NANO SYSTEMS CORP.

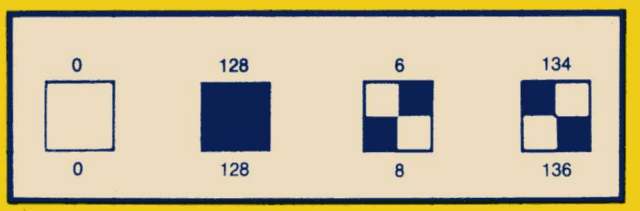
REFERENCE CARD FOR THE

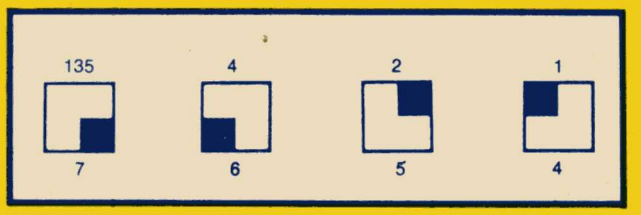
ZX80®, ZX81®

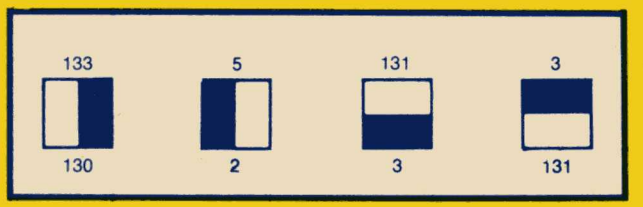
AND

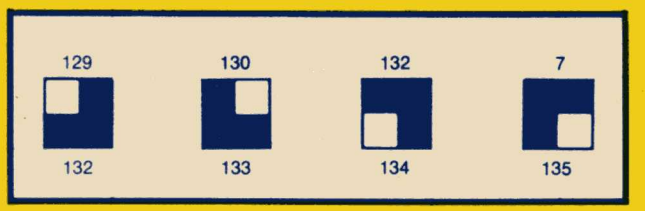
TIMEX SINCLAIR 1000™

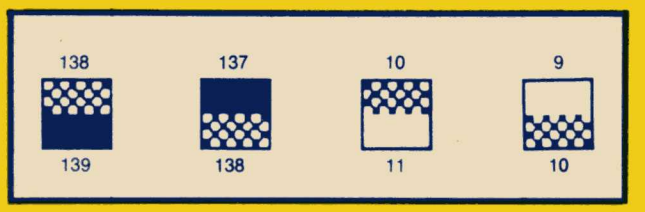
*ZX80 and ZX81 are registered trademarks of Sinclair Research LTD

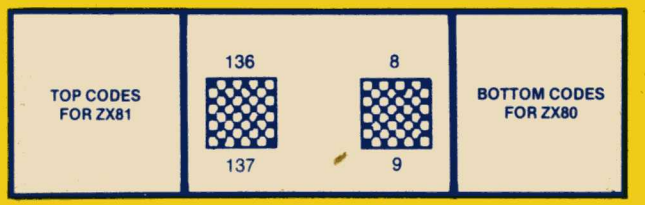












☆ = ZX80 ONLY — BASIC STATEMENTS — ★ = ZX81 ONLY

STATEMENT • DESCRIPTION

CLEAR • Clear all program variables.
CLS • Clear screen.
DIM *var (num)* • Define array "var" with "num" entries.
★**DIM *var (num* [, *num...*])** • Define array "var" with "num" entries.
FOR *var*=*num1* TO *num2* • Loop thru NEXT until var exceeds num2.
★**FOR *var*=*num1* TO *num2* STEP *incr*** • Loop thru NEXT, stepping "incr".
GOSUB *line* • Perform rtn at "line" until RETURN.
GOTO *line* • Branch to "line".
IF *cond* THEN *action* • Execute action based on condition.
INPUT *var* • Capture keyed reponse into "var".
LET *var*=*value* • Assign a value to variable.
★**LPRINT *item*** [; [,[*item*. . .] • Print data.
★**LPRINT TAB *num;item*** [; [, [**TAB**. . .] • Print with tabs.
★**LPRINT AT *line,col;item*** [; [, [**AT** . . .] • Print at position.
NEXT *var* • End of FOR loop. Increment var.
★**PAUSE *frames*** • Halt execution until time-out.
★**PAUSE 40000** • Halt execution until key pressed.
★**PLOT *(horiz, vert)*** • Turn on graphics block at this location.
POKE *addr, num* • Store number at address.
PRINT *item* [; [, [*item* . . .] • Display data.
★**PRINT TAB *num;item*** [; [, [**TAB** . . .] • Display with tabs.
★**PRINT AT *line,col;item*** [; [, [**AT** . . .] • Display at position.
RAND *num* • Re-seed random number generator.
REM • Remarks follow this statement.
RETURN • Return from a GOSUB.
★**SCROLL** • Scroll 1 line toward top of screen.
STOP • Interrupt program execution.
★**UNPLOT *(horiz, vert)*** • Turn off graphics block at this location.

