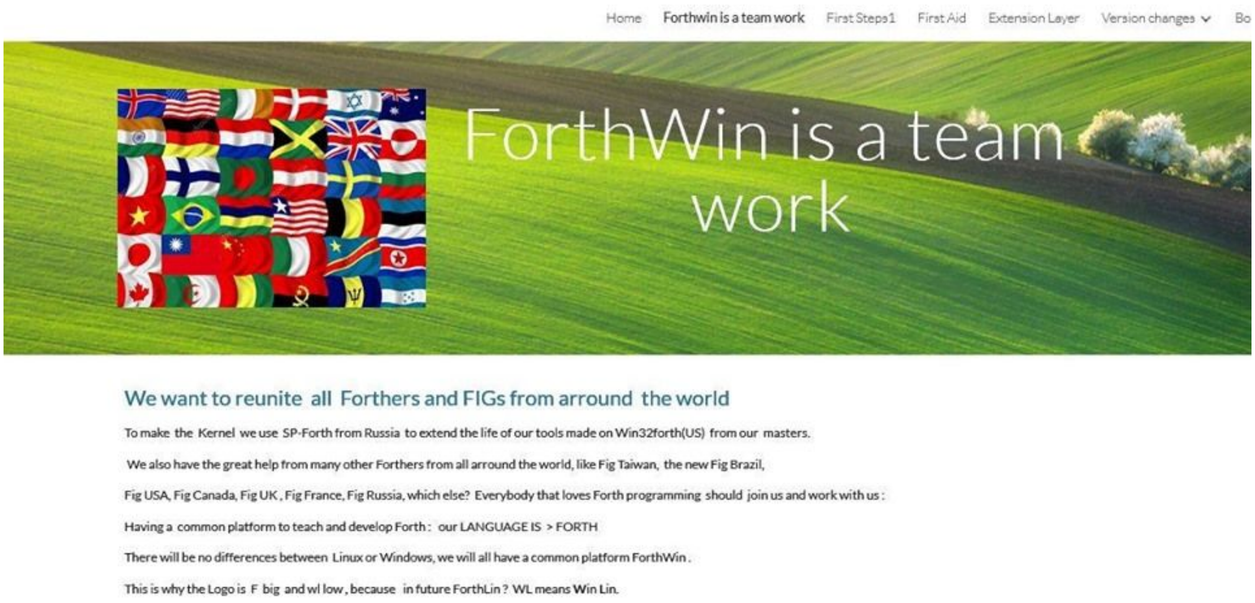

We want to reunite all Forthers and FIGs from arround the world

To make the Kernel we use SP-Forth from Russia to extend the life of our tools made on Win32forth(US) from our masters.

We also have the great help from many other Forthers from all arround the world, like Fig Taiwan, the new Fig Brazil,

Fig USA, Fig Canada, Fig UK , Fig France, Fig Russia, which else? Everybody that loves Forth programming should join us and work with us :

Having a common platform to teach and develop Forth : our LANGUAGE IS > FORTH

There will be no differences between Linux or Windows, we will all have a common platform ForthWin .

This is why the Logo is F big and wl low , because in future ForthLin ? WL means Win Lin.

Startup ForthWin

by Forth-Users-Group

6.1.2019

—

"DON'T PANIC" ***(- As the "Hichhikers Guide To The Galaxy" would say;)***

Means, GO AHEAD ! without fears, and use ForthWin as a platform to make your dreams of programming ,if learning or prototyping , come true thanks ***free Forth tools*** *!*

Many of us learned thanks Tom´s Zimmer and FIG members effort from all arround the world, with F-PC (for DOS) and later Win32forth for the windows environment "DON´T panic" was our guide for our Forth adventures and will be forever with us.

We also want to express our sincere gratitude to Sp-Forth development team from Russia, started by Alexei Cherezov and all our Russian FIG forth friends. Thanks , for all your professional work , incredible passion, and dedication !

ForthWin is a humble continuation of Win32forth+Sp-Forth, with the focus to encourage programmers all over the world to continue creating and using *Free Forths tools, make new Forth friends and share their knowledge*.

