NANO SYSTEMS CORP.

REFERENCE CARD FOR THE

APPLE® II & II PLUS

WITH DOS COMMANDS AND MESSAGES

MICROCOMPUTER BASIC

© 1982, PAUL R. DES JARDINS, and SHIRLEY A. & PAUL P. NANOS

APPLE is a Reg. TM of APPLE COMPUTER, INC.

48K MEMORY MAP

START DEC	START HEX	END DEC	END HEX	DESCRIPTION
0	$0000	255	$00FF	**ZERO-PAGE** WORKING STORAGE.
0	$0000	5	$0005	**JUMP** INSTRS FOR **RESET** RECOVERY.
10	$000A	12	$000C	**USR** FUNCTION **JUMP** INSTRUCTION.
98	$0062	102	$0066	MULTIPLY/DIVIDE RESULT.
103	$0067	104	$0068	USER PGM AREA ADDR. (2049 IF ROM, 12289 IF TAPE) **LOMEM**.
105	$0069	106	$006A	ADDR OF SIMPLE VARS. (END OF USER PGM PLUS 1 or 2).
107	$006B	108	$006C	ADDR OF ARRAY VARS.
109	$006D	110	$006E	ADDR (END OF CURR NUM STOR).
111	$006F	112	$0070	ADDR OF STRING STORAGE.
115	$0073	116	$0074	ADDR OF APPLESOFT OR DOS HIGH-MEM = 1. (**HIMEM**)
117	$0075	118	$0076	NO. OF PGM LINE BEING EXECUTED.
119	$0077	120	$0078	NO. OF PGM LINE LAST INTERRUPTED.
121	$0079	122	$007A	ADDR OF NEXT STMT TO BE EXECUTED.
123	$007B	124	$007C	LINE NUMBER OF CURRENT **DATA** ITEM.
125	$007D	126	$007E	ADDR OF CURRENT **DATA** ITEM.
127	$007F	128	$0080	ADDR OF **INPUT** SOURCE OR CURR **DATA**.
129	$0081	130	$0082	NAME OF LAST VARIABLE USED.
131	$0083	132	$0084	ADDR OF VALUE OF LAST VARIABLE USED.
157	$009D	163	$00A3	PRIMARY FLOATING-POINT ACCUM.
165	$00A5	171	$00AB	SECONDARY FLOATING-POINT ACCUM.
175	$00AF	176	$00B0	ADDR OF END OF USER PGM.
177	$00B1	200	$00C8	**GET** ROUTINE.
184	$00B8	185	$00B9	ADDR OF LAST **GET** CHARACTER.
201	$00C9	205	$00CD	CURRENT RANDOM NUMBER.
216	$00D8	223	$00DF	**ONERR** POINTERS AND SCRATCH PAD.
224	$00E0	226	$00E2	HIGH-RES X AND Y VALUES.
228	$00E4	228	$00E4	HIGH-RES COLOR CODE.
232	$00E8	234	$00E9	ADDR OF **SHAPE** TABLE.
244	$00F4	248	$00F8	**ONERR** POINTERS.
256	$0100	511	$01FF	SYSTEM STACK.
512	$0200	767	$02FF	**KEYBOARD** CHARACTER BUFFER.
768	$0300	975	$03CF	AVAIL. FOR MACH LANG PGM.
976	$03D0	1023	$03FF	DOS VECTORS.
1024	$0400	2047	$07FF	LOW-RES OR TEXT **PAGE 1** FOR VIDEO.
2048	$0800	36399	$8E0F	PGM, VARS, FREE-SPACE STORAGE.
2048	$0800	3071	$0BFF	LOW-RES OR TEXT **PAGE 2** FOR VIDEO.
3072	$0C00	8191	$1FFF	PGM, VARS, FREE-SPACE STORAGE.
8192	$2000	16383	$3FFF	HIGH-RES **PAGE 1**, OR, FREE-SPACE.
16384	$4000	24575	$5FFF	HIGH-RES **PAGE 2**, OR, FREE-SPACE.
24576	$6000	38399	$95FF	FREE-SPACE AND STRING STORAGE.
38400	$9600	49151	$BFFF	**DOS**.
9152	$C000	53247	$CFFF	HARDWARE INPUT/OUTPUT ADDRESSES.
3248	$D000	57343	$DFFF	UNUSED IN INTEGER BASIC MACHINES.
3248	$D000	63487	$F7FF	APPLESOFT, APPLESOFT ROM.
1344	$E000	63487	$F7FF	INTEGER BASIC ROM.
1488	$F800	65535	$FFFF	SYSTEM MONITOR.

☆= Integer only ★= Applesoft only ◆= Invalid in Imm. mode

BASIC STATEMENTS

STATEMENT ● DESCRIPTION

INPUT-OUTPUT

◆★**GET** *var* ● Assign value of next key pressed to "var".
IN#0 ● Establish keyboard as input device.
IN#*num* ● Establish slot "num" as input device (1-7).
◆☆**INPUT** [*string,*] *var* [,*var...*] ● Prompt for and read keyed response.
◆★**INPUT** [*string;*] *var* [,*var...*] ● Prompt for and read keyed response.
PR#0 ● Establish screen as output device.
PR#*num* ● Establish slot "num" as output device (1-7).
PRINT *item* [; | , | *item...*] ● Display data. (**TAB** & **SPC** useable).
★**RECALL** *var* ● Read numeric array from tape into array "var".
★**SPEED=***value* ● Set input/output speed. (0-255)
★**STORE** *var* ● Write numeric array "var" to tape.

MEMORY/DATA

★**CLEAR** ● Clear strings to null and variables to zero.
☆**CLR** ● Clear strings to null and variables to zero.
★**DATA** [*item* [, *item...*]] ● Store items into memory table.
◆★**DEF FN** *var (arg)* = *exp* ● Create function *var* using *arg* in *exp*.
☆**DIM** *var$(num)* [,*var$(num)...*] ● Define string var len (255 max).
☆**DIM** *var(num)* [,*var(num)...*] ● Define numeric array(s).
★**DIM** *var(num* [,*num...*)] [,*var...*] ● Define array(s).
★**HIMEM:***addr* ● Establish high memory address for Basic.
[**LET**] *var=value* ● Assign a value to a variable.
★**LOMEM:***addr* ● Establish low memory address for Basic.
POKE *addr,num* ● Store number at address specified.
★**READ** *var* [,*var...*] ● Read items into var(s) from **DATA** table.
REM ● Remarks follow this statement.
★**RESTORE** ● Reset pointer to start of **DATA** table.