PARENS NOT REQUIRED FOR PLOT/UNPLOT.

☆ = ZX80 ONLY — BASIC COMMANDS — ★ = ZX81 ONLY

COMMAND	DESCRIPTION
BREAK	Interrupt execution until CONT.
CONT	Continue after **BREAK** or **STOP.**
★**COPY**	Print screen contents on the printer.
EDIT	Edit current program line.
★**FAST**	Enter high-speed execution mode.
★**FUNCTION**	Enter function mode.
★**GRAPHICS**	Enter or exit graphics mode.
LIST [lineno]	Display program lines on screen.
★**LLIST [lineno]**	Print program lines on printer.
☆**LOAD**	Load first program found on cassette.
★**LOAD "name"**	Load program "name" from cassette.
NEW	Clear memory and restart.
RUN [lineno]	Execute program.
☆**SAVE**	Save program from memory to tape.
★**SAVE "name"**	Save program called "name" to tape.
★**SLOW**	Enter slow-speed execution mode.
EDIT SUBCOMMANDS	
⇧	Forward-space cursor.
⇧	Backspace cursor.
⇧	Scroll up.
⇧	Scroll down.
lineno	Delete line "lineno".
★**DELETE**	Backspace and erase.
☆**HOME**	Move cursor to line 0.
☆**RUBOUT**	Backspace and erase.
CURSORS AND MARKERS	
>	Current program line marker.
★F	Function prompter.
★G	Graphics prompter.
K	Keyword prompter.
L	Data/character prompter.
S	Syntax error marker.

☆ = ZX80 ONLY	BASIC FUNCTIONS	★ = ZX81 ONLY
FUNCTION	**RETURNS**	
CHARACTER/STRING FUNCTIONS		
CHR$ (num)	Character whose decimal value is num. ("A" = 38)	
★LEN (string)	Length of string.	
STR$ (num)	Character string of num.	
☆TL$ (string)	String without the first character.	
★string ([n1] [TO][n2])	Substr of string from pos. n1 to n2.	
NUMERIC FUNCTIONS		
ABS (num)	Absolute value of number. (−3.25 = 3.25)	
CODE (string)	Decimal value of first char in string.	
★INT (num)	Nearest whole number after rounding down. (2.5 = 2) (−2.5 = −3)	
★RND	Random number between 0 and 1.	
☆RND (num)	Random number between 0 and num +1.	
☆RND (0)	1.	
★SGN (num)	0 (Zero), 1 (Positive), − 1 (Negative).	
★VAL (String)	Number extracted from character string.	
MATH FUNCTIONS		
★ACS (num)	Angle whose cosine is num, in radians.	
★ASN (num)	Angle whose sine is num, in radians.	
★ATN (num)	Angle whose tangent is num, in radians.	
★COS (angle)	Cosine of angle radians.	
★EXP (num)	Inverse LN of num.	
★LN (num)	Natural logarithm of num. (Base 2.718281828)	
★PI	3.14159265.	
★SIN (angle)	Sine of angle radians.	
★SQR (num)	Square root of num.	
★TAN (angle)	Tangent of angle radians.	
Radians = Degrees/57.29577951	Degrees = Radians∗57.29577951	
I/O AND OTHER FUNCTIONS		
★INKEY$	Character value of key pressed on keyboard.	
PEEK (addr)	Decimal value of byte at location "addr".	
USR (addr)	Branch to machine language rtn at "addr".	
PARENS IN FUNCTIONS ABOVE ARE OPTIONAL IN MOST CASES.		

