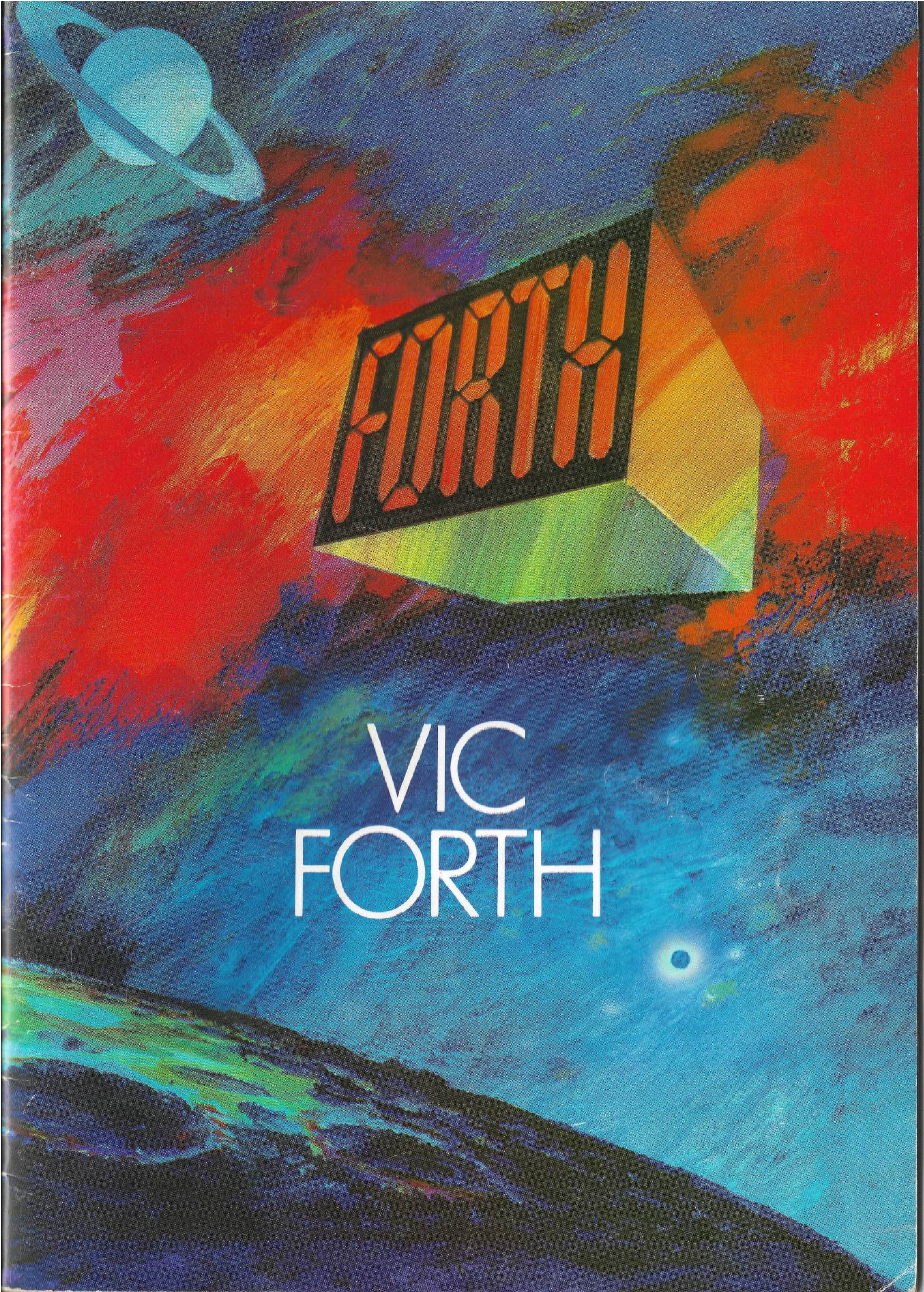
VIC FORTH



VIC-FORTH
User's Guide & Reference Manual

PREFACE

VIC-FORTH is a fourth-generation programming language that in many respects is very different from other languages. In relation to its size, its power is unequalled. In just 8K, a typical minicomputer FORTH version supplies structured, compact and extremely fast code, virtual memory, a resident macro assembler and a resident editor, as well as multitasking capability. FORTH is even able to extend itself and its compiler. It is a meta-language.

This manual does not pretend to be a textbook on the art of FORTH programming. Instead, we have assembled a literature list for your convenience.

The first section of this manual will help you to get started. VIC-FORTH's system peculiarities are described in detail. Of course VIC-FORTH is very standard, and conforms to fig-FORTH except for some minor improvements.

The second part is a glossary of all words in VIC-FORTH. Every command is described, parameters and results are shown. This section comprises the main part of this manual, since there are several hundred commands in VIC-FORTH.

PART 1 – GENERAL INFORMATION

VIC-FORTH is a highly capable language that operates from ROM and is based on fig-FORTH. It is nearly identical to PET-FORTH, a professional FORTH version which runs on the Commodore CBM business computer series. VIC-FORTH contains everything in PET-FORTH's Kernel, plus its standard System Extensions. This gives you 100% compatibility.

It is normal for a FORTH system to contain an Editor and an Assembler. These two vocabularies have been left out of your ROM module, and are instead supplied in listing form. This has been done to achieve maximum flexibility in memory usage. You may enter them from the keyboard, if desired, or decide to save memory.

FORTH, during program development, is a disk based language; it is not normally designed to work with cassette. VIC-FORTH has extensions, however, to make it possible to save and reload workspaces from cassette. If you plan to do much programming, you are strongly advised to purchase a diskette unit. You will then enjoy the full power of VIC-FORTH.

For disk users, your dealer can provide a System Extension Diskette. It contains the Assembler and Editor vocabularies, Floating Point routines, Complex Numbers, Advanced String Handling, a very complete Graphics Package, and many other useful utilities. Some demonstration programs and games are supplied as well.

VIC-FORTH will work with any quantity of memory. 3K of expansion RAM is included in the plug-in module, so the 3K RAM module should not be plugged in.

On an unexpanded VIC, you will have approximately 3400 bytes free for your programs and data. Since FORTH code is very compact, this is sufficient for many applications, and 16K and more is almost luxurious!

VIC-FORTH uses eleven disk buffers of 256 bytes each, making the total buffer space 2860 bytes. Video memory is located at $1000.

