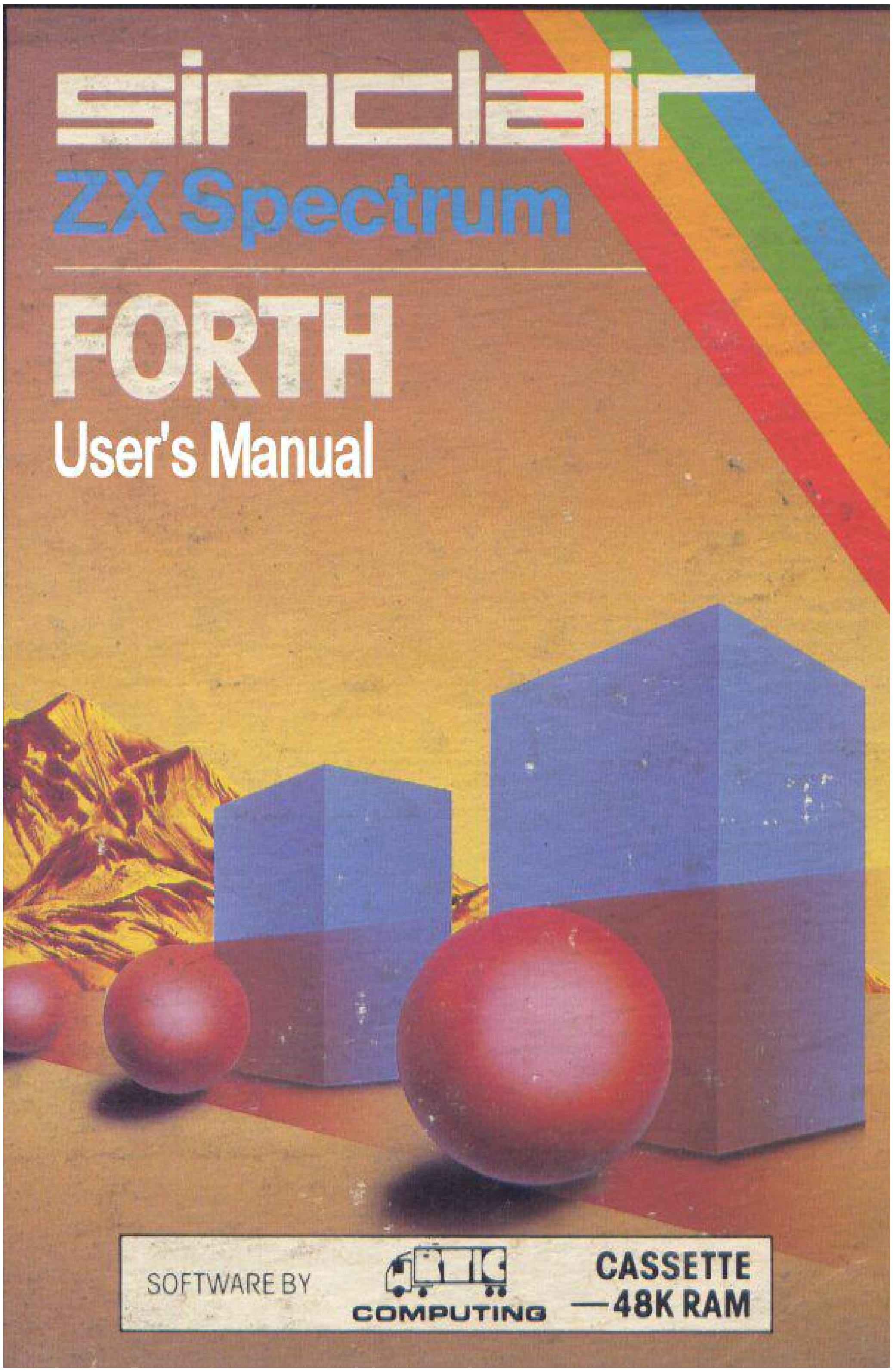

sinclair
ZX Spectrum
FORTH
User's Manual
SOFTWARE BY
ARTIC COMPUTING
CASSETTE —48K RAM

CONTENTS

Page	i		Installation Instructions
Page	1	1.0	Basic Operations
Page	1	1.1	Getting Started
Page	1	1.11	Screen Editor
Page	2	1.2	Words
Page	2	1.3	Numbers
Page	3	1.4	Parameter Stack
Page	3	1.5	Arithmetic
Page	4	1.6	Stack Manipulations
Page	5	1.7	Definitions
Page	5	1.8	Modes
Page	7		Table 1
Page	8		Table 2 and Table 3
Page	9	2.0	Data Declaration
Page	9	2.1	Constants
Page	9	2.2	Variables
Page	10	2.3	Altering Number Bases
Page	10	2.4	Arrays
Page	10	2.5	Other Memory Operations
Page	11		Table 4
Page	12	3.0	Input and Output
Page	12	3.1	Character Set & User Definable Characters
Page	12	3.2	Input of Information
Page	13	3.3	Output and Printing
Page	14	3.4	Number Printing
Page	14	3.5	Other Printing Operations
Page	15	3.6	Colour Hi Resolution and Sound
Page	16		Table 5
Page	18	4.0	Conditional Branches and Loop
Page	18	4.1	Conditional Branches
Page	19	4.2	Indefinite Loops
Page	20	4.3	Return Stack
Page	20	4.4	Controlled Loops
Page	21	4.5	Nesting Structures
Page	21		Table 6
Page	22	5.0	Tape Storage
Page	22	5.1	Saving Programs
Page	22	5.2	Loading Programs
Page	23	5.3	Screen Format
Page	23	6.0	Other Useful Commands
Page	25		Appendix ‘a’ ‘Error Codes’
Page	25		Appendix ‘b’ ‘Useful Routines’
Page	26		Glossary

SPECTRUM FORTH
INSTALLATION INSTRUCTIONS

LOAD the Forth compiler by:

    LOAD “ “ CODE (CODE extended mode ‘ I ‘).

Stop the tape when it has loaded.

The Forth Editor is next on the tape. To load the Editor, type the following:

    1 LOAD <ENTER>

The computer will respond with: READY CASSETTE.
Pressing ENTER will ititiate the cassette loading routine.
Start the tape and then press ENTER. The first screen takes a few seconds to load. Stop the tape when the characteristic blue and yellow lines finish. Wait for READY CASSETTE to appear and then start the tape and press ENTER. There are a total of 3 screens to be loaded.
If the program does not load properly, re-wind tape to the beginning of the screen.
After the third screen has loaded, ‘OK’ will appear.
N.B. Don’t be alarmed by an error **MSG # 4**
Your Editor is now loaded.
Now start programming in Forth.

HAPPY PROGRAMMING & GOOD LUCK

1.0 BASIC OPERATIONS

The easiest way to learn FORTH is to use it. As Forth Is an interactive language you can sit down and experiment. In this Users Manual, there are many examples to illustrate the capabilities of Forth. We suggest you try them for yourself.

1.1 GETTING STARTED

The Forth will announce itself and tell you how much free memory you have. The graphic ■ is your cursor and will appear when the system is ready for input from the keyboard. You are now ready to type a command terminated by ENTER. Until you have actually typed ENTER you may change your commands by using the DELETE to delete any unwanted characters, pressing it once for any character to be deleted, then re-type the remainder of the line.

The simplest command that you can give to Spectrum-Forth is an empty line. If you now type ENTER, your Spectrum-Forth should respond with OK, as it has seen there is nothing to do, finished the line and is waiting for another command. You should try this as it will show that your Spectrum-Forth is alive and welt and listening to you.

1.11 SCREEN EDITOR

Spectrum-Forth has been given a Screen Editor to help you' redefine any words you may have made an error with.
The Editor provides a copy cursor which may be moved with the Cursor Keys (CAPS SHIFT 5-8). The Editor is invoked by pressing any of the cursor keys. Position this cursor to the line you wish to Edit then press EDIT key CAPS SHIFT '1'. This will copy the character to the Forth Cursor and move both cursors across one character. As there is an auto repeat on all keys, hold the key down and you can copy a complete line.

EDITOR COMMANDS

CAPS SHIFT	5	Cursor left
" "	6	Cursor down
" "	7	Cursor up
" "	8	Cursor right
" "	1	Copy letter and move cursor across one character.

1.2 WORDS

The basic command unit of Forth is called a word. A word consists of a string of characters delimited by spaces. The only restrictions on words are that no word may contain a space, an ENTER, or a Graphics character. The word may be any length, with the first 31 characters, being significant, which allows the use of meaningful words. Words may be entered in upper or lower case.

After terminating a line of text with ENTER, the FORTH TEXT INTERPRETER scans the input breaking it up into words which will be executed in the order of entry. Each Word in Forth has a name (the way you refer to it) and a definition (the meaning i.e. what it does).

To execute a word, the interpreter searches the Dictionary to determine the definition of the word. If the word is found: the definition is interpreted. If it is not found, the interpreter attempts to convert the word to a 1 6 bit integer. If the word is not a valid number in the current base, an error message # 0 is given. The system returns with cursor for new input. This Dictionary may be extended by adding new words which call upon existing words. (See Section 1.7).

1.3 NUMBERS

Numbers can be expressed in any base (from 2 → 36). The system defaults to decimal at power up. However at any time you can use the commands (words) DECIMAL or HEX or you can define another base. This establishes the number base to treat succeeding numbers both for input and output. In general you should stick to one base throughout all your definitions to avoid confliction in interpretation.
Numbers may be typed in as positive or negative integers. Positive (unsigned) integers [0 → 65535] are accepted or signed integers [-32768 → 32767] as well. You may also use double precision numbers which are signed integers [-2147483648 → 21 47483647] . These 32 bit, double precision numbers must be prefixed by a dot '.'