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	**LN (X + SQR(X∗X + 1))**
COSINE	**LN (X + SQR(X∗X − 1))**
TANGENT	**LN ((1 +X)/(1 −X))/2**
SECANT	**LN ((SQR(−X∗X + 1) + 1)/X)**
COSECANT	**LN ((SGN(X)∗SQR(X∗X + 1) + 1)/X)**
COTANGENT	**LN ((X + 1)/(X − 1))/2**

BASIC SPECIAL CHARACTERS AND OPERATORS	
;	Suppress tab after PRINT.
,	Tab to next column after PRINT.
blank	Tab to next line after PRINT.
" "	Double-quote char. in a string.
"	Character-string delimiter.
$	Identifies character-string variable. (A$)
blank	Identifies a numeric variable. (A)
E	Scientific notation. (1.7E38)
( )	Denotes priority in order of operations.
=	Equal or assignment.
+	Addition, concatenation.
−	Subtraction or minus sign.
∗	Multiplication.
∗∗	Exponentiation.
/	Division.
>	Greater than.
<	Less than.
>=	Greater than or equal to.
<=	Less than or equal to.
<>	Not equal.
NOT	Reverses true/false result.
AND	If both expr are true, result is true.
OR	If either or both expr are true, result is true.
NON-ZERO	True.
ZERO	False.
Order of operations: **( ), ∗∗, − (negation), ∗, /, +−, =, >, <, NOT, AND, OR.**	

SCREEN LAYOUT					
LINE	**DISPLACEMENT**	**Y-COORD**	**LINE**	**DISPLACEMENT**	**Y-COORD**
0	0	43 42	12	396	19 18
1	33	41 40	13	429	17 16
2	66	39 38	14	462	15 14
3	99	37 36	15	495	13 12
4	132	35 34	16	528	11 10
5	165	33 32	17	561	9 8
6	198	31 30	18	594	7 6
7	231	29 28	19	627	5 4
8	264	27 26	20	660	3 2
9	297	25 24	21	693	1 0
10	330	23 22	22	726	
11	363	21 20	23	759	

DEC	HEX	ZX80	ZX81	DEC	HEX	ZX80	ZX81
0	00	SPACE	SPACE	64	40		RND
1	01	"	▗	65	41		INKEY$
2	02	▌	▝	66	42		PI
3	03	▄	▀	67	43		
4	04	▖	▖	68	44		
5	05	▝	▌	69	45		
6	06	▗	▚	70	46		
7	07	▟	▜	71	47		
8	08	▚	▒	72	48		
9	09	▒	▄	73	49		
10	0A	▄	▀	74	4A		
11	0B	▀	"	75	4B		
12	0C	£	£	76	4C		
13	0D	$	$	77	4D		
14	0E	:	:	78	4E		
15	0F	?	?	79	4F		
16	10	(	(	80	50		
17	11	)	)	81	51		
18	12	−	>	82	52		
19	13	+	<	83	53		
20	14	*	=	84	54		
21	15	/	+	85	55		
22	16	=	−	86	56		
23	17	>	*	87	57		
24	18	<	/	88	58		
25	19	;	;	89	59		
26	1A	,	,	90	5A		
27	1B	.	.	91	5B		
28	1C	0	0	92	5C		
29	1D	1	1	93	5D		
30	1E	2	2	94	5E		
31	1F	3	3	95	5F		
32	20	4	4	96	60		
33	21	5	5	97	61		
34	22	6	6	98	62		
35	23	7	7	99	63		
36	24	8	8	100	64		
37	25	9	9	101	65		
38	26	A	A	102	66		
39	27	B	B	103	67		
40	28	C	C	104	68		
41	29	D	D	105	69		
42	2A	E	E	106	6A		
43	2B	F	F	107	6B		
44	2C	G	G	108	6C		
45	2D	H	H	109	6D		
46	2E	I	I	110	6E		
47	2F	J	J	111	6F		
48	30	K	K	112	70		⇧
49	31	L	L	113	71		⇩
50	32	M	M	114	72		⇦
51	33	N	N	115	73		⇨
52	34	O	O	116	74	HOME	GRAPHICS
53	35	P	P	117	75	EDIT	EDIT
54	36	Q	Q	118	76	NEWLINE	ENTER
55	37	R	R	119	77	RUBOUT	DELETE
56	38	S	S	120	78		K/L mode
57	39	T	T	121	79		FUNCTION
58	3A	U	U	122	7A		
59	3B	V	V	123	7B		
60	3C	W	W	124	7C		
61	3D	X	X	125	7D		
62	3E	Y	Y	126	7E		number
63	3F	Z	Z	127	7F		cursor