VIC-FORTH differs in some minor points from standard fig-FORTH:

* In order to support the various memory sizes, User Variables och Disk Buffers are not located in high memory, but rather in the low end of it. Thus, they *precede* the dictionary. The normal user will never be affected.

* Of the same reason, VIC-FORTH will not tell you when the dictionary is full. You must keep track of this yourself. It is easily checked by subtracting *HERE* from the highest RAM address in your system. You will then get a value representing the number of bytes left.

* There are a number of extensions resident in ROM. These include DUMP, 1+!, 1—!, 1—, 2—, PICK, LINE, TEXT, and others. In addition, IEEE/file handling words and cassette save and load words are provided.

The below three points are very technical, and may or may not make sense to you:

* DOES> words are more compact; they each require two bytes less space than in fig-FORTH. DOES> now works according to FORTH-79 specifications.

* USE and PREV now are constants pointing to RAM cells. The user will not notice any difference.

* The FORTH vocabulary header is moved to RAM on cold start. This is necessary because the system is ROM-based. The user will not notice any difference.

In summary, these deviations are very minor, and source compatibility is in no way affected. We have supplied this list for those who have very through knowledge of FORTH, to document the implementation.

In cassette based VIC-FORTH systems, error messages will be given in the form "ERROR # 99". When a disk unit is used, you will automatically **get** textual error messages. Below is a list of error numbers and their textual counterparts. Note that the file handling words also use these codes as error codes.

1 —	Empty stack
2 —	Dictionary full (not used)
3 —	Has incorrect addressing mode
4 —	Isn't unique
5 —	
6 —	Disc range?
7 —	Full stack
8 —	Disc error!
9 —	Not 8-bit value or address
10 —	
11 —	
12 —	Illegal value — reenter
13 —	Too large — reenter
14 —	
15 —	Datatronic AB (c) 1982 etc.
16 —	
17 —	Compilation only, use in definition
18 —	Execution only
19 —	Conditionals not paired
20 —	Definition not finished
21 —	In protected dictionary
22 —	Use only when loading
23 —	Off current editing screen
24 —	Transfer aborted
25 —	Too many files
26 —	File already open
27 —	File not open
28 —	File not found
29 —	Device not present
30 —	Not input file
31 —	Not output file
32 —	Missing file name
33 —	Illegal device number

There is room on the disk for user expansion of error messages up to 63. If you're not using the disk, you may use any error number of your choice.

PART 2 - VIC FORTH GLOSSARY

This glossary contains all word residing in VIC-FORTH's main vocabulary. In addition, the glossary contains the words of the VIC-FORTH EDITOR vocabulary, which may be supplied in ROM or in listing form, depending on your particular version of VIC-FORTH. All words are presented in the order of their ASCII sort.

The first line of each entry shows a symbolic description of the action of the procedure on the parameter 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	ASCII character
d	32-bit signed double integer
f	boolean true/false flag
ff	boolean false flag
n	16-bit signed integer
u	16-bit unsigned integer
sf	boolean true flag

Unless otherwise noted, all references to numbers are for 16 bit signed integers. For 32-bit double integers, the most significant cell is on top.

All arithmetic is implicitly 16-bit signed integer math, with error and overflow indication unspecified.

! n addr ---

Stores 16 bits of n at address. "store".

!CSP

Save the stack position in CSP. Used as a part of the compiler security.

d1 --- d2

Generate from the double 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 < # och # >.
See #S. "sharp".

# > d --- addr count

Terminates numeric conversion by dropping d, leaving the text address and character count suitable for TYPE . "sharp-greater".

#S d1 --- d2

Generates ASCII text in the output buffer by repeated calls to # , until a double zero results. Used between <# och #>. "sharp-s".

' --- addr

Used in the form: ' nnnn
Leaves the parameter field addresses of the word nnnn. As a compiler directive, executes in a colon definition to compile the address as a literal. "tick".

(

Used in the form: (kkkk)
Ignore a comment that will be delimited by a right parenthesis on the same line. It is defined so that it may be used within a colon definition. Notice that a blank must follow the left parenthesis. "parenthesis".

(.")

The run-time procedure, compiled by .", that 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. See ;CODE .

(+LOOP) n ---

The run-time procedure, compiled by +LOOP , that increments the loop index by n and tests for loop completion. See +LOOP .

(ABORT)

Executes after an error, when WARNING is —1. The word normally just executes ABORT , but this may be altered (with care) to a user's alternative error procedure. In ROM-based systems such as VIC-FORTH, this is not possible.

(DERROR)

Executes after a disk error. Normally just executes DERROR , but may be altered to point to a user's disk error handling procedure. In ROM-based systems such as VIC-FORTH, this is not possible.

(DO)

The run-time procedure, compiled by DO , that moves the loop parameters to the return stack. See DO.

(DOES>) --- pfa (execution)

Starts the interpretation of a defining word's DOES> part by pushing IP onto the return stack, replacing IP with the indirect contents of W plus three, pushing the value of W plus two onto the stack (the member's PFA), and branching to NEXT .

(FIND) addr1 addr2 --- pfa b f (found)
addr1 addr2 --- ff (not found)

Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Returns parameter field address, length byte of the name field and a boolean true for a good match. If no match is found, a boolean false is left.

(LINE) n1 n2 --- addr count

Converts the line number n1 and the screen n2 to a disk buffer address containing the data. A count of 64 indicates the full length of text.

(LOOP)

The run-time procedure, compiled by LOOP , that increments the loop index by one and tests for loop completion. See LOOP .

(NUMBER) d1 addr1 --- d2 addr2

Converts the ASCII text at addr1+1 with regard to BASE. The new value is accumulated to the double value d2. Addr2 is the address of the first unconvertable digit. Used by NUMBER .

* n1 n2 --- n2

Leaves the product n3 of n1 and n2. "times".

*/ n1 n2 n3 --- n4

Leaves the ratio n4=(n1*n2)/n3. By using a 32-bit intermediate value, greater accuracy is achieved than would have been possible with the phrase n1 n2 * nr / . "times-divide".

*/MOD n1 n2 n3 --- n4 n5

As */ , but also leaves the remainder n4. "times-divide-mod".

+ n1 n2 --- n3

Leaves the sum n3 of n1 and n2.

+! n addr ---

Increments the cell at the address by n. "plus-store".

+- n1 n2 --- n3

Apply the sign of n2 to n1, which is left as n3. "plus-minus".