Since all the numbers are stored in binary form, you can take advantage of Numeric base selection to perform number conversions to convert a decimal number to a Hexadecimal number. For example, type : DECIMAL 258 HEX .<ENTER>! and you will receive the response 102 OK (but remember you are now in HEXADECIMAL).

1.4 THE PARAMETER STACK

All computer programs manipulate data by using an established set of Parameters. In Forth, most of the Parameters are maintained on a PUSH DOWN stack called the Parameter Stack.
A Push Down stack is a particular arrangement of memory storage; Forth Words, which refer to the Parameter Stack, only do so by accessing the topmost items. [LAST IN FIRST OUT].
To place a number on the stack, you can type it as part of your input command. The Forth Word ‘ . ‘ dot removes the top number off the stack and prints it, in the current base, on the screen. For example: To place numbers on the stack 2 4 6 8 <ENTER>

```
The stack now looks like this 8
                              6
                              4
                              2
```

If you now type **.** <ENTER> the output will be 8 OK

```
The stack now looks like this 6
                              4
                              2
```

Now type . . . ENTER 6 4 2 OK. The stack is now empty.
Suppose you type **.** <ENTER> the computer responds
. ? MSG #1 (− stack underflow errors).
Forth also has another stack which is called the ‘RETURN STACK’ which is used by the interpreter for storage of return addresses. Any error message empties both STACKS.

1.5 ARITHMETIC

Forth has a pro-defined set of Arithmetic operators (See Table 1).
Since Forth uses a Push Down Stack and reverse polish notation the Parameters must be on the stack before the operation can be performed, thus to add two numbers together and display the results type in **: 5 27 + . <ENTER>** 32 OK.
Breaking this line down into its constituent parts you will find that

5	Pushes the value 5 onto the stack.
27	Pushes the value 27 onto the stack.
+	Removes the top. two items from the stack, adds them together and places the sum back onto the stack. NOTE: The stack has a net loss of 1 item.
.	Removes the top item from the stack and displays it 32 OK

Thus you leave the stack just as it was before you started.
Processing of Comparisons may also be unfamiliar. Forth assumes the conventions of positive logic: Truth Value

Ø - False

# Ø - True

The Forth relation words (such as < > = etc) may be remembered as written between the second stack entry on the left and the top stack item. on the right. Thus A B < will test for A > B and leave only a truth value on the stack, since both A and B have been removed.

1.6 STACK MANIPULATIONS

Other frequently performed operations are classified as Stack Manipulations for which Spectrum-Forth provides a few simple words. These words (described In Table 3) are generally used to maintain discipline in the stack when it contains Parameters. Practising with these words will make them useful to you quickly.

When practising, keep in mind 2 elementary rules:

1. Keep parity — everything on - must come off.
2. Never remove more items than you have placed on the stack.

After you have become familiar with both Arithmetic Operators and Stack Manipulators, you will want to create your own words.
For example: To square a number use:
: SQUARE DUP * . ; <ENTER> OK

1.7 DEFINITIONS

Part of Forth's power lies in the ability to allow you to define your own words. For example, it might be that we often have to calculate the cube of a number. It is easy to define a new word to do the job.

: CUBE DUP DUP * * . ; <ENTER> OK

Here is what each component does:

:	Begins a definition.
CUBE	The name of the word to be added to the Dictionary.
DUP DUP* *	The Forth words defining what to do.
;	Ends a definition.

After making this definition, we can calculate and print a cube whenever we want.
For example:
4 CUBE <ENTER> 64 OK
3 CUBE <ENTER> 27 OK
What would have happened if you had used the word CUBE before it was defined ? Forth would not have allowed it. It would print CUBE ? MSG # Ø (undefined word error). Fortunately for the novice programmer. Forth has a rich vocabulary of pre-defined words.
For example:

?	which prints the contents of the memory locations addressed by the top of the stack.
?	has a simple definition.

: ? @> . ;
Another simple combination that is pre-defined is **1+** which adds 1 to the item at the top of the stack.
1 has the definition **: 1+ 1 + ;**

1.8 MODES

The Forth text interpreter operates in two modes — Immediate Execution and Compilation.
In Immediate Execution Mode, each word of the input string is looked up in the Dictionary and executed immediately.
During Compilation however, most word are not executed. Instead, a reference to them is compiled in the Dictionary.
The word **:** places the interpretation in Compile Mode, whereas **;** returns it to Immediate Execution.

The compiled form of the definition consists of pointers to the addresses of routines that will be executed by the inner-interpreter when the definition is executed. This form of interpretation is extremely fast. To distinguish between the modes of immediate execution and compilation, try the following examples:

905 . <ENTER> 905 OK

executes immediately (note the interaction).

: SHOW 905 . ; <ENTER> OK

compiles nothing happens yet.

SHOW <ENTER> 905 OK

executes the compiled routine to produce the desired result.

To learn quickly, you must practice with the basic Forth words and words you evolve out of experiments. Develop a kind of notation which will leave you with a sketch of what you have done (to help you avoid making the same mistakes twice).

EXERCISES

1.What Is the difference between DUP * DUP * and DUP DUP * * ?

2.What is the difference between OVER SWAP and SWAP OVER ?

TABLE 1
ARITHMETIC OPERATORS

Word Normal 16 Bit	Description	Example of Stack Before	Example of Stack After
		Top	Top
+	Adds	9 6 2	9 8
-	Subtracts	9 6 2	9 4
*	Multiply (signed)	9 6 2	9 12
/	Divides (signed)	9 6 2	9 3
1+	Add 1	9 6 2	9 6 3
2+	Add 2	9 6 2	9 6 4
ABS	Leaves absolute value	9 -6 -2	9 -6 2
MAX	Leaves largest of top two entries	9 6 2	9 6
MIN	Leaves smallest of top two entries	9 6 2	9 2
MINUS	(Unary minus) 2's compliment	9 6 2	9 6 -2
MOD	Leaves Modulus (division remainder)	9 6 2	9 0
*/	Multiplies 2nd & 3rd and divides by first	9 6 2	27
*/MOD	As above but leaves remainder	9 6 2	27 0
+-	Apply sign of 1st to 2nd	9 6 -2	9 -6
/MOD	Divide 2nd by 1st leaving remainder and quotient	9 6 2	9 0 3
AND	Leave bitwise logical	9 6 3	9 2
OR	Bitwise logical OR	9 6 3	9 7
XOR	Bitwise exclusive OR	9 6 3	9 5

Double precision and Mixed operator commands also exist (see glossary). They are proceeded until either D M or U.

TABLE 2
COMPARISON OPERATORS

Word	Description	Before	After
<	Compares: leaves 1 if 2nd less		
	than 1st, otherwise Ø	9 6 2	9 Ø
>	Compares: leaves 1 if 2nd greater		
	than 1st, otherwise Ø	9 6 2	9 1
Ø=	Tests for zero; leaves 1 if top entry		
	is less than zero, otherwise Ø	9 6 2	9 6 Ø
Ø >	Tests for negative; leaves 1 if top		
	entry is less than zero, otherwise Ø	9 6 2	9 6 Ø
=	Tests for number equals; leave 1 if		
	top two equal, otherwise Ø	9 6 2	9 6 Ø

TABLE 3
STACK MANIPULATION OPERATORS

Word	Description	Before	After
*	Prints item on top of stack	1 2 3	1 2
DROP	Discard top entry	3 2 1	3 2
DUP	Duplicates top entry	3 2 1	3 2 1 1
-DUP	Duplicates top entry	3 2 1 or	3 2 1 1
	if it is non zero	3 2 ∅	3 2 ∅
OVER	Copies 2nd entry over top entry	3 2 1	3 2 1 2
ROT	Rotates top 3 entries	4 3 2 1	4 2 1 3
SWAP	Swaps top 2 entries	3 2 1	3 1 2
R	Prints 2nd item right Justified		
	in a field of 1st entry	3 2 1	3
R	Copy top of Return stack	ret. ret.	
	to computation stack	2∅ 3∅ 2∅ 3∅	
		1 2 3	1 2 3 3∅

2.0 DATA DECLARATION

Forth allows you to set aside memory for constants, variable and arrays.

2.1 CONSTANTS

To assign names to constants you use the word CONSTANT. These are used as they are easier to recall than the number or are used often.
For example: **5280 CONSTANT FT/MILE** <ENTER> OK creates a new word **FT/MILE** and assigns it a value 5280. After FT/MILE has been defined you can use it as you would 5280 to place value on the STACK.
e.g. 3 FT/MILE * computes the number of FT in three miles.
NB. Once a value has been defined as a constant, its binary value is independent of the current number base.

2.2 VARIABLES

The Forth word VARIABLE names a location whose value is likely to change. Suppose we wish to keep the score of a game of Space Invaders then we can declare a variable as follows:
0 VARIABLE SCORE <ENTER> OK
↑
Initial value.
When you invoke a variable by name its address is placed on the stack. The FORTH WORD @ replaces the address on the stack by the contents of the 2 bytes at that address.
For example: To place your score on the stack you use:
SCORES @ <ENTER> OK
Sometimes you need to examine the contents of a variable. The Forth word '?' outputs the value of the variable whose address is on the top of the stack.
For example: **SCORE ? <ENTER> Ø** OK
The word '**!**' is used to store a 16 bit value into a location. '**!**' uses the value which is the 2nd item on the stack and stores it into the address which is on the stack.
For example: To set the score to 1**ØØ**
1ØØ SCORE ! <ENTER> OK
The word '+**!**' adds a value to a variable (location).
For example: To increase your score by **1ØØ**
1ØØ SCORE +! <ENTER> OK
NB. Since the Parameter is used to store intermediate values, the need for temporary variables is eliminated.