ZX80 ONLY : 16 - 26 , 64 - 212 Not available from keyboard.

DEC	HEX	ZX80	ZX81	DEC	HEX	ZX80	ZX81
128	80	■	■	192	C0		" "
129	81	"	▗	193	C1		AT
130	82	▌	▝	194	C2		TAB
131	83	▄	▀	195	C3		
132	84	▟	▜	196	C4		CODE
133	85	▙	▐	197	C5		VAL
134	86	▛	▞	198	C6		LEN
135	87	▜	▟	199	C7		SIN
136	88	▞	▒	200	C8		COS
137	89	▒	▄	201	C9		TAN
138	8A	▄	▀	202	CA		ASN
139	8B	▀	"	203	CB		ACS
140	8C	£	£	204	CC		ATN
141	8D	$	$	205	CD		LN
142	8E	:	:	206	CE		EXP
143	8F	?	?	207	CF		INT
144	90	(	(	208	D0		SQR
145	91	)	)	209	D1		SGN
146	92	−	>	210	D2		ABS
147	93	+	<	211	D3		PEEK
148	94	*	=	212	D4	"	USR
149	95	/	+	213	D5	THEN	STR$
150	96	=	−	214	D6	TO	CHR$
151	97	>	*	215	D7	;	NOT
152	98	<	/	216	D8	,	* *
153	99	;	;	217	D9	)	OR
154	9A	,	,	218	DA	(	AND
155	9B	.	.	219	DB	NOT	< =
156	9C	0	0	220	DC	−	> =
157	9D	1	1	221	DD	+	< >
158	9E	2	2	222	DE	*	THEN
159	9F	3	3	223	DF	/	TO
160	A0	4	4	224	E0	AND	STEP
161	A1	5	5	225	E1	OR	LPRINT
162	A2	6	6	226	E2	* *	LLIST
163	A3	7	7	227	E3	=	STOP
164	A4	8	8	228	E4	>	SLOW
165	A5	9	9	229	E5	<	FAST
166	A6	A	A	230	E6	LIST	NEW
167	A7	B	B	231	E7	RET	SCROLL
168	A8	C	C	232	E8	CLS	CONT
169	A9	D	D	233	E9	DIM	DIM
170	AA	E	E	234	EA	SAVE	REM
171	AB	F	F	235	EB	FOR	FOR
172	AC	G	G	236	EC	GOTO	GOTO
173	AD	H	H	237	ED	POKE	GOSUB
174	AE	I	I	238	EE	INPUT	INPUT
175	AF	J	J	239	EF	RAND	LOAD
176	B0	K	K	240	F0	LET	LIST
177	B1	L	L	241	F1		LET
178	B2	M	M	242	F2		PAUSE
179	B3	N	N	243	F3	NEXT	NEXT
180	B4	O	O	244	F4	PRINT	POKE
181	B5	P	P	245	F5		PRINT
182	B6	Q	Q	246	F6	NEW	PLOT
183	B7	R	R	247	F7	RUN	RUN
184	B8	S	S	248	F8	STOP	SAVE
185	B9	T	T	249	F9	CONT	RAND
186	BA	U	U	250	FA	IF	IF
187	BB	V	V	251	FB	GOSUB	CLS
188	BC	W	W	252	FC	LOAD	UNPLOT
189	BD	X	X	253	FD	CLEAR	CLEAR
190	BE	Y	Y	254	FE	REM	RETURN
191	BF	Z	Z	255	FF		COPY

ZX81 - SELECTED ROM CALLS

TO USE, POKE BYTES INTO ANY SAFE RAM, AND CALL VIA USR FUNCTION (LET A=USR(addr)). RESULTS RETURNED AS FUNCTION VALUE, AND IN BC REGISTER.