BUF addr1 --- addr2 f

Advance the disk buffer address addr1 to the address of the next disk buffer. Boolean f is false if addr2 points to the same buffer as PREV . "plus-buff".

+LOOP n1 --- (execution)
addr n2 --- (compilation)

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 totoal compared to the limit. The branch back to DO occurs 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. "plus-loop".

+ORIGIN n --- addr

Leaves the address of the n:th byte relative to the start of the kernel. This definition is used to access or modify the boot-up parameters at the origin area. "plus-origin".

, n ---

Store n into the next available dictionary cell, incrementing DP by two. "comma".

— n1 n2 --- n3

Leaves the difference n3 of n1 and n2. "minus".

--> Continue interpretation with the next disk screen. "next-screen".

-BCD n1 --- n2

Convert the binary value n1 to a packed BCD value n2. n1 must have a value from 0 to 99. "dash-b-c-d".

-DISC addr s t d f1 --- f2

The disk interface word used to transfer a disk block from or to a memory area. Track t, sector s, on drive d is either read or written depending on f1, which is 0 for a write and 1 for a read. Boolean f2 is true if a disk error has occurred. "dash-disk".

-DUP 0 --- 0
n1 --- n1 n1 (non-zero)

Duplicates the top stack value only if it is non-zero. Mainly used before a IF , to eliminate the need for an ELSE part to DROP it. "dash-dup".

FIND --- pfa b sf (found)
--- ff (not found)

Accepts the next word from the input stream to HERE , and searches the CONTEXT and then CURRENT volcabularies for a matching entry. If found, the parameter field address, length byte, and a boolean true is left. Otherwise, only a boolean false is left. "dash-find".

-MOVE addr n --- (EDITOR)

Moves the text at addr to line n in the current editing screen. The number of characters moved is given by the constant C/L . "dash-move".

-TRAILING addr n1 --- addr n2

Adjusts the character count of the string at addr to suppress trailing blanks. "dash-trailing".

. n ---

Prints the number n on the selected device, converted according bo BASE . The value is followed by one blank. "dot".

."	Used in the form: ." ttttttt" Compiles an in-line string ttttttt with a run-time procedure to transmit the text to the selected output device. If used outside a colon definition, ." will immediately print the characters up to the final ". The maximum text length is 255 characters. See (.") . "dot-quote".
.LINE	line screen --- Transmits the specified line of the indicated screen to the selected output device. "dot-line".
.R	n1 n2 --- Print the number n1 right-aligned in a field whose width is n2. No following blank is printed. "dot-r".
/	n1 n2 --- n3 Divides n1 by n2, leaving the quotient n3. "divide".
/MOD	n1 n2 --- rem quot Divides n1 by n2, leaving both remainder and quotient. The remainder has the sign of the quotient. "divide-mod".
0 1 2 3	--- n These four values are defined as constants, giving their own value when invoked. This saves two bytes each time they are used in a colon definition.
0 <	n --- f Leaves boolean true if n is less than zero. "zero-less".
0=	n --- f Leaves boolean true if n is zero. May also be used as a NOT function. "zero-equal".
0BRANCH	f --- The run-time procedure to branch conditionally. If f is false (zero) the following in-line parameter is added to the interpretive pointer IP to branch ahead or back. Compiled by IF , UNTIL , and WHILE . "zero-branch".
1+	n1 --- n2 Increments n1 by one. "one-plus".
1+!	addr --- Increments the cell at addr by one. "one-plus-store".
1-	n1 --- n2 Decrements n1 by one. "one-minus".
1-!	addr --- Decrements the cell at addr by one. "one-minus-store".
2!	d addr --- Stores the 32-bit value d in four bytes at addr. "two-store".

2@	addr --- d Fetches the 32-bit value from 4 bytes at addr. "two-fetch".
2+	n1 ---n2 Increments n1 by two. "two-plus".
2-	n1 --- n2 Decrements n1 by two. "two-minus".
2DROP , 2DUP , 2OVER , 2ROT , 2 SWAP , 2VARIABLE, 2 CONSTANT	Double-precision counterparts for the usual 16 bit words. "two-drop", "two-dup", "two-over", "two-rote", "two-swap", "two-variable", and "two-constant".
:	Used in the form called a colon definition: : cccc . . . ; Creates an entry in the dictionary defining cccc as equivalent to the sequence of words represented by '. . .' until the next ; or ;CODE . The compiling process is done by the next interpreter as long as STATE is non-zero. Other details are that the CONTEXT vocabulary is set to the CURRENT vocabulary, which means that the vocabulary into which the definition is linked is selected. Words that have the precedence bit set (IMMEDIATE words) are executed rather than being compiled. "colon".
;CODE	Used in the form: : cccc . . . ;CODE mnemonics END-CODE Stop compilation of and terminate a new defining word cccc by compiling (;CODE) . Set the CONTEXT to ASSEMBLER , assembling to machine code the following nmemonics, which specify the run-time behaviour for words defined by cccc. When cccc later executes in the form: cccc nnnn then word nnnn will be created with its execution procedure given by the machine code between ;CODE and END-CODE . The code field of nnnn points to the code after ;CODE . That is, when nnnn later is executed, it does so by jumping to the code after nnnn. An existing defining word must exist in cccc prior to ;CODE . "semicolon-code".
;S	Stop interpretation of a screen. ;S is also the run-time word compiled by ; at the end of a colon definition which returns control to the calling procedure. "semi-s".
<	n1 n2 --- f Leaves true if n1 is smaller than n2. "less-than".

< #	Start conversion of a double-precision number, leaving the result below PAD . "less-sharp".
<BUILDS	Used within a colon definition: : cccc <BUILDS ... DOES> ... ; Each time cccc is executed, <BUILDS creates a new dictionary entry, with a run-time behaviour in high-level. Executing cccc in the form: cccc nnnn uses <BUILDS to create nnnn, with a call to the DOES> part in cccc. When nnnn is later executed, the address of its parameter field is pushed on the stack, and the words after DOES> are executed. <BUILDS and DOES> allow run-time procedures to be written in level rather than in assembler code (as required by ;CODE. "builds".
=	n1 n2 --- f Leaves boolean true if n1 equals n2. "equals".
>	n1 n2 --- g Leaves boolean true if n1 is greater than n2. "greater-than".
>R	n --- Remove a number from the computation stack and place as the most accissible. Use should be balanced with a R> in the same definition. "to-r".
?	addr --- Prints the contents of the address in free format according to the BASE . "question".
?COMP	Issue error message if not compiling". "question-compiling".
?CSP	Issue error message if the stack position differs from the value saved in CSP. "question-c-s-p".
?DISC	Read the disk drive's status/error code and save it starting at $1295. The length of the message is stored at $1294. See COUNT and TYPE . "question-disc".
?ERROR	f n --- Issue error message n, if the boolean flag is true. "question-error".
?EXEC	Issue an error message if not executing. "question-executing".
?LOADING	Issue an error message if not loading. "question-loading".