We are also happy to receive all of you , testers of ForthWin, future developers and friends. Please help us extend this platform. All your comments, will drive this project further. You are all welcome to participate ! ***ForthWin is your new house***

Chapters : (work in progress)

List of Forth Words in ForthWin dictionary :

(Alphabetical order)

Memory

! (x a-addr --) "store" CORE

Store x at a-addr.

See: 3.3.3.1 Address alignment.

(ud1 -- ud2) "number-sign" CORE

Divide ud1 by the number in BASE giving the quotient ud2 and the remainder n. (n is the least-significant digit of ud1.) Convert n to external form and add the resulting character to the beginning of the pictured numeric output string. An ambiguous condition exists if # executes outside of a

> <# #>

delimited number conversion.

NumConv

#> (xd -- c-addr u) "number-sign-greater" CORE

Drop xd. Make the pictured numeric output string available as a character string. c-addr and u specify the resulting character string. A program may replace characters within the string.

See: 6.1.0030 #, 6.1.0050 #S, 6.1.0490 <#.

NumConv

#S (ud1 -- ud2) "number-sign-s" CORE

Convert one digit of ud1 according to the rule for #. Continue

conversion until the quotient is zero. ud2 is zero. An ambiguous

condition exists if #S executes outside of a

> <# #>

delimited number

conversion.

Execution

' ("<spaces>name" -- xt) "tick" CORE

Skip leading space delimiters. Parse name delimited by a space.

Find name and return xt, the execution token for name. An ambiguous

condition exists if name is not found.

When interpreting,

> ' xyz EXECUTE

is equivalent to

> xyz

.

See: 3.4 The Forth text interpreter, 3.4.1 Parsing, A.6.1.2033 POSTPONE,

A.6.1.2510 ['], D.6.7 Immediacy.

("paren" CORE

Compilation: Perform the execution semantics given below.

Execution: ("ccc<paren>" --)

Parse ccc delimited by) (right parenthesis). (is an immediate word. The number of characters in ccc may be zero to the number of characters in the parse area.

See: 3.4.1 Parsing, 11.6.1.0080 (.

* (n1|u1 n2|u2 -- n3|u3) "star" CORE

Multiply n1|u1 by n2|u2 giving the product n3|u3.

*/ (n1 n2 n3 -- n4) "star-slash" CORE

Multiply n1 by n2 producing the intermediate double-cell result d. Divide d by n3 giving the single-cell quotient n4. An ambiguous condition exists if n3 is zero or if the quotient n4 lies outside the range of a signed number. If d and n3 differ in sign, the implementation-defined result returned will be the same as that returned by either the phrase

> >R M* R> FM/MOD SWAP DROP

or the phrase

> >R M* R> SM/REM SWAP DROP .

Integer division.

***/MOD** (n1 n2 n3 -- n4 n5) "star-slash-mod" CORE

Multiply n1 by n2 producing the intermediate double-cell result d. Divide d by n3 producing the single-cell remainder n4 and the single-cell quotient n5. An ambiguous condition exists if n3 is zero, or if the quotient n5 lies outside the range of a single-cell signed integer. If d and n3 differ in sign, the implementation-defined result returned will be the same as that returned by either the phrase

```
> >R M* R> FM/MOD
```

or the phrase

```
> >R M* R> SM/REM.
```

See: 3.2.2.1 Integer division.

+ (n1|u1 n2|u2 -- n3|u3) "plus" CORE

Add n2|u2 to n1|u1, giving the sum n3|u3.

See: 3.3.3.1 Address alignment.

+! (n|u a-addr --) "plus-store" CORE

Add n|u to the single-cell number at a-addr.

See: 3.3.3.1 Address alignment.

Memory

+LOOP "plus-loop" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: do-sys --)

Append the run-time semantics given below to the current definition. Resolve the destination of all unresolved occurrences of LEAVE between the location given by do-sys and the next location for a transfer of control, to execute the words following +LOOP.

Run-time: (n --) (R: loop-sys1 -- | loop-sys2)

An ambiguous condition exists if the loop control parameters are unavailable. Add n to the loop index. If the loop index did not cross the boundary between the loop limit minus one and the loop limit, continue execution at the beginning of the loop. Otherwise, discard the current loop control parameters and continue execution immediately following the loop.