PROGRAM CONTROL

CALL *addr* ● Invoke machine language program at "addr".
END ● End program execution normally.
FOR *var=num1* **TO** *num2* [**STEP** *incr*] ● Loop thru **NEXT** until var greater than num.
☆**GOSUB** *var* ● Perform routine at line or "var" until **RETURN**.
★**GOSUB** *line* ● Perform routine at "line" until **RETURN**.
☆**GOTO** *var* ● Branch to line or "var".
★**GOTO** *line* ● Branch to "line".
☆**IF** *cond* **THEN** *action* ● Do action based on condition(s).
★**IF** *cond* **THEN** *action* [:*action...*] ● Do action(s) based on cond(s).
★**IF** *cond* [**THEN** [**GOTO**] *line* ● Branch to "line" based on cond(s).
☆**NEXT** *var* ● Indicate end of **FOR** loop.
★**NEXT** [*var* [,*var...*]] ● Indicate end of **FOR** loop.
◆★**ON** *num* **GOSUB** *line* [,*line...*] ● Perform rtn "num" in "line" list.
◆★**ON** *num* **GOTO** *line* [,*line...*] ● Branch to rtn "num" in "line" list.
◆★**ONERR GOTO** *line* ● If error occurs, branch to "line".
POP ● **RETURN** to prior **GOSUB** return point.
◆★**RESUME** ● End err-routine and return to statement in error.
RETURN ● Return from a **GOSUB**.
★**STOP** ● Interrupt program execution until **CONT** entered.
★**WAIT** *addr,exp1* [,*exp2*] ● Pause until bit compares are satisfied.

TEXT GRAPHICS (x=1-40, y=1-24)

★**FLASH** ● Set video output mode to flash characters.
★**HOME** ● Clear window and move cursor to upper left.
★**HTAB** *num* ● Tab cursor 1-256 pos from start of curr. line.(0=256)
★**INVERSE** ● Set video output mode to inverse characters.
★**NORMAL** ● Set video input/output mode to normal.
◇★**TAB** *num* ● Tab cursor 1-255 positions from start of curr. line.
TEXT ● Set screen **TEXT** mode (40×24). Move cursor to last line.
VTAB *num* ● Tab cursor "num" (1-24) lines vertically.

LOW-RESOLUTION GRAPHICS (x=0-39, y=0-39 or 47)

COLOR=*num* ● Set plotting color to "num" color code (0-15).
GR ● Set screen to low-res graphics mode (40 × 40). Clear screen.
HLIN *x1,x2* **AT** *y* ● Draw horiz. line on line "y" from "x1" to "x2".
PLOT *x,y* ● Change block at "x,y" to current color.
VLIN *y1,y2* **AT** *x* ● Draw vert. line at pos. "x" from "y1" to "y2".

HIGH-RESOLUTION GRAPHICS (x=0-279, y=0-159 or 191)

★**DRAW** *defno* **AT** *x,y* ● Draw shape "defno" (0-255) at point "x,y".
★**HCOLOR=***num* ● Set graphics color to "num" (0-7).
★**HGR** ● Set high-res mode (280×160). Display page 1 (8K-16K).
★**HGR2** ● Set high-res mode (280×192). Display page 2 (16K-24K).
★**HPLOT** [*x1,y1* | **TO** *x2,y2* [**TO** ...]] ● Plot point(s) with current color.
★**ROT=***num* ● Set angle of shape to **DRAW/XDRAW**. (0-255)
0=normal, 16=90°, 32=180°, 48=270°, 64=360°.
★**SCALE=***num* ● Set scale size of shape. (0-255)
1=normal, 0=maximum.
★**SHLOAD** ● Load shape table into memory from cassette.
(If on disk, **BLOAD** shape table & **POKE** low/hi addr.)
★**XDRAW** *defno* **AT** *x,y* ● Erase shape "defno" (0-255) at point "x,y".

(☆=Integer only) **BASIC FUNCTIONS** (★=APPLESOFT only)

FUNCTION	RETURNS
CHARACTER/STRING FUNCTIONS	
★**CHR$(num)**	Character value of "num". (65="A")
★**LEFT$(str,num)**	Substring of "num" chars from left side of string.
LEN(str)	Length of string.
★**MID$(str,start,[,len])**	Substring of "str" at "start" pos, length "len".
★**RIGHT$(str,num)**	Substring of "num" chars from right side of string.
★**STR$(num)**	Character string of a number, including sign.
NUMERIC FUNCTIONS	
ABS(num)	Absolute value of num. (−3.25=3.25)
ASC(str)	Decimal value of first character. ("A"=65)
★**INT(num)**	Largest whole number not exceeding num. (2.5=2) (−2.5=3)
☆**num MOD num1**	Remainder after dividing num by num1. (10 MOD 7 = 3)
RND(num)	Random number between 0 and 1.
RND(0)	Last random number generated.
SGN(num)	0(zero), 1(positive), −1(negative).
★**VAL(str)**	Number extracted from character string.
MATH FUNCTIONS	
★**ATN(angle)**	Arctangent of angle(degrees), in radians.
★**COS(angle)**	Cosine of angle(radians), in radians.
★**EXP(num)**	Inverse LOG of num.
★**LOG(num)**	Natural Logarithm of "num".
★**SIN(angle)**	Sine of angle(radians), in radians.
★**SQR(num)**	Square root of "num".
★**TAN(angle)**	Tangent of angle(radians), in radians.