?PAIRS	n1 n2 --- Issue an error message if n1 is unequal to n2. The message indicated that compiled conditionals do not match. "question-pairs".
?STACK	Issue an error message if the stack is out of bounds. "question-stack".
?TERMINAL	--- f Perform a test of the STOP key. A true flag indicates actuation. "question-terminal".
@	addr --- n Leave the 16-bit contents of the address. "fetch".
ABORT	Empty both stacks and enter execution state. Return control to the user.
ABS	n --- u Leaves the absolute value of n.
AGAIN	addr n --- (vid kompilering) Used within a colon definition in the form: BEGIN ... AGAIN At run-time, AGAIN forces a brach back to the corresponding BEGIN. The stack is not affected. Execution cannot leave this loop, other than by executing the word EXIT, that is defined on the VIC-FORTH diskette. At compile-time, AGAIN compiles an unconditional branch from address HERE to addr. n is used for error checkin during the compilation.
ALLOT	n --- Add n to the dictionary pointer DP. May be used to reserve dictionary space or re-origin memory. n is a signed number.
AND	n1 n2 --- n3 Leaves the logical bitwise AND result of n1 and n2.
ASSEMBLER	Sets CONTEXT to ASSEMBLER, making dictionary searches begin here. Defind IMMEDIATE making selection of the ASSEMBLER possible even during compilation.
B	(EDITOR) Used to select the screen next lower in sequence as the current editing screen. See N and L. "back".
B.	n --- Prints the number n in binary form, regardless of BASE, which remains unaffected. "b-dot".

B/BUF --- n

This constant leaves the number of bytes per disk buffer, 256. "bytes-per-buffer".

B/SCR --- n

This constant leaves the number of blocks per screen, that is 4. By convention, an editing screen is organized as 16 lines of 64 characters each. "block-per-screen".

BACK addr ---

Calculate the backward branch offset from HERE to addr and compile into the next available dictionary cell.

BACKUP (EDITOR)

Copies the diskette in drive 0 to drive 1, after FLUSH :ing the buffers.

BASE --- addr

A USER variable containing the numeric I/O conversion base.

BEGIN --- addr n

Used 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 .
At compile time, BEGIN leaves its return address and n for compiler error checking.

BL --- c

A constant that leaves the ASCII code for a blank. "blank".

BLANKS addr count ---

Fill an area of memory beginning at addr with blanks.

BLK --- addr

A USER variable that points to the block being interpreted. If zero, text is taken from the keyboard.

BLOCK n --- addr

Leaves the address of block n. If the block is not already in memory, it is transferred from disk to which ever buffer was least recently written. If this buffer contains an updated block, it is first rewritten to disk.
See BUFFER , R/W , -DISC , UPDATE etc.

BOOT-UP

System boot-up. Initializes disks, empties all buffers, and loads the electives screen (screen 1). In the case of the VIC-FORTH System Extension Diskette, this will load the EDITOR and ASSEMBLER vocabularies.

BRANCH

The run-time procedure to unconditionally branch. The cell after BRANCH , containing a offset, is added to the interpretive pointer IP , causing a branch. Compiled by ELSE , AGAIN , and REPEAT .

BUFFER n --- addr

Obtain the next memory buffer, assigning it to block n. If the contents of the buffer is updated, it is written to disk. The block is *not* read from disk. Leaves the address of the first data byte in the block.

C! b addr ---

Store the 8-bit value b at addr. "c-store".

C, b ---

Stores the 8-bit value b in the nest cell in the dictionary, incrementing DP y one. "c-comma".

C@ addr --- b

Fetches one byte from the address. "c-fetch".

CFA pfa --- cfa

Convert the parameter field address of a definition to its code field address. "c-f-a".

CLEAR n --- (EDITOR)

Fills screen n with blanks, preparing it for editing. See ZERO .

CLIT --- b (execution)

A 8-bit counterpart for LIT . "c-lit".

CLOAD --- f

Load a saved dicitonary from cassette. A flag is left on the stack which is false if the load was successful. Otherwise it is an error code.

CLOSE n ---

Close file n, disassociating n from the external device specified in OPEN .

CMOVE orig dest count ---

Moves the specified quantity of bytes specified from address orig to address dest. The byte in orig is moved first, proceeding towards high memory. "c-move".

COLD

Causes a hardware reset. VIC-FORTH will be restarted, with the dictionary pointer set to the minimum standard. May be called to remove application programs and restart.

COMPILE

When the colon definition containing COMPILE is executed, the address of the word following COMPILE will be 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 a constant named cccc, which when executed will push n onto the stack.

CONTEXT --- addr

A USER variable containing a pointer to the vocabulary within which dictionary searches will first begin.

COPY n1 n2 --- (EDITOR)

Copies screen n1 to screen n2.

COUNT addr1 --- addr2 n

Leaves the address addr2 and the byte count n of a string starting at addr1. It is presumed that the string is stored in the usual FORTH way, with the length in the first byte. COUNT is often followed by TYPE .

CR

Transmits a carriage return to the selected output device. "c-r".

CREATE

A defining word used in the form:

```
CREATE cccc
```

by such words as CODE and VARIABLE to create a dictionary header for a FORTH word. The code field contains the address of the words parameter field. The new word is created in the CURRENT vocabulary.

CSAVE --- f

Saves the dictionary and USER-variables to cassette. The workspace, which may be given a name (by NAME), can be reloaded by CLOAD . CSAVE leaves a flag which is false if the save was successful, otherwise it is an error code.

CSP --- addr

A USER variable temporarily storing the stack pointer position, for compilation error checking. "c-s-p".

D n--- (EDITOR)

Deletes line n from the current editing screen. All lines under n will move up one line, and line 15 is blanked. "delete".

D+ d1 d2 --- d3

Leaves the double-precision sum of the double-precision numbers d1 and d2. "d-plus".

D+— d1 n --- d2

Apply the sign of n to the 32-bit value d1, giving d2. "d-plus-minus".

