B2	53682	GETSUB	Convert an expression to an int number if positive
D1BF	53695	MAKINT	Convert floating point to signed integer
D1D1	53713	ARY	Locate an array, create if not found
D245	53829	BADSUB	Display BAD SUBSCRIPT message
D248	53832	ILQUAN	Display ILLEGAL QUANTITY message
D24D	53837	ARY2	Checks if subscript is within the range

D261	53857	ARY6	Routine: Create an array
D2EA	53994	ARY14	Locate a specific array element
D34C	54092	M16	Calculate multidimensional array size
D37D	54141	FRE	Routine: BASIC FRE (free memory calculation)
D391	54161*	MAKFP	Convert Y and A integer to floating point
D39E	54174	POS	Routine: BASIC POS
D3A2	54178		Convert .Y to floating pt in FAC (0-255)
D3A6	54182	NODIRM	Check if statement is entered in direct mode
D3AE	54190	UNDEF	Display UNDEF'D FUNCTION message
D3B3	54195	DEF	Routine: BASIC DEF
D3E1	54241	FN	Check DEF FN and FN syntax
D3F4	54260	EVALFN	Routine: BASIC FN
D44F	54351	EVFN3	Store DEF FN values into stack
D465	54373	STR	Routine: BASIC STR$
D475	54389	ALC1	Calculate new string length and vector
D487	54407	MAKSTR	Scan and set up a string
D4F4	54516	ALCSPC	Allocate memory for a string
D526	54566	GRBCOL	Garbage collection for new or altrd string GRBCOL
D5BD	54717-54844	GCOL13	String garbage routines
D606	54790	COLECT	Garbage collection for string
D63D	54845	ADDSTR	BASIC add to string, check for string too long
D67A	54906	XFERSTR	Routine: Move string in memory
D6A3	54947	DELST	Discard a temporary string
D6DB	55003	DELTSD	Routine to delete temporary string stack
D6EC	55020	CHR	Routine: BASIC CHR$
D700	55040	LEFT	Routine: BASIC LEFT$
D72C	55084	RIGHT	Routine: BASIC RIGHT$
D737	55095	MID	Routine: BASIC MID$
D761	55137	FINLMR	Obtain string param for LEFT$, MID$, RIGHT$
D77C	55164	LEN	Routine: BASIC LEN, get the length param of string
D782	55170	GSINFO	Get string information
D78B	55179	ASC	Routine: BASIC ASC
D79B	55195	GETBYT	Get a one-byte parameter 0 to 255 in .X
D79E	55086		Get Acc#1 LSB to .X register
D7AD	55213	VAL	Routine: BASIC VAL
D7B5	55221		Evaluate string (VAL)
D7B9	55225		Evaluate string from X,Y (above +4)
D7EB	55275	GETAD	Get two-byte address for POKE and WAIT
D7F7	55287	MAKADR	Convert floating point into two-byte address integer
D80D	55309	PEEK	Routine: BASIC PEEK
D824	55332	POKE	Routine: BASIC POKE
D82D	55341	WAIT	Routine: BASIC WAIT
D849	55369	ADD05	Routine: round number by adding .5
D850	55376	LAMIN	SUBTRACT mem contents from floating point
D853	55379	SUB	BASIC – (subtract)
D862	55394	PLUS1	Routine for processing exponents

D867	55399*	LAPLUS	ADD memory contents to floating point accum.
D86A	55402	PLUS	BASIC + (add)
D8A7	55463	PLUS6	Adjust exponent to negative
D8F7	55543	ZERFAC	Set FAC to zero and sign to positive
D8FE	55550	NORMLZ	Add fractions and normalize result
D947	55623	COMFAC	Complement FAC by reversing bits with EOR +1
D97E	55678	OVERFL	Display OVERFLOW message
D983	55683	ASRRES	Shift exponents to equal prior to math function
D9BC	55740	FPC1	Constant of one for floating point accumulator
D9C1	55745	LOGCON	Constants for LOG function
D9EA	55786	LOG	Routine: BASIC LOG
DA28	55848*	TIMES	BASIC * (MULTIPLY)
DA59	55897	TIMES3	Multiply a byte subroutine
DA8C	55948	LODARG	Move floating point mem loc to FAC2
DAB7	55991	MULDIV	Add exponents pf FAC to FAC2
DAE2	56034*	MULTEN	Multiply FAC by 10
DAF9	56057	FPCTEN	+10 floating point constant
DAFE	56062	DIVTEN	DIVIDE FAC by 10
DB07	56071		Divide (AFAC) by MFLPT (A/Y), sign (X), ans (FAC)
DB0F	56079	LADIV	Move floating point in memory to FAC2
DB12	56082	DIVIDE	BASIC / (divide)
DBA2	56226*	LODFAC	Move floating point in memory to FAC
DBC7	56263	FACTF2	Move FAC2 to memory ($5C-60)
DBCA	56266	FACTF1	Move FAC to memory ($57-5B)
DBD0	56272	FACTFP	Move FAC in variable to memory ($49,4A)
DBD4	56276	STORFAC	Perform move of FAC to memory
DBD7	56279*		Copy Acc#1 to (X,Y) location
DBFC	56316	ATOF	Transfer FAC2 to FAC
DC0C	56332	RFTOA	Move FAC to FAC2 with rounding
DC0F	56335	FTOA	Move FAC to FAC2 without rounding
DC1B	56347	ROUND	Round FAC by adjusting the rounding byte
DC2B	56363	SGNFAC	Test the sign of FAC
DC39	56377	SGN	Routine: BASIC SGN
DC3C	56380	INTFP	Convert sign to 0 or -1
DC44	56388	INTFP1	Convert 2byte integer to floating point in FAC
DC58	56408	ABS	Routine: BASIC ABS
DC5B	56411	CMPFAC	Compare FAC to memory
DC9B	56475	FPINT	Convert FAC floating point to signed integer
DCCC	56524	INT	Routine: BASIC INT
DCE9	56553	FILFAC	Convert an ASCII string to a floating point number
DCF3	56563	ASCFLT	Convert ASCII string to floating point no. in FAC
DD7E	56702	ASC18	Add .A to FAC
DDB3	56755	FPC12	String to floating point conversion constants
DDC2	56770	PRTIN	Issue message IN
DDCD	56781*	PRTFIX	Routine: Display decimal number; Print integer in.X
DDD7	56791		Print FAC as ASCII string

DDDD	56797*	FLTASC	Routine: Convert FAC to ASCII string (or TI$)
DF11	57105	FLP05	Constant 0.5 for rounding and SQR
DF16	57110	FLTCON	Powers of 10 table in 4byte fixed integer
DF3A	57146	HMSCON	Constants for TI$ division conversion
DF71	57201	SQR	Routine: BASIC SQR
DF7B	57211	EXPONT	BASIC power (up arrow),exponent
DFB4	57268	NEGFAC	BASIC monadic
DFBF	57279	EXPCON	Table for EXP in floating point format
DFED	57325	EXP	Routine: BASIC EXP
E000	57344-65535		8K KERNAL ROM (BASIC spillover -58527)
E040	57408	SEREVL	Series evaluation subroutine
E056	57430	SER2	Math series evaluation routine
E08A	57482	RNDC1	Table of constants for RND
E092	57490??	COLECT	Garbage collection for string
E094	57492	RND	Routine: BASIC RND
E0F6	57590	PATCHBA	BASIC patch routines (CLR when RS-232 opened)
E127	57639	SYSTEM	BASIC SYS
E153	57683	SAVE	Routine: BASIC SAVE
E162	57698	BVERIF	Routine: BASIC VERIFY
E165	57701	BLOAD	Routine: BASIC LOAD
E1BB	57787	FOPEN	Routine: BASIC OPEN
E1C4	57796	FCLOSE	Routine: BASIC CLOSE
E1D1	57809	PARSL	Set parameters for SAVE, VERIFY, and LOAD
E203	57859	IFCHRG	Check for more characters in current statement
E20B	57867	SKPCOM	Skip any comma in parameters being scanned
E20E	57870	CHRERR	Check for parameter after comma
E216	57878	PAROC	Routine: set parameters for OPEN and CLOSE
E261	57953	COS	Routine: BASIC COS
E268	57960	SIN	Routine: BASIC SIN
E2B1	58033	TAN	Routine: BASIC TAN
E2DD	58077	FPC20	Trigonometric constants for COS, SIN, and TAN
E30B	58123	ATN	Routine: BASIC ATN
E33B	58171	ATNCON	Constants for ATN evaluation
E378	58232	COLDBA	Routine: Cold start (SYS 58232 restarts BASIC) calls 58459 INITVCTRS, initialize vectors at 768- calls 58276 INITBA, initialize CHRGET, zero page calls 58372 FREMSG, displays CBM messages jumps to 50754, NEW routine
E387	58247	CGIMAG	CHRGET rout, RND seed copied to zero RAM
E3A4	58276	INITBA	Initialize BASIC: restore page zero pointers
E404	58372	FREMSG	Display routine for startup; calculates free memory
E429	58409-	CBMMSG	Vic BASIC start up screen in ASCII
E44F	58447	BASVCTR	6 BASIC vectors copied to 768
E45B	58459	INITVCTR	Copy BASIC vectors from ROM to RAM
E467	58471	WARMBAS	Routine: Warm start BASIC
E476	58486	PATCHER	Program patch area

E4A0	58528-65535		KERNAL ROM
E4A0	58528	SEROUT1	Serial output a 1
E4A9	58537	SEROUT0	Serial output a 0
E4B2	58546	SERGET	Serial get input bit from VIA1 and stabilize
E4BC	58556	PATCHES	Program patch area
E4C1	58561		Transfer 195-196 and display LOAD/VERIFY msg
E4CF	58575		Addendum to close tape
E4CF	58624	IOBASE	Retrieve address of I/O memory page
E505	58629	SCRN	Max screen columns and rows; comp check
E50A	58634	PLOT	Read or set current cursor column and line
E518	58648	INITSK	Initialize 6550 VIC chip, screen, and pointers
E55F	58719	CLSR	Clear the screen
E581	58753	HOME	Move cursor to home position
E587	58759	SETSLINK	Reset screen line link table pointers
E5B5	58805	UNUSDNM	NMI entry for restore key
E5BB	58811	SETIODEF	Reset device numbers to defaults
E5C3	58819	INITVIC	Reset VIC chip registers from table at 60900
E5CF	58831	LP2	Get a char from keyboard buff and shift queue
E5E5	58853	GETQUE	Wait for char in key buffer
E619	58905	GET2RTN	Purge the keyboard buffer by routine at 58959
E64F	58959	GETSCRN	Get INPUT from screen
E6B8	59064	QUOTECK	Test for quotes and set flag
E6C5	59077	SETCHAR	Set up display of character on screen
E6EA	59114	SCROLL	Move cursor on screen, add line, scroll
E72D	59181	RETREAT	Mover cursor to previous logical line and column
E742	59202*	SCRNOUT	Handle characters going to the screen
E8C3	59587	NEXTLINE	Mover cursor to next screen line
E8D8	59608	RTRN	Carriage return, turns off quote and revers, etc.
E8E8	59624	BACKUP	Move cursor to end of previous line
E8FA	59642	FORWARD	Mover cursor to start of next line
E912	59666	COLORSE	Set foreground color
E921	59681	COLORTB	Color code key table
E929	59689	CNVRTCD	Code conversion table
E975	59765	SCRL	Routine: Scroll the screen
E9EE	59886	OPENLIN	Routine: Open a blank line
EA56	59990	MOVLINE	Routine: Move screen line
EA6E	60014	SETADDR	Address of screen line and color set in memory
EA7E	60030	LINPTR	Set pointer to address of start of screen line
EA8D	60045	CLRALINE	Routine: Blank out a screen line
EAA1	60065	SYNPRT	Sync color to byte and store char on screen
EAAA	60074	PUTSCRN	Store character on screen
EAB2	60082	COLORSY	Address of color map byte for screen map byte
EABF	60095	IRQ	IRQ interrupt handler
EB1E	60190	SCNKEY	Scan keyboard for press using 6522 VIA2
EBDC	60380	SETKEYS	Set keyboard decode table address in 245-6
EC46	60486	KEYVCTR	Keyboard decode table addresses

EC5E	60510	NORMKEY	Table for decoding unshifted keys into ASCII
EC9F	60575	SHFTKEY	Table for decoding shifted keys into ASCII
ECE0	60640	LOGOKEY	Table for decoding C= shifted keys into ASCII
ED21	60705	CHARSET	Set uppercase / graphics character set
ED30	60720	GRAPHMO	Set graphic mode specified by certain ASCII chars
ED5B	60763	WRAPLIN	Wrap line on screen
ED69	60777	WHATKEY	Keyboard decoding table (unused?)
EDA3	60835	CTRLKEY	Table for decoding control shifted keys into ASCII
EDE4	60900	VICINIT	Initial values for video interface chip registers
EDF4	60916	RUNTB	Shift RUN keys put LOAD and RUN in buffer
EDFD	60925	LDTB2	Screen line link table LSB of lines in screen map
EE14	60948*	TALK	Instruct the serial device to talk to computer
EE17	60951*	LISTEN	Instruct the serial device to listen to computer
EE1C	60956	LISTI	Prepare to send serial command
EE49	61001	SRSEND	Send command or data to serial devices
EEB4	61108	SRBAD	Set ST for timeout or DEVICE NOT PRESENT
EEC0	61220	SECOND	Send secondary address after listen command
EEC5	61125	SCATN	Serial: clear attention
EECE	61134	TKSA	Serial: send secondary address after talk command
EEE4	61156*	CIOUT	Send byte on serial bus
EEF6	61174	UNTLK	Serial: send untalk command to serial devices
EF04	61188	UNLSN	Send unlisten command to serial devices
EF19	61209	ACPTR	Receive byte from serial device
EF84	61316	SRCLKHI	Serial: set clock line high
EF84	61325	SRCLKLO	Serial: set clock line low
EF96	61334	WAITABIT	Delay one millisecond
EFA3	61347	RSNXTBIT	RS-232: send next bit
EFBF	61375	RSPRTY	RS-232:calculate parity and stop bits value
EFE8	61416	RSSTOPS	RS-232: transmit stop bits
EFEE	61422	RSNXTBY	RS-232: prepare next byte to be sent from buffer
F016	61462	RSMISSN	RS-232: clear to send OR data set ready missing
F027	61479	RSCPTBIT	RS-232: compute desired word length bit count
F036	61494	RSINBIT	RS-232: receive an input bit NMI
F04B	61515	RSSTPBIT	RS-232: determine if all stop bits received
F05B	61531	RSPREPIN	RS-232: prepare to receive the next input byte
F068	61544	RSSTRBIT	RS-232: check for start bit in receive mode
F06F	61551	RSINBYTE	RS-232: put constructed byte into receive buffer
F09B	61579	RSINPRTY	RS-232: parity check for the input byte
F09D	61597	RSPRTYE	RS-232: parity error on input byte
F0A2	61602	RSOVERU	RS-232: buffer overrun on input byte
F0A5	61605	RSBREAK	RS-232: break detected on input
F0A8	61608	RSFRAME	RS-232: framing error on input
F0AA	61610	RSINERR	RS-232: set input error status and continue
F0B9	61625	RSDVCER	RS-232: ILLEGAL DEVICE message
F0BC	61628	RSOPNOU	RS-232: open channel for output
FOED	61677	RSOUTSA	RS-232: store character in transmit buffer

F102	61698	RSPREPO	RS-232: set up NMI for transmission
F116	61718	RSOPNIN	RS-232: open channel for input
F14F	61775	RSNXTIN	RS-232: receive next character from buffer
F160	61792	RSPAUSE	RS-232: check that serial and tape are idle
F174	61812	KMSGTBL	Table of Kernal messages
F1E2	61922	SPMSG	Display LOADING or VERIFYING messages
F1E6	61926*	KMSGSH	Print Kernal control messages
F1F5	61941	GETIN	Routing routine for obtaining character input data
F20E	61966	CHRIN	Input characters from current input device
F230	62000	CHRINTP	Obtain byte from tape buffer
F250	62032	CHRINTP2	Load .A with next tape character
F264	62052	CHRINSR	Obtain byte from the serial line
F26F	62063	CHRINRS	Obtain byte from RS-232
F27A	62074	CHROUT	Output character to current output device
F290	62096	CHROUTT	Output a character to tape
F2C7	62151	CHKIN	Open .X file number channel for input
F309	62217	CHKOUT	Open .X file number channel for output
F34A	62282	CLOSE	Close logical file number in .A
F3CF	62415	FNDFLNO	Find file number (.X) in file table at 601($259)
F3DF	62431	SETFLCH	Set file characteristics of file (.X) into 184-186
F3EF	62447	CLALL	Abort all open files with no close
F3F3	62451	CLRCHN	Abort all open channels
F40A	62474	OPEN	Open logical file, file number in 184
F495	62613	SERNAME	Send secondary address and filename to serial
F4C7	62663	OPENRS	Open RS-232 device
F542	62786	LOAD	Load or verify to RAM from device (number in 186)
F55C	62812	LOADSER	Load and verify RAM from a serial device
F5D1	62929	LOADTP	Load or verify RAM from tape
F647	63047	SRCHING	Display SEARCHING message for tape
F659	63065	FILENAME	Routine: Display filename
F66A	63082	LDVRMSG	Display LOADING or VERIFYING message
F675	63093	SAVE	Save RAM to device number specified in 186
F692	63122	SAVESER	Save RAM to serial device
F6F1	63217	SAVETP	Save RAM to tape
F728	63272	SAVING	Display SAVING message
F734	63284	UDTIM	Increment jiffy clock (160-162)
F760	63328	RDTIM	Put jiffy clock from 160-2 into .Y,.X,and.A
F767	63335	SETTIM	Set time into jiffy clock 160-2 from .Y,.X,and.A
F770	63344	STOP	Check for the STOP key, purge key buffer
F77E	63358	FILEMSG	I/O error file message handler
F7AF	53407	FAH	Find next tape header, .X (header ID number)
F7E7	63463	TAPEH	Build an output tape header in tape buffer
F84D	63565	TPBUFA	Load tape buffer address from 178-179 into .X, .Y
F854	63572	LDAD1	Set load/save start and end pointers to tape buffer
F867	63591	FNDHDR	Find the tape header for specified filename or next
F88A	63626	JTP20	Increment tape buffer character counter