Radians=Degrees/57.29577951 Degrees=Radians*57.29577951

I/O AND OTHER FUNCTIONS

★**FRE(0)**	Number of available bytes in memory.
★**FN var(arg)**	User function result. (var defined in DEF FN stmt)
PDL(num)	0-255, paddle value. (num is 0, 1, 2, or 3)
PEEK(addr)	Decimal value of byte at loc. "addr".
★**POS(0)**	Current horiz. position of cursor.
SCRN(horiz, vert)	Code (LOW-RES) of color at specified screen pos.
★**SPC(num)**	"num" spaces (PRINT stmt. only).
★**TAB(num)**	Tab cursor to pos. "num" (PRINT stmt. only). num is 0-255. Backward tabbing not allowed.
★**TAB(0)**	Tab cursor to pos. 256 (PRINT stmt. only).
★**USR(exp)**	Pass argument to mach. lang. routine.

All functions are effective in immediate mode.

DERIVED FUNCTIONS (X is in radians.)

FUNCTION	EXPRESSION
SECANT	1/COS(X)
COSECANT	1/SIN(X)
COTANGENT	1/TAN(X)
INVERSE	
SINE	ATN(X/SQR(−X*X + 1))
COSINE	−ATN(X/SQR(−X*X + 1)) + 1.570796
SECANT	ATN(SQR(X*X − 1)) + (SGN(X) −1) *1.570796
COSECANT	ATN(1/SQR(X*X −1)) + (SGN(X) −1)*1.570796
COTANGENT	−ATN(X) + 1.570796
HYPERBOLIC	
SINE	(EXP(X) − EXP(−X))/2
COSINE	(EXP(X) + EXP(−X))/2
TANGENT	−EXP(−X)/(EXP(X) + EXP(−X))*2 + 1
SECANT	2/(EXP(X) + EXP(−X))
COSECANT	2/(EXP(X) − EXP(−X))
COTANGENT	EXP(−X)/(EXP(X) − EXP(−X))*2 + 1
INVERSE HYPERBOLIC	
SINE	LOG(X + SQR(X*X + 1))
COSINE	LOG(X + SQR(X*X − 1))
TANGENT	LOG((1 +X)/(1 −X))/2
SECANT	LOG(SQR(−X*X + 1) +1)/X)
COSECANT	LOG((SGN(X)*SQR(X*X + 1) + 1)/X)
COTANGENT	LOG((X + 1)/(X − 1))/2

☆=Integer only ★=APPLESOFT only

BASIC SPECIAL CHARACTERS AND OPERATORS

;	Supress tab after PRINT.
,	Tab to next column after PRINT.
blank	Tab to next line after PRINT.
?	Use instead of "PRINT".
:	End of Basic statement.
" "	Delimits a character string.
$	Identifies a character string variable. (A$)
★**%**	Identifies an integer variable. (A%)
☆**blank**	Identifies an integer variable. (A)
★**blank**	Identifies a real variable. (A)
E	Floating-point notation. (1.7E38)
( )	Denotes priority in order of operations.
+	Addition, concatenation, or plus sign.
−	Subtraction or minus sign.
*	Multiplication.
/	Division.
☆**MOD**	Modulo. (13 MOD 5=3, division remainder)
^	Exponentiation
=	Equal or assignment.
☆#	Not equal.
★<> or ><	Not equal.
>	Greater than or follows.
<	Less than or precedes.
>= or =>	Greater than or equal to (follows or equals).
<= or =<	Less than or equal (precedes or equals).
NOT	Reverses true/false result.
AND	If both exp are true, result is true.
OR	If either or both exp are true, result is true.
Non-zero	True.
0	False.

☆Order of operations:
( ), {NOT,−(neg)}, ^,{*,/,MOD}, {+,−}, {>,<,=,>=,=>, <=,=<,<>,><},AND,OR

★Order of operations:
( ), {NOT,−(neg),+(pos)}, ^, {*, /},{+,−}, {>,<, =,>=,=>, <=,=<, <>,><},AND,OR

★=APPLESOFT-ONLY ☆=INTEGER-ONLY ◇=INVALID IN A PGM

SYSTEM AND UTILITY COMMANDS

FORMAT	DESCRIPTION
◇☆**AUTO n1 [,incr]**	Automatic program line numbering.
★**CLEAR**	Clear strings to null and vars to 0.
☆**CLR**	Clear strings to null and vars to 0.
☆**CON**	Continue after STOP, END, or ©C.
★**CONT**	Continue after STOP, END, or ©C.
DEL n1 [,n2]	Delete line numbers n1 thru n2.
◇**DSP var**	Display variable contents.
LIST n1	List line number n1.
LIST n1 [,n2 [-n2]	List lines n1 thru n2.
LIST ,n2	List lines thru n2.
◇☆**MAN**	Return to manual line numbering.
NEW	Clear existing program from memory.
NOTRACE	Turn off program trace.
RUN [n]	Run program (beginning at line n).
SAVE	Store program on cassette tape.
TRACE	Turn on program trace.

PROMPT CHARACTERS

*=Monitor >=Integer Basic]=Applesoft Basic !=Assembler ?=INPUT, etc.