D. d ---

Print the double-precision value d on the selected output device in a free format, followed by a blank. "d-dot".

D.R. d n ---

Print a signed double value d right aligned in a field n characters wide. "d-dot-r".

DABS d --- ud

Leave the absolute value ud of a double value. "d-abs".

DECIMAL

Sets the numeric conversion BASE for decimal I/O.

DEFINITIONS

Used in the form

```
cccc DEFINITIONS
```

Sets the CURRENT vocabulary to the CONTEXT vocabulary. In the example, executing cccc made it the CONTEXT and DEFINITIONS made both specify the vocabulary cccc.

DERROR

Vectored to from (DERROR) when a disk error has occurred. Sets WARNING to zero. "d-error".

DIGIT

```
c  n1 --- n2  tf  (ok)
c  n1 --- ff      (not ok)
```

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  (execution)
d ---    (compilation)
```

If compiling, compile a stack double number into a literal. Later execution will push it onto the stack. If executing, nothing happens. "d-literal".

DMINUS d1 --- d2

Convert d1 to its double number 2s-complement. "d-minus".

DISC

Enable the virtual memory facility by initializing all drives and opening file 15 as command channel and file 13 for data.

DO

```
n1  n2 ---      (execution)
--- addr  n  (compilation)
```

Used in a colon definition in the 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 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. See I , I' , J , K , LOOP , +LOOP , and LEAVE.

When compiling within the colon definition, DO compiles (DO) , leaves the following address addr and n for later error checking.

DOES> Defines the run-time behaviour for members of a user defined word class. When the new defining word is compiled, DOES> will compile (;CODE) and a machine code branch to the routine (DOES>) . When this defining word later is exexuted, (;CODE) will let the defined word's CFA point to the machine code branch to (DOES>). When *this* new member is executed, (DOES>) starts the interpretation of the words following DOES> in the defining word, after having pushed the address of the members parameter field. This allows manipulation with this area. Typical uses include the FORTH assembler, multidimensional arrays, and compiler generation. "does".

DP --- addr

A USER variable, the dictionary pointer, which contains the address of the next free byte above the dictionary. It may be read by HERE and altered by ALLOT . "d-p".

DPL --- addr

A USER variable containing the number of digits to the left of the last decimal point in the most recently converted input number. If no decimal point was present, it has the value -1. DPL may also be used to control the position of the decimal point in a user defined output formatting word. "d-p-l".

DR0, DR1

Selects drive by setting or resetting OFFSET . OFFSET is added to the block number in BLOCK to allow for this selection. Offset i suppressed for error text so that it may always originate from Drive 0. "drive-zero", "drive-one".

DROP n ---

Removes the 16-bit value from the stack.

DUMP addr1 addr2 ---

Dumps the contents of addr1 to addr2 on the output device in hex and ASCII form. The dump may be interrupted by the STOP key.

DUP n --- n n

Duplicates the top stack value.

E n --- (EDITOR)

Erases line n in the current editing screen, that is, fills it with blanks. "erase".

ELSE addr1 n1 --- addr2 n2 (compilation)

Used in a colon definition in the form:
IF . . . ELSE . . . THEN
When executing, ELSE will be executed after the words between IF and ELSE , forcing a branch to the word following THEN . It has no stack effect.
When compiling, ELSE will compile BRANCH and leave the address addr2 and n2 for error checking. ELSE will also resolve the forward reference from IF by calculating the distance from addr1 to HERE and storing this value in addr1.

EMIT c ---

Transfer the ASCII code c to the output device. OUT will be incremented by one for each character output. This word is the basic output word used in FORTH.

EMPTY-BUFFERS

Erases all block buffers to zeroes. Updated blocks are not written to disk. This word is included in the definition of DISC , to prevent garbage from being written to disk.

ENCLOSE addr1 c --- addr1 n1 n2 n3

The next 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 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

This is an earlier name for UNTIL . It is supported by VIC-FORTH, but should not be used.

ENDIF

This is an earlier name for THEN . It is supported by VIC-FORTH, but should not be used.

ERASE addr n --

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

ERROR line --- in blk

Execute error notification and restart of the system. WARNING is ffirst examined. I 1, the text or line n, relative to screen 4 of drive 0 is printed. This line number may be negative, and beyond just screen 4. If WARNING = 0, n is just printed as a message number, since this means that no disk is available. If WARNING is -1, the definition (ABORT) is executed, which executes the system ABORT . The user may cautiously modify this by altering (ABORT). VIC-

FORTH saves the contents of IN and BLK ont the stack 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.

EXPECT addr count ---

Transfer characters from the input device until a "return" or the count of characters have been recieved. One or more nulls are added at the end of the next.

FENCE --- addr

A USER variable containing an address below which FORGET is impossible. To FORGET below this point, FENCE must be changed.

FILL addr count b ---

Fill memory at the address with the specified quantity of bytes b.

FIRST --- n

A constant that leaves the address of the first block buffer (the lowest).

FLD --- addr

A USER variable for control of number output field width. Presently unused in VIC-FORTH. "field".

FORGET

Executed in the form FORGET cccc
Removes the entry cccc from the dictionary and all entries following it.

FORTH

The name of the primary vocabulary.
Execution makes FORTH the CONTEXT vocabulary. FORTH is defined IMMEDIATE , which means it will execute even inside a colon definition, to select FORTH at compile time.

FREE s1 s2 --- (EDITOR)

Prints the numbers of all free screens in the interval screen s1 to screen s2. A screen is regarded as empty if its first cell contains 0. See CLEAR and ZERO.

H n --- (EDITOR)

Copies line n in the current editing screen to PAD . The text will be padded with blanks to 64 characters length.

H. n ---

Prints n in hexadecimal. BASE remains unaffected. "h-dot".

HERE --- addr

Leaves the address of the next free dictionary byte. See DP and ALLOT .

HEX

Set the numeric conversion base to sixteen, i.e. hexadecimal.

HLD --- addr

A USER variable containing the address of the latest character of text during numeric output conversion. "h-l-d".

HOLD c ---

Used between <# and #> to insert an ASCII charter c into a pictured numeric conversion string. E.g. 2E HOLD will place a decimal point.

HPIN b --- f

Selects file b as the current input device. Returns false if possible, otherwise returns an error code.