F894	63636	CSTEL	Display PRESS PLAY ON TAPE message
F8AB	63659	CSIO	Check tape play/rewind/forward button status
F8B7	63671	CSTE2	Display PRESS RECORD & PLAY ON TAPE
F8C0	63680	RDTPBLK	Initiate tape header read
F8C9	63689	RBLK	Read blocks from tape
F8E3	63715	WBLK	Write blocks to tape
F8F4	63732	TAPE	Common tape read/write, start tape operations
F94B	63819	TSTOP	Check for tape STOP key
F95D	63837	STTI	Set time limit for tape dipole
F98E	63886	READT	Read tape data bits into location 191 ($BF)
FAAD	64173	TPSTORE	Determine to store the input character from tape
FBD2	64466	RD300	Reset tape read pointer
FBDB	64475	NEWCH	New tape character setup
FBEA	64490	TPTOGLE	Toggle tape write line to invert output signal
FC06	64518	BLKEND	End of block write processing
FC08	64523	WRITE	IRQ driven tape data write
FC95	64661	WRTNI	block leader write
FCA8	64680	WRTZ	IRQ driven tape leader write
FCCF	64719	TNIF	Restore IRQ vector, call TNOFF
FCF6	64758	BSIV	Reset current IRQ vector
FD08	64776	TNOFF	Routine: Stop tape motor
FD11	64776	VPRTY	Compare current to end of load/save pointers
FD1B	64785	WRT62	Increment current load/save pointer
FD22	64802	START	Power on / reset (SYS 64802); check for cartridge
			SYS 64812 will skip the auto start cartridge test
FD3F	64831	CHKAUTO	Check for auto starting program at 40960(A000)
FD4D	64845	AOCBM	$41,30,C3,C2,CD (“A0CBM”)
FD52	64850	RESTOR	Reset RAM vectors to defaults
FD57	64855	VECTOR	Read or set system RAM vectors
FD6D	64877	VECTORS	Default system vector address storage table
FD8D	65909	INITMEM	Initialize system memory
FDF1	65009	IRQVCTRS	IRQ vectors table
FDF9	65017	INITVIA	Initialize the 6522 VIA registers
FE49	65097	SETNAM	Filename pointer and length stored to .X, .Y, .A
FE50	65104	SETLFS	Set current file number, device, and secondary add
FE57	65111	READST	Reset RS-232 status. Branch to 65128 for non
FE66	65126	SETMSG	Set byte to enable/disable Kernal message display
FE68	65128	READIOST	Load .A with the non-RS-232 I/O status ST
FE6A	65130	ORIOST	OR .A with contents of 144 ($90) ST and store
FE6F	65135	SETTMO	Set timeout value for IEEE-488
FE73	65139	MEMTOP	Read or set the top of memory pointer
FE82	65154	MEMBOT	Read or set the bottom of memory pointer
FE91	65169	TSTMEM	Test a memory location
FEA9	65193	NMI	NMI handler routine
FED2	65234	BREAK	Break interrupt entry
FEDE	65246	RSNMI	RS-232: NMI sequences

FF56	65366	RTI	Restore 6502 registers from stack, return interrupt
FF5C	65372	BAUDTBL	RS-232 VIA timer 2 values for baud rate table
FF72	65394	IRQROUT	IRQ routine initial 6502 entry point
FF84	65412	VECTOR	Read / set vectored I/O
FF85	65413	C4FFS	Five unused bytes (255)
FF87	65415*	RESTOR	Restore default I/O vectors
FF8A	65418*	CRESTOR	Jump to 64850 ($5D52) RESTOR
FF8D	65421*	CVECTOR	Jump to 64855 ($FF8D) VECTOR
FF90	65424*	CSETMSG	Jump to 65126 ($FE66) SETMSG
FF93	65427*	CSECOND	Send secondary address after listen
FF96	65430*	CTKSA	Send secondary address after talk
FF99	65433*	CMEMTOP	Read / set the top of memory
FF9C	65436*	CMEMBOT	Read / set the bottom of memory
FF9F	65439*	CSCNKEY	Scan keyboard
FFA2	65442*	CSETTMO	Set timeout on serial bus
FFA5	65445*	CACPTR	Input byte from serial port
FFA8	65448*	CCIOUT	Output byte to serial port
FFAB	65451*	CUNTALK	Command serial bus to untalk
FFAE	65454	CUNLSN	Command serial bus to unlisten
FFB1	65457*	CLISTEN	Listen to serial bus
FFB4	65460*	CTALK	Command serial bus device to talk
FFB7	65463*	CREADST	Read I/O status word
FFBA	65466*	CSETLFS	Set logical, first, second addresses
FFBD	65469*	CSETNAM	Set filename
FFC0	65472*	COPEN	Open a logical file
FFC3	65475*	CCLOSE	Close specified logical file
FFC6	65478*	CHKIN	Set Input channel
FFC9	65481*	CHKOUT	Set Output channel
FFCC	65484*	CLRCHN	Close/clear input and output channels
FFCF	65487*	CHRIN	Input character from channel
FFD2	65490*	CHROUT	Output character to channel
FFD5	65493*	LOAD	Load RAM from a device
FFD8	65496*	SAVE	Save RAM to device
FFDB	65499*	SETTIM	Set real time clock
FFDE	65502*	RDTIM	Read real time clock
FFE1	65505*	STOP	Test Stop key
FFE4	65508*	IGETIN	Indirect to get input from keyboard
FFE7	65511*	CLALL	Close all channels and files
FFE8	65512	GETIN	Get character from keyboard buffer
FFEA	65514*	UDTIM	Increment real time clock
FFED	65517*	SCREEN	Return XY organization of screen
FFF0	65520*	PLOT	Read / set XY cursor position
FFF3	65523*	IOBASE	Returns base address of I/O devices
FFFA	65530-65531	VCTRNMI	B NMI
FFFC	65532-65533	VCTRRST	D RESET
FFFE	65534-65535	VCTRIRQ	F IRQ

ALPHABETICAL CROSS REFERENCE TO MEMORY MAP LABELS

ABS	56408	BREAK	65234	CLALL	62447	CTKSA	65430
ACPTR	61209	BSIV	64758	CLALL	65511	CTRLKEY	60835
ADD05	55369	BSOUR	149	CLISTEN	65457	CUNLSN	65454
ADDPRC	1	BSTOP	51247	CLOSE	62282	CUNTALK	65451
ADDSTR	54845	BUF	512	CLR	50782	CURLIN	57
ADRAY1	3	BUFPNT	166	CLRALINE	60045	CVECTOR	65421
ADRAY2	5	BUMPTP	51451	CLRCHN	62451	DATLIN	63
ALC1	54389	BVECTOR	768	CLRCHN	65484	DATPTR	65
ALCSPC	54516	BVERIF	57698	CLSR	58719	DECBIN	51563
ANDD	53225	C3PO	148	CMD	51846	DEF	54195
AOCBM	64845	C4FFS	65413	CMEMBOT	65436	DEFPNT	78
ARISGN	111	CACPTR	65445	CMEMTOP	65433	DELAY	652
ARY	53713	CASELRV	35840	CMPFAC	56411	DELST	54947
ARY14	53994	CASEU	32768	CMPO	176	DELTSD	55003
ARY2	53837	CASEURV	33792	CMPST	53294	DFLTN	153
ARY6	53857	CASI	192	CNTDN	165	DFLTO	154
ARYHED	53652	CBINV	790	CNVRTCD	59689	DIM	53377
ARYTAB	47	CBMBASI	49156	COLDBA	58232	DIMFLG	12
ASC	55179	CBMMSG	58409	COLDST	49152	DIVIDE	56082
ASC18	56702	CCIOUT	65448	COLECT	57490	DIVTEN	56062
ASCFLT	56563	CCLOSE	65475	COLOR	646	DPSW	156
ASCII	215	CGIMAG	58247	COLORMA	37888	DSCPTN	80
ASRRES	55683	CHANNL	19	COLORSE	59666	EAL	174
ATN	58123	CHARAC	7	COLORSY	60082	END	51249
ATNCON	58171	CHARSET	60705	COMCHK	52989	ENDCHR	8
ATOF	56316	CHKAUTO	64831	COMFAC	55623	ERROR	50231
AUTODN	658	CHKIN	62151	COMPAR	53270	ERRTAB	49565
BACKUP	59624	CHKIN	65478	CONT	51287	EVAL	52867
BAD	256	CHKOUT	62217	COPEN	65472	EVALFN	54260
BADSUB	53829	CHKOUT	65481	COS	57953	EVFN3	54351
BASTACK	320	CHR	55020	COUNT	11	EVLVAR	53387
BASVCTR	58447	CHRERR	57870	CREADST	65463	EXP	57325
BASZPT	255	CHRGET	115	CRESTOR	65418	EXPCON	57279
BAUDOF	665	CHRIN	61966	CRNCH	50553	EXPONT	57211
BAUDTBL	65372	CHRIN	65487	CRSW	208	EXTRA	52476
BITCI	168	CHRINRS	62063	CSCNKEY	65439	FA	186
BITNUM	664	CHRINSR	62052	CSECOND	65427	FAC2	105
BITS	104	CHRINTP	62000	CSETLFS	65466	FACOV	112
BITTS	180	CHRINTP2	62032	CSETMSG	65424	FACT10	53005
BLKEND	64518	CHROUT	62074	CSETNAM	65469	FACT12	53032
BLNCT	205	CHROUT	65490	CSETTMO	65442	FACT17	53159
BLNON	207	CHROUTT	62096	CSIO	63659	FACTF1	56266
BLNSW	204	CHRTST	53523	CSTE2	63671	FACTF2	56263
BLOAD	57701	CINV	788	CSTEL	63636	FACTFP	56272
BMSGS	49960	CIOUT	61156	CTALK	65460	FAH	53407

ALPHABETICAL CROSS REFERENCE TO MEMORY MAP LABELS

FAT	611	GETBYT	55195	INPPTR	67	LINNUM	20
FBUFPT	113	GETIN	61941	INPUT	52159	LINPTR	60030
FCLOSE	57796	GETIN	65512	INPUTN	52133	LIST	50844
FILEMSG	63358	GETLIN	50528	INSRT	216	LISTEN	60951
FILENAME	63065	GETQUE	58853	INT	56524	LISTI	60956
FILFAC	56553	GETSCRN	58959	INTFLG	14	LNKPRG	50483
FIND2	51462	GETSUB	53682	INTFP	56380	LNMX	213
FINLIN	50707	GONE	51172	INTFP1	56388	LOAD	62786
FINLMR	55137	GOSUB	51331	INTIDX	53674	LOAD	65493
FIRT	164	GOTO	51360	IOBASE	58624	LOADSER	62812
FLP05	57105	GRAPHMO	60720	IOBASE	65523	LOADTP	62929
FLTASC	56797	GRBCOL	54566	IOPEN	794	LODARG	55948
FLTCON	57110	GSINFO	55170	IQPLOP	774	LODFAC	56226
FN	54241	HIBASE	648	IRQ	60095	LOG	55786
FNADR	187	HMSCON	57146	IRQROUT	65394	LOGCON	55745
FNDFLNO	62415	HOME	58753	IRQTMP	671	LOGOKEY	60640
FNDHDR	63591	IBASIN	804	IRQVCTRS	65009	LP2	58831
FNDVAR	53479	IBSOUT	806	ISAVE	818	LPACHK	52986
FNLEN	183	ICHKIN	798	ISTOP	808	LSTSHF	654
FOPEN	57787	ICHKOUT	800	JMPER	84	LSTX	197
FOR	51010	ICLALL	812	JTP20	63626	LXSP	201
FORPNT	73	ICLOSE	796	KEYD	631	M16	54092
FORWARD	59642	ICLRCH	802	KEYLOG	655	M51AJB	661
FOUR6	83	ICRNCH	772	KEYTAB	245	M51CDR	660
FPC1	55740	IERROR	768	KEYVCTR	60486	M51CTR	659
FPC12	56755	IEVAL	778	KMSGSH	61926	MAIN	50304
FPC20	58077	IF	51496	KMSGTBL	61812	MAKADR	55287
FPCTEN	56057	IFCHRG	57859	KOUNT	651	MAKFP	54161
FPINT	56475	IGETIN	810	LA	184	MAKINT	53695
FRE	54141	IGETIN	65508	LADIV	56079	MAKSPC	50104
FREEKZP	251	IGONE	776	LAMIN	55376	MAKSTR	54407
FREMSG	58372	IGRERR	52045	LAPLUS	55399	MAKVAR	53533
FRESPC	53	ILOAD	816	LASTPT	23	MAXINT	53669
FRETOP	51	ILQUAN	53832	LAT	601	MEMBOT	65154
FRMEVL	52638	IMAIN	770	LDAD1	63572	MEMERR	50229
FSBLK	190	INBIT	167	LDTB1	217	MEMHIGH	643
FTOA	56335	INDEX	34	LDTB2	60925	MEMSIZ	55
FUNDSP	49234	INDX	200	LDTND	152	MEMSTR	641
GARBFL	15	INITBA	58276	LDVRMSG	63082	MEMTOP	65139
GCOL13	54717	INITMEM	65909	LEFT	55040	MEMUSS	195
GDBLN	206	INITSK	58648	LEN	55164	MID	55095
GDCOL	647	INITVCTR	58459	LET	51621	MISCMSG	50020
GET	52091	INITVIA	65017	LET2	51650	MODE	657
GET2RTN	58905	INITVIC	58819	LET5	51674	MOVEBL	50111
GETAD	55275	INPFLG	17	LET9	51756	MOVLINE	59990

ALPHABETICAL CROSS REFERENCE TO MEMORY MAP LABELS

MSGFLG	157	PNT	209	RETVP	53637	RSPREPO	61698
MULDIV	55991	PNTR	211	RFTOA	56332	RSPRTY	61375
MULTEN	56034	POKE	55332	RIBUF	247	RSPRTYE	61597
MYCH	191	POS	54174	RIDATA	170	RSSTAT	663
NDX	198	PRDY	50281	RIDBE	667	RSSTOPS	61416
NEGFAC	57268	PRINT	51872	RIDBS	668	RSSTPBIT	61515
NEW	50754	PRINTN	51840	RIGHT	55084	RSSTRBIT	61544
NEWCH	64475	PRT1	51866	RINONE	169	RTI	65366
NEWLIN	50332	PRT6	51944	RIPRTY	171	RTRN	59608
NEWSTT	51118	PRT7	51960	RND	57492	RUN	51313
NEXT	52510	PRTFIX	56781	RNDC1	57482	RUNTB	60916
NEXTLINE	59587	PRTIN	56770	RNDX	139	RVS	199
NMI	65193	PRTOS	52027	ROBUF	249	SA	185
NMINV	792	PRTSTR	51998	RODATA	182	SAL	172
NODIRM	54182	PRTY	155	RODBE	670	SAREG	780
NORMKEY	60510	PTR2	159	RODBS	669	SAT	621
NORMLZ	55550	PTRI	158	ROPRTY	189	SAVE	57683
NXTBIT	181	PUTSCRN	60074	ROUND	56347	SAVE	63093
OLDLIN	59	QPLOP	50970	RETVP	53637	SAVE	65496
OLDTXT	61	QTSW	212	RFTOA	56332	SAVESER	63122
ON	51531	QUOTECK	59064	RIBUF	247	SAVETP	63217
OPEN	62474	RAMBLK0	1024	RIDATA	170	SAVING	63272
OPENLIN	59886	RAMBLK1	8192	RIDBE	667	SCATN	61125
OPENRS	62663	RAMBLK2	16384	RIDBS	668	SCNKEY	60190
OPMASK	77	RAMBLK3	24576	RPACHK	52983	SCNSTK	50058
OPPTR	75	RAMBLK4	40960	RPTFLG	650	SCREEN	7680
OPTAB	49280	RAMSPC	50184	RSBREAK	61605	SCREEN	65517
ORIOST	65130	RBLK	63689	RSCPTBIT	61479	SCREENX	4096
ORR	53222	RD300	64466	RSDVCER	61625	SCRL	59765
OVERFL	55678	RDTIM	63328	RSFRAME	61608	SCRN	58629
PAREXP	52977	RDTIM	65502	RSINBIT	61494	SCRNOUT	59202
PAROC	57878	RDTPBLK	63680	RSINBYTE	61551	SCROLL	59114
PARSL	57809	READ	52230	RSINERR	61610	SECOND	61220
PATCHBA	57590	READIOST	65128	RSINPRTY	61579	SER2	57430
PATCHER	58486	READST	65111	RSMISSN	61462	SEREVL	57408
PATCHES	58556	READT	63886	RSNMI	65246	SERGET	58546
PCNTR	163	READY	50292	RSNXTBIT	61347	SERNAME	62613
PEEK	55309	REM	51515	RSNXTBY	61422	SEROUT0	58537
PG3FREE	784	RESHO	38	RSNXTIN	61775	SEROUT1	58528
PIVAL	52904	RESLST	49310	RSOPNIN	61718	SETADDR	60014
PLOT	58634	RESTOR	64850	RSOPNOU	61628	SETCHAR	59077
PLOT	65520	RESTOR	65415	RSOUTSA	61677	SETFLCH	62431
PLUS	55402	RESTORE	51229	RSOVERU	61602	SETIODEF	58811
PLUS1	55394	RETREAT	59181	RSPAUSE	61792	SETKEYS	60380
PLUS6	55463	RETURN	51410	RSPREPIN	61531	SETLFS	65104