2.3 ALTERING NUMBER BASES

Forth has a user variable called base which stores the current 'number base'. You may alter this variable to any value between 2 and 36 to select bases other than decimal and hex. For example, suppose you wish to work in binary, then you may do this by:

2 BASE ! <ENTER> OK

then all the following numbers will be printed in binary. Remember, input must also be in binary.

2.4 ARRAYS

Arrays of data are important in many applications.
For example, instead of having 1Ø variables T0, T1, T2 etc. it would be better to use 10 successive data elements TEMP. Through suitable addressing arithmetic, you may compute the required elements address. This is more flexible to program as well as more economical of Dictionary space.
To set aside space in the Dictionary for arrays, you use the Forth word 'ALLOT'. In the case of temp above you write

0 VARIABLE TEMP 18 ALLOT <ENTER>OK

Where 0 VARIABLE TEMP	- defines a variable (2 bytes wide) named -TEMP
18	- puts 18 on stack
ALLOT	-allocates a further 18 bytes for Temp

To assess the n'th element, place n on top of the stack and follow it with: **2 * TEMP + @ <ENTER>** OK

NOTE Elements numbered 0 - 9 **NOT** 1-10
values for n outside this range will give unpredictable results.
To initialise the n'th element type:

(value) n 2 * TEMP + ! <ENTER> OK

2.5 OTHER MEMORY OPERATIONS

There are 4 words which can be used to manipulate memory locations.

1.CMOVE	Move the specified quantity of bytes (1st on stack) from address (3rd on stack) to address (2nd on stack). Contents of lowest address moved first.
Example:	**16396 8000 64 CMOVE <ENTER>** OK In theory, moves 64 bytes from 16396 to 8000.
2.FILL	Fills memory at address (3rd on stack) with quantity (2nd on stack) of bytes (1st on stack).
Example:	**8000 64 0 FILL <ENTER>** OK Puts 64 bytes of 0 from 8000 onwards.
3.ERASE	Fills a block of memory with zero's, equivalent to 0 FILL
Example:	**8000 64 ERASE <ENTER>** OK Erases 64 bytes starting from 8000.
4.BLANKS	Fills a block with spaces (=32 FILL) (ASCII) (See Chapter 3)

Example: 8000 64 BLANKS <EIMTER> OK
Puts 64 32's from 8000 onward

TABLE 4
MEMORY OPERATIONS

Word	Description	Before	After
@	Fetch contents of item whose address is at top of stack	100	—236
!	Stores the 2nd item on stack into location whose address is on top of stack	3 20000	empty
?	Fetches and prints contents of item whose address is on top of stack	100	empty
+!	Increments the location whose address is on top of stack by 2nd item on stack	701 2000	empty
C@	Fetch a byte whose address at top of stack	100	20
c!	Stores a byte (2nd) into the location at top of stack	254 2000	empty

CMOVE
FILL
ERASE
BLANK

} As 2.4

(Double Precision) see glossary.

EXERCISES

1. Define EXCHANGE to exchange the contents of 2 variables that is if:
   A and B are variables then the result of the command A B Exchange should be to place the value of A in B and value of B in A.
2. Define TRANSFER to move data between two arrays of the same length.

3.0 INPUT AND OUTPUT

In order to perform any function, it is necessary To input data into the computer and obtain the results. Forth has various ways of doing this.

3.1 CHARACTER SET

The User Definable Graphics (144 - 164) may be defined using the word DEF. To define a character, put 8 bytes on the stack representing bit patterns of the bottom row of the character to the top row, then put the character code you wish to change, then the word DEF. This may seem complex, but here's how to define a man. e.g **HEX .** <ENTER>
81 81 66 3C FF 7E 18 18 <EIMTER>
DECIMAL <ENTER>
144 DEF <ENTER> OK
The character 144 is now a little man.

3.2 INPUT OF INFORMATION

Forth has no real input commands as such, because the numbers of Parameters for each command is stored on the Parameter stack. This is usually placed on the stack preceding the execution of the command. For example, suppose we wish to calculate:
($4x^3$ -3x +2) for any value x. Then it is easy to define a command which finds x^3 and then one to calculate the rest of the cubic.
Example

: CUBE DUP DUP * * ;
: CUBIC DUP CUBE 4 * SWAP3* - 2 + . ; <ENTER> OK

Define a function CUBIC to calculate the cubic. The value of the Parameter x to be calculated is placed on the stack before the command is used. This is done as follows:

8 CUBIC <ENTER> 2026 OK

Thus any parameter may be passed into a command and no input command is needed.
However, Forth has got a command to accept a key from the key-board. This Forth word is 'KEY'. This is similar to the INKEY$ in Basic. However, 'KEY' waits for key to be pressed, whereas INKEY$ does not.
Suppose, during the execution of a program, you need some input from the keyboard.
For example: 'DO YOU WANT INSTRUCTIONS Y OR N?'
KEY then places the ASCII value of the key pressed into the Parameter stack where it may be examined. In Forth there is no routine to input a number more than 1 digit from the keyboard, However, there is a routine listed in the Appendix to input a number called 'INPUT'. Try and write a routine to do this yourself.

3.3 OUTPUT AND PRINTING

Forth offers you several different ways of outputting information. The most frequently used is output of a line of text. The Forth word used for this is . " followed by the message and terminated by "

For example . " THIS IS A LINE OF TEXT "   < ENTER>

prints 'THIS IS A LINE OF TEXT: on the screen. Any further output will appear on the same line. The Forth word 'CR' performs an 'ENTER' or 'Carriage Return' and all proceeding output starts at the beginning of a new line. The Spectrum Forth word AT is the same as Basic and may be used to position the cursor for output.

e.g. to print 3rd line, 2nd column: 3 2 AT . " HELLO "

Spectrum Forth automatically scrolls the screen if the current position is at the bottom of the screen.

A further way to print a character can be achieved by the Forth Word 'EMIT' which prints the character whose ASCII value is at the top of the Parameter stack. This may have been put there by the 'KEY' command. This also allows you to print a character, which is not directly available from the keyboard.

For example: 143 **EMIT <ENTER>** ☐ OK prints the character with ASCII value 143, which in this case is a 'graphic space'. Output may also be sent to the printer.

There is a variable in Spectrum Forth called PRINT. If this variable is set to Ø then the output is sent to the screen. If it is set to 1, then it is sent to the printer and the screen.

For example, to send the output to the printer, type:

1 PRINT ! <ENTER> and

Ø PRINT ! <ENTER> to turn the printer off.

The equivalent to Copy in Basic is also supplied by the Spectrum Forth word COPY.

Type: **COPY <ENTER>** to make a copy of the screen onto the printer.

3.4 NUMBER PRINTING

The simplest way to print a number is to use the Forth word ','
dot, which you have already met. This prints the top number on the stack in the minimum field width I.e. no proceeding zero's and one space after the number. Number formatting may be achieved by the following Forth words. ' .R' prints the number in a right justified field of a given width.

For example: **103 4 .R <ENTER>** <u>1030</u>K prints **103** in a field of **4** characters. Forth also gives you the ability of pictured output which enables you to format the output as required.

' <# ' starts the pictured output definition and it expects a double precision number on the stack. As you usually use single precision numbers, a single precision number may be converted to a double precision one by the Forth word "**S - > D**" which converts the top number on the stack to a double precision one. Within the pictured output you may use the following Forth words:

' # ' puts the next digit into the output buffer starting from the lowest value i.e. 112 first # puts 2 into buffer then the next #puts 1 into the buffer etc.

' # **S**' puts the remaining digits into the output buffer if non-zero.

'**HOLD**' used as '**46 HOLD**' which puts the character whose character code is **46** into the next part of the buffer.

' #> ' terminate pictured output leaves the address and length of the output buffer on the stack. Hence the string may be outputted using the '**TYPE**' command.

<u>Example</u>: Suppose you define a new command 'PIC' to produce pictured output.

: PIC < # # # 46 HOLD # # S # >; <ENTER > <u>OK</u>

This will print any double precision numbers as **(. . . . x . x x).**

<u>Example</u>: **11473. PIC TYPE <ENTER><u>11 4.73</u>** <u>OK</u>

0. PIC TYPE <ENTER> <u>0.00</u> <u>OK</u>

This needs practice to master the art of using numeric pictured output. Double precision commands exist (See Glossary)

.

3.5 OTHER PRINTING OPERATIONS

ZX-Forth has four further commands to aid formatting on the screen.

'SPACE'	This word as it suggests, prints a space on the screen. (Equivalent to 32 EMIT).
'SPACES'	This word Is used to print a given number of spaces specified by the number on the top of the stack.
For example**:**	**5 SPACES <ENTER> .....** <u>OK</u> prints **5** spaces on the screen.
'HOME'	This word moves the print position to the top left hand corner of the screen. All further output starts at the top of the screen.
'CLS'	This word clears the screen and moves the print position to the top left hand corner of the screen.