See OPEN . "h-p-in".

HPOFF

Restores the default I/O devices, that is, the keyboard and video screen. "h-p-off".

HPOUT b --- f

**Selects file b as the current output device. Returns false if possible, otherwise returns an error code.
See OPEN. "h-p-out".**

I h --- (EDITOR)

Insert the contents of PAD as line n in the current editing screen. The old line n and all subsequent lines are moved down, and line 15 is lost. PAD remains unaffected. "insert".

I --- n (FORTH)

Used within a DO - LOOP structure to copy the loop index to the parameter stack. See R , I' , J , and K.

I' --- n

Used within a DO - LOOP construction to copy the loop limit to the stack. See I , J , and K. "i-limit".

ID. nfa ---

Print a definition's name from its name field address. "i-d-dot".

IF f --- (run-time)
--- addr n (compile)

Used in a colon definition in the form:
IF (true) . . . THEN
IF (true) . . . ELSE (false . . . THEN
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 THEN . ELSE and its false part are optional; if missing, false execution skips to just after THEN .
At compile-time IF compiles 0BRANCH and reserves space for an offset at addr. addr and n are later used for resolution of the offset and error testing.

IMMEDIATE

Mark the most recently made definition so that when encountered at compile-time, it will be executed rather than being compiled. The precedence bit in the header is set. This method allows *definitions* to handle unusal compiling situations, rather than build them into the fundamental compiler. The user may force compilation of an IMMEDIATE definition by preceding it with [COMPILE]. Examples of IMMEDIATE-definitioner are IF , DO , DOES> , ; , --> , and LOOP .

IN --- addr

A USER variable containing the byte offset within the current input text buffer (terminal or disk) from which the next text will be accepted. WORD uses and moves the value of IN .

INDEX s1 s2 ---

Print the first line of each screen over the range s1 to s2. This is used to view comment lines of an area of disk screens.

INTERPRET

The outer text interpreter which sequentially executes or compiles text from the input stream (terminal or disk) depending on STATE . If the word name cannot be found in the dictionary it is converted to a value according to BASE . That also failing, an error message is given. Text input will be taken according to the convention for WORD . If a decimal point is found as a part of a number, a double number will be left; the number of decimal is left in DPL. The decimal point has no other purpose than to force this action. See NUMBER .

J --- n

Used within a nested DO - LOOP to fetch the value of the next innermost index. See I , I' , and K .

K --- n

Used within a nested DO-LOOP to fetch the next next innermost loop index. See I , I' , and J .

L (EDITOR)

Used to re-view the current editing screen. Each line is preceded ny its line number and a P to facilitate on-screen editing.

LATEST --- nfa

Leave the name field address of the topmost word in the CURRENT vocabulary.

LEAVE

Force termination of a DO-LOOP by setting the loop limit to the index. The index itself remains unchanged, and execution proceeds normally until the next LOOP or +LOOP .

LFA pfa --- lfa

Convert a word's parameter field address to its link field address. "l-f-a".

LIMIT --- addr

A constant leaving the address of the first byte after the disk buffer area.

LINE n --- addr (EDITOR)

Leaves the address of line n within the current editing screen. An error message is given if n is illegal.

LIST n ---

Display screen n as ASCII text on the selected output device. SCR contains n after this process. See L.

LIT --- n

Within a colon definition, LIT is automatically compiled before each 16 bit literal number encountered in the text. Later execution causes the contents of the next dictionary address to be pushed to the stack.

LITERAL n --- (compilation)

If compiling, then compile the stack value as a 16-bit literal. This definition is immediate so that it will execute during a colon definition.The intended use is:
: xxxx [calculate] LITERAL ... ;
Compilation is suspended for the compile time calculation of a value. Compilation is resumed and LITERAL compiles this value.

LOAD n ---

Begin interpretation of screen n. Loading will terminate at the end of the screen or at ;S. See ;S and -->.

LOOP addr n --- (compiling)

Occurs in a colon definition in the 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 precision operator which leaves the signed double product of two signed numbers. "m-times"
M*/	d1 n u --- d2 A mixed precision operator. Multiplies the signed 32-bit value d1 by the signed 16-bit value n, then divides this by the unsigned value u. The result is a 32-bit signed value. A 48-bit intermediate result is used.
M/	d n1 --- n2 n3 A mixed precision operator which leaves the signed remainder n2 and signed quotient n3, from a double number dividend and a single divisor n1. The remainder takes its sign from the dividend. "m-divide".
M/MOD	ud1 u2 --- u3 ud4 An unsigned mixed precision math operation which leaves a double qoutient ud4 and remainder u3 from a double dividend ud1 and single divisor u2. "m-divide-mod".
MAX	n1 n2 --- n3 Leaves the greater of two numbers.
MESSAGE	n --- Print on the selected output device the text of line n relative to screen 4 of drive 0. n may be positive or negative. MESSAGE may be used to print incidental text such as report headers. If WARNING is zero, the message will simply be printed as a number (disk unavailable).
MIN	n1 n2 --- n3 Leaves the smaller of two values.
MINUS	n1 --- n2 Change sign of a value.
MOD	n1 n2 --- n3 Leaves the remainder of the division n1 n2 / , with the same sign as n1.
N	(EDITOR) Increments SCR by one, making the next lower numbered screen the current editing screen.
NAME	addr len --- This word sets a file name for a subsequent IEEE file operation. It must be used before an OPEN, CSAVE, or CLOAD. If no file name is to be sent, both arguments must be zero.
NEXT	This is the inner interpreter that uses the interpretive pointer IP to execute compiled FORTH definitions. It is not directly executable, but is the return point for all CODE procedures.It acts by fetching the address pointer to by IP, storing this value in the register W. It then jumps to the address pointed to by the address pointer to by W. W points to the code field of a definition which contains the address of the code which executes for that defintion. This usage of *indirect threaded code* is a major contributor to the power, portability, and extensibility of FORTH.
NFA	pfa --- nfa Convert the parameter field address of a word to its name field address. "n-f-a".
NUMBER	addr --- d Convert a character string left at addr with a preceding length byte, to a signed double value using the current numeric BASE. If a decimal point is encountered in the text, its position is stored in DPL, but no other effect occurs. If numeric conversion is not possible, an error message will be given.