SETMSG	65126	SYNPRT	60065	USRPGM8	4608	VIA2T1LL	37158
SETNAM	65097	SYREG	782	USRPOK	0	VIA2T2CH	37161
SETSLINK	58759	SYSTEM	57639	USRVCTR	673	VIA2T2CL	37160
SETTIM	63335	TALK	60948	VAL	55213	VIAIDDRA	37139
SETTIM	65499	TAN	58033	VALTYP	13	VICCR0	36864
SETTMO	65135	TANSGN	18	VARNAM	69	VICCR1	36865
SFDX	203	TAPE	63732	VARPNT	71	VICCR2	36866
SGN	56377	TAPEH	63463	VARRANG	53012	VICCR3	36867
SGNFAC	56363	TAPEI	178	VARTAB	45	VICCR4	36868
SGNFLG	103	TBLX	214	VCTRIRQ	65534	VICCR5	36869
SHFLAG	653	TBUFFR	828	VCTRNMI	65530	VICCR6	36870
SHFTKEY	60575	TEMPF3	87	VCTRRST	65532	VICCR7	36871
SIN	57960	TEMPI	177	VECTOR	64855	VICCR8	36872
SIZE	82	TEMPPT	22	VECTOR	65412	VICCR9	36873
SKIPST	51448	TEMPST	25	VECTORS	64877	VICCRA	36874
SKPCOM	57867	TIME	160	VERCHK	10	VICCRB	36875
SPMSG	61922	TIMES	55848	VERCK	147	VICCRC	36876
SPREG	783	TIMES3	55897	VIA11ER	37150	VICCRD	36877
SQR	57201	TIMOUT	645	VIA11FR	37149	VICCRE	36878
SRBAD	61108	TKSA	61134	VIA1ACR	37147	VICCRF	36879
SRCHING	63047	TNIF	64719	VIA1DDRB	37138	VICINIT	60900
SRCLKHI	61316	TNOFF	64776	VIA1PA1	37137	VPRTY	64776
SRCLKLO	61325	TPBLOCK	829	VIA1PA2	37151	WAIT	55341
SRSEND	61001	TPBUFA	63565	VIA1PB	37136	WAITABIT	61334
STACK	256	TPHBGN	829	VIA1PCR	37148	WARMBAS	58471
STAL	193	TPHDRID	828	VIA1SR	37146	WARMST	49154
START	64802	TPHNAME	831	VIA1T1CL	37140	WBLK	63715
STATUS	144	TPSTORE	64173	VIA1T1LH	37143	WHATKEY	60777
STKEY	145	TPTOGLE	64490	VIA1T1LL	37142	WRAPLIN	60763
STKSPC	50171	TRMPOS	9	VIA1T2CH	37145	WRITE	64523
STMDSP	49164	TSTMEM	65169	VIA1T2CL	37144	WRT62	64785
STOP	63344	TSTOP	63819	VIA1TICH	37141	WRTNI	64661
STOP	65505	TSTSTOP	51244	VIA2ACR	37163	WRTZ	64680
STORFAC	56276	TXTTAB	43	VIA2DDRA	37155	XFERSTR	54906
STR	54373	TYPCHK	52618	VIA2DDRB	37154	XMAX	649
STREND	49	UDTIM	63284	VIA2IER	37166	XSAV	151
STTI	63837	UDTIM	65514	VIA2IFR	37165	ZERFAC	55543
STXTPT	50830	UNDEF	54190	VIA2PA1	37153		
SUB	55379	UNLSN	61188	VIA2PA2	37167	PROG AID	28681
SUBFLG	16	UNTLK	61156	VIA2PB	37152	RESET	64802
SVXT	146	UNUSDNM	58805	VIA2PCR	37164	RESET w/o	64812
SXREG	781	USER	243	VIA2SR	37162	SCOTT AD	32592
SYNCHR	52991	USRCMD	814	VIA2T1CH	37157	IEEE-488	45065
SYNERR	53000	USRCMDS	820	VIA2T1CL	37156	VICMON	24576
SYNO	150	USRPGM3	4096	VIA2T1HL	37159	VICMON	64096

ADC	add mem to acc with carry	A + M + C → A, C	2.2.1
AND	AND mem with accumulator	A AND M → A	2.2.3.0
ASL	arithmetic shift left	C ← [76543210] ← 0	10.2
BCC	branch on carry clear	branch on C=0	4.1.1.3
BCS	branch on carry set	branch on C=1	4.1.1.4
BEQ	branch on equal (zero set)	branch on Z=1	4.1.1.5
BIT	bit test	A AND M, M7 → N, M6 → V	4.2.1.1
BMI	branch on minus (neg set)	branch on N=1	4.1.1.1
BNE	branch on not equal (0 clear)	branch on Z=0	4.1.1.6
BPL	branch on plus (neg clear)	branch on N=0	4.1.1.2
BRK	force break / interrupt	interrupt PC+2 ↓ SR ↓	9.11
BVC	branch on overflow clear	branch on V=0	4.1.1.8
BVS	branch on overflow set	branch on V=1	4.1.1.7
CLC	clear carry	0 → C	3.0.2
CLD	clear decimal	0 → D	3.3.2
CLI	clear interrupt	0 → I	3.2.2
CLV	clear overflow	0 → V	3.6.1
CMP	compare mem with acc	A - M	4.2.1
CPX	compare mem with X	X - M	7.8
CPY	compare mem with Y	Y - M	7.9
DEC	decrement memory by 1	M - 1 → M	10.7
DEX	decrement X by 1	X - 1 → X	7.6
DEY	decrement Y by 1	Y - 1 → Y	7.7
EOR	exclusive OR with acc	A EOR M → A	2.2.3.2
INC	increment memory by 1	M + 1 → M	10.6
INX	increment X by 1	X + 1 → X	7.4
INY	increment Y by 1	Y + 1 → Y	7.5
JMP	jump to new location	(PC+1)→ PCL, (PC+2)→ PCH	4.0.2 9.8.1
JSR	jump subroutine saving return	(PC+2)↓,(PC+1)→ PCL, (PC+2)→ PCH	8.1
LDA	load accumulator with memory	M → A	2.1.1
LDX	load X with memory	M → X	7.0
LDY	load Y with memory	M → Y	7.1
LSR	logical shift right	0 → [76543210] → C	10.1
NOP	no operation	---	---
ORA	OR with accumulator	A OR M → A	2.2.3.1

PHA	push accumulator on stack	A ↓	8.5
PHP	push processor status (SR)	P ↓	8.11
PLA	pull accumulator from stack	A ↑	8.6
PLP	pull processor status (SR)	P ↑	8.12
ROL	rotate left 1 bit	C ← [76543210] ← C	10.3
ROR	rotate right 1 bit	C → [76543210] → C	10.4
RTI	return from interrupt	P ↑ PC ↑	9.6
RTS	return from subroutine	PC ↑, PC + 1 → PC	8.2
SBC	subtract with carry with borrow	A - M − $\overline{C}$ → A, $\overline{C}$	2.2.2
SEC	set carry flag	1 → C	3.0.1
SED	set decimal mode	1 → D	3.3.1
SEI	set interrupt disable status	1 → I	3.2.1
STA	store accumulator in memory	A → M	2.1.2
STX	store X in memory	X → M	7.2
STY	store Y in memory	Y → M	7.3
TAX	transfer accumulator to X	A → X	7.11
TAY	transfer accumulator to Y	A → Y	7.13
TSX	transfer stack pointer to X	S → X	8.9
TXA	transfer X to accumulator	X → A	7.12
TXS	transfer X to stack pointer	X → S	8.8
TYA	transfer Y to accumulator	Y → A	7.14

ML monitor on page 110

ADDRESSING MODES

AC	ACCUMULATOR	directs operation to accumulator
#	IMMEDIATE	operand is in the byte following instruction
Z	ZERO PAGE	8-bit address assuming high byte is zero
ZX	INDEXED ZERO PAGE	zero page adding value of X index register
ZY	INDEXED ZERO PAGE	zero page adding value of Y index register
AB	ABSOLUTE	16-bit address with 2nd byte is low order 3rd is high
AX	INDEXED ABSOLUTE	16-bit address adding value of X or Y index register
AY	INDEXED ABSOLUTE	16-bit address adding value of X or Y index register
I	IMPLIED	no addressing, implied by the operation
R	RELATIVE	8-bit signed offset added to counter on next instruction
,X)	INDEXED INDIRECT	X selects zero page vector, takes address from pointer
),Y	INDIRECT INDEXED	adds Y to pointer for address
AI	ABSOLUTE INDIRECT	data is accessed using a pointer

N	V	-	B	D	I	Z	C
negative	overflow		break	decimal	interrupt	zero	carry

	REGISTERS	CYCLES												
	N V B D I Z C	**AC**	**#**	**Z**	**ZX**	**ZY**	**AB**	**AX**	**AY**	**I**	**R**	**,X)**	**),Y**	**AI**
ADC	N V - - - Z C		2	3	4		4	4*	4*			6	5*	
AND	N - - - - Z -		2	3	4		4	4*	4*			6	5*	
ASL	N - - - - Z C	2		5	6		6	7						
BCC	- - - - - - -										2*			
BCS	- - - - - - -										2*			
BEQ	- - - - - - -										2*			
BIT	M7 M6 - - - Z -			3			4							
BMI	- - - - - - -										2*			
BNE	- - - - - - -										2*			
BPL	- - - - - - -										2*			
BRK	- - - - 1 - -									7				
BVC	- - - - - - -										2*			
BVS	- - - - - - -										2*			
CLC	- - - - - - 0									2				
CLD	- - - 0 - - -									2				
CLI	- - - - 0 - -									2				
CLV	- 0 - - - - -									2				
CMP	N - - - - Z C		2	3	4		4	4*	4*			6	5*	
CPX	N - - - - Z C		2	3			4							
CPY	N - - - - Z C		2	3			4							
DEC	N - - - - Z -			5	6		6	7						
DEX	N - - - - Z -									2				
DEY	N - - - - Z -									2				
EOR	N - - - - Z -		2	3	4		4	4*	4*			6	5	
INC	N - - - - Z -			5	6		6	7						
INX	N - - - - Z -									2				
INY	N - - - - Z -									2				
JMP	- - - - - - -						3							5
	N V B D I Z C	**AC**	**#**	**Z**	**ZX**	**ZY**	**AB**	**AX**	**AY**	**I**	**R**	**,X)**	**),Y**	**AI**
	BYTES	1	2	2	2	2	3	3	3	1	2	2	2	3

N	V	-	B	D	I	Z	C
negative	overflow		break	decimal	interrupt	zero	carry

	REGISTERS	CYCLES												
	N V B D I Z C	**AC**	**#**	**Z**	**ZX**	**ZY**	**AB**	**AX**	**AY**	**I**	**R**	**,X)**	**),Y**	**AI**
JSR	- - - - - - -						6							
LDA	N - - - - Z -		2	3	4		4	4*	4*			6	5*	
LDX	N - - - - Z -		2	3		4	4		4*					
LDY	N - - - - Z -		2	3	4		4	4*						
LSR	0 - - - - Z C	2		5	6		6	7						
NOP	- - - - - - -									2				
ORA	N - - - - Z -		2	3	4		4	4*	4*			6	5*	
PHA	- - - - - - -									3				
PHP	- - - - - - -									3				
PLA	N - - - - Z -									4				
PLP	*from stack*									4				
ROL	N - - - - Z C	2		5	6		6	7						
ROR	N - - - - Z C	2		5	6		6	7						
RTI	*from stack*									6				
RTS	- - - - - - -									6				
SBC	N V - - - Z C		2	3	4		4	4*	4*			6	5*	
SEC	- - - - - - 1									2				
SED	- - - 1 - - -									2				
SEI	- - - - 1 - -									2				
STA	- - - - - - -			3	4		4	5	5			6	6	
STX	- - - - - - -			3		4	4							
STY	- - - - - - -			3	4		4							
TAX	N - - - - Z -									2				
TAY	N - - - - Z -									2				
TSX	N - - - - Z -									2				
TXA	N - - - - Z -									2				
TXS	- - - - - - -									2				
TYA	N - - - - Z -									2				
	N V B D I Z C	**AC**	**#**	**Z**	**ZX**	**ZY**	**AB**	**AX**	**AY**	**I**	**R**	**,X)**	**),Y**	**AI**
	BYTES	1	2	2	2	2	3	3	3	1	2	2	2	3

DEC	A	#	ZP	AB	ABX	ABY	ZPX	ZPY	,X)	),Y
ADC		105	101	109	125	121	117		97	113
AND		41	37	45	61	57	53		33	49
ASL	10		6	14	30		22			
BIT			36	44						
CMP		201	197	205	221	217	213		193	209
CPX		224	228	236						
CPY		192	196	204						
DEC			198	206	221		214			
EOR		73	69	77	93	89	85		65	81
INC			230	238	253		246			
LDA		169	165	173	189	185	181		161	177
LDX		162	166	174		190		182		
LDY		160	164	172	188		180			
LSR	74		70	78	94		86			
ORA		9	5	13	29	25	21		1	17
ROL	42		38	46	62		54			
ROR	106		102	110	126		118			
SBC		233	229	237	253	249	245		225	241
STA			133	141	157	153	149		129	145
STX			134	142				150		
STY			132	140			148			
	A	**#**	**ZP**	**AB**	**ABX**	**ABY**	**ZPX**	**ZPY**	**,X)**	**),Y**
bytes	1	2	2	3	3	3	2	2	2	2

BPL	BMI	BVC	BVS	BCC	BCS	BNE	BEO
16	48	80	112	144	176	208	240

TXA	TAX	TYA	TAY	TSX	TXS
138	170	152	168	186	154

PHP	PLP	PHA	PLA
8	40	72	104

BRK	JSR	RTI	RTS	JMP	JMP()	NOP
0	32	64	96	76	108	234

CLC	SEC	CLI	SEI	CLV	CLD	SED
24	56	88	120	184	216	248

DEY	INY	DEX	INX
136	200	202	232

ML monitor on page 110

HEX	A	#	ZP	AB	ABX	ABY	ZPX	ZPY	,X)	),Y
ADC		69	65	6D	7D	79	75		61	71
AND		29	25	2D	3D	39	35		21	31
ASL	0A		06	0E	1E		16			
BIT			24	2C						
CMP		C9	C5	CD	DD	D9	D5		C1	D1
CPX		E0	E4	EC						
CPY		C0	C4	CC						
DEC			C6	CE	DE		D6			
EOR		49	45	4D	5D	59	55		41	51
INC			E6	EE	FE		F6			
LDA		A9	A5	AD	BD	B9	B5		A1	B1
LDX		A2	A6	AE		BE		B6		
LDY		A0	A4	AC	BC		B4			
LSR	4A		46	4E	5E		56			
ORA		09	05	0D	1D	19	15		01	11
ROL	2A		26	2E	3E		36			
ROR	6A		66	6E	7E		76			
SBC		E9	E5	ED	FD	F9	F5		E1	F1
STA			85	8D	9D	99	95		81	91
STX			86	8E				96		
STY			84	8C			94			
	A	#	ZP	AB	ABX	ABY	ZPX	ZPY	,X)	),Y
bytes	1	2	2	3	3	3	2	2	2	2

BPL	BMI	BVC	BVS	BCC	BCS	BNE	BEQ
10	30	50	70	90	B0	D0	F0

TXA	TAX	TYA	TAY	TSX	TXS	
8A	AA	98	A8	BA	9A	

PHP	PLP	PHA	PLA
08	28	48	68

BRK	JSR	RTI	RTS	JMP	JMP()	NOP
00	20	40	60	4C	6C	EA

CLC	SEC	CLI	SEI	CLV	CLD	SED
18	38	58	78	B8	D8	F8

DEY	INY	DEX	INX
88	C8	CA	E8

ML monitor on page 110

7	6	5	4	3	2	1	0
N	V	1	B	D	I	Z	C
negative	overflow	always 1	break	decimal	interrupt	zero	carry

ML monitor on page 110

ADC

A + M + C → A, C — add mem to acc with carry

N	V	D	I	Z	C
√	√	-	-	√	√

#	ADC #oper	$69	105	2B	2
ZP	ADC addr	$65	101	2B	3
ZPX	ADC addr, X	$75	117	2B	4
AB	ADC ADDR	$6D	109	3B	4
ABX	ADC ADDR, X	$7D	125	4B	4/5
ABY	ADC ADDR, Y	$79	121	3B	4/5
,X)	ADC (addr, X)	$61	97	2B	6
), Y	ADC (addr), Y	$71	113	2B	5/6

*add 1 if page boundary is crossed

011bbb01

AND

A AND M → A — and mem with accumulator

N	V	D	I	Z	C
√	-	-	-	√	-

#	AND #oper	$29	41	2B	2
ZP	AND addr	$25	37	2B	3
ZPX	AND addr, X	$35	53	2B	4
AB	AND ADDR	$2D	45	3B	4
ABX	AND ADDR, X	$3D	61	3B	4/5
ABY	AND ADDR, Y	$39	57	3B	4/5
,X)	AND (addr, X)	$21	33	2B	6
), Y	AND (addr), Y	$31	49	2B	5

*add 1 if page boundary is crossed

001bbb01

ASL

C ← [76543210] ← 0 — arithmetic shift left

N	V	D	I	Z	C
√	-	-	-	√	√

ACC	ASL #oper	$0A	10	1B	2
ZP	ASL addr	$06	6	2B	5
ZPX	ASL addr, X	$16	22	2B	6
AB	ASL ADDR	$0E	14	3B	6
ABX	ASL ADDR, X	$1E	30	3B	7

000bbb10

BCC

Branch on C=0 — branch on carry clear

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BCC Oper	$90	144	2B	2*

*add 1 if branch is to same page

*add 2 if branch is to different page

1001000

BCS

Branch on C=1 — branch on carry set

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BCS Oper	$B0	176	2B	2*

*add 1 if branch is to same page

*add 2 if branch is to next page

1011000

BEQ Branch on Z=1 branch on result zero

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BEQ Oper	$F0	240	2B	2*
*add 1 if branch occurs to same page					11110000
*add 2 if branch occurs to next page					

BIT A AND M, M7 → N, M6 → V bit test

N	V	D	I	Z	C
M7	M6	-	-	√	-

ZP	BIT addr	$24	36	2B	3
AB	BIT ADDR	$2C	44	3B	4
					0010b100

BMI Branch on N=1 branch on result minus

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BMI Oper	$30	48	2B	2*
*add 1 if branch occurs to same page					00110000
*add 2 if branch occurs to different page					

BNE Branch on Z=0 branch on result not zero

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BNE Oper	$D0	208	2B	2*
*add 1 if branch occurs to same page					11010000
*add 2 if branch occurs to different page					

BPL Branch on N=0 branch on result plus

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BPL Oper	$10	16	2B	2*
*add 1 if branch occurs to same page					00010000
*add 2 if branch occurs to different page					

BRK interrupt PC+2 ↓ SR ↓ force break

N	V	D	I	Z	C
-	-	-	1	-	-

IMPL	BRK	$00	0	1B	7
A BRK command cannot be masked by setting 1					00000000

BVC Branch on V=0 branch on overflow clear

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BVC Oper	$50	80	2B	2*
*add 1 if branch occurs to same page					01010000
*add 2 if branch occurs to different page					

BVS Branch on V=1 branch on overflow set

N	V	D	I	Z	C
-	-	-	-	-	-

REL	BVS Oper	$70	112	2B	2*
*add 1 if branch occurs to same page					01110000
*add 2 if branch occurs to different page					