3.6 COLOUR HI RESOLUTION AND SOUND

Most of the Colour and Hi Resolution graphics commands are available in Standard Spectrum Forth. The colour codes are the same as Basic: 1 = Red etc. The only difference using these commands in Forth is that the parameters are proceeding the command rather than following the command.

i.e. to change the ink colour to red;

BASIC	FORTH
INK 1 <ENTER>	1 INK <ENTER>

N.B. In Forth, these colours are only temporary and revert back to the power up colours when a CLS is executed. To make these colours permanent, type the word PERM.

e.g. Blue paper, White ink

2 Paper 7 Ink PERM <ENTER>

Colour Sound and Hi Resolution Commands:

Basic		Forth
INK	X	X INK
PAPER	X	X PAPER
FLASH	X	X FLASH
BRIGHT	X	X BRIGHT
INVERSE	X	X INV
OVER	X	X GOVER
BORDER	X	X BORDER
PLOT	X, Y	X, Y PLOT
CIRCLE	X.Y. radius	radius. X,Y CIRCLE
DRAW	X,Y	X,Y DRAW
BEEP	X,Y	X,Y BEEP

TABLE 5

CHARACTER CODES

0 - 7	CONTROL CHARACTERS	
8	DELETE CAPS – shift `0`	
9 - 12	CONTROL CHARACTERS	
13	ENTER	
14 - 31	NOT USED	
32	SPACE	
33	!	Symbol- shift `1`
34	“	Symbol- shift `P`
35	#	Symbol- shift `3`
36	$	Symbol- shift `5`
37	%	
38	-	Symbol- shift `6`
39	`	Symbol- shift `7`
40	(	Symbol- shift `8`
41	)	Symbol- shift `9`
42	*	Symbol- shift `B`
43	+	Symbol- shift `K`
44	‘	Symbol- shift `N`
45	-	Symbol- shift `J`
46	.	Symbol- shift `M`
47	/	Symbol- shift `V`
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	

54	6	
55	7	
56	8	
57	9	
58	:	Symbol- shift `Z`
59	;	Symbol- shift `O`
60	<	Symbol- shift `R`
61	=	Symbol- shift `L`
62	>	`T`
63	?	`C`
64	@	`2`
65	A	
66	B	
67	C	
68	D	
69	E	
70	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
80	P	
81	Q	
82	R	
83	S	
84	T	
85	U	
86	V	
87	W	
88	X	
89	Y	
90	Z	

91	[	Symbol- shift `Y`		
92	\	Symbol- shift `D`		
93	]	Symbol- shift `U`		
94	↑	Symbol- shift `H`		
95	-	Symbol- shift `Ø`		
96	£			
97	a			
98	b			
99	c			
100	d			
101	e			
102	f			
103	g			
104	h			
105	i			
106	j			
107	k			
108	l		127	©
109	m		128	
110	n		129	
111	o		130	
112	p		131	
113	q		132	
114	r		133	
115	s		134	
116	t		135	
117	u		136	
118	v		137	
119	w		138	
120	x		139	
121	y		140	
122	z		141	
123	{		142	
124	|		143	
125	}		144	- 164 User Definable Graphics
126	“		165	- 255 Key Words in Basic not used

4.0 CONDITIONAL BRANCHES AND LOOPS

Forth provides conditional branching statements, which alter the order in which commands are executed depending upon a given condition. Forth also provides looping atructures to repeat a sequence of commands a given number of times.
N.B. Condition branches and loops cannot be executed directly and must be included with a definition.

4.1 CONDITIONAL BRANCHES

3 compiling words 'IF' ELSE ENDIF (or THEN) are used to compile conditional branches in a definition. In Forth, the 'IF' command examines the top of the stack t& determine which branch will be taken. A conditional branch has the following structure:
: DEFINITION condition IF (true) this ELSE (false) that THEN continue ; where

: DEFINITION	- begins the definition.
condition	- places a condition (non-zero/zero) on the stack.
IF	- removes and tests the number on the stack.
this	- executes this if number non-zero (true).
ELSE	
that	- executes that if the number is zero (false).
THEN	
continue	- continues from both lines.

IF marks the place where the top of the stack is popped and examined. If the value is non-zero, everything up to ELSE is executed and at ELSE execution skips to THEN. On the other hand, if the stack value is zero, everything up to ELSE is skipped after ELSE is executed.
The 'ELSE that' bit is optional and may be omitted if not needed.
The "truth" value on the stack is often the result of a comparison that uses one of the Forth comparison operators i.e. < > == etc. (See Chapter 1).
The two truth values may be combined by the Forth words 'AND OR XOR". For example:

AND	- leaves true if top two truth values on stack are true.
OR	- leaves true 1 or both are true.
XOR	- leaves true if 1 is true and other is false.

FOR example:
1 1 AND gives 1   1 Ø AND gives Ø   (See truth tables in Table 6)
For example: Suppose we wish to define a Forth word to mark examination papers, say 50% Pass, less than 50% Fail.
We may define a word as follows:

: EXAM 50 < IF ." FAIL " ELSE' ." PASS " THEN CR ;
< ENTER> <u>OK</u>

Then to use the word use   Mark EXAM <ENTER >
If less than 50 then computer prints FAIL
If mark greater than or equal to, the computer prints PASS.

4.2 INDEFINITE LOOPS

Forth also includes a series of Looping structures which repeat a set of commands either until a given condition is satisfied or not, or a set number of times. In this section we will look at the former.
The first type is
: EXAMPLE BEGIN process condition UNTI L continue;
<ENTER> OK where

: EXAMPLE	- begins definition
BEGIN	- marks the beginning of an indefinite loop.
process	- defines the action to be executed.
condition	- leaves a "truth" value on the stack.
UNTIL	- pops the value off the stack and returns to BEGIN if the condition is zero (false).
continue ;	- continue execution when value is true (non-zero).

NB END and UNTIL may be used interchangeably,
e.g. Suppose you wish to search through memory for a given 16 bit number and print the address of the number, we can define a word SEARCH to do this for us. Suppose we are searching for the first occurrence of zero from the beginning of memory.

: SEARCH Ø BEGIN DUP @ SWAP 1+ SWAP Ø = UNTIL 1- . ;
<ENTER> OK

The code between BEGIN and UNTIL is repeatedly executed until the value found is **Ø**. The **Ø** before the loop is initialisation in the case of the start address of the search.
The code after UNTIL prints the address of the occurrence of **Ø**.
A second form of indefinite looping is
: EX1 BEGIN condition WHILE process REPEAT continue
<ENTER> OK
where

: EX1	- begins definition.
BEGIN	- marks the beginning of an indefinite loop.
condition	- leaves a truth value on the stack.
WHILE	- if the truth value is true (non-zero).
process	- then process executed.
REPEAT	- returns to begin.
continue;	- if truth value false (zero) then continue is executed.

Example to search through memory as before.

: SEARCH 1 0 DUP @ WHILE 1+ REPEAT . ; <ENTER> OK

try and work this out for yourself.
The final kind of indefinite loop is of the form

: EXZ BEGIN process AGAIN ; <ENTER> .

which is repeated indefinitely. This is an Infinite loop and can only be terminated by the break key.

4.3 RETURN STACK

Forth uses two stacks, the Parameter Stack and the Return Stack. This is because otherwise Parameters and Return addresses may get confused. There are several commands to transfer Parameters from one stack to another.

> R	removes a number from the Parameter stack and places it on the Return stack.
R >	removes a number from the Return stack and places it on the Parameter stack.
NB	If used in a definition, both must be used to compliment each other.
R or I	make a copy of the number or the top of the Return stack onto the Parameter stack. The return stack is not altered.

In order to try to use these commands, try the following: Define a Forth word 2SWAP to swap the first 2 numbers of the stack with the third and forth, that is after
1 2 3 4 5 2SWAP the stack should contain
1 4 5 2 3 (with 3 on the top).

4.4 CONTROLLED LOOPS

A controlled loop is one which is repeated a certain number of times. Forth provides a **DO ... LOOP** structure for this. This takes the form
: TEN-TIMES 1Ø Ø DO process LOOP ; <ENTER> OK

: TEN-TIMES	- begins definition
1Ø	- gives terminating value
Ø	- gives starting value
DO	- transfers loop Parameters to the Return stack.
Process	-
LOOP ;	- repeats the loop 1Ø times.

Within the loop, the loop index may be accessed by the Forth word ' I '
Suppose we wish to print the numbers Ø -9 on the screen one per line, we can define a Forth word NO. to do this.
: NO 1Ø Ø DO CR I . LOOP ; <ENTER > OK
If you wish to increment by a step other than 1, you may use the
DO . . . + LOOP structure. For example
: BY2 1Ø Ø DO 2 +LOOP ; <ENTER>OK

$$\underset{\text{step}}{\underset{\uparrow}{2}}$$

where

: BY2	- begins definition
1ØØ	- Parameters of loop.
DO	-begin loop (as before).
2	- puts step onto stack.
+ LOOP	- removes step from stack and adds to current loop index.

If result is less than terminating value, loop is repeated with new index value. Otherwise loop is terminated.
The step may be positive or negative. By using a negative step, the loop will count down. However, the Parameters must be placed in

reverse order   i.e. Ø 1Ø rather than 1Ø Ø.

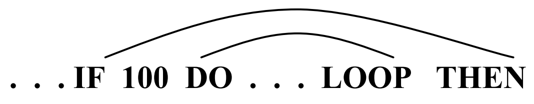
For example, to print the numbers 10 to 1, the following Forth word may be defined
: TENT01 Ø 1Ø DO CR I . -1 +LOOP ; <ENTER> OK
In this case LOOP checks if the loop index is greater than the terminating value and repeats if it is true. If you need to leave a **DO . . . LOOP** construct before the loop has finished.
For example: If a certain condition is met, then the command '**LEAVE**' will cause the loop to terminate at the next LOOP or + LOOP.   **Example:**
: EX3 1Ø 0 DO I DUP 6 = IF LEAVE ELSE . THEN LOOP ;
<ENTER> is a rather clumsy way to print the numbers Ø -5 but illustrates the use of this word.

Exercise

1 Define POWER so that m n POWER computes the n'th power of m, for non-negative n.

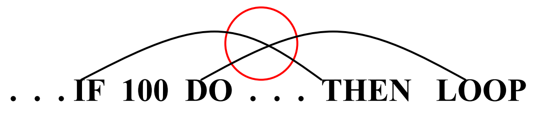
4.5 NESTING STRUCTURES

DO . . . LOOP and IF ... THEN sequence may contain either such sequence but only if they are properly nested, that is one entire DO . . . LOOP may be inside another but they may not overlap I
For example:
RIGHT:



. . . IF 100 DO . . . LOOP THEN

WRONG



. . . IF 100 DO . . . THEN LOOP

Exercise

1. How would you define MAX MIN and ABS ? (All supplied as SPECTRUM-Forth).
2. Define FACTORIAL to compute the factorial of a number.