OFFSET	--- addr A USER variable which may contain a block offset to disk drives. The contents of OFFSET is added to the stack number by BLOCK. Messages by MESSAGE are independent of OFFSET. See BLOCK, DR0, and MESSAGE.
OPEN	fnr dv sa --- f Open file fnr on logical device dv with secondary address sa. Before executing this word, a file name must be set by NAME. A flag is left which is false if the open was successful and otherwise is an error code.
OR	n1 n2 --- n3 Leave the result of a 16-bit bitwise OR.
OUT	--- addr A USER variable that contains a value incremented by EMIT. The user may alter and examine OUT to control display formatting etc. Since all routines ultimately use EMIT for output, the value reflects the total number of characters output
OVER	n1 n2 --- n1 n2 n1 Copy the second stack value, pushing it on top.
P	n --- (EDITOR) Used in the form: n P xxxx.....xxxxx to store the text xxxx on line n in the current editing screen and in PAD. See L. "put".
PAD	--- addr Leave the address of the text buffer, which is a fixed offset above HERE.

PAPER (EDITOR)

Open file 4 to the printer and select it as the current output device. See VIDEO.

PFA nfa---pfa

Convert the name field address of a word to its parameter field address. "p-f-a".

PREV --- addr

A USER variable containing the address of the disk buffer most recently referenced. the UPDATE command marks this buffer to be later rewritten to disk.

PROGRAM (EDITOR)

Copies screens 0 through 60 from Drive 0 to Drive 1, the area most commonly used for program text.

QUERY

Input 80 characters (or until a "return") from the current input device. 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 --- (EDITOR)

Replace line n in the current editing screen the contents of PAD. "replace".

R --- n (FORTH)

Copy the top of the return stack to the parameter stack.

R# --- addr

A USER variable which may contain the location of an editing cursor, or other file related function. "r-sharp".

R/W addr blk f ---

The FORTH standard disc read-write linkage. addr specifies the source or destination block buffer, blk is the sequential number of the referenced disk 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 checks for errors. "read-slash-write".

R> --- n

Remove the top value from the return stack and leave it on the parameter stack. See >R and R."r-form".

RO --- n

A USER variable containing the initial location of the return stack. See RP!. "r-zero".

REPEAT addr n --- (compiling)

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. At compile-time, REPEAT compiles BRANCH and the offset from HERE to addr. n is used for error testing.

ROT n1 n2 n3 --- n2 n3 n1

Rotate the top three stack values, bringing the *third* to the top. "rote".

RP!

Restores the return stack pointer from the variable RO, thus initializing it."r-p-store".

S n --- (EDITOR)

Spreads the current editing screen at line n, making line n blank and moving all subsequent lines down. Line 15 is lost."spread"

S->D n --- d

Sign extend a single number to form a double number. "s-to-d".

S0 --- addr

A USER variable containing the initial value for the parameter stack. See SP!. "s-zero".

SCR --- addr

A USER variable containing the number of the current editing screen. "s-c-r"

SCREENS fr to count --- (EDITOR)

Copies the indicated number of screens from fr to to.

SHOW s1 s2 ---

Prints all used TRIADS in the range s1 to s2. A TRIAD containing three zero screens will not be listed.

SIGN n d --- d

Stores an ASCII "-" sign just before a converted numeric output string of n is negative. n is removed, but double number d is maintained on the stack. Must be used between <# and #>.

SMUDGE

Used during word definition to toggle the "smudge bit" in a definition header. This prevents an uncompleted definition from being found during dictionary searches until compilation is completed without error. A bad word must therefor be SMUDGEd again before you can FORGET it.

SP!

Empties the parameter stack by loading the parameter stack pointer from S0."s-p-store".

SP@ --- addr

Leave the address of the top value of the stack, as it was *beforeSP@*was executed. That is, the phrase 1 2 SP@ @ ... will type 2 2 1 . "s-p-fetch".

SPACE

Transmit an ASCII blank to the output device.

SPACES
n ---

Transmit n ASCII blanks to the output device.

STATE
--- addr

A USER variable containing the compilation state. A non - zero value indicates compilation.

STATUS
--- n

Push the IEEE file status code. Here is a table of the individual bits in the number.

0	Time out on write
1	Time out on read
2	Short block
3	Long block
4	Unrecoverable read error
5	Checksum error
6	EOI (End of file)
7	Device not present

SWAP
n1 n2 --- n2 n1

Swap the top two stack values.

T*
ud u --- u-triple

Multiplies the unsigned 32-bit number ud by the unsigned 16-bit value u, yielding a 48-bit unsigned result.

T/
u triple u --- ud

Divides an unsigned 48-bit value by an unsigned 16-bit value u, giving a 32-bit unsigned result.

TEXT
c ---

Accept text from the input stream to PAD, after clearing PAD. TEXT will stop accepting after C/L characters, or when an ASCII c character is encountered, or at a RETURN. Initial occurences of c are ignored.

THEN
addr n --- (compiling)

Used within a colon definition in the form:
```
  IF ... THEN
  IF ... ELSE ... THEN
```
At run-time, THEN serves as a destination for a branch from IF or ELSE. It marks the end of a conditional stucture.
At compile-time, THEN calculates the offset from addr to HERE, storing this at addr. n is used for error checking. See IF and ELSE.

TIB
--- addr

A USER variable containing the address of the terminal input buffer. "t-i-b".

TOGGLE
addr b ---

Complement the contents of addr by the bit pattern b.

TRAVERSE
addr1 n --- addr2

Move across the name field of a FORTH variable length name field. Addr1 is the address of either the length byte or the last character in the name.If n=1 the motion is toward hi memory; if n=—1 the motion is toward low memory. The addr2 resulting is the address of the other en of the name field.

TRIAD
scr ---

Display on the selected output device the three screens which include that numbered scr, beginning with a screen evenly divisible by three. Output is suitable for source text records, and includes a reference line at the bottom taken from line 15 of screen 4.

TYPE
addr count ---

Transmit count characters from addr to the selected output device.

U*
u1 u2 --- ud

Leave the unsigned double product of two unsigned single numbers. "u-times".

U.
u ---

Print the value u as a unsigned value. "u-dot".

U/
ud u1 --- u2 u3

Leave the unsigned remainder u2 and the unsigned quotient u3 from the unsigned double dividend ud and the unsigned divisor u1. "u-divide".

UM*/
ud1 u1 u2 --- ud2

As M*/, but all values are unsigned.

UNTIL
```
        f ---  (execution)
addr    n ---  (compiling)
```
Used within a colon definition in the form:
```
  BEGIN ... UNTIL
```
At run-time, UNTIL controls the conditional branch back to BEGIN. Ir r is false, execution returns to just after BEGIN; if true, execution continues ahead.