CLC 0 → C clear carry flag

N	V	D	I	Z	C
-	-	-	-	-	0

IMPL	CLC	$18	24	1B	2
					00011000

CLD `0 → D` clear decimal mode

N	V	D	I	Z	C
-	-	0	-	-	-

IMPL	CLD	$D8	216	1B	2
					11011000

CLI `0 → I` clear interrupt disable bit

N	V	D	I	Z	C
-	-	-	0	-	-

IMPL	CLI	$58	88	1B	2
					01011000

CLV `0 → V` clear overflow flag

N	V	D	I	Z	C
-	0	-	-	-	-

IMPL	CLV	$B8	184	1B	2
					10111000

CMP `A - M` compare with accumulator

N	V	D	I	Z	C
√	-	-	-	√	√

#	CMP #oper	$C9	201	2B	2
ZP	CMP addr	$C5	197	2B	3
ZPX	CMP addr, X	$D5	213	2B	4
AB	CMP ADDR	$CD	205	3B	4
ABX	CMP ADDR, X	$DD	221	3B	4/5
ABY	CMP ADDR, Y	$D9	217	3B	4/5
,X)	CMP (addr, X)	$C1	193	2B	6
), Y	CMP (addr), Y	$D1	209	2B	5/6
*add 1 if page boundary is crossed					110bbb01

CPX `X - M` compare with X

N	V	D	I	Z	C
√	-	-	-	√	√

#	CPX #oper	$E0	224	2B	2
ZP	CPX addr	$E4	228	2B	3
AB	CPX ADDR	$EC	236	3B	4
					1110bb00

CPY `Y - M` compare with Y

N	V	D	I	Z	C
√	-	-	-	√	√

#	CPY #oper	$C0	192	2B	2
ZP	CPY addr	$C4	196	2B	3
AB	CPY ADDR	$CC	204	3B	4
					1100bb00

<=>	N	Z	C	
	1	0	0	Register LESS than data
	0	1	1	Register EQUAL to data
	0	0	1	Register GREATER than data

DEC `M - 1 → M` decrement memory by one

N	V	D	I	Z	C
√	-	-	-	√	-

ZP	CMP addr	$C6	198	2B	5
ZPX	CMP addr, X	$D6	214	2B	6
AB	CMP ADDR	$CE	206	3B	6
ABX	CMP ADDR, X	$DE	222	3B	7
					110bb110

DEX X - 1 → X decrement index X by one

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	DEX	$CA	202	1B	2

11001010

DEY Y - 1 → Y decrement index Y by one

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	DEY	$88	136	1B	2

10001000

EOR A EOR M → A exclusive or with accumulator

N	V	D	I	Z	C
√	-	-	-	√	-

#	EOR #oper	$49	73	2B	2
ZP	EOR addr	$45	69	2B	3
ZPX	EOR addr, X	$55	85	2B	4/5
AB	EOR ADDR	$4D	77	3B	4
ABX	EOR ADDR, X	$5D	93	3B	4/5
ABY	EOR ADDR, Y	$59	89	3B	4/5
,X)	EOR (addr, X)	$41	65	2B	6
), Y	EOR (addr), Y	$51	81	2B	5/6

*add 1 if page boundary is crossed

010bbb01

INC M + 1 → M increment memory by one

N	V	D	I	Z	C
√	-	-	-	√	-

ZP	INC addr	$E6	230	2B	5
ZPX	INC addr, X	$F6	246	2B	6
AB	INC ADDR	$EE	238	3B	6
ABX	INC ADDR, X	$FE	254	3B	7

111bb110

INX X + 1 → X increment index X by one

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	INX	$E8	232	1B	2

11101000

INY Y + 1 → Y increment index Y by one

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	INY	$C8	200	1B	2

11001000

JMP (PC+1) → PCL, (PC+2) → PCH jump to new location

N	V	D	I	Z	C
-	-	-	-	-	-

AB	JMP ADDR	4C	76	3B	3
INDRCT	JMP (ADDR)	6C	108	3B	5

01b01100

JSR

(PC+2) ↓,(PC+1)→PCL,(PC+2)→PCH jump save return

N	V	D	I	Z	C
-	-	-	-	-	-

AB	JSR ADDR	$20	32	3B	6
					00100000

LDA

M → A load accumulator with memory

N	V	D	I	Z	C
√	-	-	-	√	-

#	LDA #oper	$A9	169	2B	2
ZP	LDA addr	$A5	165	2B	3
ZPX	LDA addr, X	$B5	181	2B	4
AB	LDA ADDR	$AD	173	3B	4
ABX	LDA ADDR, X	$BD	189	3B	4/5
ABY	LDA ADDR, Y	$B9	185	3B	4/5
,X)	LDA (addr, X)	$A1	161	2B	6
), Y	LDA (addr), Y	$B1	177	2B	5/6
*add 1 if page boundary is crossed					101bbb01

LDX

M → X load index X with memory

N	V	D	I	Z	C
√	-	-	-	√	-

#	LDX #oper	$A2	162	2B	2
ZP	LDX addr	$A6	166	2B	3
ZPX	LDX addr, X	$B6	182	2B	4
AB	LDX ADDR	$AE	174	3B	4
ABX	LDX ADDR, X	$BE	190	3B	4/5
*add 1 if page boundary is crossed					101bbb10

LDY

M → Y load index Y with memory

N	V	D	I	Z	C
√	-	-	-	√	-

#	LDY #oper	$A0	160	2B	2
ZP	LDY addr	$A4	164	2B	3
ZPX	LDY addr, X	$B4	180	2B	4
AB	LDY ADDR	$AC	172	3B	4
ABX	LDY ADDR, X	$BC	188	3B	4/5
*add 1 if page boundary is crossed					101bbb00

LSR

0 → [76543210] → C logical shift right

N	V	D	I	Z	C
0	-	-	-	√	√

ACC	LSR #oper	$4A	74	1B	2
ZP	LSR addr	$46	70	2B	5
ZPX	LSR addr, X	$56	86	2B	6
AB	LSR ADDR	$4E	78	3B	6
ABX	LSR ADDR, X	$5E	94	3B	7
					010bbb10

NOP

NO OPERATION no operation

N	V	D	I	Z	C
-	-	-	-	-	-

IMPL	NOP	$EA	234	1B	2
2 cycle delay					11101010

ORA

A OR M → A — OR with accumulator

N	V	D	I	Z	C
✓	-	-	-	✓	-

#	ORA #oper	$09	9	2B	2
ZP	ORA addr	$05	5	2B	3
ZPX	ORA addr, X	$15	21	2B	4
AB	ORA ADDR	$0D	13	3B	4
ABX	ORA ADDR, X	$1D	29	3B	4/5
ABY	ORA ADDR, Y	$19	25	3B	4/5
,X)	ORA (addr, X)	$01	1	2B	6
), Y	ORA (addr), Y	$11	17	2B	5

*add 1 on page crossing

000bbb01

PHA

A ↓ — push proc status on stack

N	V	D	I	Z	C
-	-	-	-	-	-

IMPL	PHA	$48	72	1B	3

01001000

PHP

P ↓ — push accumulator from stack

N	V	D	I	Z	C
-	-	-	-	-	-

IMPL	PHP	$08	8	1B	3

00001000

PLA

A ↑ — pull accumulator on stack

N	V	D	I	Z	C
✓	-	-	-	✓	-

IMPL	PLA	$68	104	1B	4

01101000

PLP

P ↑ — pull proc status from stack

N	V	D	I	Z	C
from stack					

IMPL	PLP	$28	40	1B	4

00101000

ROL

C ← [76543210] ← C — rotate one bit left

N	V	D	I	Z	C
✓	-	-	-	✓	✓

ACC	ROL #oper	$2A	42	1B	2
ZP	ROL addr	$26	38	2B	5
ZPX	ROL addr, X	$36	54	2B	6
AB	ROL ADDR	$2E	46	3B	6
ABX	ROL ADDR, X	$3E	62	3B	7

001bbb10

ROR

C → [76543210] → C — rotate one bit right

N	V	D	I	Z	C
✓	-	-	-	✓	✓

ACC	ROR #oper	$6A	106	1B	2
ZP	ROR addr	$66	102	2B	5
ZPX	ROR addr, X	$76	118	2B	6
AB	ROR ADDR	$6E	110	3B	6
ABX	ROR ADDR, X	$7E	126	3B	7

011bbb10

RTI — P ↑ PC ↑ — return from interrupt

N	V	D	I	Z	C
from stack					

IMPL	RTI	$40	64	1B	6
					01000000

RTS — PC ↑, PC + 1 → PC — return from subroutine

N	V	D	I	Z	C
-	-	-	-	-	-

IMPL	RTS	$60	96	1B	6
					01100000

SBC — A - M − $\overline{C}$ → A, $\overline{C}$ — subtract mem from acc w/ borrow

N	V	D	I	Z	C
✓	✓	-	-	✓	✓

#	SBC #oper	$E9	233	2B	2
ZP	SBC addr	$E5	229	2B	3
ZPX	SBC addr, X	$F5	245	2B	4
AB	SBC ADDR	$ED	237	3B	4
ABX	SBC ADDR, X	$FD	253	3B	4/5
ABY	SBC ADDR, Y	$F9	249	3B	4/5
,X)	SBC (addr, X)	$E1	225	2B	6
), Y	SBC (addr), Y	$F1	241	2B	5/6
					111bbb01

SEC — 1 → C — set carry flag

N	V	D	I	Z	C
-	-	-	-	-	1

IMPL	SEC	$38	56	1B	2
					00111000

SED — 1 → D — set decimal mode

N	V	D	I	Z	C
-	-	1	-	-	-

IMPL	SED	$F8	248	1B	2
					11111000

SEI — 1 → I — set interrupt disable status

N	V	D	I	Z	C
-	-	-	1	-	-

IMPL	SEI	$78	120	1B	2
					01111000

STA — A → M — store accumulator in memory

N	V	D	I	Z	C
-	-	-	-	-	-

ZP	STA addr	$85	133	2B	3
ZPX	STA addr, X	$95	149	2B	4
AB	STA ADDR	$8D	141	3B	4
ABX	STA ADDR, X	$9D	157	3B	5
ABY	STA ADDR, Y	$99	153	3B	5
,X)	STA (addr, X)	$81	129	2B	6
), Y	STA (addr), Y	$91	145	2B	6
					100bbb01

STX — X → M — store index X in memory

N	V	D	I	Z	C
-	-	-	-	-	-

ZP	STX addr	$86	134	2B	3
ZPX	STX addr, X	$96	150	2B	4
AB	STX ADDR	$8E	142	3B	4
					100bb110

STY Y → M store index Y in memory

N	V	D	I	Z	C
-	-	-	-	-	-

ZP	STY addr	$84	132	2B	3
ZPX	STY addr, X	$94	148	2B	4
AB	STY ADDR	$8C	140	3B	4

100bb100

TAX A → X transfer accumulator to X

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	TAX	$AA	170	1B	2

10101010

TAY A → Y transfer accumulator to Y

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	TAY	$A8	168	1B	2

10101000

TSX S → X transfer stack pointer to X

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	TSX	$BA	186	1B	2

10111010

TXA X → A transfer X to accumulator

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	TXA	$8A	138	1B	2

10001010

TXS X → S transfer X to stack pointer

N	V	D	I	Z	C
-	-	-	-	-	-

IMPL	TXS	$9A	154	1B	2

10011010

TYA Y → A transfer Y to accumulator

N	V	D	I	Z	C
√	-	-	-	√	-

IMPL	TYA	$98	152	1B	2

10011000

RELATIVE BRANCH BACKWARDS

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8_	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113
9_	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
A_	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81
B_	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65
C_	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
D_	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
E_	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
F_	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

UNINTENDED / UNDOCUMENTED OPCODES

ALR (ASR)

AND op + SLR

N	V	D	I	Z	C
√	-	-	-	√	√

#	ALR #oper	4B	75	2B	2

ANC (AAC)

AND op + set C as ASL

N	V	D	I	Z	C
√	-	-	-	√	√

#	ANC #oper	0B	11	2B	2

ANC2

AND op + set C as ROL

N	V	D	I	Z	C
√	-	-	-	√	√

#	ANC #oper	2B	43	2B	2

highly unstable

ANE (XAA)

* AND X + AND op

N	V	D	I	Z	C
√	-	-	-	√	-

#	ANE #oper	8B	139	2B	2

highly unstable

ARR

AND op + ROR

N	V	D	I	Z	C
√	√	-	-	√	√

#	ARR #oper	6B	107	2B	2

DCP (DCM)

DEC op + CMP op

N	V	D	I	Z	C
√	-	-	-	√	√

ZP	DCP addr	C7	119	2B	5
ZPX	DCP addr, X	D7	215	2B	6
AB	DCP ADDR	CF	207	3B	6
ABX	DCP ADDR, X	DF	223	3B	7
ABY	DCP ADDR, Y	DB	219	3B	7
,X)	DCP (addr, X)	C3	195	2B	8
), Y	DCP (addr), Y	D3	211	2B	8

ISC (ISB,INS)

INC op + SBC op

N	V	D	I	Z	C
√	√	-	-	√	√

ZP	ISC addr	E7	231	2B	5
ZPX	ISC addr, X	F7	247	2B	6
AB	ISC ADDR	EF	239	3B	6
ABX	ISC ADDR, X	FF	255	3B	7
ABY	ISC ADDR, Y	FB	251	3B	7
,X)	ISC (addr, X)	E3	227	2B	8
), Y	ISC (addr), Y	F3	243	2B	4

JAM (KIL,HLT)

freezes CPU

HEX	02	12	22	62	72	92	32	42	52	B2	D2	FE
DEC	2	18	34	98	114	146	50	66	82	178	210	254

LAS (LAR,LAE)

LDA / TSX op

N	V	D	I	Z	C
√	-	-	-	√	-

ABY	LAS ADDR, Y	BB	187	3B	4/5

LAX

LDA op + LDX op

N	V	D	I	Z	C
√	-	-	-	√	-

ZP	LAX addr	A7	167	2B	3
ZPX	LAX addr, X	B7	183	2B	4
AB	LAX ADDR	AF	175	3B	4
ABY	LAX ADDR, Y	BF	191	3B	4/5
,X)	LAX (addr, X)	A3	163	2B	6
), Y	LAX (addr), Y	B3	179	2B	5/6

LXA

Store * AND oper in A and X

N	V	D	I	Z	C
√	-	-	-	√	-

#	LXA #oper	AB	171	2B	2

highly unstable

RLA

ROL op + AND op

N	V	D	I	Z	C
√	-	-	-	√	√

ZP	RLA addr	27	39	2B	5
ZPX	RLA addr, X	37	55	2B	6
AB	RLA ADDR	2F	47	3B	6
ABX	RLA ADDR, X	3F	63	3B	7
ABY	RLA ADDR, Y	3B	59	3B	7
,X)	RLA (addr, X)	23	35	2B	8
), Y	RLA (addr), Y	33	51	2B	8

RRA

ROR op + ADC opp

N	V	D	I	Z	C
√	√	-	-	√	√

ZP	RRA addr	67	103	2B	5
ZPX	RRA addr, X	77	119	2B	6
AB	RRA ADDR	6F	111	3B	6
ABX	RRA ADDR, X	7F	127	3B	7
ABY	RRA ADDR, Y	7B	123	3B	7
,X)	RRA (addr, X)	63	99	2B	8
), Y	RRA (addr), Y	73	115	2B	8

SAX (AAX,AXS)

A AND X → M

N	V	D	I	Z	C
-	-	-	-	-	-

ZP	SAX addr	87	135	2B	3
ZPX	SAX addr, X	97	151	2B	4
AB	SAX ADDR	8F	143	3B	4
ABX	SAX ADDR, X	83	131	2B	6

SBX (AXS,SAX)

(A AND X) - op → X

N	V	D	I	Z	C
√	-	-	-	√	√

#	SBX #oper	CB	203	2B	2

SHA (AHX,AXA) A AND X AND (H + 1) → M

N	V	D	I	Z	C
-	-	-	-	-	-

ABY	SHA ADDR, Y	9F	159	3B	5
), Y	SHA (addr), X	93	147	2B	6

notably unstable

SHX (A11,SXA,XAS) X AND (H + 1) → M

N	V	D	I	Z	C
-	-	-	-	-	-

ABY	SHX ADDR, Y	9E	158	3B	5

notably unstable

SHY (A11,SYA,SAY) Y AND (H + 1) → M

N	V	D	I	Z	C
-	-	-	-	-	-

ABX	SHY ADDR, X	9C	156	3B	5

notably unstable

SLO (ASO) ASL op + ORA op

N	V	D	I	Z	C
✓	-	-	-	✓	✓

ZP	SLO addr	07	7	2B	5
ZPX	SLO addr, X	17	23	2B	6
AB	SLO ADDR	0F	15	3B	6
ABX	SLO ADDR, X	1F	31	3B	7
ABY	SLO ADDR, Y	1B	27	3B	7
,X)	SLO (addr, X)	03	3	2B	8
), Y	SLO (addr), Y	13	19	2B	8

SRE (LSE) LSR op + EOR op

N	V	D	I	Z	C
✓	-	-	-	✓	✓

ZP	SRE addr	47	71	2B	5
ZPX	SRE addr, X	57	87	2B	6
AB	SRE ADDR	4F	79	3B	6
ABX	SRE ADDR, X	5F	95	3B	7
ABY	SRE ADDR, Y	5B	91	3B	7
,X)	SRE (addr, X)	43	67	2B	8
), Y	SRE (addr), Y	53	83	2B	8

TAS (XAS,SHS) A AND X → SP, A AND X AND (H + 1) → M

N	V	D	I	Z	C
-	-	-	-	-	-

ABY	TAS ADDR, Y	9B	155	3B	5

notably unstable

USBC (SBC) SBC op + NOP

N	V	D	I	Z	C
✓	✓	-	-	✓	✓

#	USBC #oper	EB	235	2B	2

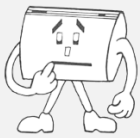
NOP (DOP,TOP) NO OPERATION

N	V	D	I	Z	C
-	-	-	-	-	-

IMPL	3A 5A 7A DA FA	1A	26	1B	2
#	82 89 C2 E2	80	128	2B	2
ZP	44 64	04	4	2B	3
ZPX	34 54 74 D4 F4	14	14	2B	4
AB		0C	12	3B	4
ABX	3C 5C 7C DC FC	1C	28	3B	4/5

MACHINE LANGUAGE OPTIMIZATION TIPS *(courtesy of wimoos)*

- ☐ Change JSRs, followed by RTS into JMPs
- ☐ Change JMPs into branches on known conditions
- ☐ A BEQ-branch taken after a CMP, CPY or CPX, will have the carry set
- ☐ A BEQ-branch taken after an EOR will have 0 in A
- ☐ A comparison using EOR affects A and Z, but not the carry
- ☐ An ASL/ROL may help to save b7 (negative) a little longer in the carry
- ☐ An LSR/ROR may help to save b0 a little longer in the carry