TABLE 6

AND

	10
1	10
0	00

OR

	10
1	11
0	10

XOR

	10
1	01
0	10

5.0 TAPE STORAGE

Forth normally works interactively and once a definition has been typed in, there is no way of changing it without re-typing it. Forth also provides a method of storing the code on. a numbered screen. A screen consists of 16 lines of 64 characters. Programs are stored on a screen using the Editor — see Editor Manual.

5.1 SAVING PROGRAMS

Once the program has been written on the screen, it is possible to save it onto cassette. To do this, you must set up your computer for saving (see relevant chapter In computer manual). The command FLUSH tells SPECTRUM. Forth to save the current screen onto cassette. If you type **FLUSH** <ENTER> , SPECTRUM-Forth will respond **READY CASSETTE**. Now press record on your cassette recorder and press <ENTER> . One screen takes 30 seconds to save or F LUSH. If you type any character other than <ENTER> then the command is aborted and the cursor re-appears.

5.2 LOADING PROGRAMS

SPECTRUM- FORTH gives you two words to load a program from tape. 'LIST' is used to list a screen in the form n LIST where n is the screen number. If screen n is already in memory, it is listed onto the screen. If it is not, SPECTRUM- Forth will try to load it from cassette. Connect your computer for loading (see manual) and position the tape to the silent part immediately preceding the screen you want to load. Press <ENTER> and then play on the cassette recorder. If the screen loads correctly it will be listed on the screen. If however, it does not or you attempted to load the wrong screen, then the 'READY CASSETTE' will be repeated. Try again, use <ENTER>' or any other key to abort as with FLUSH.
The variable FIRST contains the screen number of any screen currently in memory. This can be examined by:
FIRST ? <ENTER>
If you wish to stop loading, press 'space key'.
'**LOAD**' is used to compile the definition in a screen. This must be preceded by the screen number as with '**LIST**'. If the screen is not already in memory, it is treated as if it had been typed from the keyboard.

5.3 SCREEN FORMAT

Each screen has a screen number, and consists of 16 lines of 64 characters. However, so that Forth can use these efficiently, it is necessary to terminate the screen by special Forth words.
` — — > ` at the end of the final line of the screen commands SPECTRUM-Forth to 'LOAD' the next consecutive screen when the screen is loaded from tape using the 'LOAD' command.
This is used when programs occupy more than one Forth screen.
SPECTRUM-Forth compiles the screen in memory and then prints READY CASSETTE to LOAD on the next screen.
' **; S** ' at the end of the final line of the screen terminates the 'LOAD' command. This is used for the last screen of program.
Failure to place a terminator at the end of the screen could cause the system to crash.
N.B. Only one screen may be stored in memory at any time and will be overwritten when another screen is loaded.

6.0 OTHER USEFUL COMMANDS

As Forth has such a rich vocabulary, it is impossible to list them all here. The only way to get used to them is practice by reading the GLOSSARY and trying the commands.
` FORGET ` This word is used to forget a definition. It is used in the form FORGET word <ENTER>
N.B. This will forget the word 'word' and all the following words defined after it. For example, if we define the following words;

```
: WD1  .  " HELLO "  ;     <ENTER>
: WD2  .  " HI " ;         <ENTER>
: WD3  .  " BYE  "  ;      <ENTER>
```

then **FORGET WD2** <ENTER> OK will remove WD2 and WD3 from the Dictionary but not WD1.

` BYE '	This word is used to exit SPECTRUM-Forth and return to Basic.
' IMMEDIATE '	Normally when a word is encountered within a definition, it is compiled as part of the definition. If you require a word to be executed when it is encountered, even within a definition, then the word may be declared to be 'Immediate' by following its definition.
' TASK '	This word is a dummy definition which is conventionally used to start a program, so whenever you wish to forget a program, you know where to forget from.
' VLIST '	This word will list all the words in the Dictionary. The most recently defined word is listed first. The listing can be stopped at any time by use of the break key (SHIFT SPACE).

For example:
 : WD3 . " IT IS NOW COMPILING" ; IMMEDIATE <ENTER> OK
This then prints the message whenever WD3 is encountered and it is not compiled.
" ["and "]" and 'LITERAL'
Sometimes it is convenient to calculate a constant with a definition without having to calculate it each time the definition is executed. The Forth word '[' temporarily places the computer into interpret mode and anything typed will be executed immediately. The word '] ' places the computer back into compile mode, in a definition.
The Forth word 'LITERAL' places the top value on the stack into the current definition as a constant. Literal is 'immediate' and so executed when encountered.
For example: The following two definitions are equivalent:
: PT1 3 [1 3 + 2 *] LITERAL + . ; <ENTER>
: PT2 3 8 + . ; <ENTER>
This is used when the result of a calculation is not known and saves you from working it out.
' **VOCABULARY** ' Forth lets you create your own vocabularies so that all the words for one program may be kept together.
Vocabularies should be declared as immediate.
For example: to define a vocabulary called mine
VOCABULARY MINE IMMEDIATE <ENTER>
To place definitions within your vocabulary, you type vocabulary name definitions.
For example: **MINE DEFINITIONS** <ENTER>
All definitions preceding this will be placed in the vocabulary '**MINE**' until you again change vocabulary. From within a vocabulary, you may assess all the words within the vocabulary and all words in the vocabulary the current vocabulary was defined in. If you wish to use a word in a different vocabulary, then you precede the word with the vocabulary name.
The basic vocabulary is FORTH.
'**MEM**' This word prints out your remaining free memory. The number of bytes printed in the current base.

APPENDIX 'A' SPECTRUM-FORTH ERROR CODES

ERROR CODE	ERROR MESSAGE
0	COMMAND NOT IN DICTIONARY
1	STACK EMPTY
2	DICTIONARY FULL (OUT OF MEMORY)
3	INCORRECT ADDRESS MODE
4	WARNING: NAME NOT UNIQUE
7	STACK FULL (OUT OF MEMORY)
17	WORD MUST BE USED IN DEFINITION
18	EXECUTION ONLY
19	UNMATCHED CONDITIONAL
20	DEFINITION NOT FINISHED
21	IN PROTECTED DICTIONARY
22	USE ONLY WHEN LOADING
23	OFF CURRENT EDITING SCREEN
24	DECLARE VOCABULARY

APPENDIX 'B' - USEFUL ROUTINES

The INPUT routine listed here inputs a number from the keyboard when executed, terminated by an ENTER, leaving it on the stack.

```
: INPUT PAD 1+ 64 EXPECT . Ø PAD
(NUMBER) DROP DROP ;
```

INPUT, will input a double precision number.

```
: INPUT . PAD 1+ 64 EXPECT .Ø PAD
(NUMBER) DROP ;
```

SPECTRUM FORTH GLOSSARY

This glossary contains all of the word definitions in Release 1 of SPECTRUM-Forth. The definitions are presented in the order of their ASCII sort. The first line of each entry shows a symbolic description of the action of the procedures on the parameter stack. The symbols indicate the order in which input parameters have been placed on the stack. Three dashes " - - - " indicate the execution point; any parameters left on the stack are listed. In this notation, the top of the stack is to the right.

The symbols include:

addr	memory address
b	8 bit byte (i.e. hi 8 bits zero)
c	8 bit ASCII character (hi 8 bits zero)
d	32 bit signed double integer, most significant portion with sign on top of stack.
f	boolean flag. 0-false, non-zero-true.
ff	boolean false flag-0.
n	16 bit signed integer number.
u	16 bit unsigned integer.
tf	boolean true flag-non-zero.

The capital letters on the right show definition characteristics;

C	May only be used within a colon definition. A digit indicates number of memory addresses used, if other than one.
E	Intended for execution only.
LO	Level Zero definition of FORTH-78.
L1	Level One definition of FORTH-78.
P	Has precedence bit set. Will execute even when compiling.
U	A user variable.

Unless otherwise noted, all references to numbers are for 16 bit signed integers. For 32 bit signed double numbers, the most significant part (with the sign) is on top of the stack. All arithmetic is implicitly 16 bit signed integer math, with error and under-flow indication unspecified.

!	n addd- - -	LO
	Store 1 6 bits of n at address. Pronounced "store",	
!CSP	Save the stack position in CSP. Used as part of the compiler security.	
#	d1 - - - d2	LO
	Generate from a double number d1, the next ASCII character which is placed in an output string. Result d2 is the quotient after division by BASE, and is maintained for further processing. Used between <# and #> . See # S.	
#>	d - - - addr count	LO
	Terminates numeric output conversion by dropping d, leaving the text address and character count suitable for TYPE.	

#S	d1 - - - d2 Generates ASCII text in the text output buffer, by the use of # until a zero double number n2 results. Used between <# and # >	LO
’	- - - addr Used in the form: ’ nnnn Leaves the parameter field address of dictionary word nnnn. As a compiler directive, executes in a colon-definition to compile the address as a literal.	P.LO
(	Used in the form: (cccc) Ignore a comment that will be delimited by a right parenthesis on the same line. May occur during execution or in a colon-definition. A blank after the leading parenthesis is required.	P.LO C+
(.")	The run-time procedure, compiled by ." which transmits the following in-line text to the selected output device. See ."	
(; CODE)	The run-time procedure, compiled by , CODE, that rewrites the code field of the most recently defined word to point to the following machine code sequence. See , CODE.	
(+LOOP)	n - - - The run-time procedure compiled by +LOOP, which increments the loop index by n and tests for loop completion. See +LOOP.	C2
(ABORT)	Executes after an error when WARNING is -1. This word normally executes ABORT, but may be altered (with care) to a user’s alternative procedure.	
(DO)	The run-time procedure compiled by DO which moves the loop control parameters to the return stack. See DO.	
(FIND)	addr1 addr2 - - - pfa b tf (ok) addr1 addr2 - - - ff (bad) Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Return parameter field address, length byte of name field and boolean true for a good match. If no match is found, only a boolean false is left.	
(LINE) t	n1 n2 - - - addr count Convert the line number n1 and the screen n2 to the tape buffer address containing the. data. A count of 64 indicates he full line text length.	C2
(LOOP)	The run-time procedure compiled by LOOP which increments the loop index and tests for loop completion. See LOOP.	