See: 6.1.1240 DO, 6.1.1680 I, 6.1.1760 LEAVE.

Dictionary

, (x --) "comma" CORE

Reserve one cell of data space and store x in the cell. If the data-space pointer is aligned when , begins execution, it will remain aligned when , finishes execution. An ambiguous condition exists if the data-space pointer is not aligned prior to execution of ,.

See: 3.3.3 Data space, 3.3.3.1 Address alignment.

- (n1|u1 n2|u2 -- n3|u3) "minus" CORE

Subtract n2|u2 from n1|u1, giving the difference n3|u3.

See: 3.3.3.1 Address alignment.

Group: ArithLog

. (n --) "dot" CORE

Display n in free field format.

See: 3.2.1.2 Digit conversion, 3.2.1.3 Free-field number display.

."

." "dot-quote" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by " (double-quote). Append the run-time semantics given below to the current definition.

Run-time: (--)
Display ccc.

See: 3.4.1 Parsing, 6.2.0200 .(.

/ (n1 n2 -- n3) "slash" CORE

Divide n1 by n2, giving the single-cell quotient n3. An ambiguous condition exists if n2 is zero. If n1 and n2 differ in sign, the implementation-defined result returned will be the same as that returned by either the phrase >R S>D R> FM/MOD SWAP DROP or the phrase >R S>D R> SM/REM SWAP DROP.

See: 3.2.2.1 Integer division.

ArithLog

/MOD (n1 n2 -- n3 n4) "slash-mod" CORE

Divide n1 by n2, giving the single-cell remainder n3 and the single-cell quotient n4. An ambiguous condition exists if n2 is zero. If n1 and n2 differ in sign, the implementation-defined result returned will be the same as that returned by either the phrase >R S>D R> FM/MOD or the phrase >R S>D R> SM/REM.

See: 3.2.2.1 Integer division.

ArithLog

0< (n -- flag) "zero-less" CORE

flag is true if and only if n is less than zero.

0= (x -- flag) "zero-equals" CORE

flag is true if and only if x is equal to zero.

1+ (n1|u1 -- n2|u2) "one-plus" CORE

Add one (1) to n1|u1 giving the sum n2|u2.

ArithLog

1- (n1|u1 -- n2|u2) "one-minus" CORE

Subtract one (1) from n1|u1 giving the difference n2|u2.

ArithLog

2! (x1 x2 a-addr --) "two-store" CORE

Store the cell pair x1 x2 at a-addr, with x2 at a-addr and x1 at the next consecutive cell. It is equivalent to the sequence SWAP OVER ! CELL+ !.

See: 3.3.3.1 Address alignment.

Memory

2* (x1 -- x2) "two-star" CORE

x2 is the result of shifting x1 one bit toward the most-significant bit, filling the vacated least-significant bit with zero.

ArithLog

2/

2/ (x1 -- x2) "two-slash" CORE

x2 is the result of shifting x1 one bit toward the least-significant bit, leaving the most-significant bit unchanged.

ArithLog

2@

2@ (a-addr -- x1 x2) "two-fetch" CORE

Fetch the cell pair x1 x2 stored at a-addr. x2 is stored at a-addr and x1 at the next consecutive cell. It is equivalent to the sequence DUP CELL+ @ SWAP @.

See: 3.3.3.1 Address alignment, 6.1.0310 2!.

Memory

2DROP

2DROP (x1 x2 --) "two-drop" CORE

Drop cell pair x1 x2 from the stack.

DataStack

2DUP

2DUP (x1 x2 -- x1 x2 x1 x2) "two-dupe" CORE

Duplicate cell pair x1 x2.

Group: DataStack

2OVER

2OVER "two-over" CORE

(x1 x2 x3 x4 -- x1 x2 x3 x4 x1 x2)

Copy cell pair x1 x2 to the top of the stack.

DataStack

2SWAP

2SWAP "two-swap" CORE

(x1 x2 x3 x4 -- x3 x4 x1 x2)

Exchange the top two cell pairs.