BASIC INDIRECT JUMPS		
USRADD	$0001 - 0002	1-2
ADRAY1	$0003 - 0004	3-4
ADRAY2	$0005 - 0006	5-6
INPPTR	$0043 - 0044	67-68
KEYTAB	$00F5 - 00F6	245-246
IERROR	$0300 - 0301	768-769
IMAIN	$0302 - 0303	770-771
ICRNCH	$0304 - 0305	772-773
IQPLOP	$0306 - 0307	774-775
IGONE	$0308 - 0309	776-777
IEVAL	$030A – 030B	778-779
Register temp storage for SYS		
SAREG	$030C	780 .A
SXREG	$030D	781 .X
SYREG	$030E	782 .Y
SPREG	$030F	783 .P

KERNAL VECTOR ADDRESSES		
CINV	$0314 - 0315	788-789
CBINV	$0316 - 0317	790-791
NMINV	$0318 - 0319	792-793
IOPEN	$031A - 031B	794-795
ICLOSE	$031C – 031D	796-797
ICHKIN	$031E – 031F	798-799
ICKOUT	$0320 - 0321	800-801
ICLRCH	$0322 – 0323	802-803
IBASIN	$0324 - 0325	804-805
IBSOUT	$0326 - 0327	806-807
ISTOP	$0328 - 0329	808-809
IGETIN	$032A – 032B	810-811
ICLALL	$032C - 032D	812-813
USRCMD	$032E – 032F	814-815
ILOAD	$0330 - 0331	816-817
ISAVE	$0332 - 0333	818-819

INTERRUPT VECTORS					
HEX		DEC		FUNCTION	REGS PUSHED
$0314	$0315	788	789	INT	A X Y
$0316	$0317	790	791	BRK	A X Y
$0318	$0319	792	793	NMI	

KERNAL ERROR CODES *	carry bit is set, number of error returned in accumulator		
0	routine terminated by STOP key	**5**	device not present
1	too many open files	**6**	file is not an input file
2	file already open	**7**	file is not an output file
3	file not open	**8**	file name is missing
4	file not found	**9**	illegal device number

USER CALLABLE KERNAL ROUTINES				STACK	ERROR
ACPTR	FFA5	65445	Input byte from serial port	13	READST
CHKIN*	FFC6	65478	Open channel for input	-	3,5,6
CHKOUT*	FFC9	65481	Open channel for output	-	3,5,7
CHRIN*	FFCF	65487	Input character from channel	-	READST
CHROUT*	FFD2	65490	Output character to channel	-	READST
CIOUT	FFA8	65448	Output byte to serial port	-	READST
CLALL*	FFE7	65511	Close all channels and files	11	none
CLOSE*	FFC3	65475	Close a specified logical file	-	none
CLRCHN*	FFCC	65484	Restore default I/O devices	9	none
GETIN*	FFE4	65512	Get char from current input dev	-	none
IOBASE	FFF3	65523	Return base address of I/O device	2	none
LISTEN	FFB1	65457	Command devices on serial to listen	-	READST
LOAD*	FFD5	65493	Load (A=0) Verify (A=1) ram	-	0,4,5,8,9
MEMBOT	FF9C	65436	Read (C=1) Set(C=0) bottom of memory	-	none
MEMTOP	FF99	65433	Read (C=1) Set(C=0) top of memory	2	none
OPEN*	FFC0	65472	Open a logical file	-	1,2,4,5,6
PLOT	FFF0	65520	Read (C=1) Set(C=0) x y cursor pos	2	none
RAMTAS	FF87	65415	Init RAM, allocate tape buff, set screen	2	none
RDTIM	FFDE	65502	Read real time clock	2	none
READST	FFB7	65463	Read I/O status word (see page 114)	2	none
RESTOR	FF8A	65415	Restore default I/O vectors	2	none
SAVE*	FFD8	65496	Save RAM to device from $2B to .X, .Y	-	5,8,9
SCNKEY	FF9F	65439	Scan keyboard	-	none
SCREEN	FFED	65517	Return screen size in rows columns	2	none
SECOND	FF93	65427	Send secondary address after LISTEN	-	READST
SETLFS	FFBA	65466	Set logical, first, and second address	2	none
SETMSG	FF90	65424	Enable/disable KERNAL messages	2	none
SETNAM	FFBD	65469	Set file name	-	none
SETTIM	FFD8	65499	Set real time clock	2	none
SETTMO	FFA2	65442	Set (A<#128) Reset (A>#127) timeout	2	none
STOP*	FFE1	65505	Scan STOP key	-	none
TALK	FFB4	65460	Command serial to TALK	-	READST
TKSA	FF96	65430	Send secondary address after TALK	-	READST
UDTIM	FFEA	65514	Update (increment) real time clock	2	none
UNLSN	FFAE	65454	Command serial to UNLISTEN	-	READST
UNTLK	FFAB	65451	Command serial to UNTALK	-	READST
VECTOR	FF8D	65412	Store (C=1) Restore (C=0) vectors	2	none

*through RAM vector

	PREP. ROUTINES	IN			OUT		
		.A	.X	.Y	.A	.X	.Y
ACPTR	TALK [TSKA]				data	alt	
CHKIN*	(OPEN)		LF#		alt		
CHKOUT*	(OPEN)		LF#		alt		
CHRIN*	(OPEN CHKIN)				data	alt	
CHROUT*	(CHKOUT OPEN)	data					
CIOUT	LISTEN [SECOND]	data					
CLALL*					alt	alt	
CLOSE*		LF#			alt	alt	alt
CLRCHN*					alt	alt	
GETIN*					data	alt	alt
IOBASE						addr lo	addr hi
LISTEN		DEV#					
LOAD*	SETLFS SETNAM	load/ver	start lo	start hi		end lo +1	end hi
MEMBOT		C=0	bot lo	bot hi	C=1	bot lo	bot hi
MEMTOP		C=0	top lo	top hi	C=1	top lo	top hi
OPEN*	SETLFS SETNAM				alt	alt	alt
PLOT		C=0	row	col	C=1	row	col
RAMTAS					alt	alt	alt
RDTIM					MSB	MSB2	LSB
READST					ST		
RESTOR					alt	alt	alt
SAVE*	SETLFS SETNAM	#<txttab (=#$28)	end lo	end hi		end lo +1	end hi
SCNKEY					alt	alt	alt
SCREEN						#rows	#cols
SECOND	LISTEN	SA/$60					
SETLFS		LF#	DEV#	SA			
SETMSG		.A val $40 control msgs on, $80 error msgs on.					
SETNAM		len	addr lo	addr hi			
SETTIM		MSB	MSB2	LSB			
SETTMO							
STOP*		yes: .Z = 1 no: .A = last row keyboard scan					
TALK		DEV#					
TKSA	TALK	SA/$60					
UDTIM					alt	alt	
UNLSN					alt		
UNTLK					alt		
VECTOR		C=1	table lo	table hi	C=0	table lo	table hi

* through RAM vector

USER CALLABLE ROM SUBROUTINES (abridged)

MAKSPC	C3BB	50107	Open up space in BASIC text
RAMSPC	C408	50184	Check available memory (called by 1)
ERROR	C437	50231	Send BASIC error message
	C48A	50314	Main CHRGET entry
NEWLIN	C49C	50220	Crunch tokens, insert line
GETLIN	C560	50528	Fix chaining, CLR & READY
CRNCH	C579	50553	Fix chaining
FINLIN	C613	50707	Receive line from keyboard
STXTPT	C68E	50830	Crunch tokens (called by 7)
CONT	C857	51287	Find line in BASIC
DECBIN	C96B	49771?	Purge stack of all returns and nexts (POP)
	CAD3	51923	Continue BASIC execution (CONT)
	CAD7	51927	Get fixed point number from BASIC text
PRTSTR	CB1E	51998	Send RETURN, LF if in screen mode
	CB24	52004	Send RETURN, LINEFEED
	CB47	52039	Print string from A, Y
FRMEVL $	CD9E	52638	Print precomputed string
FRMEVL #	CD9E	52638	Print char (output .A to device)
SYNCHR	CEFF	52991	Eva string $0D =#$FF ($07) | Eva number $0D =#$00 ($07)
LPACHK	CEFA	52986	Check for (
RPACHK	CEF7	52983	Check for)
FNDVAR	D0E7	53479	Find floating point variable, given name
RETVP	D185	53637	Bump variable address by 2 (called by 31)
	D79E	55086	Get Acc#1 LSB to X register
	D7B5	55221	Evaluate string (VAL)
	D7B9	55225	Evaluate string from X,Y (above +4)
GETAD	D7EB	55275	Get two parameters for POKE, WAIT
LAPLUS	D867	55399	Add (from memory)
TIMES	DA28	55848	Multiply by memory location
MULTEN	DAE2	56034	Multiply Acc#1 by 10
LODFAC	DBA2	56226	Unpack memory variable to Acc#1
	DBD7	56279	Copy Acc#1 to (X,Y) location
PRTFIX	DDCD	56781	Print fixed point value
FLTASC	DDDD	56797	Convert number to string at $0100 (called by 48)
SCRNOUT	E742	59202	Print a character
KMSGSH	F1E6	61926	Print system message
TALK	EE14	60948	Send talk to IEEE/Serial
LISTEN	EE17	60951	Send listen to IEEE/Serial
CSECOND	FF93	65427	Send secondary address
CIOUT	EEE4	61156	Send char to IEEE/Serial

		IN			OUT		
		.A	.X	.Y	.A	.X	.Y
MAKSPC	C3BB	Ary Top Hi		Ary Top Lo	unaltered		
RAMSPC	C408	same as above, start address of move in $5F, 60 (5C, 5D)					
ERROR	C437	error #					
	C48A	(direct call) $7A=FF,$7B=01($77,78) :01FF=basic lnbuf-1					
NEWLIN	C49C		Lnbf len				
GETLIN	C560	(direct call) $7A=FF,$7B=01($77,78) :01FF=basic lnbuf-1					
CRNCH	C579	.X=lnbuf len ($0200,X)=#$00					
FINLIN	C613	StrBasLo	StrBasHi				
STXTPT	C68E	direct call			StrBasHi		
CONT	C857	Cur Lin Lo		Cur Lin Hi			
DECBIN	C96B	address of text in Chrget ptr: $7A,7B ($77,78)					
	CAD3	direct call			LF(#$0A)		
	CAD7	direct call			LF(#$0A)		
PRTSTR	CB1E	addr Lo		addr Hi			
	CB24	length	addr in $22,23 (1F,20)				
	CB47	char			char		
FRMEVL $	CD9E	address of expression			addr Lo		addr Hi
FRMEVL #	CD9E	in Chrget pointer			result in Acc #1		
SYNCHR	CEFF	direct call			char		
LPACHK	CEFA	direct call			char		
RPACHK	CEF7	direct call			char		
FNDVAR	D0E7				VarAddrLo		VarAddrHi
RETVP	D185	name in $45,46 (42,43)			VarAddrLo		VarAddrHi
	D79E					data	
	D7B5	address= (Chrget ptr)			float pt result in Acc#1		
	D7B9		addr Lo	addr Hi	float pt result in Acc#1		
GETAD	D7EB	address= (Chrget ptr)			.X=pram2,pram1 in Acc#1 (fxd)		
LAPLUS	D867	addr Lo		addr Hi	float pt result in Acc#1		
TIMES	DA28	addr Lo		addr Hi	float pt result in Acc#1		
MULTEN	DAE2				result in Acc#1		
LODFAC	DBA2	addr Lo		addr Hi			
	DBD7	addr Lo	addr Hi				
PRTFIX	DDCD	Value Hi	Value Lo				
FLTASC	DDDD	#$00		#$01			
SCRNOUT	E742	char					
KMSGSH	F1E6			offset			
TALK	EE14	dev#					
LISTEN	EE17	dev#					
CSECOND	FF93	SA OR $60					
CIOUT	EEE4	char					

VIA1

← = OUT → = IN courtesy of srowe

PA		PB		CA	
serial ATN out ←	7	RS-232 DSR in →	7	cassette motor ←	2
cassette switch →	6	RS-232 CTS in →	6	restore key →	1
joy fire/lightpen →	5			**CB**	
joystick left →	4	RS-232 DCD in →	4	RS-232 Tx data ←	2
joystick down →	3	RS-232 RI in →	3	RS-232 Tx data →	1
joystick up →	2	RS-232 DTR out ←	2		
serial CLK in →	1	RS-232 RTS out ←	1	RS-232 Tx	T1
serial DATA in →	0	RS-232 Rx data →	0	RS-232 Tx	T2

VIA2

PA		PB		CA	
keyboard row →	7	joystick right ⇆	7	serial CLK out ←	2
keyboard row →	6	keyboard column ←	6	cassette read →	1
keyboard row →	5	keyboard column ←	5	**CB**	
keyboard row →	4	keyboard column ←	4	serial DATA out ←	2
keyboard row →	3	keyboard column/ cassette write ←	3	serial SRQ in →	1
keyboard row →	2	keyboard column ←	2		
keyboard row →	1	keyboard column ←	1	jiffy clock, cassette read	T1
keyboard row →	0	keyboard column ←	0	serial,cassette read/write	T2

RS-232

$293 $294 $FFC0

OPEN <file number>,2,0,"<control><command>"

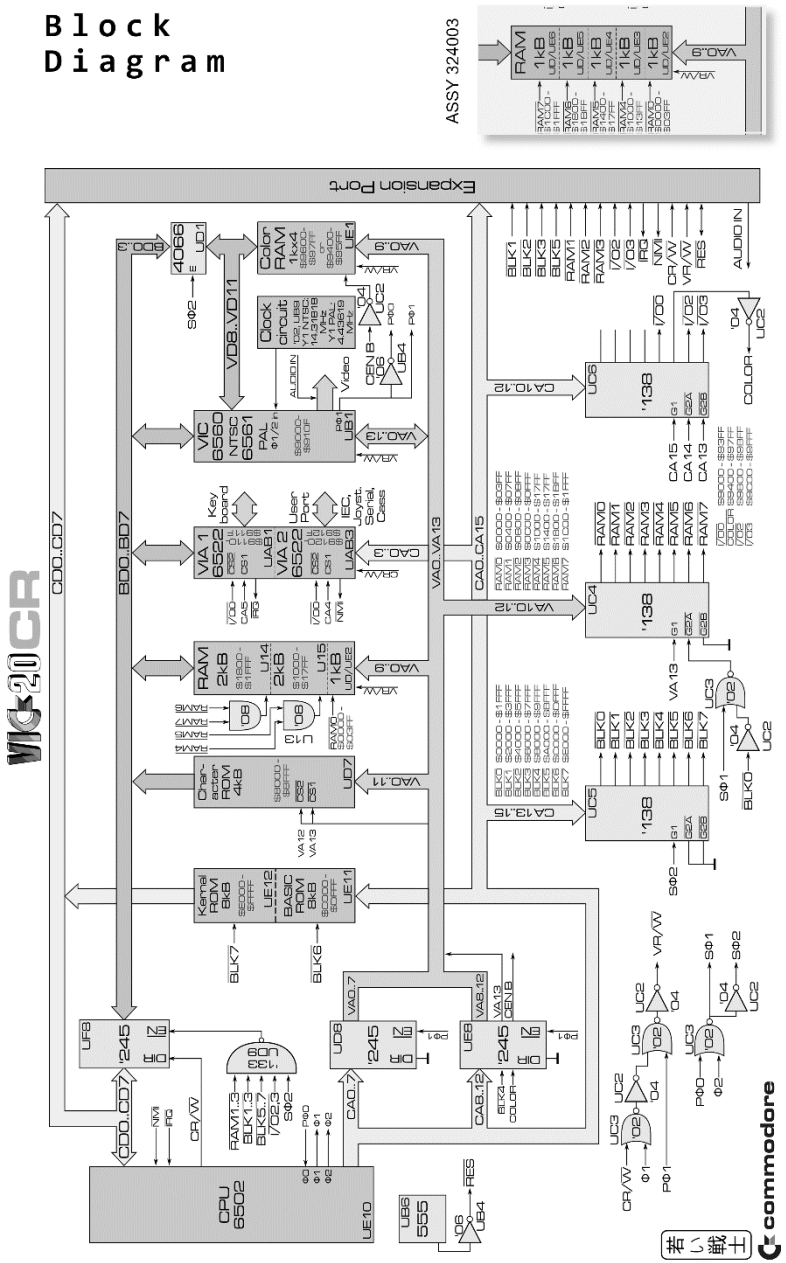
RS-232 CONTROL

BIT	7	6	5	4	3	2	1	0	
	|	|	|	x	|	|	|	|	**BAUD**
0: 1 stop bit		|	|		0	0	0	1	50
1: 2 stop bits		|	|		0	0	1	0	75
		|	|		0	0	1	1	110
WORD LENGTH		|	|		0	1	0	0	134.5
8 bits	0	0			0	1	0	1	150
7 bits	0	1			0	1	1	0	300
6 bits	1	0			0	1	1	1	600
5 bits	1	1			1	0	0	0	1200
					1	0	0	1	1800
					1	0	1	0	2400
					1	0	1	1	3600

RS-232 COMMAND

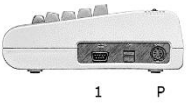
BIT	7	6	5	4	3	2	1	0	
	|	|	|	|	x	x	x	|	
PARITY DISABLED	-	-	0	|				0	3 LINE
ODD PARITY	0	0	1	|				1	X LINE
EVEN PARITY	0	1	1	\					
MARK TRANSMIT	1	0	1		0	FULL DUPLEX			
SPACE TRANSMIT	1	1	1		1	HALF DUPLEX			

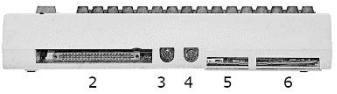
Block Diagram



courtesy of Sven Petersen
http://tech.guitarsite.de/vic-20_block.html

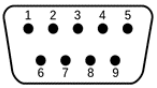
PORTS





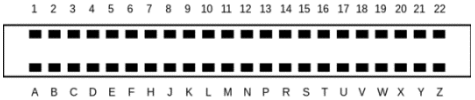
1 GAME PORT
2 EXPANSION PORT
3 AUDIO / VIDEO
4 SERIAL I / O
5 CASSETTE PORT
6 USER PORT

GAME PORT 1



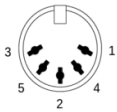
1	VIA #1 PA2 (JOY 0 up)
2	VIA #1 PA3 (JOY 1 down)
3	VIA #1 PA4 (2 left), paddle Left fire
4	VIA #2 PB7 (3 right), paddle Right fire
5	POT Y
6	VIA #2 PA7 (joy button), (light pen)
7	+5V (Max. 100mA)
8	GND
9	POT X