(NUMBER)	d1 addr1 - - - d2 addr2 Convert the ascil text beginning at addr1+1 with regard to BASE. The new value is accumulated into double number d1, being left as d2. Addr2 is the address of the first unconvertible digit. Used by NUMBER.	
*	n1 n2 — — — prod Leave the signed product of two signed numbers.	LO
*/	n1 n2 n3 - - - n4 Leave the ratio n4 - n1*n2/n3 where all are signed numbers. Retention of an intermediate 31-bit product permits greater accuracy than would be available with the sequence: n1 n2 * n3 /	LO
*/MOD	n1 n2 n3 - - - n4 n5 Leave the quotient n5 and remainder n4 of the operation n1*n2/n3 A 31 bit intermediate product is used as for */.	LO
+	n1 n2 — — — sum Leave the sum of n1+n2	LO
+!	n addr — — — Add n to the value at the address. Pronounced "plus-store".	LO
+-	n1 n2 - - - n3 Apply the sign of n2 to n1, which is left as n3.	
+LOOP	n1 - - - (run) Used in a colon-definition in the form: DO . . . n1 +LOOP At run-time, LOOP selectively controls branching back to the corresponding DO based on n1, the loop index and the loop limit. The signed increment n1 is added to the index and the total compared to the limit. The branch back to DO occurs until the new index is equal to or greater than the limit (n1 > 0), or until the new index is equal to or less than the limit (n1 < 0). Upon exiting the loop, the parameters are discarded and execution continues ahead.	
+ORIGIN	n - - - addr Leave the memory address relative by n to the origin parameter area. n is the minimum address unit, either byte or word.	
	n - - - Store n into the next available dictionary memory cell, advancing the dictionary pointer, (comma)	LO
-	n1 n2 - - - diff Leave the difference of n1 — n2	LO
-->	Continue interpretation with the next screen. (Pronounced next-screen).	P,LO

- DUP	n1 - - n1 (if zero) n1 - - n1 n1 (non zero) Reproduce n1 only if it is non-zero. This is usually used to copy a value just before IF, to eliminate the need for an E LSE part to drop it.	LO
-FIND	- - - pfa b tf (found) - - - ft (not found) Accepts the next text word (delimited by blanks) in the input stream to HERE, and searches the CONTEXT and then CURRENT vocabularies for a matching entry. If found, the dictionary entry's parameter field address, its length byte, and a boolean true is left. Otherwise, only a boolean false is left.	
-TRAILING	addr n1 - - - addr n2 Adjusts the character count n1 of a text string beginning address to suppress the output of trailing blanks, i.e. the characters at addr+n1 to addr+n2 are blanks.	
.	n - - - Print a number from a signed 16 bit two's complement value, converted according to the numeric BASE. A trailing blank follows. Pronounced "dot".	LO
.”	Used in the form: ." cccc" Compiles an in-line string cccc (delimited by the trailing") with an execution procedure to transmit the text to the selected output device. If executed outside a definition, **.**" will immediately print the text until the final".	p,LO
.R	n1 n2 - - - Print the number n1 right aligned in a field whose width is n2. No following blank is printed.	
/	n1 n2 — — — quot Leave the signed quotient of n1/n2.	LO
/MOD	n1 n2 - - - rem quot Leave the remainder and signed quotient of n1/n2. The remainder has the sign of the dividend	LO
0 1 2 3	- - - n These small numbers are used so often that it is attractive to define them by name in the dictionary as constants.	
0<	n - - - f Leave a true flag if the number is less than zero (negative), otherwise leave a false flag.	LO
0=	n - - - f Leave a true flag if the number is equal to zero, otherwise leave a false flag.	LO

OBRANCH	f - - - The run-time procedures to conditionally branch. If f is false (zero), the following in-line parameter is added to the interpretive pointer to branch ahead or back Compiled by IF, UNTIL, and WHILE	C2
1+	n1 - - - n2 Increment n1 by 1.	L1
2+	n1 - - - n2 Leave n1 incremented by 2.	
:	Used in the form called a colon-definition: **: cccc ... ;** Creates a dictionary entry defining cccc as equivalent to the following sequence of Forth word definitions '**. . .** ' until the next ` ; ` or ` **;** CODE '. The compiling process is done by the text interpreter as long as STATE is non-zero. Other details are that the CONTEXT vocabulary is set to the CURRENT vocabulary and that words with the precedence bit set (P) are executed rather than being compiled.	P.E.LO P C LO
;	Terminate a colon-definition and stop further compilation. Compiles the run-time ; **S.**	
; CODE	Used with Forth assembler.	
; s	Stop interpretation of a screen ; S is also the run-time word compiled at the end of a colon-definition which returns execution to the calling procedure.	
<	n1 n2 — — — f Leave a true flag if n1 is less than n2; otherwise leave a false flag. Set up for pictured numeric output formatting using the words < # # # S SIGN # > The conversion is done on a double number producing ext at PAD.	LO
<BUILDS	Used within a colon-definition: : cccc < BUILDS . . . **DOES> . . . ;** Each time cccc is executed, <BUILDS defines a new word with a high-level execution procedure. Executing cccc in the form: cccc nnnn uses < BUILDS to create a dictionary entry for nnnn with a call to the DOES> part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in cccc. <**BUI LDS and DOES**> allow run-time procedures to be written in high-level rather than in assembler code	C, LO
-	n1 n2 - - - f Leave a true flag if n1=n2; otherwise leave a false flag	LO
>	n1 n2 - — - f Leave a true flag if n1 is greater than n2; otherwise a false flag.	LO

>R	n--- Remove a number from the computation stack and place as the most accessible on the return stack. Use should be balanced with R > in the same definition	c.LO
?	ADDR - - Print the value contained at the address in free format according to the current base.	LO
?COMP	Issue error message if not compiling.	
?CSP	Issue error message if stack position differs from value in CSP.	
?ERROR	f n - - - Issue an error message number n, if the boolean flag is true.	
?EXEC	Issue an error message if not executing.	
?LOADING	Issue an error message if not loading	
?PAIRS	n1 n2 - - -. Issue an error message if n1 does not equal n2. The message indicates that compiled conditionals do not match.	
?STACK	Issue an error message if The stack is out of bounds. This definition may be installation dependent	
@	addr - - - n Leave the 16 bit contents of address.	LO
ABORT	Clear the stacks and enter the execution state. Return control to the operators terminal, printing a message appropriate to the installation.	
ABS	n - - - u. Leave the absolute value of n as u.	LO
AGAIN	Used in a colon-definition in the form: BEGIN . . . AGAIN At run-time, AGAIN forces execution to return to corresponding BEGIN. There is no effect on the stack. Execution cannot leave this loop (unless R > DROP is executed one level below)	
ALLOT	n - - - Add the signed number to the dictionary pointer DP. May be used to reserve dictionary space or re-origin memory, n is with regard to computer address type (byte or word).	LO
AND	n1 n2 - - - n3 Leave the bit wise logical and of n1 and n2 as n3.	LO
AT	n1 n2- - - Position printer cursor on screen to Line n1 Column n2.	
B/SCR	- - - n This constant leaves the number of blocks per editing screen. By convention, an editing screen is 1024 bytes organised as 16 lines of 64 characters each.	

BACK	addr - - -	Calculate the backward branch offset from HERE to addr and compile into the next available dictionary memory address.	
BASE	- - - addr	A user variable containing the current number base used for input and output conversion.	U.LO
BEEP	n1 n2 - - -	Beep Duration n1 Pitch n2.	

BEGIN		Occurs in a colon-definition in form-
		BEGIN **. . .** UNTIL
		BEGIN **. . .** AGAIN
		BEGIN **. . .** WHILE **. . .** REPEAT
		At run-time, BEGIN marks the start of a sequence that may be repetitively executed. It serves as a return point from the corresponding UNTIL. AGAIN or REPEAT. When executing UNTIL, a return to BEGIN will occur if the top of the stack is false; for AGAIN and REPEAT a return to BEGIN always occurs.	

BL	- - - c	A constant that leaves the ASCII value for "blank".	
BLANKS	addr count - - -	Fill an area of memory beginning at addr with blanks.	
BLK	- - -addr	A user variable containing the block number being interpreted. If zero, input is being taken from the terminal input buffer.	U,LO
BLOCK	n - - - addr	Leave the memory address of the block buffer containing block n. If the block is not already in memory, it is transferred from tape.	LO
BORDER	n - - -	Set border colour to n.	
BRANCH		The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer If to branch ahead or back. BRANCH is compiled by ELSE, AGAIN, REPEAT.	C2,LO
BRIGHT	n - - -	Brightness of Characters 0 - normal, 1 - bright.	
BYE		Return to basic.	
C!	b addr - - -	Store 8 bits at address. On word addressing computers, further specification is necessary regarding byte addressing.	