SPECIAL POKES, PEEKS, CALLS

FUNCTION	HOW TO DO IT
ANNUNCIATOR, Ann 0, Pin 15	POKE 49240,1 (Off) POKE 49241,1 (On)
ANNUNCIATOR, Ann 1, Pin 14	POKE 49242,1 (Off) POKE 49243,1 (On)
ANNUNCIATOR, Ann 2, Pin 13	POKE 49244,1 (Off) POKE 49245,1 (On)
ANNUNCIATOR, Ann 3, Pin 12	POKE 49246,1 (Off) POKE 49247,1 (On)
CASSETTE, Output	PEEK(49184)
CASSETTE, Input	PEEK(49256)
CLEAR, LO RES Page 1, 40 Lines	CALL 63542 (-994)
CLEAR, LO RES Page 1, 48 lines	CALL 63538 (-1998)
CLEAR, Curr HI RES Page	CALL 62450 (-3086)
CLEAR, Curr HI RES Page to HCOLOR	CALL 62454 (-3082)
INVERSE, Set Screen Mode.	POKE 50,127
INVERSE, Reset to Normal.	POKE 50,255
KEYBOARD, Last Byte in	PEEK(49152)
KEYBOARD, Clear Strobe	POKE 49168,1 = 16368
PADDLE, Button 0 (GC Pin 2) RED	PEEK(49249) Off: 0 On: 128
PADDLE, Button 1 (GC Pin 3) BLUE	PEEK(49250) Off: 0 On: 128
PADDLE, Button 2 (GC Pin 4)	PEEK(49251) Off: 0 On: 128
PADDLE, Game Ctlr Strobe (Pin 5)	POKE 49264,1 (Before PEEK)
PADDLE, Game Ctlr 0 (Pin 6)	PEEK(49252) 0-255 —
PADDLE, Game Ctlr 1 (Pin 10)	PEEK(49253) 0-255 —
PADDLE, Game Ctlr 2 (Pin 7)	PEEK(49254) 0-255 —
PADDLE, Game Ctlr 3 (Pin 11)	PEEK(49255) 0-255 —
SET MODE, GRAPHICS	POKE 49232,1
SET MODE, TEXT	POKE 49233,1
SET MODE, All TEXT or GR	POKE 49234,1
SET MODE, Mix TEXT & GR	POKE 49235,1
SET MODE, Select Page 1	POKE 49236,1
SET MODE, Select Page 2	POKE 49237,1
SET MODE, LO RES GRAPHICS	POKE 49238,1
SET MODE, HI RES GRAPHICS	POKE 49239,1
SPEAKER, Toggle	PEEK(49200)
STROBE, Utility	PEEK(49216)
WINDOW, Set left margin	POKE 32,a (0-39) Default 0.
WINDOW, Set width	POKE 33,a (1-40) Default 40.
WINDOW, Set top margin	POKE 34,a (0-24) Default 0.
WINDOW, Set bottom margin	POKE 35,a (0-24) Default 24.
WINDOW, Home cursor	CALL 64600
WINDOW, Erase to end of window	CALL 64578
WINDOW, Erase to end of line	CALL 64668 - 768
WINDOW, Scroll in window	CALL 64924

RESTORE HI-RES SCREEN CALL -3100

MONITOR COMMANDS (Use hex address and values)

FORMAT	DESCRIPTION
Call 65385	Invoke the MONITOR.
©C	Exit the MONITOR. Re-enter ROM lang.
3D0G	Exit the MONITOR. (DOS)
I	Set inverse display mode.
N	Set normal display mode.
addr[.addr1]	Display memory.
RETURN	Display next 8 memory locations.
[addr:]val[,val...]	Store value(s) into memory.
addr<addr1.addr2[M|V]	Copy(M) or Compare(V) memory.
addr.addr1[W|R]	Write(W) to cassette from memory, or, Read(R) from cassette into memory.
addrG	Branch to "addr".
addrL	Display 20 instr in assembled format.
L	Display next 20 instr in assembled format.
val+val	Add 2 hex values and display the result.
val−val	Subtract in hex and display the result.
©B	Enter ROM language.
©E	Display 6502 regs.
n©K	Set input to go to slot n.
n©P	Set output to go to slot n. 6 up to 6 out
©Y	Branch to $3F8. (User JMP statement at this loc.)
addrS	* Display and execute instr at "addr" and display regs.
S	* Display and execute next instr and display regs.
addrT	* Set Trace on.
F666G	** Invoke mini-assembler.
$comd	** Invoke Monitor command from mini-assembler.
$FF69G	** Exit from mini-assembler.

*—Not available with Autostart ROM.
**—Not available with APPLE II Plus.
Δ = Space.

KEY AND CONTROL FUNCTIONS

KEY	*FUNCTION*
←	Backspace cursor and erase.
→	Forward space cursor and retype.
©B	Return to BASIC.
©C	Break.
©G	Ring bell.
©S	†Pause execution.
©X	Cancel line being typed.
0G	Return to BASIC after reset (Applesoft).
ESC,@	Clear screen, home cursor.
ESC,A	Forward space cursor, without retype.
ESC,B	Backspace cursor without erase.
ESC,C	Downspace cursor.
ESC,D	Upspace cursor.
ESC,E	Clear to end-of-line.
ESC,F	Clear to end of window.
ESC,I	†Upspace cursor. Stay in ESC mode.
ESC,J	†Backspace cursor. Stay in ESC mode.
ESC,K	†Forward space cursor. Stay in ESC mode.
ESC,M	Downspace cursor. Stay in ESC mode.

† Available only with Autostart Monitor ROM.

APPLESOFT INTERNAL CODES

DEC	HEX	KEYWORD	DEC	HEX	KEYWORD	DEC	HEX	KEYWORD
128	80	END	164	A4	LOMEM:	200	C8	+
129	81	FOR	165	A5	ONERR	201	C9	−
130	82	NEXT	166	A6	RESUME	202	CA	*
131	83	DATA	167	A7	RECALL	203	CB	/
132	84	INPUT	168	A8	STORE	204	CC	^
133	85	DEL	169	A9	SPEED=	205	CD	AND
134	86	DIM	170	AA	LET	206	CE	OR
135	87	READ	171	AB	GOTO	207	CF	>
136	88	GR	172	AC	RUN	208	D0	=
137	89	TEXT	173	AD	IF	209	D1	<
138	8A	PR#	174	AE	RESTORE	210	D2	SGN
139	8B	IN#	175	AF	&	211	D3	INT
140	8C	CALL	176	B0	GOSUB	212	D4	ABS
141	8D	PLOT	177	B1	RETURN	213	D5	USR
142	8E	HLIN	178	B2	REM	214	D6	FRE
143	8F	VLIN	179	B3	STOP	215	D7	SCRN(
144	90	HGR2	180	B4	ON	216	D8	PDL
145	91	HGR	181	B5	WAIT	217	D9	POS
146	92	HCOLOR=	182	B6	LOAD	218	DA	SQR
147	93	HPLOT	183	B7	SAVE	219	DB	RND
148	94	DRAW	184	B8	DEF	220	DC	LOG
149	95	XDRAW	185	B9	POKE	221	DD	EXP
150	96	HTAB	186	BA	PRINT	222	DE	COS
151	97	HOME	187	BB	CONT	223	DF	SIN
152	98	ROT=	188	BC	LIST	224	E0	TAN
153	99	SCALE=	189	BD	CLEAR	225	E1	ATN
154	9A	SHLOAD	190	BE	GET	226	E2	PEEK
155	9B	TRACE	191	BF	HIMEM:	227	E3	LEN
156	9C	NOTRACE	192	C0	TAB(	228	E4	STR$
157	9D	NORMAL	193	C1	TO	229	E5	VAL
158	9E	INVERSE	194	C2	FN	230	E6	ASC
159	9F	FLASH	195	C3	SPC(	231	E7	CHR$
160	A0	COLOR=	196	C4	THEN	232	E8	LEFT$
161	A1	POP	197	C5	AT	233	E9	RIGHT$
162	A2	VTAB	198	C6	NOT	234	EA	MID$
163	A3	HIMEM:	199	C7	STEP			