EXPANSION PORT 2



1	GND	system ground	A	GND	system ground
2	CD0	data line 0	B	CA0	address line 0
3	CD1	data line 1	C	CA1	address line 1
4	CD2	data line 2	D	CA2	address line 2
5	CD3	data line 3	E	CA3	address line 3
6	CD4	data line 4	F	CA4	address line 4
7	CD5	data line 5	H	CA5	address line 5
8	CD6	data line 6	J	CA6	address line 6
9	CD7	data line 7	K	CA7	address line 7
10	/BLK 1	Memory block 1 - $2000	L	CA8	address line 8
11	/BLK 2	Memory block 2 - $4000	M	CA9	address line 9
12	/BLK 3	Memory block 3 - $6000	N	CA10	address line 10
13	/BLK 5	Memory block 5 - $A000	P	CA11	address line 11
14	/RAM 1	1K RAM 1 - $0400	R	CA12	address line 12
15	/RAM 2	1K RAM 2 - $0800	S	CA13	address line 13
16	/RAM 3	1K RAM 3 - $0C00	T	/I/O2	input/output 2 - $9800
17	V R/W	VIC HI read - / LO write	U	/I/O3	input/output 3 - $9C00
18	C R/W	CPU HI read - / LO write	V	SØ2	S Phi 2
19	/IRQ	interrupt request	W	/NMI	non maskable interrupt
20	NC	not connected	X	/RESET	reset pin
21	+5V DC	supply voltage +5V DC	Y	NC / …	not connected (CR audio IN)
22	GND	system ground	Z	GND	system ground

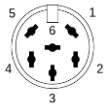
AUDIO / VIDEO 3



1	+5V REG (euro +6V DC 10ma)	RED
2	GND	
3	Audio	GREY
4	Video Low (composite)	BLACK
5	Video High (composite)	WHITE

Radio Shack Part 42-2394

SERIAL I/O 6 pin DIN 4



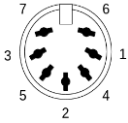
1	SRQ IN
2	GND
3	VIA #1 PA7, ATN out
4	VIA #2 CA2, Clock in/out
5	VIA #2 CB2, DATA in/out
6	NC (Reset)

CASSETTE PORT 5

A-1	GND	Ground
B-2	+ 5V	5 Volt DC
C-3	MOTOR	6 Volt motor CA2
D-4	READ	Data Input, Read CA1
E-5	WRITE	Data Output, Write PB3
-6	SENSE	Key press detection PA6

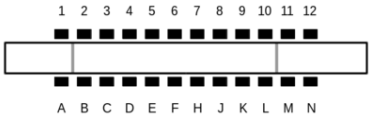
Tape information on page 119

CR POWER P



1	GND (5 Volt)
2	GND (5 Volt)
3	GND (5 Volt)
4	+5V DC or NC
5	+5V DC
6	9V AC1
7	9V AC2

USER PORT 6



MODES:
3 line (*high)
X line
user defined

1	GND	Ground	A	GND	(AA)	GND		3	X	
2	+5V	(100mA max)	B	CB1	(BB)	Rx Sin	IN	3	X	
3	RESET	COLD start	C	PB0	(BB)	Rx Sin	IN	3	X	
4	PA2	Joy UP	D	PB1	(CA)	RTS	OUT	3*	X	
5	PA3	Joy DOWN	E	PB2	(CD)	DTR	OUT	3*	X	
6	PA4	Joy LEFT, Pad X fire	F	PB3	(CE)	DRI	IN			U
7	PA 5	pen and JOY fire	H	PB4	(CF)	DCD	IN		X	
8	PA6	Tape sense switch	J	PB5	( )	XXX	IN			U
9	PA7	Serial ATN out	K	PB6	(CB)	CTS	IN		X	
10	9VAC	VIC transformer	L	PB7	(CC)	DSR	IN		X	
11	9VAC	VIC transformer	M	CB2	(BA)	Sout	OUT	3	X	
12	GND	Ground	N	GND	(AB)	GND		3	X	U

RS-232 information on page 104

KEYBOARD	$00C5 $028D
PEEK(197) or PEEK(203)	wait until a key is pressed: WAIT 197, 64, 64 wait until no key is pressed: WAIT 197, 64

0	1	12	I	26	X	39	F1	52	O
1	3	13	P	27	V	41	S	53	@
2	5	14	*	28	N	42	F	54	↑
3	7	15	Rt	29	,	43	H	55	F5
4	9	17	A	30	/	44	K	56	2
5	+	18	D	31	↕	45	:	57	4
6	£	19	G	32	Sp	46	=	58	6
7	De	20	J	33	Z	47	F3	59	8
8	←	21	L	34	C	48	Q	60	0
9	W	22	;	35	B	49	E	61	–
10	R	23	↔	36	M	50	T	62	Ho
11	Y	24	St	37	.	51	U	63	F7

←	1	2	3	4	5	6	7	8	9	0	+	–	£	CLR	INS	F1
8	0	56	1	57	2	58	3	59	4	60	5	61	6	62	7	39
	Q	W	E	R	T	Y	U	I	O	P	@	*	↑			F3
	48	9	49	10	50	11	51	12	52	13	53	14	54			47
RUN STOP	A	S	D	F	G	H	J	K	L	:	;	=	return			F5
24	17	41	18	42	19	43	20	44	21	45	22	46	15			55
	Z	X	C	V	B	N	M	,	.	/			↕	↔		F7
	33	26	34	27	35	28	36	29	37	30			31	23		63
					space 32								no key 64			

additional keyboard information on page 64

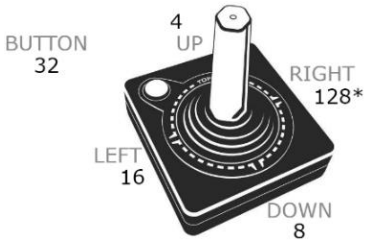
Keyboard Matrix

	0	1	2	3	4	5	6	7	8
A	1	←	ct	st	sp	C=	Q	2	
B	3	W	A	Ls	Z	S	E	4	
C	5	R	D	X	C	F	T	6	
D	7	Y	G	V	B	H	U	8	
E	9	I	J	N	M	K	O	0	
F	+	P	L	,	.	:	@	-	
G	£	*	;	/	Rs	=	↑	Ho	
H	De	Rt	⇔	↕	F1	F3	F5	F7	
I									Rs

Write to Port B ($9120) Read from Port A ($9121)

Keyboard Connector Pins

1	Ground	11	Column 1
2	[key]	12	Column 0
3	RESTORE key	13	Row 7
4	+5 volts	14	Row 6
5	Column 7, Joy 3	15	Row 5
6	Column 6	16	Row 4
7	Column 5	17	Row 3
8	Column 4	18	Row 2
9	Col 3, Tape Write(E5)	19	Row 1
10	Column 2	20	Row 0



JOYSTICK			$911F $9120 $9122
$911F 37151	Port A	R-------	Joy Right
$9120 37152	Port B	--FLDU--	Joy Fire, Left, Down, Up

UP	(PEEK(37151)AND4)=0	**4**
DOWN	(PEEK(37151)AND8)=0	**8**
LEFT	(PEEK(37151)AND16)=0	**16**
RIGHT *	(PEEK(37152)AND128)=0	**128**
BUTTON	(PEEK(37151)AND32)=0	**32**

* enclose between POKE37154,127 and POKE37154,255

PADDLES	$9008 $9009
PADDLE A	PEEK(36872)
PADDLE B	PEEK(36873)
BUTTON A	(PEEK(37151)AND16)=0
BUTTON B *	(PEEK(37152)AND128)=0

JITTER REDUCTION A	XF=A*XF+B*PEEK(36872):X=INT(XF+.5)
JITTER REDUCTION B	YF=A*YF+B*PEEK(36873):Y=INT(YF+.5)

* enclose between POKE37154,127 and POKE37154,255

LIGHT PEN	$9006 $9007
Pen X axis	INT((PEEK(36870)-28)/4)
Pen Y axis	INT((PEEK(36871)-24)/4)

WARNING: Atari compatible, but some pin-compatible controllers such as Sega may damage a VIC

PC	SR	AC	XR	YR	SP
program counter	status register	accumulator	index register X	index register Y	stack pointer

A	ASSEMBLE .A <address><opcode><operand>
B	BREAKPOINT .B <address>[,<times>]
D	DISASSEMBLE .D <address>[,<address>]
E	ENABLE VIRTUAL ZERO PAGE .E <address>
F	FILL MEMORY .F <address>,<address>,<value>
G	GO .G [<address>]
H	HUNT .H <address>,<address>,<data>
I	INTERPRET .I <address>[,<address>]
J	JUMP .J
L	LOAD .L "<filename>",<device>
M	MEMORY DISPLAY .M <address>[,<address>]
N	NUMBER *(W is optional command for word table)* .N <address>,<address>,<offset>,<low limit>,<high limit>[,W]
Q	QUICK TRACE .Q [<address>]
R	REGISTERS .R
RB	REMOVE BREAKPOINT .RB
S	SAVE .S "<filename>",<device>,<address>,<address>
T	TRANSFER .T <address>,<address>,<destination address>
W	WALK .W [<address>]
X	EXIT TO BASIC .X

$	hex convert	Z	switch case	!	zero page address
#	decimal convert	+	add	<	shift bits left
%	binary convert	−	subtract	>	shift bits right
"	ascii convert	&	checksum	( )	indirect addressing

machine language instructions on page 82

SUPER EXPANDER	
CHAR	Puts text on the graphic screen `CHAR<row>,<column>,"<text>"`
CIRCLE	Draws a circle, ellipse, or arc `CIRCLE<color>,x,y,xradius,yradius [,<as>,<af>]`
COLOR	Select colors `COLOR<screen>,<border>,<character>,<auxiliary>`
DRAW	Plots a line between two points `DRAW<color>[TO],x1,y1 TO x2,y2 [TO … xn, yn]`
GRAPHIC	Prepares the screen for graphics `(0 text,1 multi,2 hi-res,3 mixed,4 text)` `GRAPHIC<mode>`
KEY	Sets or displays function key shortcuts `KEY[<key number>,"<string>"]`
PAINT	Fills an enclosed area in a color `PAINT<color>,x,y`
POINT	Plots a single point `POINT<color>,x1,y1 x2,y2, …`
REGION	Select character color `REGION<color>`
SCNCLR	Clears a graphic screen `SCNCLR`
SOUND	Sets sound registers `SOUND<voice 1>,<voice 2>,<voice 3>,<voice 4>,<volume>`

RCOLR(X)	Returns the value in a color register `RCOLR(X)`
RDOT(X,Y)	Returns the color of a point on the screen `RDOT(X,Y)`
RGR(X)	Returns the current graphic mode `RGR(X)`
RJOY(X)	Returns status of the joystick `RJOY(X)`
RPEN(X)	Returns the position of the light pen `RPEN(X)`
RPOT(X)	Returns the position of the game paddle `RPOT(X)`
RSND(X)	Returns the value in a sound register `RSND(X)`

CTRL ←	MUSIC MODE		
CDEFGAB	music notes	O	select music octave
P	display all music chars	T	set tempo duration
Q	cancel display of P	R	play a musical rest
V	set volume	#	play a sharp note
S	select music voice	$	play a flat note

music and sound information on page 46

PROGRAMMER'S AID	SYS 28681
AUTO	Display line being executed `AUTO<start line>,<interval>`
CHANGE	Display line being executed `CHANGE<old text>,<new text>,<line>[-<line>,<line>-]`
DELETE	Display line being executed `DELETE<line>[-<line>,<line>-]`
DUMP	Display all variables `DUMP`
EDIT	Change to EDIT mode `EDIT`
FIND	Display line being executed (FIND "<string>" will find text strings only) `FIND<code>,<line>[-<line>,<line>-]`
HELP	Highlight error `HELP`
KEY	Display line being executed `KEY<key number>,"<code>"`
KILL	Stop AID cartridge `KILL`
MERGE	Display line being executed `MERGE"<program>",<device>`
OFF	Cancel TRACE or STEP mode `OFF`
PROG	Change to PROGRAM mode `PROG`
RENUMBER	Display line being executed `RENUMBER<start line>,<interval>`
STEP	Halt after each line is executed `STEP`
TRACE	Display line being executed `TRACE`

Ctrl-A	Scroll up	**Ctrl-N**	Erase all
Ctrl-E	Cancel quote	**Ctrl-Q**	Scroll down
Ctrl-L	Erase after cursor	**Ctrl-U**	Erase line

		EDIT MODE			EDIT MODE
F1	LIST	LIST	**F2**	MID$(	AUTO
F3	RUN	RUN	**F4**	LEFT$(	DELETE
F5	GOTO	FIND	**F6**	RIGHT$(	CHANGE
F7	INPUT	TRACE	**F8**	CHR$(	STEP

Hold CONTROL key for F9 – F12

F9	EDIT	PROG	**F10**	GOSUB	RENUMBER
F11	RETURN	MERGE	**F12**	STR$(	OFF

READ ERROR CHANNEL	F = error code. E$ = error string, T = track, S = sector

```
1 OPEN15,8,15:INPUT#15,F,E$,T,S;:PRINTF;E$;T;S:CLOSE 15
```

DOS ERROR CODES			
1	files scratched response.	**39**	command file not found
2-19	unused error messages.	**50**	record not present
20	block header not found on disk	**51**	overflow in record
21	sync character not found.	**52**	file too large
22	data block not present	**60**	file open for write
23	checksum error in data	**61**	file not open
24	byte decoding error	**62**	file not found
25	write-verify error	**63**	file exists
26	attempt with write protect on	**64**	file type mismatch
27	checksum error in header	**65**	no block
28	data extends into next block	**66**	illegal track or sector
29	disk ID mismatch	**67**	illegal system track or sector
30	general syntax error	**70**	no channels available
31	invalid command	**71**	directory error
32	command too long	**72**	disk full or directory full
33	invalid filename	**73**	power up, or write w/ DOS mismatch
34	no file name given	**74**	drive not ready

BASIC 4.0							
APPEND	BACKUP	CATALOG	COLLECT	CONCAT	COPY	DCLOSE	DIRECTORY
DLOAD	DOPEN	DSAVE	HEADER	RECORD	RENAME	SCRATCH	
BUTI TOOLKIT							
APPEND	AUTO	DELETE	DUMP	EDIT	FIND	HELP	KILL
OFF	RENUMBER	REPEAT	STEP	TRACE	UNNEW	VIC	
CASSETTE TURBO							
AUTO	CLOAD	CMERGE	CSAVE	CVERIFY	DELETE	DUMP	
FIND	HELP	KILL	RENUMBER	RESET	TRACE		
COMBASIC							
AINST	CLI	CLINE	COPY	CURS	DEX	ERR	EXEC	FLINE
GTSTR	HELP	INIT	INSTR	IRQCL	LIM	LINE	LOD	MEM
MOVE	NOSTP	OFF	PRT	PUT	REST	SAV	SEI	SET
SIM	SRT	STACK	STAY	SUB	USER	WR	:	
HI-RES TOOL KIT SYS 45056							
AU$	CIRCLE	CLEAR	DEH	DRAW	DUMP	FILL	
FIND	GRAPHICS	HELP	INVERT	LINE	OFF	POT	
PUT	RENUMBER	REST	SET	TEXT	TRACE		
VIC TOOL SYS 45056							
AU$	AUTO	DE?	DUMP	FIND	HELP	OFF	RENUMBER	TRACE
WATERLOO BASIC							
CALL	ELSE	ENDIF	ENDLOOP	ENDPROC	LOOP	PROC	QUIT	UNTIL	WHILE

DEVICE NUMBER			PAGE
0	Keyboard	IN	106
1	Cassette	IN & OUT	119
2	RS-232	IN & OUT	104
3	Screen	IN & OUT	30
4	Printer	OUT	24
5	Printer	OUT	24
6	Plotter	IN & OUT	--
7	Plotter	IN & OUT	--
8-30	Disk	IN & OUT	115

COMMAND		
Tape	0	Read tape file
Tape	1	Write tape file
Tape	2	Write w/ END marker
Disk	0	Load from disk
Disk	1	Save to disk
Disk	2-14	Open data channel
Disk	15	Command channel
Printer	0	Upper case / graphics
Printer	7	Upper case / lower

1541 DIP	1/A	2/B
DEVICE 8	ON	ON
DEVICE 9	OFF	ON
DEVICE 10	ON	OFF
DEVICE 11	OFF	OFF

FILENAME EXTENSIONS			
PRG	program file	DEL	deleted file
SEQ	sequential file	*	splat file
USR	user file	<	protected file
REL	relative file	DIR	directory

CHANGE DISK DEVICE NUMBER (TEMPORARY)

```
OPEN15,8,15:PRINT#15,"M-W";CHR$(119);CHR$(0);CHR$(2);
CHR$(<device number+64>):CLOSE15
```

DEVICE STATUS				ST $FFB7
BIT	VALUE	**TAPE**	**SERIAL**	**RS-232**
0	0	OK	OK	OK
0	1		Write time out	Parity error
1	2		Read time out	Framing error
2	4	Data block too short		Rec buffer overrun
3	8	Data block too long		
4	16	Verify read error		CTS signal missing
5	32	Checksum error		
6	64	End of file (EOI)	End of file (EOI)	RTS signal missing
7	128		Device not present	Break detected

BASIC LOAD COMMANDS

Load a BASIC program

```
LOAD"<filename>",<device>,[<command>]
```

Command 1:absolute location

Load first or next program

```
LOAD"*",<device>
```

Load program starting with specified letter (wildcard)

```
LOAD"A*",<device>
```

Load program starting and ending with letters (wildcard)

```
LOAD"A?B",<device>,[<command>]
```

DIRECTORY COMMANDS

Load disk directory

```
LOAD"$",<device>
```

Name Wildcard: Load directory of programs starting with specified letter (A)

```
LOAD"$A*",<device>
```

Type Wildcard: Load directory of specified file type

```
LOAD"$*=S",<device>    Sequential files
LOAD"$*=P",<device>    Program files
LOAD"$*=R",<device>    Relative files
LOAD"$*=U",<device>    User files
```

Load list of subdirectories

```
LOAD"$*=D",<device>
```

Load disk directory name

```
LOAD"$$",<device>
```

SAVE AND VERIFY COMMANDS

Save a BASIC program @ overwrite not recommended

```
SAVE"<filename>",<device>,[<command>]
```