C,	b - - -	Store 8 bits of b into the next available dictionary byte, advancing the dictionary pointer	
C@	addr - - - b	Leave the 8 bit contents of memory address	
CFA	pfa - - - cfa	Convert the parameter field address of a definition to its code field address.	
CIRCLE	n1 n2 n3 - - -	Draw a circle of radius n1 at co-ordinates (n2, n3).	
CLS		Clears the screen.	
CMOVE	from to count - - -	Move the specified quantity of bytes beginning at address from to address to. The contents of address from is moved first proceeding toward high memory.	
COLD		The cold start procedure to adjust the dictionary pointer to. the minimum standard and restart via ABORT. May be called from the terminal to remove application programs and restart.	C2
COMPILE		When the word containing COMPILE executes, the execution address of the word following COMPILE is copied (compiled) into the dictionary. This allows specific compilation situations to be handled in addition to simply compiling an execution address (which the interpreter already does).	
CONSTANT	n - - -	A defining word used in the form: n CONSTANT cccc to create word cccc, with its parameter field containing n. When cccc is later executed, it will push the value of n to the stack.	LO
CONTEXT	- - - addr	A user variable containing a pointer to the vocabulary within which dictionary searches will first begin.	U, LO
COPY		Send a copy of screen to the printer.	
COUNT	addr1 - - - addr2 and byte.	Leave the byte address addr2 and byte count n of a message text beginning at address addr1. It is pre-summed that the first byte at addr1 contains the text byte count and the actual text starts with the second byte. Typically COUNT Is followed by TYPE.	LO
CR		Transmit a carriage return and line feed to the selected output device.	LO
CREATE		A defining word used in the form: CREATE cccc by Such words as CODE and CONSTANT to create a dictionary header for a Forth definition. The code field contains the address of the words parameter field. The new word is created in the CURRENT vocabulary.	

CSP	- - - addr A user variable temporarily storing the stack pointer position, for compilation error checking	U
D+	d1 d2 - - -dsum Leave the double number sum of two double numbers	
D+ -	d1 n - - - d2 Apply the sign of n to the double number d1, leaving It as d2.	
D.	d - - - Print a signed double number from a 32 bit two's complement value. The high-order 16 bits are most accessible on the stack. Conversion is performed according to the current BASE. A blank follows Pronounced D - dot.	L1
D.R	d n - - - Print a signed double number d right aligned in a field n characters wide.	
DABS	d - - - ud Leaves the absolute value ud of a double number.	
DECIMAL	Set the numeric conversion BASE for decimal input-output.	LO
DEF	b1 b2 b3 . . . b8 n1 - - - Define a user definable graphic.	
DEFINITIONS	Used in the form: cccc DEFINITIONS Set the Current vocabulary to the CONTEXT vocabulary. In the example executing vocabulary name cccc made it the CONTEXT vocabulary and executing DEFINITIONS made both specify vocabulary cccc	L1
DIGIT	cn1 - - - n2 tf (ok) c n1 - - - ff (bad) Converts the ASCII character c (using base n1) to its binary, equivalent n2, accompanied by a true flag If the conversion is invalid, leaves only a false flag.	
DLITERAL	d - - - d (executing) d - - - (compiling) If compiling, compile a stack double number into a literal. Later execution of the definition containing the literal will push it to the stack. If executing, the number will remain on the stack	P
DMINUS	d1 - - - d2 Convert d1 to its double number two's compliment.	

DO	n1 n2 - - - (execute) Occurs in a colon-definition in form- DO . . . LOOP DO . . . +LOOP At run-time, DO begins a sequence with repetitive execution controlled by a loop limit n1 and an index with initial value n2. DO removes these from the stack Upon reaching LOOP the index is incremented by one' Until the new index equals or exceeds the limit, execution Loops back to just after DO; otherwise the loop parameters are discarded and execution continues ahead Both n1 and n2 are determined at run-time and may be the result of other operations. Within a loop 'I' will copy the current value of the index to the stack See l, LOOP, +LOOP, LEAVE	P, C2, LO
DOES>	A word which defines the run-time action within a high level defining word. DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES > Used in combination with BUI LDS. When the DOES> part executes it begins with the address of the first parameter of the new word on the stack. This allows interpretation using this area or its contents. Typical uses include the Forth assembler, mulit-dimensial arrays, and compiler generation.	LO
DP	- - - addr A user variable, the dictionary pointer, which contains the address of the next free memory above the dictionary The value may be read by HERE and altered by ALLOT	U,L
DPL	- - - addr A user variable containing the number of digits to the right of the decimal on double integer input. It may also be used to hold output column location of a decimal point in user generated formatting. The default value on single number input is –1.	U,LO
DRAW	n1 n2 - - - Draw a line from current plot position n1 in x-direction n2 in Y-direction.	
DROP	n - - - Drop the number from the stack	LO
DUP	n - - - n n Duplicate the value on the stack.	LO
ELSE	Occurs within a colon-definition in the form- IF . . . ELSE . . . ENDIF At run-time, ELSE executes after the true following IP. ELSE forces execution to skip over the following false part and resumes execution after the ENDIF. It has no stack effect.	

EMIT c--- LO
Transmit ASCII character c to the selected output for each character output.

EMPTY-BUFFERS
Mark all block-buffers as empty, not necessarily affecting the contents. This is also an initialization procedure before first use of the tape.

ENCLOSE addr c - - -
addr1 n1 n2 n3
The text scanning primitive used by WORD. From the text address addr1 and an ASCII delimiting character c, is determined the byte offset to the first non-delimiter character n1, the offset to the first delimiter after the text n2, and the offset to the first character not included. This procedure will not process past an ASCII 'null', Treating it as an unconditional delimiter.

END P,C2,LO
This is an 'alias' or duplicate definition for UNTIL.

ENDIF
Occurs in a colon-definition form:
IF . . . ENDIF IF . . . ELSE . . . ENDIF
At run-time, ENDIF serves only as the destination of a forward branch from IF or ELSE. It marks the conclusion of the conditional structure. THEN is another name for ENDIF. Both names are supported in ZX-FORTH.
See also IF and ELSE.

ERASE addr n - - -
Clear a region of memory to zero from addr over n addresses.

ERROR line - - - in b1k
Execute error notification and re-start of system. WARNING is first examined. If 1, the text of line n, relative to screen 4 of drive 0 is printed. This line number may be positive or negative, and beyond Just screen 4. If WARNING-0, n is just printed as a message number (non disc installation). If WARNING is -1, the definition (ABORT) is executed which executes the system ABORT. The user may cautiously modify this execution by altering (ABORT). ZX-FORTH saves the contents of IN and B LK to assist in determining the location of the error. Final action is execution of QUIT.

EXECUTE addr- -
Execute the definition whose code field address is on the stack. The code field address is also called the compilation address.

EXPECT addr count - - - LO
Transfer characters from the terminal to address, until a "return" or the count of characters have been received. One or more nulls are added at the end of the text.

FENCE	- - - addr A user variable containing an address below which FORGETTING is trapped. To forget below this point the user must alter the contents of FENCE.	U
FILL	addr quan b - - - Fill memory at the address with the specified quantity of byte b.	
FIRST	- - - n A constant that leaves the address of the block buffer	
FLASH	n1 - - - Defines whether character is flashing or steady. 0 -Steady 1 - Flashing	
FORGET	Executed in the form: FORGET cccc Delete definition named cccc from the dictionary with all entries physically following it. I n SPECTRUM-Forth an error message will occur if the CURRENT and CONTEXT vocabularies are not currently the same	E, LO
FORTH	The name of the primary vocabulary. Execution makes FORTH the CONTEXT vocabulary. Until additional user vocabularies are defined, new user definitions become a part of FORTH. Forth is immediate, so it will execute during the creation of a colon-definition, to select this vocabulary at compile time.	P, LI
GOVER	n - - - Controls overprinting. n = 0 -Character obliterated by other characters. n = 1 -New characters XOR with previous characters.	
HERE	- - - addr Leave the address of the next available dictionary location.	LO
HEX	Set the numeric conversion base to sixteen (hexa-decimal).	LO
HLD	- - - addr A user variable that holds the addresses of the latest character of text during numeric output conversion	LO
HOLD	c - - - Used between < #and-# > to insert an ASCII character into a pictured numeric output string, e.g. 2E HOLD will place a decimal point.	LO
HOME	Move cursor to top left of screen.	
I	- - - n Used with a DO-LOOP to copy the loop index to the stack. Other use is implementation dependent. See R.	C, LO

ID	addr - - - print a definition's name from its name field address
	f - - - (run-time) P ,C2, LO Occurs in a colon-definition in form- IF (tp) . . . ENDIF IF (tp) . . . ELSE(fp) . . . ENDIF At run-time. IF selects execution based on a boolean flag. If f is true (non-zero), execution continues ahead through the true part. If f is false (zero) execution skips till just after ELSE to execute the false part. After either part, execution resumes after ENDIF. ELSE and its false part are optional **.;** if Missing, false execution skips to just after ENDIF.
IMMEDIATE	Mark the most recently made definition so that when encountered at compile time, it will be executed rather than being compiled, i.e. the precedence bit in its header is set. This method allows definitions to handle unusual compiling situations, rather than build them into the fundamental compiler. The user may force compilation of an immediate definition by preceding it with (COMPILE)
IN	- - - addr LO A user variable containing the byte offset within the current input text buffer (terminal or disc) from which the next text will be accepted. WORD uses and moves the value of IN.
INK	n1 - - - Set ink (foreground) colour.
INTERPRET	The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or disc) depending on STATE. If the word name cannot be after a search of CONTEXT and then CURRENT it is converted to a number according to the current base. That also failing an error message echoing the name with a `?` will be given. Text input will be taken according to the convention for WORD If a decimal point is found as part of a number a double number value will be left. The decimal point has no other purpose than to force this action. See NUMBER
INV	n - - - Controls inversion of characters. n = Ø - normal n = 1 - inverse video.
KEY	- - - c LO Leave the ASCII of the text terminal key struck.
LATEST	- - - addr Leave the name field address of the topmost word in the CURRENT vocabulary, C, LO