APPLESOFT RESERVED WORDS

&	END	HTAB	NOT	RESUME	STORE
ABS	EXP	IF	NOTRACE	RETURN	STR$
AND	FLASH	IN#	ON	RIGHT$	TAB(
ASC	FN	INPUT	ONERR	RND	TAN
AT	FOR	INT	OR	ROT=	TEXT
ATN	FRE	INVERSE	PDL	RUN	THEN
CALL	GET	LET$	PEEK	SAVE	TO
CHR$	GOSUB	LEN	PLOT	SCALE=	TRACE
CLEAR	GOTO	LET	POKE	SCRN(	USR
COLOR=	GR	LIST	POP	SGN	VAL
CONT	HCOLOR=	LOAD	POS	SHLOAD	VLIN
COS	HGR	LOG	PR#	SIN	VTAB
DATA	HGR2	LOMEM:	PRINT	SPC(	WAIT
DEF	HIMEM:	MID$	READ	SPEED=	XPLOT
DEL	HLIN	NEW	RECALL	SQR	XDRAW
DIM	HOME	NEXT	REM	STEP	
DRAW	HPLOT	NORMAL	RESTORE	STOP	

INTEGER BASIC ADDRESSING

In Integer Basic, addresses above 32767 must be expressed using the negative values of their two's complement. This is done by adding the address to −65536.

Example: For address 40000
−65536
+40000
−25536 ← two's complement address.

DOS 3.3 COMMAND SUMMARY

Command	Options SDV	RB	AL	Description
APPEND f	ooo	. .	. .	Append output to file f.
BLOAD p	ooo	. .	o .	Load binary program p.
BRUN p	ooo	. .	o .	Run binary program p.
BSAVE p	ooo	. .	r r	Save binary prog.
CATALOG	oo .	. .	. .	List disk catalog.
CHAIN p	ooo	. .	. .	Chain to program p. (INT only).
CLOSE [f]	. . .	. .	. .	Close DOS file(s).
DELETE f	ooo	. .	. .	Delete file f from disk.
EXEC p	ooo	o .	. .	Execute Text file from disk.
FP [p]	ooo	. .	. .	Select FP Basic (APPLESOFT).
INIT p	ooo	. .	. .	Init. disk w/'p' as HELLO pgm.
INT	. . .	. .	. .	Select Integer Basic.
LOAD p	ooo	. .	. .	Load program p from disk.
LOCK f	ooo	. .	. .	Lock file f on disk.
MAXFILES n	. . .	. .	. .	Specify max no. of open files.
MON [C] [I] [O]	. . .	. .	. .	Set DOS trace options.
NOMON [C] [I] [O]	. . .	. .	. .	Reset DOS trace options.
OPEN f	ooo	. .	. .	Open sequential file.
OPEN f	ooo	. .	r .	Open random file f.
POSITION f	. . .	r .	. .	Position file f at record no. r.
READ f	. . .	. o	. .	Read sequential file.
READ f	. . .	x x	. .	Read random file.
RENAME f1, f2	ooo	. .	. .	Rename file f1 to f2 on disk.
RUN p	ooo	. .	. .	LOAD p from disk and RUN it.
SAVE p	ooo	. .	. .	Save current pgm to disk as p.
UNLOCK f	ooo	. .	. .	Unlock file f on disk.
VERIFY f	ooo	. .	. .	Verify file f.
WRITE f	. . .	o .	. .	Write sequential file.
WRITE f	. . .	x x	. .	Write random file.

Options are entered as a comma, option code and parameter.
For example: LOAD p,S6,D2, etc....

LEGEND:

SDV — Length; RB — Address; AL — Byte, Record, Volume, Drive, Slot

o=Optional.
r=Required.
xx=One of two required.

COLOR SELECTION

BINARY	DEC	MONOCHROME	LO-RES	HI-RES
0000	0	BLACK	BLACK	BLACK1
0001	1	¼WHITE	MAGENTA	BLUE
0010	2	¼WHITE	DARK BLUE	BLUE
0011 | 3 | ½WHITE | PURPLE | WHITE1
0100 | 4 | ¼WHITE | DARK GREEN | BLACK2
0101 | 5 | GREY | GREY1 | RED
0110 | 6 | ½WHITE | MEDIUM BLUE | VIOLET
0111 | 7 | ¾WHITE | LIGHT BLUE | WHITE2
1000 | 8 | ¼WHITE | BROWN | illegal
1001 | 9 | ½WHITE | ORANGE | illegal
1010 | 10 | GREY | GREY2 | illegal
1011 | 11 | ¾WHITE | PINK | illegal
1100 | 12 | ½WHITE | GREEN | illegal
1101 | 13 | ¾WHITE | YELLOW | illegal
1110 | 14 | ¾WHITE | AQUA | illegal
1111 | 15 | WHITE | WHITE | illegal

Published 1982 by Nanos Systems Corp., P.O. Box 24344, Speedway, IN 46224 (317) 244-4078.
Printed in U.S.A.