TO SCAN KEYBOARD FASTER THAN INKEY$

HEX	DEC	CODE
CD BB 02	205 187 2	CALL 02BBH
7C	124	LD A,H
C6 02	198 2	ADD A,2
38 09	56 9	JR C,+9
44	68	LD B,H
4D	77	LD C,L
CD BD 07	205 189 7	CALL 07BDH
06 00	6 0	LD B,0
4E	78	LD C,(HL)
D8	216	RET C
01 00 00	1 0 0	LD BC,0
C9	201	RET

TO MOVE CURSOR TO A ROW, COLUMN

01 cl rw	1 cl rw	LD BC,row col
CD F5 08	205 245 8	CALL 08F5H
C9	201	RET

TO OUTPUT A CHARACTER TO SCREEN

3E nn	62 nn	LD A,nn (nn=character)
D7	215	RST 0010H
C9	201	RET

TO OUTPUT CHARACTER STRING TO SCREEN

11 dd dd	17 dd dd	LD DE,addr of string (low byte first)
01 dd dd	1 dd dd	LD BC,length of string (low byte first)
CD 6B 0B	205 107 11	CALL 0B6BH
C9	201	RET

TO PLOT

01 xxyy	1 xx yy	LD BC,yyxx
3E 9B	62 155	LD A,9BH
CD B2 0B	205 178 11	CALL 0BB2H
C9	201	RET

TO UNPLOT

01 xx yy	1 xx yy	LD BC,yyxx
3E A0	62 160	LD A,A0H
CD B2 0B	205 178 11	CALL 0BB2H
C9	201	RET

TO SET "FAST"

CD 20 0F	205 32 15	CALL 0F20H
C9	201	RET

TO SET 'SLOW"

CD 28 0F	205 40 15	CALL 0F28H
C9	201	RET

HOW TO USE *FOR . . . NEXT*

The *FOR* statement sets up conditions for executing a series of BASIC lines over and over again until the ending conditions are met.

A SAMPLE OF USAGE:
```
10 FOR A = 0 TO 30
20 Z = Z + Y
30 R = R + Z
40 NEXT A
50 PRINT M
```
When you enter: FOR A = 0 TO 30
You are really saying:
Assign the value "0" to A.
Execute code beginning at line 20.
When you get to the "NEXT A" statement, add 1 to A.
Then, if A is greater than 30, go to the next line (50).
Otherwise, go back to line 20.

YOU CAN ENTER THE STATEMENT MANY WAYS:
```
FOR C = A TO B
FOR X = 1 TO W
FOR P = L TO 10
```

HOW TO USE *FOR . . . STEP . . . NEXT*

USAGE IS THE SAME AS **FOR . . . NEXT** ABOVE,
EXCEPT: **FOR A = 0 TO 30 STEP 2**
WHEN YOU GET TO THE "NEXT A" STATEMENT,
2 IS ADDED TO A.
OTHER FORMS CAN STEP IN A NEGATIVE INCREMENT,
FOR EXAMPLE:
FOR A = 30 TO 0 STEP −2

HOW TO USE MORE THAN 1 *FOR* AT A TIME

WHEN YOU USE MORE THAN ONE **FOR** AT A TIME, IT IS CALLED "NESTING".
WHEN NESTING, EACH **FOR** HAS ITS OWN **NEXT.**
THE LAST **FOR** ENTERED MUST FIND ITS OWN **NEXT** FIRST, OR YOU HAVE BAD PROCESSING.
```
FOR A=1 TO 4
FOR B =1 TO 7
FOR C=5 TO 10
NEXT C
NEXT B
NEXT A
```
What happens here, is:
The **A** loop is entered. It loops 4 times.
The **B** loop is called 4 times by the **A** loop.
Each time the **A** loop calls it, the **B** loop goes 7 times.
The **C** loop is called 7 times by the **B** loop.
Each time the **B** loop calls it, the **C** loop goes 6 times.
So, **A** loops 4 times, **B** loops 28 times, **C** loops 168 times.