LEAVE	Force termination of a DO-LOOP at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged and execution proceeds normally until LOOP or +LOOP is encountered	
LFA	pfa - - - _1fa Convert the parameter field address of a dictionary definition to its field address.	
LIMIT	- - - n A constant leaving the address just above the highest memory available for a tape buffer. Usually this is the highest system memory.	
LIST	n - - - Display the ASCII text of screen n on the selected out-put device. SCR contains the screen number during and after this process.	LO
LIT	- - - n Within a colon-definition, LIT is automatically com-piled before each 16 bit literal number encountered in input text. Later execution of LIT causes the contents of the next dictionary address to be pushed to the	C2, LO
LITERAL	n - - - (compiling) It compiling, then compile the stack value n as a 16 bit literal. This definition is immediate so that it will execute during a colon definition. The intended use is: : xxx (calculate) LITERAL ; Compilation is suspended for the compile time calculation of a value. compilation is resumed and LITERAL compile this value	P, C2, LO
LOAD	n - - - Begin interpretation of screen n Loading will terminate at ;S. See ; Sand -->.	LO
LOOP	Occurs in a colon-definition in form: DO LOOP At run-time, LOOP selectively controls 'branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues ahead	

M *	n1 n2 - - - d	A mixed magnitude math operation which leaves the double number signed product of two signed numbers.	
M/	d n1 - - - n2 n3	A mixed magnitude math operator which leaves the signed remainder n2 and signed quotient n3, from a double number dividend and devisor nl. The remainder takes its sign from the dividend.	
M/MOD	ud1 u2 - - - u3 ud4	An unsigned mixed magnitude math operation which leaves a double quotient ud4 and remainder u3, from a double dividend ud1 and single divisor u2.	
MAX	n1 n2 - - - max	Leave the greater of two numbers.	LO
MESSAGE	n - - -	Print ? MSG # n	
MIN	n1 n2 - - - min	Leave the smaller of two numbers.	LO
MINUS	n1 - - - n2	Leave the two's complement of a number.	LO
MOD	n1 - - - n2 mod	Leave the remainder of n1/n2, with the same sign as n1.	LO.
NEXT		Used with FORTH assembler.	
NFA	pfa - - - nfa	Convert the parameter field address of a definition to its name field.	
NUMBER	addr - - - d	Convert a character string left at addr with a preceding count, to a signed double number, using the current numeric base. If a decimal point is encountered in the text, the position will be given in DPL, but no other effect occurs. If numeric conversion is net possible, an error message will be given.	
OR	n1 n2 - - or	Leave the bit-wise logical or of two 16 bit values.	LO
OUT	- - - addr	A user variable that contains a value incremented by EMIT. The user may alter and examine OUT to control display formatting.	U
OVER	n1 n2 - - - n1 n2 n1	Copy the second stack value, placing it as the new top.	LO
PAD	- - - addr	Leave the address of the text output buffer, which is a fixed offset above HERE.	LO
PAPER	n - - -	Control paper (background) colour.	
PERM	- - -	Makes all temporary colours permanent.	

PLOT	n1 n2 - - - Prints an ink spot at (n1 n2) and moves the PLOT position.	
PFA	nfa - - - pfa Convert the name field address of a compiled definition to its parameter field address.	
QUERY	Input 80 characters of text (or until a "return") from the operators terminal. Text is positioned at the address contained in TIB with IN set to zero.	
QUIT	Clear the return stack, stop compilation, and return control to the operators terminal. No message is given	
R	- - - n Copy the top of the return stack to the computation stack.	
R #	- - - addr A user variable which may contain the location of an editing cursor, or other file related function.	U
R/W	addr bl k f - - - The fig-FORTH standard disc read-write linkage, addr specifies the source or destination block buffer, b1 k is the sequential number of the referenced block; and f is a flag for f-0 write and f-1 read. R/W determines the location on mass storage, performs the read-write and performs any error checking.	
R >	- - - n Remove the top value from the return stack and leave it on the computation stack. See >R and R.	LO
RO	- - - addr A user -variable containing the initial location of the return stack. Pronounce R-zero. See RP .'	U
REPEAT	Used within a colon-definition in the form: BEGIN ... WHILE ... REPEAT At run-time, REPEAT forces an unconditional branch back to just after the corresponding BEGIN.	P,C2
RSMUDGE	Reset smudge bit of the most recent entry in the dictionary.	
ROT	n1 n2 n3 - - - n2 n3 n1 Rotate the top three values on the stack, bringing the third to the top.	LO
RP !	A computer dependent procedure to initialize the return stack pointer from user variable RO	
S-> D	n---d Sign extend a single number to form a double number	
SO	- - - addr A user variable that contains the initial value for the stack pointer. Pronounced S-zero. See SP !	U

SCR	---addr	A user variable containing the screen number most recently referenced by LIST	U
SIGN	nd - - - d	Stores an ASCII "-" sign just before a converted numeric output string in the text output buffer when n is negative, n is discarded, but double number d is maintained. Must be used between <# and #>	LO
SMUDGE		Used during word definition to toggle the "smudge bit" in a definitions name field. This prevents an uncompleted definition from being found during dictionary searches, until compiling is compiled without error.	
sp !		A computer dependent procedure to initialize the stack pointer from SO.	
SP@	- - - addr	A computer dependent procedure to return the address of the stack position to the top of the stack as it was before SP@ was executed. (e.g. 1 2 SP@ . . . would type 2 2 1)	
SPACE		Transmit an ASCII blank to the output device.	LO
SPACES	n - - -	Transmit n ASCII blanks to the output device	LO
STATE	- - - addr	A user variable containing the compilation state. A non-2ero value indicates compilation. The value itself may be implementation dependent	LO, U
SWAP	n1 n2 - - - n2 n1	Exchange the top two values on the stack.	LO
TASK		A no-operation word which can mark the boundary between applications. By forgetting TASK and re-compiling, an application can be discarded in its entirety.	
THEN		An alias for ENDIF.	P, CO, LO
TIB	- - - addr	A user variable containing the address of the terminal input buffer.	U
TOGGLE	addr b - - -	Complement the contents of addr by the bit pattern b	
TRAVERSE	addr1 n - - - addr2	Move across the name field of a ZX-FORTH variable length name field, addr1 is the address of either the length byte or the last letter. If n-1, the motion is toward hi memory; if n--1, the motion is toward low memory. The addr2 resulting is address of the other end of the name.	

TYPE	addr count - - -	LO	Transmit count characters from addr to the selected output device.
U*	u1 u2 - - - ud		Leave the unsigned double number product of two unsigned numbers.
U/	ud u1 - - - u2 u3		Leave the unsigned remainder u2 and unsigned quotient u3 from the unsigned ed double dividend ud and unsigned divisor u1
UNTIL	f - - - (run-time)		occurs within a colon-definition in the form: BEGIN .. . UNTIL At run-time, UNTIL controls the conditional I branch back to the corresponding BEGIN. If f is false, execution returns to just after BEGIN; if true execution continues ahead
USER	n - - -	LO	A defining word used in the form: n USER cccc which creates a user variable cccc. The parameter field of cccc contains n as a fixed offset relative to the user pointer register UP for this user variable. When cccc is later executed, it places the sum of its offset and the user area base address on the stack as the storage address of that particular variable
VARIABLE		E, Lu	A defining word used in the form: n VARIABLE cccc When VARIABLE is executed, it creates the definitions cccc with its parameter field initialized to n. When cccc is later executed, the address of its parameter field (containing n) is left on the stack, so that a fetch or store may access this location
VOC-LINK	- - - addr	U	A user variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked by these fields to allow control for FORGETTING thru multiple vocabularies.
VOCABULARY		E,Lu	A defining word used in the form: n VOCABULARY cccc To create a vocabulary definition cccc. Subsequent use of cccc will make it the CONTEXT vocabulary which is searched first by INTERPRET. The sequence "cccc DEFINITIONS" will also make cccc the CURRENT vocabulary into which new definitions are placed. In SPECTRUM-Forth cccc will be so chained as to include all definitions of the vocabulary in which cccc is itself defined. All

	vocabularies ultimately chain to Forth. By convention, vocabulary names are to be declared IMMEDIATE. See VOC-LINK.	
VLIST	List the names of the definitions in the context vocabulary. "Break" will terminate the listing.	
WARNING	Must be 0 for no disc installation.	
WHILE	f - - - (run-time) Occurs in a colon-definition in the form: BEGIN . . . WHILE (tp) . . . REPEAT At run-time, WHILE selects conditional execution based on boolean flag f. If f is true (non-zero), WHILE continues execution of the true part thru to REPEAT, which then branches back to BEGIN. If f is false (zero), execution skips to just after REPEAT, exiting the structure.	P,C2
WIDTH	Maximum length of word name, 31 in ZX-Forth.	
WORD	c - - - Readthe next text characters from the input stream being interpreted, until a delimiter c is found, storing the packed character string beginning at the dictionary buffer HERE. WORD leaves the character count in the first two or more blanks. Leading occurrences of c are ignored. If BLK is zero, text is taken from the terminal input buffer, otherwise from the block stored in BLK. See BLK, IN.	
XOR	n1 n2 - - -- xor Leave the bitwise logical Exclusive Or of two values.	L1
[	Used in a colon-definition in form: : xxx [words] more ; Suspend complication. The words after (are executed, not compiled. This allows calculation or compilation exceptions before resuming compilation with I . See LITERAL,).	P.L1
[COMPILE]	Used in a colon-definition in form: ; xxx [COMPILE] FORTH ; [COMPILE] will force the compilation of an	P,C