APPLESOFT ERROR MESSAGES

MESSAGE	MEANING	CODE
none.	Bad response to INPUT	**254**
none.	©C interrupt was keyed in.	**255**
BAD SUBSCRIPT	Addressed array w/invalid subscript.	**107**
CAN'T CONTINUE	CONT is not possible at this time.	**----**
DIVISION BY ZERO	Divided by number too small.	**133**
EXTRA IGNORED	Extra INPUT data ignored, or, GET input was ":" or ",".	**----**
FORMULA TOO COMPLEX	String oper. too complex.	**191**
ILLEGAL DIRECT	Tried Basic-only stmt in immed. mode.	**----**
ILLEGAL QUANTITY	Used a bad arg for function.	**53**
NEXT WITHOUT FOR	Used NEXT before FOR.	**0**
OUT OF DATA	READ can't find DATA.	**42**
OUT OF MEMORY	Memory exhausted, nesting too deep, or HIMEM/LOMEM spec invalid.	**77**
OVERFLOW	Number too large for computer.	**69**
REDIM'D ARRAY	Dimensioned same array twice.	**120**
REENTER	Entered invalid data type for INPUT.	**----**
RETURN WITHOUT GOSUB	Used RETURN or POP before GOSUB.	**22**
STRING TOO LONG	Exceeded 255 characters in string.	**176**
SYNTAX	Punctuation, spelling, etc. error.	**16**
TYPE MISMATCH	Wrong data type used. (num, char)	**163**
UNDEF'D FUNCTION	User function not defined (DEF FN).	**224**
UNDEF'D STATEMENT	Referenced line which does not exist.	**90**

INTEGER BASIC ERROR MESSAGES

MESSAGE	MEANING
BAD BRANCH ERR	Tried to GOTO or GOSUB to a non-existing line.
BAD NEXT ERR	Used NEXT before FOR.
BAD RETURN ERR	Used RETURN before GOSUB.
DIM ERR	DIMensioned same array twice.
ERR	Bad data read from cassette tape.
MEM FULL ERR	Memory is full.
NO END ERR	END statement missing from program.
RANGE ERR	Parameter outside of legal range.
RETYPE LINE	Bad response to INPUT.
STOPPED AT nnnn	Ended program execution at line "nnnn".
STR OVFL ERR	String exceeds length in DIM statement.
STRING ERR	Invalid or incorrect string statement.
SYNTAX ERR	Bad punctuation, spelling, format, etc.
TOO LONG ERR	Statement or data exceeds max. char. length.
16 FORS ERR	Nexted FOR statements exceed 16 maximum.
16 GOSUBS ERR	Nested GOSUB statements exceed 16 maximum.
255 ERR	Number exceeds 255 or precedes 0.
32767 ERR	Number exceeds 32767 or precedes −32767.

DOS ERROR MESSAGES

MESSAGE	CODE	MESSAGE	CODE
DISK FULL	**9**	**NOT DIRECT COMMAND**	**15**
END OF DATA	**5**	**PROGRAM TOO LARGE**	**14**
FILE LOCKED	**10**	**RANGE ERROR**	**2**
FILE NOT FOUND	**6**	**RANGE ERROR**	**3**
FILE TYPE MISMATCH	**13**	**SYNTAX ERROR**	**11**
I/O ERROR	**8**	**VOLUME MISMATCH**	**7**
LANGUAGE NOT AVAILABLE	**1**	**WRITE PROTECTED**	**4**
NO BUFFERS AVAILABLE	**12**		

APPLESOFT ERROR HANDLING

If an ONERR statement has been executed, the following Zero Page locations are of interest:

DEC	HEX	DESCRIPTION
129	$81	Name of last variable used.
130	$82	
216	$D8	$80 = An ONERR Statement is active. $00 = No active ONERR.
218	$DA	Error Line Number (low).
219	$DB	Error Line Number (high).
222	$DE	Error Code.

READING MACHINE LANGUAGE

MACHINE LANGUAGE INSTRUCTIONS ARE CODED INTERNALLY AS FOLLOWS:

Length 1—op —Where op is a single opcode with no operands.
Length 2—opaa —Where op is the opcode, and aa is the operand.
Length 3—opaabb —Where op is the opcode, and aabb is the 2-byte operand.

INSTRUCTION SET NOTATION

#imm = Immediate operand.
*+disp = Instruction (program) counter plus displacement.
addr = Absolute or Zero-page address.
abaddr = Absolute address only.
zaddr = Zero-page address only.
(abaddr) = Address is obtained from location abaddr.
(zaddr,X) = Address is obtained from location calculated by adding "zaddr" to contents of reg. X.
(zaddr),Y = Address is obtained by adding contents of reg. Y to contents of location zaddr.
A = Accumulator (Reg. A).
P = Processor Status Reg.
S = Stack Pointer Reg.
X = Index reg. X.
Y = Index reg. Y.

HEX/DEC CONVERSION CHART

8 4 2 1 HEX	DEC	8 4 2 1 HEX	DEC	8 4 2 1 HEX	DEC	8 4 2 1 HEX	DEC
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15
65535		4095		255		15	