HOW TO USE *IF . . . THEN*

The **IF** statement allows you to compare items against each other and **THEN** take an action based upon the results of the compare.
You can test for equal, less than, greater than, or any combination of the three.
You can also combine tests for a complex compare.
You can also combine **IF**s by making the next one the action.
This is called "NESTING".

Example of a simple **IF** statement:
```
IF A = B THEN LET C = D
IF A NOT = B THEN GOTO 1000
IF A > B THEN PRINT L$;
IF A < B THEN STOP
IF A<>B THEN GOSUB 2000
```

Example of a complex **IF** statement:
```
IF A = B AND C = D THEN GOSUB 3000        (Both required)
IF A = B OR C = D THEN GOTO 4000          (Only one required)
IF A = B AND C = D OR E = F THEN PRINT M$;
                  (First two, or last one required)
```
When using **AND**, then every test must be true to take the action.
When using **OR**, then only one of the tests must be true.
Of course, if there are **AND**s, then they must all be true before the **OR** makes its decision.
When none of the tests fits the condition for the **THEN**, control of your program falls through to the next line.

Example of combined **IF**s:
```
IF A = B THEN IF C = D THEN IF E = F THEN GOTO 5000
```
This is easier to see if written as follows:
```
IF A = B THEN
  IF C = D THEN
    IF E = F THEN
      GOTO 5000
```
At each level, when the condition is not true, then the next **IF** is not tested. Instead, all of this code is skipped, and the computer goes to the next line.
When "NESTING" **IF**s, it is the same as using **AND** without the nesting.

Another example:
```
IF A = B THEN IF C NOT = D OR E > F THEN PRINT R$;
```
If **A** equals **B** then we ask the next **IF** question, otherwise we go to the next line.
In the next **IF** question, if **C** is not equal to **D** then we will print.
If **C** is equal to **D**, then we ask if **E** is greater than **F**.
If **E** is greater than **F**, then we will print.
Otherwise, we will go to the next line.

ZX80 MEMORY MAP

ADDRESS		
DECIMAL	HEX	DESCRIPTION
16384	4000H	**ERROR-CODE MINUS ONE.**
16385	4001H	*BASIC SYSTEM CONTROL FLAG BITS.
16386	4002H	CURRENT BASIC STATEMENT NUMBER.
16388	4004H	**ADDRESS OF K OR L CURSOR.**
16390	4006H	**BASIC STATEMENT NUMBER AT ❯ CURSOR.**
16392	4008H	*ADDRESS OF PROGRAM VARIABLES.
16394	400AH	*ADDRESS OF WORKING STORAGE (KEY INPUT)
16396	400CH	***ADDRESS OF UPPER SCREEN.**
16398	400EH	***ADDRESS OF LOWER SCREEN.**
16400	4010H	***ADDRESS OF END-OF-SCREEN.**
16402	4012H	***NUMBER OF LOWER-SCREEN LINES.**
16403	4013H	NUMBER OF FIRST BASIC STMT ON SCREEN.
16405	4015H	**ADDRESS OF S MARK MINUS ONE.**
16407	4017H	**NUMBER OF STMT TO "CONTINUE" AT.**
16409	4019H	SYNTAX FLAG BITS.
16410	401AH	SYNTAX TABLE POINTER.
16412	401CH	**RANDOM NUMBER SEED.**
16414	401EH	SCREEN FRAME DISPLAY COUNT.
16416	4020H	ADDRESS OF FIRST CHAR OF FIRST VAR NAME IN LAST DIM, FOR, INPUT, LET, NEXT.
16418	4022H	VALUE OF LAST VAR OR EXPRESSION.
16420	4024H	***LINE POS OF NEXT SCREEN CHAR:** FROM 33 (LEFT) TO 2 (RIGHT) 1 = FIRST COL, NEXT LINE (LINE FULL) 0 = FIRST COL, (E-O-LINE.)
16421	4025H	***CURRENT SCREEN LINE (0=BOT, 23=TOP)**
16422	4026H	*ADDRESS OF CHAR AFTER PEEK OR POKE STMT.
16424	4028H	**USER PROGRAM AREA.**

* = DO NOT POKE. UNPREDICTABLE RESULTS.

ERROR CODES

ERROR CODES APPEAR AS: xx/yy
WHERE: xx is the error code,
AND: yy is the number of the last statement executed.