Command 1:absolute location, 2:end-of-tape marker, 3:command 1 and 2

Verify a file

```
VERIFY"<filename>",<device>,[<command>]
```

Command 1:absolute location

COMMAND CHANNEL	or PRINT#15, . . .
Enter VIC 20 mode (faster)	
OPEN15,8,15,"UI-":CLOSE15	OFF: OPEN15,8,15,"UI+":CLOSE15
Create folder	
OPEN15,8,15,"C0:<filename>":CLOSE15	
Go to parent folder	
OPEN15,8,15,"CD:←":CLOSE15	
Initialize the disk drive	
OPEN15,8,15,"I0":CLOSE15	
Format a new disk	
OPEN15,8,15,"N0:<disk name>,<2 character ID>"	
Validate a disk	
OPEN15,8,15,"V0":CLOSE15	
Copy a file on the same disk	
OPEN15,8,15,"C0:<new name>,<old name>"	
Delete (scratch) a file (or files)	
OPEN15,8,15,"S0:<filename>":CLOSE15	
Rename a file	
OPEN15,8,15,"R0:<new name>,<old name>"	
Merge (combine) files	
PRINT#15,"C0:<newname>=0:<file1>,[0:<file2>,0:<file3>…],"D0=D1"	

RELATIVE FILES
OPEN file for READ and WRITE
OPEN2,8,3,"<filename>"
Create a file with X records
OPEN2,8,3,"<filename>,L,"+CHR$(X)
Extend file with E records
OPEN2,8,3,"<filename>,L,"+CHR$(X+E)
POSITION to record
PRINT#2,"P"CHR$(3)CHR$(<LO>)CHR$(<HI>)
POSITION to a character within a record
PRINT#2,"P"CHR$(3)CHR$(<LO>)CHR$(<HI>)CHR$(<POSITION>)

SEQUENTIAL FILES	
OPEN file for READ	
OPEN2,8,2,"<filename>,S,R"	
OPEN file for WRITE	@ overwrite not recommended
OPEN2,8,2,"<filename>,S,W"	
OPEN file for APPEND	
OPEN2,8,2,"<filename>,S,A"	
OPEN file for EMERGENCY (read improperly closed file)	
OPEN2,8,2,"<filename>,S,M"	

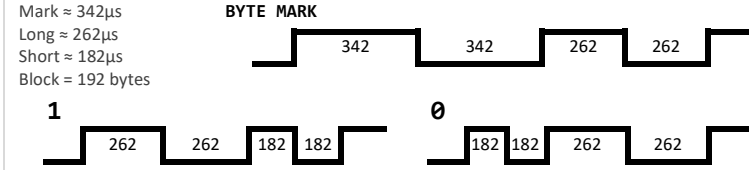
DISK DIRECT COMMANDS	OPEN15,8,15, ... :CLOSE15
Block-Allocate	
PRINT#15,"B-A";<channel>;<drive>;<track>;<sector>	
Block-Execute	
PRINT#15,"B-E";<channel>;<drive>;<track>;<sector>	
Block-Free	
PRINT#15,"B-F";<channel>;<track>;<sector>	
Buffer-Pointer	
PRINT#15,"B-P";<channel>;<byte>	
Block-Read	
PRINT#15,"B-R";<channel>;<drive>;<track>;<sector>	
Block-Write	
PRINT#15,"B-W";<channel>;<drive>;<track>;<sector>	
Mem-Execute	
PRINT#15,"M-E";CHR$(<addr lo>)CHR$(<addr hi>)	
Memory-Read	
PRINT#15,"M-R";CHR$(<addr lo>)CHR$(<addr hi>)CHR$(#)	
Memory-Write	
PRINT#15,"M-W";CHR$(<ad lo>)CHR$(<ad hi>)CHR$(#); CHR$(<data>)	
Memory-Write: “Disk Drive Knock Reduction” command	
PRINT#15,"M-W";CHR$(106)CHR$(0)CHR$(1); CHR$(113)	
User Command	
PRINT#15,"U<x>";<channel>;<drive>;<track>;<sector>	

JiffyDOS	SYS 58492
@	read the disk drive error channel
@C:newfile=file	copy a file on the same diskette
@I	initialize the disk drive
@N:diskname,ID	format a new diskette
@N:diskname	short new
@Q	disable the JiffyDOS commands
@R:[newname]=[oldname]	rename a file
@S:[file1],[file2]...	scratch a file (files)
@UJ	reset the disk drive
@V	validate a disk
@$	display a disk directory
@#device	set the default device number
/[filename]	load a BASIC program
↑[filename]	load and run a BASIC program
%[filename]	load a machine language program
←[filename]	save a BASIC program
@B	disable the 1541 head rattle
@D:[filename]	list a BASIC program from disk
@F	disable the function keys
@G	set interleave gap size
@L:[filename]	lock/unlock a file
@O	un-new a BASIC program
@P	toggle printer output
@T:[filename]	list an ASCII file from disk
@X	set destination device number
*"filename"type	copy a file
F[filename]	load and run machine language file
´[filename]	verify a file
CONTROL + S	stop / freeze listing
CONTROL + A	toggle all files for copy
CONTROL + D	default drive toggle
CONTROL + P	screen dump
CONTROL + W	toggle single file for copy
SHIFT RUN/STOP	load and run 1st program on disk
SYS 58492	re-enable JiffyDOS commands

TAPE WAVEFORMS AND SEQUENCE

Bit 1 = LLSS, Bit 0 = SSLL

Mark ≈ 342µs
Long ≈ 262µs
Short ≈ 182µs
Block = 192 bytes



10s Leader	Header block	Inter-block gap	Header Block repeated	3s Leader	Program	Inter-clock Gap	Program repeated	End	End repeated

USEFUL MEMORY UTILITIES

courtesy of Mike via Denial forums

SAVE memory block

```
SYS57809(N$),<device>:POKE193,<start_lo>:POKE194,<start_hi>
POKE780,193:POKE781,<end_lo>:POKE782,<end_hi>:SYS65496
```

SAVE memory block (and force load address on tape)

```
SYS57809(N$),<device>,1:POKE193,<start_lo>:POKE194,<start_hi>
POKE780,193:POKE781,<end_lo>:POKE782,<end_hi>:SYS65496
```

LOAD memory block ('relative' or 'relocating' load)

```
SYS57809(N$),<device>:POKE780,0:POKE781,<start_lo>:POKE782,
<start_hi>:SYS65493
```

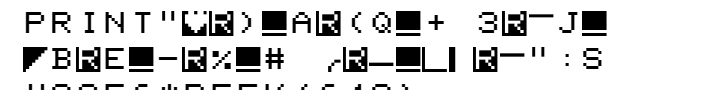
LOAD memory block ('absolute' load)

```
SYS57809(N$),<device>,1:POKE780,0:SYS65493
```

UNNEW

```
PRINT"{CLR,RVS ON}){RVS OFF}A{RVS ON}(Q{RVS OFF}+{SPACE}3
{RVS ON,SHIFT-E}J{RVS OFF,SHIFT-POUND}B{RVS ON}E{RVS OFF}-
{RVS ON}%{RVS OFF}#{SPACE,SHIFT-U,RVS ON,SHIFT-F,RVS OFF,SHIFT-
L,C=-H,RVS ON,SHIFT-D}":SYS256*PEEK(648)
```

For visual reference:



As homage to the original VIC 20 user manual, I made several tiny games, each restricted to five lines of BASIC code (one full screen). I always found type-in programs to be a little intimidating, so a screenshot of each program's LIST is included to assist proofreading.

B-5 BOMBER

By Jeffrey Daniels

Clear the skyline to land your bomber aircraft before colliding with a building. This is a one button game. Use any key of the keyboard, OR the joystick fire button, OR the paddle fire button.

	{clr}	SHIFT HOME
	{yellow}	CONTROL 8
	{down}	CRSR DOWN
	{right}	CRSR RIGHT

B-5 BOMBER PROGRAM LISTING:

```
1 E=36876:POKEE+3,105:PRINT"{clr}{yellow}{down}
  {right}BOMBER":D=30720:H=32:FORA=8167TO8183:
  FORC=0TO1+RND(A)*9
2 B=A-C*22:POKEB+D,2+(AAND3):POKEB,189:NEXT:NEXT:
  A=7745:C=8186:B=C:POKEE-2,140:G=E+1
3 POKEA,H:A=A+1:POKEA,62:POKEB,H:IFPEEK(A+1)>HTHEN
  POKEE+2,0:WAIT197,64,64:RUN
4 B=B-22*(B<C):POKEG,F*-(PEEK(B)=189):IF(PEEK(197)+
  PEEK(37151)<190)ANDB>=CTHENB=A:F=239
5 POKEB,42:POKEB+D,1:POKE198,0:POKEE+2,9:POKEC-1,233:
  POKEE,F:F=F+7*(F>128):POKEG,0:GOTO3
```

UFO RUSH

By Jeffrey Daniels

Move to the top of the screen while avoiding collision. Beat the High Score.

Use any two keyboard keys recommended below to move the UFO left or right:

Z	left	X	right	S	left	D	right	L	left	;	right
Z	left	/	right	E	left	R	right	.	left	/	right
Z	left	C	right	S	left	F	right	:	left	;	right

	{clr}	SHIFT HOME
	{white}	CONTROL 2
	{cyan}	CONTROL 4
	{cmd *}	CMD *
	{rvs on}	CONTROL 9
	{space}	SPACE BAR
	{rvs off}	CONTROL 0
	{shift £}	SHIFT £
	{yellow}	CONTROL 8
	{down}	CRSR DOWN
	{shift A}	SHIFT A
	{shift Z}	SHIFT Z

UFO RUSH PROGRAM LISTING:

```
1 E=36879:POKEE,11:POKEE-2,0:PRINT"{clr}{white}"A
  TAB(5)"{cyan}{cmd *}{rvs on}{space}UFO{space}RUSH
  {space}{rvs off}{shift £}{yellow}"TAB(18):K=1:
  IFF<ATHENF=A
2 PRINTF"{down}":POKE646,9:FORT=0TO17:PRINT"{shift A}
  {shift Z}"SPC(RND(T)*28):NEXT:A=A+1:C=8196:H=E-3:
  J=32
3 I=PEEK(197)AND3:B=C-22+(I=K)-(I=2):D=PEEK(B):POKE
  B,65:POKEC,J:POKEH,.:IFC<7724THEN1
4 G=(G+K)AND3:POKEE-K,9+G*J:C=B:POKEH,190:FORT=.TO40-
  A*2:NEXT:IFD=JTHEN3
5 POKEH+1,220:FORI=15TO0STEP-.1:POKEE-1,I:POKEE,262-
  PEEK(E):NEXT:WAIT197,64,64:A=0:GOTO1
```

LIGHTS OUT

By Jeffrey Daniels

Toggle the lights until they are all in the same state (all on or all off) in the fewest moves possible. This puzzle game starts with a random pattern on a 5 by 5 grid.

Use the keyboard to trigger the corresponding letter's light and its orthogonally adjacent neighbors. Press any other key to reset the game for a new puzzle.

	{clr}	SHIFT HOME
	{red}	CONTROL 3
	{cmd *}	CMD *
	{rvs on}	CONTROL 9
	{space}	SPACE BAR
	{black}	CONTROL 1
	{home}	HOME
	{down}	CRSR DOWN

LIGHTS OUT PROGRAM LISTING:

```
1 B=36876:POKEB+3,26:POKEB+2,9:PRINT"{clr}{red}",
  "{cmd *}{rvson}LIGHTS OUT{black}":FORC=0TO24
2 GOSUB4:POKEA,C+1:POKEA+30720,6:NEXT:FORE=1TO25:
  C=INT(RND(1)*20):GOSUB4:NEXT
3 GETA$:ON-(A$="")GOTO3:C=ASC(A$)-65:GOSUB4:W=W+1:
  PRINT"{home}{down}"W:ON-(C>-1ANDC<25)GOTO3:RUN
4 A=7840+INT(C/5)*44+(C/5-INT(C/5))*10:IFE=0THEN
  A(1)=2:A(2)=-2:A(3)=44:A(4)=-44:RETURN
5 POKEB,173+C*2:FORD=0TO4:F=A+A(D):POKEF,128+PEEK(F)
  AND255:NEXT:POKEB,0:RETURN
```

Ignore this: it is a personal cheat sheet and fills an empty space

W	move / str	**U**	undo / del	**R**	rom / all	**H**	horizontal	**G**	song / tone	**F1**	chr / col
A	move / str	**P**	pas / mrg	**8**	828 / 3k	**V**	vertical	**£**	music / rec	**F3**	screen
S	move / str	**T**	trade / tr+	**9**	r90 / rvs	**F**	flip	**=**	reset	**F7**	toggle
D	move / str	**E**	ext / brw	**+**	double	**Z**	spin	*****	isolate	**Rt**	enter
1	select/show								45 248		46 29

TRAILS

By Jeffrey Daniels

Collect the clovers. Avoid running off screen or into yourself.

Use the default WASD keys OR modify the highlighted areas of line 3.

UP	W	9	P	9	Q	48	*	14
DOWN	S	41	.	41	A	17	=	46
LEFT	A	17	L	17	O	52	A	17
RIGHT	D	18	;	18	P	13	S	41

	{clr}	SHIFT HOME
	{yellow}	CONTROL 8
	{white}	CONTROL 2
	{shift X}	SHIFT X
	{home}	HOME
	{down}	CRSR DOWN

Highlighted areas in line 3 may be changed to one of the suggested combinations listed above or any key code (see page 108).

TREASURE TRAILS PROGRAM LISTING:

```
1 DIMA(255):A=36875:PRINT"{clr}{yellow}"SPC(30)
  "TRAILS"SPC(200)"{white}{shift X}{shift X}
  {home}{down}":C=7910: B=1:A(1)=C:POKEA+3,4
2 D=D+1:I=7724+RND(1)*400:ON-(PEEK(I)><32)GOTO2:POKE
  A+4,10+RND(1)*5:POKEI,88:POKEA,163
3 C=C+B:E=PEEK(C):POKEC,9↑2:I=PEEK(197):IFI<63THEN
  B=((I=9)-(I=41))*22+(I=17)-(I=18)
4 F=F+1:POKEA,(FAND25)+215:A(G)=C:G=(G+1)*-(G<D):POKE
  A(G),32:ON-(E=88)-2*(E=32)GOTO2,3
5 PRINTD:POKEA+4,8:FORI=15TO0STEP-.1:POKEA+3,I:POKE
  A,99+I*9:NEXT:WAIT197,64,64:RUN
```

$00	0	0	$40	64	16384	$80	128	32768	$C0	192	49152
$01	1	256	$41	65	16640	$81	129	33024	$C1	193	49408
$02	2	512	$42	66	16896	$82	130	33280	$C2	194	49664
$03	3	768	$43	67	17152	$83	131	33536	$C3	195	49920
$04	4	1024	$44	68	17408	$84	132	33792	$C4	196	50176
$05	5	1280	$45	69	17664	$85	133	34048	$C5	197	50432
$06	6	1536	$46	70	17920	$86	134	34304	$C6	198	50688
$07	7	1792	$47	71	18176	$87	135	34560	$C7	199	50944
$08	8	2048	$48	72	18432	$88	136	34816	$C8	200	51200
$09	9	2304	$49	73	18688	$89	137	35072	$C9	201	51456
$0A	10	2560	$4A	74	18944	$8A	138	35328	$CA	202	51712
$0B	11	2816	$4B	75	19200	$8B	139	35584	$CB	203	51968
$0C	12	3072	$4C	76	19456	$8C	140	35840	$CC	204	52224
$0D	13	3328	$4D	77	19712	$8D	141	36096	$CD	205	52480
$0E	14	3584	$4E	78	19968	$8E	142	36352	$CE	206	52736
$0F	15	3840	$4F	79	20224	$8F	143	36608	$CF	207	52992
$10	16	4096	$50	80	20480	$90	144	36864	$D0	208	53248
$11	17	4352	$51	81	20736	$91	145	37120	$D1	209	53504
$12	18	4608	$52	82	20992	$92	146	37376	$D2	210	53760
$13	19	4864	$53	83	21248	$93	147	37632	$D3	211	54016
$14	20	5120	$54	84	21504	$94	148	37888	$D4	212	54272
$15	21	5376	$55	85	21760	$95	149	38144	$D5	213	54528
$16	22	5632	$56	86	22016	$96	150	38400	$D6	214	54784
$17	23	5888	$57	87	22272	$97	151	38656	$D7	215	55040
$18	24	6144	$58	88	22528	$98	152	38912	$D8	216	55296
$19	25	6400	$59	89	22784	$99	153	39168	$D9	217	55552
$1A	26	6656	$5A	90	23040	$9A	154	39424	$DA	218	55808
$1B	27	6912	$5B	91	23296	$9B	155	39680	$DB	219	56064
$1C	28	7168	$5C	92	23552	$9C	156	39936	$DC	220	56320
$1D	29	7424	$5D	93	23808	$9D	157	40192	$DD	221	56576
$1E	30	7680	$5E	94	24064	$9E	158	40448	$DE	222	56832
$1F	31	7936	$5F	95	24320	$9F	159	40704	$DF	223	57088
$20	32	8192	$60	96	24576	$A0	160	40960	$E0	224	57344
$21	33	8448	$61	97	24832	$A1	161	41216	$E1	225	57600
$22	34	8704	$62	98	25088	$A2	162	41472	$E2	226	57856
$23	35	8960	$63	99	25344	$A3	163	41728	$E3	227	58112
$24	36	9216	$64	100	25600	$A4	164	41984	$E4	228	58368
$25	37	9472	$65	101	25856	$A5	165	42240	$E5	229	58624
$26	38	9728	$66	102	26112	$A6	166	42496	$E6	230	58880
$27	39	9984	$67	103	26368	$A7	167	42752	$E7	231	59136
$28	40	10240	$68	104	26624	$A8	168	43008	$E8	232	59392
$29	41	10496	$69	105	26880	$A9	169	43264	$E9	233	59648
$2A	42	10752	$6A	106	27136	$AA	170	43520	$EA	234	59904
$2B	43	11008	$6B	107	27392	$AB	171	43776	$EB	235	60160
$2C	44	11264	$6C	108	27648	$AC	172	44032	$EC	236	60416
$2D	45	11520	$6D	109	27904	$AD	173	44288	$ED	237	60672
$2E	46	11776	$6E	110	28160	$AE	174	44544	$EE	238	60928
$2F	47	12032	$6F	111	28416	$AF	175	44800	$EF	239	61184
$30	48	12288	$70	112	28672	$B0	176	45056	$F0	240	61440
$31	49	12544	$71	113	28928	$B1	177	45312	$F1	241	61696
$32	50	12800	$72	114	29184	$B2	178	45568	$F2	242	61952
$33	51	13056	$73	115	29440	$B3	179	45824	$F3	243	62208
$34	52	13312	$74	116	29696	$B4	180	46080	$F4	244	62464
$35	53	13568	$75	117	29952	$B5	181	46336	$F5	245	62720
$36	54	13824	$76	118	30208	$B6	182	46592	$F6	246	62976
$37	55	14080	$77	119	30464	$B7	183	46848	$F7	247	63232
$38	56	14336	$78	120	30720	$B8	184	47104	$F8	248	63488
$39	57	14592	$79	121	30976	$B9	185	47360	$F9	249	63744
$3A	58	14848	$7A	122	31232	$BA	186	47616	$FA	250	64000
$3B	59	15104	$7B	123	31488	$BB	187	47872	$FB	251	64256
$3C	60	15360	$7C	124	31744	$BC	188	48128	$FC	252	64512
$3D	61	15616	$7D	125	32000	$BD	188	48384	$FD	253	64768
$3E	62	15872	$7E	126	32256	$BE	190	48640	$FE	254	65024
$3F	63	16128	$7F	127	32512	$BF	191	48896	$FF	255	65536