‡Undefined opcode

◆=Reverse Graphics

DEC	HEX	PRINT	VIDEO	6502	INTEGER
0	00	NUL	◆@◆	BRK	
1	01	SOH	◆A◆	ORA (zaddr,X)	eol
2	02	STX	◆B◆	‡	
3	03	ETX	◆C◆	‡	:
4	04	EOT	◆D◆	‡	
5	05	ENQ	◆E◆	ORA zaddr	
6	06	ACK	◆F◆	ASL zaddr	
7	07	BEL	◆G◆	‡	
8	08	BS	◆H◆	PHP	
9	09	HT	◆I◆	ORA imm	
10	0A	LF	◆J◆	ASL	
11	0B	VT	◆K◆	‡	
12	0C	FF	◆L◆	‡	
13	0D	CR	◆M◆	ORA abaddr	
14	0E	SO	◆N◆	ASL abaddr	
15	0F	SI	◆O◆	‡	
16	10	DLE	◆P◆	BPL *+disp	
17	11	DC1	◆Q◆	ORA (zaddr),Y	
18	12	DC2	◆R◆	‡	+
19	13	DC3	◆S◆	‡	−
20	14	DC4	◆T◆	‡	*
21	15	NAK	◆U◆	ORA zaddr,X	/
22	16	SYN	◆V◆	ASL zaddr,X	= (rel)
23	17	ETB	◆W◆	‡	# (rel)
24	18	CAN	◆X◆	CLC	>= (rel)
25	19	EM	◆Y◆	ORA abaddr,Y	> (rel)
26	1A	SUB	◆Z◆	‡	<= (rel)
27	1B	ESC	◆[◆	‡	<> (rel)
28	1C	FS	◆\◆	‡	< (rel)
29	1D	GS	◆]◆	ORA abaddr,X	AND
30	1E	RS	◆←◆	ASL abaddr,X	OR
31	1F	US	◆_◆	‡	MOD
32	20	SPACE	◆ ◆	JSR abaddr	^
33	21	!	◆!◆	AND (zaddr,X)	
34	22	"	◆"◆	‡	(str dim)
35	23	#	◆#◆	‡	, (str subscr)
36	24	$	◆$◆	BIT zaddr	THEN (line)
37	25	%	◆%◆	AND zaddr	THEN
38	26	&	◆&◆	ROL zaddr	
39	27	'	◆'◆	‡	, (INPUT)
40	28	(	◆(◆	PLP	" (left)
41	29	)	◆)◆	AND imm	" (right)
42	2A	*	◆*◆	ROL	(str subscr)
43	2B	+	◆+◆	‡	
44	2C	,	◆,◆	BIT abaddr	
45	2D	−	◆-◆	AND abaddr	(
46	2E	.	◆.◆	ROL abaddr	PEEK
47	2F	/	◆/◆	‡	RND
48	30	0	◆0◆	BMI *+disp	SGN
49	31	1	◆1◆	AND (zaddr),Y	ABS
50	32	2	◆2◆	‡	PDL
51	33	3	◆3◆	‡	
52	34	4	◆4◆	‡	((dim)
53	35	5	◆5◆	AND zaddr,X	
54	36	6	◆6◆	ROL zaddr,X	− (unary)
55	37	7	◆7◆	‡	NOT
56	38	8	◆8◆	SEC	((expr)
57	39	9	◆9◆	AND abaddr,Y	= (str rel)
58	3A	:	◆:◆	‡	# (str rel)
59	3B	;	◆;◆	‡	LEN(
60	3C	<	◆<◆	‡	ASC(
61	3D	=	◆=◆	AND abaddr,X	SCRN(
62	3E	>	◆>◆	ROL abaddr,X	, (SCRN)
63	3F	?	◆?◆	‡	(

‡Undefined opcode

◆=Flashing Graphics

DEC	HEX	PRINT	VIDEO	6502	INTEGER
64	40	@	=@◆	RTI	$ (var)
65	41	A	=A◆	EOR (zaddr,X)	
66	42	B	=B◆	‡	
67	43	C	=C◆	‡	
68	44	D	=D◆	‡	, (DIM)
69	45	E	=E◆	EOR zaddr	; (PRINT)
70	46	F	=F◆	LSR zaddr	; (PRINT)
71	47	G	=G◆	‡	
72	48	H	=H◆	PHA	, (PRINT)
73	49	I	=I◆	EOR imm	, (PRINT)
74	4A	J	=J◆	LSR	
75	4B	K	=K◆	‡	TEXT
76	4C	L	=L◆	JMP abaddr	GR
77	4D	M	=M◆	EOR abaddr	CALL
78	4E	N	=N◆	LSR abaddr	DIM (str)
79	4F	O	=O◆	‡	DIM (int)
80	50	P	=P◆	BVC	TAB
81	51	Q	=Q◆	EOR (zaddr),Y	END
82	52	R	=R◆	‡	
83	53	S	=S◆	‡	INPUT (msg)
84	54	T	=T◆	‡	INPUT
85	55	U	=U◆	EOR zaddr,X	FOR
86	56	V	=V◆	LSR zaddr,X	= (FOR)
87	57	W	=W◆	‡	TO
88	58	X	=X◆	CLI	STEP
89	59	Y	=Y◆	EOR abaddr,Y	NEXT
90	5A	Z	=Z◆	‡	(NEXT)
91	5B	[	=[◆	‡	RETURN
92	5C	\	=\◆	‡	GOSUB
93	5D	]	=]◆	EOR abaddr,X	REM
94	5E	^	=^◆	LSR abaddr,X	LET
95	5F	_	=_◆	‡	GOTO
96	60	`	=`◆	RTS	IF
97	61	a	=!◆	ADC (zaddr,X)	PRINT
98	62	b	="◆	‡	PRINT
99	63	c	=#◆	‡	PRINT
100	64	d	=$◆	‡	POKE
101	65	e	=%◆	ADC zaddr	, (POKE)
102	66	f	=&◆	ROR zaddr	COLOR=
103	67	g	='◆	‡	PLOT
104	68	h	=(◆	PLA	, (PLOT)
105	69	i	=)◆	ADC imm	HLIN
106	6A	j	=*◆	ROR	, (HLIN)
107	6B	k	=+◆	‡	AT (HLIN)
108	6C	l	=,◆	JMP (abaddr)	VLIN
109	6D	m	=-◆	ADC abaddr	, (VLIN)
110	6E	n	=.◆	ROR abaddr	AT (VLIN)
111	6F	o	=/◆	‡	VTAB
112	70	p	=0◆	BVS	= (asgn str)
113	71	q	=1◆	ADC (zaddr),Y	= (asgn num)
114	72	r	=2◆	‡	)
115	73	s	=3◆	‡	
116	74	t	=4◆	‡	LIST
117	75	u	=5◆	ADC zaddr	, (LIST)
118	76	v	=6◆	ROR zaddr	
119	77	w	=7◆	‡	POP
120	78	x	=8◆	SEI	NODSP (str)
121	79	y	=9◆	ADC abaddr,Y	NODSP (val)
122	7A	z	=:◆	‡	NOTRACE
123	7B	{	=;◆	‡	DSP (str)
124	7C	;	=<◆	‡	DSP (val)
125	7D	}	==◆	ADC abaddr,X	TRACE
126	7E	~	=>◆	ROR abaddr,X	PR#
127	7F	RUB	=?◆	‡	IN#