At compile time, UNTIL compiles 0BRANCH and an offset from HERE to addr. n is used for error tests.

UPDATE

Marks the most recently referenced block (pointed to by PREV) as altered. The block will subsequently be transferred automatically to disk should its buffer be required for a different block.

USE --- addr

A USER variable containing the address of the block buffer to use next, as the least recently written.

USER n ---

A defining word used in the form:
n USER ccc
to create a USER variable called cccc.The parameter field of cccc contains n as a fixed offset relative to the user pointer register UP. When cccc is later executed, the sum of its offsets and the user area base address is pushed to the stack as the storage address of that variable.

VARIABLE

A defining word used in the form:
n VARIABLE cccc
When VARIABLE is executed, it creates the definition 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.

VIDEO (EDITOR)

Restor the default 1/O devices. Close file 4 to the printer.

VOC-LINK --- addr

A USER variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked via these fields to allow for FORGET trough multiple vocabularies.

VOCABULARY

A defining word used in the form:
VOCABULARY vvvv
to create a vocabulary definition vvvv. Subsequent use of vvvv will make it the CONTEXT vocabulary which is searched first. The phrase vvvv DEFINITIONS will also make it the CURRENT vocabulary into which new definitions are linked.
In VIC-FORTH, vvvv will be so chained as to include all definitions of the vocabulary in which cccc itself is defined. All vocabularies therefor ultimately chain to FORTH. By convention, vocabularies are to be defined IMMEDIATE.

VLIST

List the name of the definitions in the CONTEXT vocabulary. STOP will terminate the listing.

WARNING --- addr

A USER variable containing a value controlling messages. If=1, disk is present, and screen 4 of Drive 0 is the base location for messages. If zero, no disk is present and

messages will be presented by number. If=—1, execute (ABORT) for a user specified procedure.

WHERE in blk ---

Prints the line interpreted when an error was detected from in and blk. An up-arrow marks the position. If blk is zero, the error occurred during interpretation from the key-board.

WHILE f --- (execution)
addr1 n --- addr1 n1 addr2 n2

Occurs in a colon definition in the form:
BEGIN ... WHILE ... REPEAT
At run-time , WHILE selects conditional execution based on boolean flag f. If f is true (non-zero), WHILE continues execution of the part up to REPEAT, which then jumps back to BEGIN. If false, the structure is terminated. When compiling, WHILE emplaces 0BRANCH and leaves addr2 of the reserved offset. The stack values will be resolved by REPEAT.

WIDTH --- addr

A USER variable containing the maximum number of letters saved in the compilation of a words' name. It must be 1 through 31, with a default value of 31. It may be changed by the user.

WORD c ---

Read the next text characters from the input stream being interpreted, until a delimiter c is found, storing the packed character string beginning at HERE. WORD leaves the character count in the first byte,the characters, and ends with two or more blanks. Leading occurences of c are ignored. If BLK is non-zero, it specifies from what disk block to interpret. See BLK and IN.

x

This is a pseudonym for the 'null' or dictionary entry for a name of ASCII null. It is the execution procedure to terminate interpretation of a line of text from the terminal or within a disk buffer, as both buffers end with nulls.

XOR n1 n2 --- n3

Leave the result of a 16-bit exclusive-or of n1 and n2.

ZERO n --- (EDITOR)

Fills screen n with ASCII nulls, marking it as unoccupied. See CLEAR and FREE.

Used in a colon definition in the form:
: xxxx ... [...] ... ;
Suspend compilation . The subsequent words are *interpreted* and not compiled. This allows calculation or compilation exceptions before resuming compilation with]. See LITERAL.

[COMPILE]	Used in a colon definition in the form: : xxxx ... [COMPILE] FORTH ... ; [COMPILE] will force compilation of the following IMMEDIATE word, that otherwise would have been *executed* instead of compiled. The above example will select FORTH during execution rather than during the compilation of the word xxxx.
]	Start compiling, to the completion of the colon definition or until [terminates execution See [.

LITERATURE

This literature list includes all sorts of texts on FORTH: some are textbooks, and some are introductory texts. We recommend you to read som of them.

PET-FORTH MANUAL, Datatronic AB, Box 42094, S-126 12 STOCKHOLM, SWEDEN. Textbook and User's Reference Manual for PET-FORTH, which is a fig-FORTH, and the elder brother of VIC-FORTH. This manual also describes IEEE file handling, the assembler, and most extensions of VIC-FORTH. 326 pp paperback. Covers most aspects of FORTH programming.

STARTING FORTH, Leo Brodie, FORTH, Inc., Prentice-Hall. ISBN 0-13-842930-8 harbound, ISBN 0-13-842922-7 paperback. 350 pp. A very good textbook on FORTH, covering most aspects of the language.

SYSTEMS GUIDE TO FIG-FORTH, Forth Interest Group, PO Box 1105, San Carlos, CA 94070. Technical dissectation of a fig-FORTH system. Most words are described and their inner workings explained. Two assemblers are described in detail. 201 pp softbound.

KITT PEAK FORTH PRIMER, Forth Interest Group. This textbook describes the FORTH system at Kitt Peak National Observatory, which is very close to polyFORTH and figFORTH, and thereby to VIC-FORTH. Chapters on Floating Point and Interrupts are included. 200 pp softbound.

BYTE MAGAZINE, August 1980 issue. This is an exellent issue of BYTE, that is exclusively devoted to the FORTH language and programming method. Artecles range from pure introductions to more technical. A demonstration program (BreakForth, similar to Breakout) is described with source code for a TRS-80.

FORTH-79 STANDARD, Forth Interest Group. This paper describes the FORTH-79 standard, which is intended to allow transportability of standard FORTH program in source form among standard FORTH systems. 45 pp ringbound.

PROCEEDINGS 1980 FORML (FORTH Modification Lab) CONFERENCE, Forth Interest Group. A lot of technical data and suggestions are contained in this paper. You will find many interesting concepts to think about and work out further.

VIC-FORTH is a thoroughly tested system, and errors are most unlikely to occur. If any errors or weaknesses are found, however, we would appreciate to hear from you. Also, drop a line if you have any ideas on improvement.

Send all correspondence to:

DATATRONIC AB
System Department VIC-FORTH
Box 42094
S-126 12 STOCKHOLM
SWEDEN

Be sure to include the number of the version of VIC-FORTH (it is visible on the screen after power-on), the complete configuration of your system, your name, address, and telephone number. We regret that we cannot handle your errand unless it is in writing and includes all of the above data.

commodore
COMPUTER