BASIC ERROR CODES	$C437
BAD SUBSCRIPT	attempt outside of array DIM range
CAN'T CONTINUE	cannot recall spot to resume program
DEVICE NOT PRESENT	attempt to access unavailable device
DIVISION BY ZERO	disallowed math
EXTRA IGNORED	too many items of data entered
FILE DATA	received string data when numeric expected
FILE NOT FOUND	failed to find file on device
FILE NOT OPEN	command sent to device not open
FILE OPEN	attempt to open file already open
FORMULA TOO COMPLEX	expression too long to process
ILLEGAL DIRECT	statement used in program mode
ILLEGAL DEVICE NUMBER	attempted file operation on unsuitable device
ILLEGAL QUANTITY	a number falls outside of range allowed
LOAD	device failed to LOAD
NEXT WITHOUT FOR	cannot find corresponding FOR
NOT INPUT FILE	file specified for output only
NOT OUTPUT FILE	file specified for input only
OUT OF DATA	could not find corresponding DATA
OUT OF MEMORY	no more RAM available (stack overflow)
OVERFLOW	number exceeds allowable size
REDIM'D ARRAY	array may only have one defining DIM
REDO FROM START	INPUT does not match expected value
RETURN W/O GOSUB	cannot find corresponding GOSUB
STRING TOO LONG	string exceeds 255 characters
SYNTAX	unrecognized command
TYPE MISMATCH	expected string or number but got other
UNDEF'D FUNCTION	FN employed without DEF
UNDEF'D STATEMENT	attempted GOTO or GOSUB line not found
VERIFY	program does not match file on device

DOS error codes on page 113 Kernal error codes on page 99

ABS	A shift B	A|	**INPUT**			**RETURN**	RE shift T	RE|
AND	A shift N	A/	**INPUT#**	I shift N	I/	**RIGHT$**	R shift I	R┐
ASC	A shift S	A♥	**INT**			**RND**	R shift N	R/
ATN	A shift T	A|	**LEFT$**	LE shift F	LE_	**RUN**	R shift U	R┌
CHR$	C shift H	C|	**LEN**			**SAVE**	S shift A	S♠
CLOSE	CL shift O	CL┌	**LET**	L shift E	L¯	**SGN**	S shift G	S|
CLR	C shift L	C_	**LIST**	L shift I	L┐	**SIN**	S shift I	S┐
CMD	C shift M	C\	**LOAD**	L shift O	L┌	**SPC(**	S shift P	S┐
CONT	C shift O	C┌	**LOG**			**SQR**	S shift Q	S●
COS			**MID$**	M shift I	M┐	**STATUS**	ST	ST
DATA	D shift A	D♠	**NEW**			**STEP**	ST shift E	ST¯
DEF	D shift E	D¯	**NEXT**	N shift E	N¯	**STOP**	S shift T	S|
DIM	D shift I	D┐	**NOT**	N shift O	N┌	**STR$**	ST shift R	ST_
END	E shift N	E/	**ON**			**SYS**	S shift Y	S |
EXP	E shift X	E↑	**OPEN**	O shift P	O┐	**TAB**	T shift A	T♠
FN			**OR**			**TAN**		
FOR	F shift O	F┌	**PEEK**	P shift E	P¯	**THEN**	T shift H	T |
FRE	F shift R	F_	**POKE**	P shift O	P┌	**TIME**	TI	TI
GET	G shift E	G¯	**POS**			**TIME$**	TI shift 4	TI$
GET#			**PRINT**	shift /	?	**TO**		
GOSUB	GO shift S	GO♥	**PRINT#**	P shift R	P_	**USR**	U shift S	U♥
GO			**READ**	R shift E	R¯	**VAL**	V shift A	V♠
GOTO	G shift O	G┌	**REM**			**VERIFY**	V shift E	V¯
IF			**RESTORE**	RE shift S	RE♥	**WAIT**	W shift A	W♠

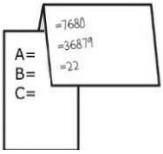
.A	Assemble
.B	Breakpoint
.D	Disassemble
.E	Enable Zero Page
.F	Fill Memory
.G	Go
.H	Hunt
.I	Interpret
.J	Jump
.L	Load
.M	Memory Display
.N	Number
.Q	Quick Trace
.R	Registers
.RB	Remove Breakpoint
.S	Save
.T	Transfer
.W	Walk
.X	Exit to BASIC

ADC	CPY	PLP	ANC/2	
AND	DEC	ROL	ANE	
ASL	DEX	ROR	ARR	
BCC	DEY	RTI	DCP	
BCS	EOR	RTS	ISC	
BEQ	INC	SBC	LAS	
BIT	INX	SEC	LAX	
BMI	INY	SED	LXA	
BNE	JMP	SEI	NOP	
BPL	JSR	STA	RLA	
BRK	LDA	STX	RRA	
BVC	LDX	STY	SAX	
BVS	LDY	TAX	SBX	
CLC	LSR	TAY	SHA	
CLD	NOP	TSX	SHX	
CLI	ORA	TXA	SHY	
CLV	PHA	TXS	SLO	
CMP	PHP	TYA	TAS	
CPX	PLA	ALR	USBC	

43-44	Prog start pointer L/H
45-46	Num var point L/H (prg end)
55-56	Top of free RAM point L/H
197	Key currently pressed [64]
198	# of chars in buffer
631	Keyboard buffer (10)
646	Current cursor color
653	Shift 1, Control 2, C= 4
828-	Tape buffer (192)
4096-	User map 8K +30208 col
7168-	User Char mem
7680-	Screen map 0K +30720 col
32768	Character ROM (upper)
36869	Video Matrix [240,255]
36874	Sound 75, 76, 77 noise
36878	Aux color / volume
36879	Screen color
37151	Joystick UDLF (page 109)
64802	Reset {64812 skip cart}

A	1
B	2
C	3
D	4
E	5
F	6
G	7
H	8
I	9
J	10
K	11
L	12
M	13
N	14
O	15
P	16
Q	17
R	18
S	19
T	20
U	21
V	22
W	23
X	24
Y	25
Z	26

VARIABLE LOG



7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

0	□□□□	0
16	□□□■	1
32	□□■□	2
48	□□■■	3
64	□■□□	4
80	□■□■	5
96	□■■□	6
112	□■■■	7
128	■□□□	8
144	■□□■	9
160	■□■□	10
176	■□■■	11
192	■■□□	12
208	■■□■	13
224	■■■□	14
240	■■■■	15

INDEX

1541 Disk drive 1, 114

Abbreviations 12, 126
Addressing modes 83
AND 21
ASCII 4, 22
Assembly 82
Audio 46
Auto start 67
Auxiliary color 36, 40, 130
AV port 106

BASIC keywords 4, 12,16
BASIC program location 17, 34
Binary 4, 39, 42, 130
Block diagrams 102
Boolean functions 20
Border color 37, 62, 129

Cassette port 106
Change case 22, 24
Character base location 36, 61
Character codes 4, 25, 28, 35
Character editor 38
CHR$ codes 4, 22
Clock 15, 18, 65
Cold start 67
Color codes 40, 130
Color map location 34
Columns 36, 61
Combine files 114
Commodore key 55, 108
Control key 57, 108
Cursor 24, 40
Custom characters 38
Cycles 84, 88

Datacassette 106, 119
Device numbers 114
Device status 114
Dimensions 3
DIP switches 48, 114
Directory 115
Disk blocks 112
Disk commands 115
Disk drive 1, 112,
Disk error codes 116
Display 1, 30, 32, 36
DOS 113, 118
Double character mode 36, 61
Dump 19

Errors 113, 125
Expansion port 106

File names 114
Floating point variables 18
Folders 116
Function keys 24, 108

Game port 104, 107
Graphic printer 22, 114
Graphics 38, 44

Hardware 3, 104
Hexadecimal 4, 42, 124, 130
Hi-res graphics 38, 44
Horizontal alignment 36, 60

Illegal opcodes 96
Integers 18
Interrupt vectors 99
Interlace mode 36

Jiffy Dos 118
Joystick 64, 109

KERNAL 72, 99, 100
Keyboard 55, 64, 106, 130
Keywords 4, 16, 67
Knock reduction 117

Labels 78
Light pen 60, 107
Limit memory 34, 50
Line structure 17
List to printer 24
LOAD 2, 113, 115
Logical operands 20, 21
Lower case 22, 24
LSB 4, 124

Machines language 82, 110
Math functions 20, 67
Memory blocks 119
Memory expansion 48
Memory map 0, 48
Merge files 116
MSB 4, 124
Multi-color mode 40,43
Multi-part files 1, 48
Music 46

NOP 98
NOR 20
NOT 21

Opcodes 82, 86, 88, 126
Optimization 19, 99
OR 21
Order of operations 20

Paddle 61, 109
PETSCII 4, 22
Ports 3, 104
Power connection 106
Printer 22, 24
Programmers aid 81, 112
Program location 0, 50

Quick start 1
Quotation mode 24

Ram vectors 99
Raster beam 60
Registers 58, 108
Relative branch backward 95
Relative files 114, 116
Reset 81
Rom images 48
ROM subroutines 102
Row 36, 61
RS-232 56, 104
RUN STOP 58

SAVE 115
Save memory block 117
Scott Adams games 1, 81
Scrap paper 127
Screen color 37, 129
Screen location 34, 61
Screen map 30, 32, 35
Screen position 36
Scroll 57
Sequential files 114, 116
Serial port 106
Shift key 57, 106
Sound 46
Specifications 3
Status ST 15. 114
Start of memory 50
Strings 18
Super expander 111
Syntax error 125

Tape 106, 119
Time (TI and TI$) 15, 18, 65
Tokens 4, 12, 16
Top of memory pointer 34, 50
Type-in programs 120

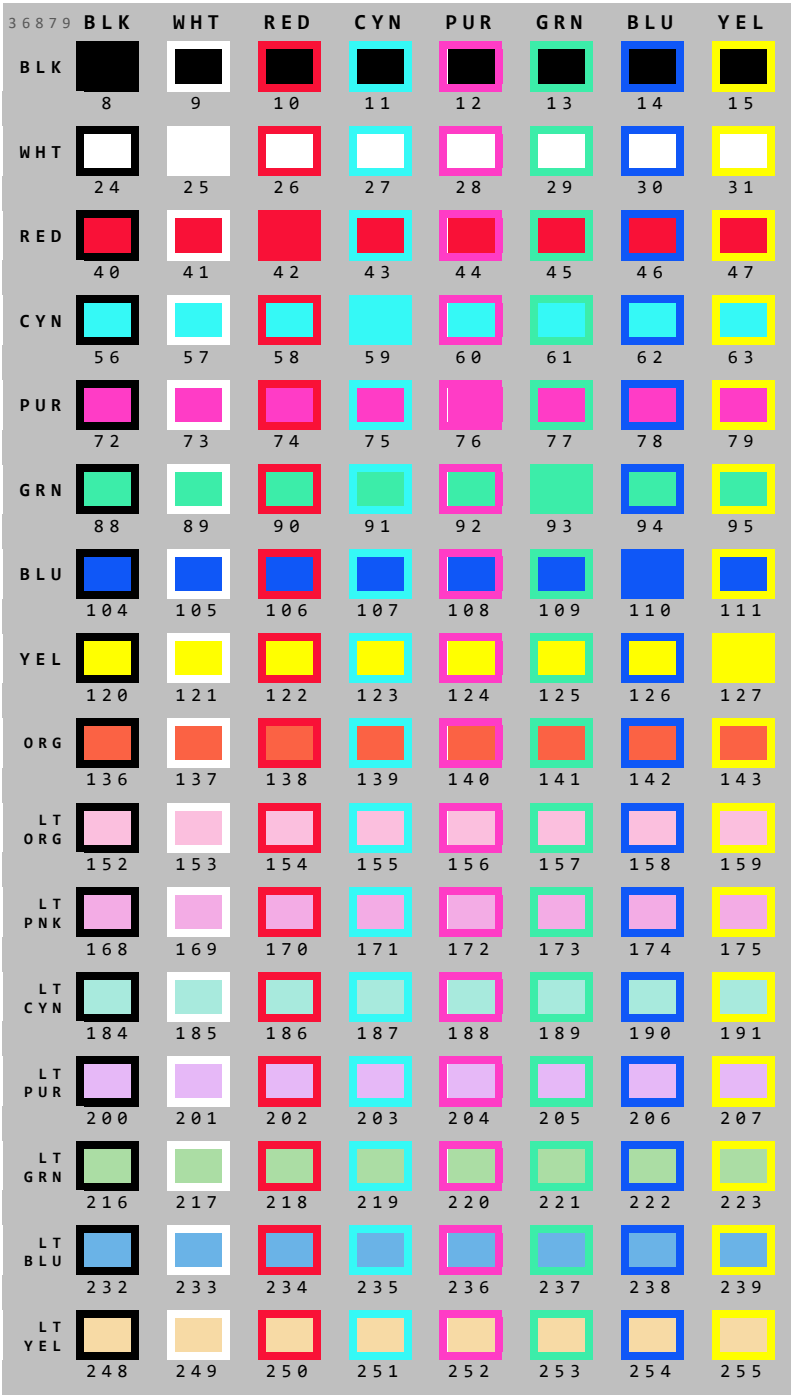
Undocumented opcodes 96
UNNEW 117
User defined characters 38, 44
User files 114
User port 106

Variables 18, 127
Vector addressing 99
Vertical alignment 36, 60
VIA 36, 60, 62, 64, 104
VICMON 1,81, 110, 126
Voices 62

Warm start 72, 81

XOR 20

Zero page 49


36879
BLK
WHT
RED
CYN
PUR
GRN
BLU
YEL
BLK 8 9 10 11 12 13 14 15
WHT 24 25 26 27 28 29 30 31
RED 40 41 42 43 44 45 46 47
CYN 56 57 58 59 60 61 62 63
PUR 72 73 74 75 76 77 78 79
GRN 88 89 90 91 92 93 94 95
BLU 104 105 106 107 108 109 110 111
YEL 120 121 122 123 124 125 126 127
ORG 136 137 138 139 140 141 142 143
LT ORG 152 153 154 155 156 157 158 159
LT PNK 168 169 170 171 172 173 174 175
LT CYN 184 185 186 187 188 189 190 191
LT PUR 200 201 202 203 204 205 206 207
LT GRN 216 217 218 219 220 221 222 223
LT BLU 232 233 234 235 236 237 238 239
LT YEL 248 249 250 251 252 253 254 255

7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1
C 192 B		C 48 B		C 12 B		C 3 B	

←	1	2	3	4	5	6	7	8	9	0	+	–	£	cl	de	F1
8	0	56	1	57	2	58	3	59	4	60	5	61	6	62	7	39
	Q	W	E	R	T	Y	U	I	O	P	@	*	↑			F3
	48	9	49	10	50	11	51	12	52	13	53	14	54			47
rs	A	S	D	F	G	H	J	K	L	:	;	=	return			F5
24	17	41	18	42	19	43	20	44	21	45	22	46	15			55
	Z	X	C	V	B	N	M	,	.	/			↕	↔		F7
	33	26	34	27	35	28	36	29	37	30			31	23		63

197 space 32 no key 64

1	7680	00	4096	0
2	7702	16	4118	22
3	7724	2C	4140	44
4	7746	42	4162	66
5	7768	58	4184	88
6	7790	6E	4206	110
7	7812	84	4228	132
8	7834	9A	4250	154
9	7856	B0	4272	176
10	7878	C6	4294	198
11	7900	DC	4316	220
12	7922	F2	4338	242
13	7944	08	4360	264
14	7966	1E	4382	286
15	7988	34	4404	308
16	8010	4A	4426	330
17	8032	60	4448	352
18	8054	76	4470	374
19	8076	8C	4492	396
20	8098	A2	4514	418
21	8120	B8	4536	440
22	8142	CE	4558	462
23	8164	E4	4580	484

HEX	MSB	
0	0	0
1	4096	256
2	8192	512
3	12288	768
4	16384	1024
5	20480	1280
6	24576	1536
7	28672	1792
8	32768	2048
9	36864	2304
A	40960	2560
B	45056	2816
C	49152	3072
D	53248	3328
E	57344	3584
F	61440	3840

	7654		3210	
0	0	□□□□	0	0
1	16	□□□■	1	1
2	32	□□■□	2	2
3	48	□□■■	3	3
4	64	□■□□	4	4
5	80	□■□■	5	5
6	96	□■■□	6	6
7	112	□■■■	7	7
8	128	■□□□	8	8
9	144	■□□■	9	9
A	160	■□■□	10	A
B	176	■□■■	11	B
C	192	■■□□	12	C
D	208	■■□■	13	D
E	224	■■■□	14	E
F	240	■■■■	15	F

SCREEN	30
BINARY	39
SOUND	46
MEMORY	48
LABELS	78
ASSEMBLY	84
VICMON	110
STORAGE	114
ERRORS	125

			AUX	MUL
0	BLK	0	0	8
1	WHT	1	16	9
2	RED	2	32	10
3	CYN	3	44	11
4	PUR	4	64	12
5	GRN	5	80	13
6	BLU	6	96	14
7	YEL	7	112	15

36878	AUX
ORG	128
Lt. ORG	144
PNK	160
Lt. CYN	176
Lt. PUR	192
Lt. GRN	208
Lt. BLU	224
Lt. YEL	240

rvs	F1
off	F2
home	F3
clear	F4
up	F5
down	F6
left	F7
right	F8
insrt	del