‡Undefined opcode

DEC	HEX	PRINT	VIDEO	6502	APPLESOFT
128	80	@	@	‡	END
129	81	A	A	STA (zaddr,X)	FOR
130	82	B	B	‡	N EXT
131	83	C	C	‡	DATA
132	84	D	D	STY zaddr	INPUT
133	85	E	E	STA zaddr	DEL
134	86	F	F	STX zaddr	DIM
135	87	G	G	‡	READ
136	88	→,©H	H	DEY	GR
137	89	I	I	‡	TEXT
138	8A	J	J	TXA	PR#
139	8B	K	K	‡	IN#
140	8C	L	L	STY abaddr	CALL
141	8D	M,RET	M	STA abaddr	PLOT
142	8E	N	N	STX abaddr	HLIN
143	8F	O	O	‡	VLIN
144	90	P	P	BCC	HGR2
145	91	Q	Q	STA (zaddr),Y	HGR
146	92	R	R	‡	HCOLOR=
147	93	S	S	‡	HPLOT
148	94	T	T	STY zaddr,X	DRAW
149	95	←,©U	U	STA zaddr,X	XDRAW
150	96	V	V	STX zaddr,Y	HTAB
151	97	W	W	‡	HOME
152	98	X	X	TYA	ROT=
153	99	Y	Y	STA abaddr,Y	SCALE=
154	9A	Z	Z	TXS	SHLOAD
155	9B	ESC	‡	‡	TRACE
156	9C	\	\	‡	NOTRACE
157	9D	⊗©M	]	STA abaddr,X	NORMAL
158	9E	© ^	^	‡	INVERSE
159	9F		_	‡	FLASH
160	A0	SPACE	SPACE	LDY imm	COLOR=
161	A1	!	!	LDA (zaddr,X)	POP
162	A2	"	"	LDX imm	VTAB
163	A3	#	#	‡	HIMEM:
164	A4	$	$	LDY zaddr	LOMEM:
165	A5	%	%	LDA zaddr	ONERR
166	A6	&	&	LDX zaddr	RESUME
167	A7	'	'	‡	RECALL
168	A8	(	(	TAY	STORE
169	A9	)	)	LDA imm	SPEED=
170	AA	*	*	TAX	LET
171	AB	+	+	‡	GOTO
172	AC	,	,	LDY abaddr	RUN
173	AD	−	−	LDA abaddr	IF
174	AE	.	.	LDX abaddr	RESTORE
175	AF	/	/	‡	&
176	B0	0	0	BCS	GOSUB
177	B1	1	1	LDA (zaddr),Y	RETURN
178	B2	2	2	‡	REM
179	B3	3	3	‡	STOP
180	B4	4	4	LDY zaddr,X	ON
181	B5	5	5	LDA zaddr,X	WAIT
182	B6	6	6	LDX zaddr,Y	LOAD
183	B7	7	7	‡	SAVE
184	B8	8	8	CLV	DEF
185	B9	9	9	LDA abaddr,Y	POKE
186	BA	:	:	TSX	PRINT
187	BB	;	;	‡	CONT
188	BC	<	<	LDY abaddr,X	LIST
189	BD	=	=	LDA abaddr,X	CLEAR
190	BE	>	>	LDX abaddr,Y	GET
191	BF	?	?	‡	NEW

©=CTRL KEY ⊗=Shift Key ‡Undefined opcode

DEC	HEX	KEYS	VIDEO	6502	APPLESOFT
192	C0	@	@	CPY imm	TAB(
193	C1	A	A	CMP (zaddr,X)	TO
194	C2	B	B	‡	FN
195	C3	C	C	‡	SPC(
196	C4	D	D	CPY zaddr	THEN
197	C5	E	E	CMP zaddr	AT
198	C6	F	F	DEC zaddr	NOT
199	C7	G	G	‡	STEP
200	C8	H	H	INY	+
201	C9	I	I	CMP imm	−
202	CA	J	J	DEX	*
203	CB	K	K	‡	/
204	CC	L	L	CPY abaddr	^
205	CD	M	M	CMP abaddr	AND
206	CE	N	N	DEC A	OR
207	CF	O	O	‡	>
208	D0	P	P	BNE *+disp	=
209	D1	Q	Q	CMP (zaddr),Y	<
210	D2	R	R	‡	SGN
211	D3	S	S	‡	INT
212	D4	T	T	‡	ABS
213	D5	U	U	CMP zaddr,X	USR
214	D6	V	V	DEC zaddr,X	FRE
215	D7	W	W	‡	SCRN(
216	D8	X	X	CLD	PDL
217	D9	Y	Y	CMP abaddr,Y	POS
218	DA	Z	Z	‡	SQR
219	DB	[	[	‡	RND
220	DC	\	\	‡	LOG
221	DD	⊗M	]	CMP abaddr,X	EXP
222	DE	^	^	DEC abaddr,X	COS
223	DF	_	_	‡	SIN
224	E0			CPX imm	TAN
225	E1		!	SBC (zaddr,X)	ATN
226	E2		"	‡	PEEK
227	E3		#	‡	LEN
228	E4		$	CPX zaddr	STR$
229	E5		%	SBC zaddr	VAL
230	E6		&	INC zaddr	ASC
231	E7		'	‡	CHR$
232	E8		(	INX	LEFT$
233	E9		)	SBC imm	RIGHT$
234	EA		*	NOP	MID$
235	EB		+	‡	
236	EC		,	CPX abaddr	
237	ED		−	SBC abaddr	
238	EE		.	INC abaddr	
239	EF		/	‡	
240	F0		0	BEQ *+disp	
241	F1		1	SBC (zaddr),Y	
242	F2		2	‡	
243	F3		3	‡	
244	F4		4	‡	
245	F5		5	SBC zaddr,X	
246	F6		6	INC zaddr,X	
247	F7		7	‡	
248	F8		8	SED	
249	F9		9	SBC abaddr,Y	
250	FA		:	‡	
251	FB		;	‡	
252	FC		<	‡	
253	FD		=	SBC abaddr,X	
254	FE		>	INC abaddr,X	
255	FF		?	‡	

⊗=Shift Key ‡Undefined opcode