CODE	MEANING
0	Successful execution, or, GOTO-line too big.
1	NEXT has invalid variable, but, variable is assigned.
2	Variable not assigned, or, DIMensioned.
3	Bad subscript.
4	Memory exhausted.
5	Screen full.
6	Arithmetic number too large.
7	RETURN before GOSUB.
8	INPUT attempted in command mode. Illegal.
9	STOP was executed.
ZX81 ONLY	
A	Invalid parameter.
B	Invalid integer.
C	Invalid data in VAL string.
D	BREAK was pressed.
E	Unused.
F	SAVE name is a null string. Illegal.

ZX81 MEMORY MAP

ADDRESS		
DECIMAL	HEX	DESCRIPTION
0	0000H	MONITOR ROM.
8192	2000H	NOTHING. (USED FOR ROM IN SOME ADD-ON DEVICES).
16384	4000H	**ERROR-CODE MINUS ONE.**
16385	4001H	*BASIC SYSTEM CONTROL FLAG BITS.
16386	4002H	*ADDR OF NEXT INSTR AFTER "RETURN"ING.
16388	4004H	**ADDR OF LAST AVAIL BASIC BYTE +1.**
16390	4006H	**CURSOR MODE – K, L, F, OR G.**
16391	4007H	**CURRENT BASIC STMT NUMBER.**
16393	4009H	ROM VERSION CODE (0 = 8K).
16394	400AH	**BASIC STMT NUMBER AT ❯ CURSOR.**
16396	400CH	***ADDRESS OF SCREEN.**
16398	400EH	**ADDRESS OF NEXT SCREEN PRINT POS.**
16400	4010H	*ADDRESS OF PROGRAM VARIABLES.
16402	4012H	ADDRESS OF ASSIGNMENT VARIABLE.
16404	4014H	*ADDR OF WORKING STORAGE (KEY INPUT).
16406	4016H	*ADDR OF BYTE AFTER PEEK OR POKE.
16408	4018H	**ADDRESS OF S MARK MINUS ONE.**
16410	401AH	*ADDRESS OF MATH CALC STACK.
16412	401CH	*ADDR OF END OF MATH CALC STACK.
16414	401EH	B-REGISTER OF CALCULATOR.
16415	401FH	ADDRESS OF CALCULATOR MEMORY.
16417	4021H	NOT USED.
16418	4022H	***NUMBER OF LOWER-SCREEN LINES.**
16419	4023H	**NUMBER OF FIRST BASIC STMT ON SCREEN.**
16421	4025H	**LAST KEY PRESSED.**
16423	4027H	KEYBOARD DEBOUNCE STATUS.
16424	4028H	NUMBER OF BLANK LINES ABOVE AND BELOW MOVING GRAPHICS.
16425	4029H	***ADDR OF NEXT BASIC STMT LINE.**
16427	402BH	**NUMBER OF STMT TO "CONT"INUE AT.**
16429	402DH	SYSTEM FLAG BITS.
16430	402EH	STRING-TYPE LENGTH IN ASSIGNMENT.
16432	4030H	ADDR OF NEXT SYNTAX TABLE ENTRY.
16434	4032H	**RANDOM NUMBER SEED.**
16436	4034H	SCREEN FRAME DISPLAY COUNT.
16438	4036H	**LAST "PLOT" X-COORDINATE.**
16439	4037H	**LAST "PLOT" Y-COORDINATE.**
16440	4038H	**LSB OF ADDR OF NEXT "LPRINT" POSITION.**
16441	4039H	***"PRINT" COLUMN NUMBER.**
16442	403AH	***"PRINT" LINE NUMBER.**
16443	403BH	INTERNAL FLAG BITS.
16444	403CH	PRINTER BUFFER.
16477	405DH	CALCULATOR AUXILIARY MEMORY AREA.
16507	407BH	NOT USED.
16509	407DH	**USER PROGRAM AREA.**
17407	43FFH	END OF 1K SYSTEMS.
18431	47FFH	END OF 2K SYSTEMS.
32767	7FFFH	END OF 16K SYSTEMS.

* = DO NOT POKE. UNPREDICTABLE RESULTS.

Addr 16393 (4009H) thru 16508 (407BH) are always SAVEd with the program.

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