DataStack

: (C: "<spaces>name" -- colon-sys) "colon" CORE

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name, called a "colon definition". Enter compilation state and start the current definition, producing colon-sys. Append the initiation semantics given below to the current definition.
The execution semantics of name will be determined by the words compiled into the body of the definition. The current definition shall not be findable in the dictionary until it is ended (or until the execution of DOES> in some systems).

Initiation: (i*x -- i*x) (R: -- nest-sys)

Save implementation-dependent information nest-sys about the calling definition. The stack effects i*x represent arguments to name.

name Execution: (i*x -- j*x)

Execute the definition name. The stack effects i*x and j*x represent arguments to and results from name, respectively.

See: 3.4 The Forth text interpreter, 3.4.1 Parsing, 3.4.5 Compilation,
6.1.1250 DOES>, 6.1.2500 [, 6.1.2540], 15.6.2.0470 ;CODE.

;

; "semicolon" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: colon-sys --)

Append the run-time semantics below to the current definition. End the current definition, allow it to be found in the dictionary and enter interpretation state, consuming colon-sys. If the data-space pointer is not aligned, reserve enough data space to align it.

Run-time: (--) (R: nest-sys --)

Return to the calling definition specified by nest-sys.

See: 3.4 The Forth text interpreter, 3.4.5 Compilation.

<

<	(n1 n2 -- flag)	"less-than"	CORE

flag is true if and only if n1 is less than n2.

See: 6.1.2340 U<

NumConv

<#	(--)	"less-number-sign"	CORE

Initialize the pictured numeric output conversion process.

See: 6.1.0030 #, 6.1.0040 #>, 6.1.0050 #S.

= (x1 x2 -- flag) "equals" CORE

flag is true if and only if x1 is bit-for-bit the same as x2.

> (n1 n2 -- flag) "greater-than" CORE

flag is true if and only if n1 is greater than n2.

See: 6.2.2350 U>.

>BODY (xt -- a-addr) "to-body" CORE

a-addr is the data-field address corresponding to xt. An ambiguous condition exists if xt is not for a word defined via CREATE.

See: 3.3.3 Data space.

Execution

>IN (-- a-addr) "to-in" CORE

a-addr is the address of a cell containing the offset in characters from the start of the input buffer to the start of the parse area.

>NUMBER

>NUMBER "to-number" CORE

(ud1 c-addr1 u1 -- ud2 c-addr2 u2)

ud2 is the unsigned result of converting the characters within the string specified by c-addr1 u1 into digits, using the number in BASE, and adding each into ud1 after multiplying ud1 by the number in BASE. Conversion continues left-to-right until a character that is not convertible, including any "+" or "-", is encountered or the string is entirely converted. c-addr2 is the location of the first unconverted character or the first character past the end of the string if the string was entirely converted. u2 is the number of unconverted characters in the string. An ambiguous condition exists if ud2 overflows during the conversion.

See: 3.2.1.2 Digit conversion.

NumConv

>R

Interpretation: Interpretation semantics for this word are undefined.

Execution: (x --) (R: -- x)

Move x to the return stack.

See: 3.2.3.3 Return stack, 6.1.2060 R>, 6.1.2070 R@, 6.2.0340 2>R, 6.2.0410 2R>, 6.2.0415 2R@.

RStack

?DUP (x -- 0 | x x) "question-dupe" CORE

Duplicate x if it is non-zero.

DataStack

@

@ (a-addr -- x) "fetch" CORE

x is the value stored at a-addr.

See: 3.3.3.1 Address alignment.

Memory

ABORT (i*x --) (R: j*x --) CORE

Empty the data stack and perform the function of QUIT, which includes emptying the return stack, without displaying a message.

See: 9.6.2.0670 ABORT.

ABORT" "abort-quote" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by a " (double-quote). Append the run-time semantics given below to the current definition.

Run-time: (i*x x1 -- | i*x) (R: j*x -- | j*x)

Remove x1 from the stack. If any bit of x1 is not zero, display ccc and perform an implementation-defined abort sequence that includes the function of ABORT.

See: 3.4.1 Parsing, 9.6.2.0680 ABORT".

ABS (n -- u) "abs" CORE

u is the absolute value of n.

ArithLog

ACCEPT (c-addr +n1 -- +n2) CORE

Receive a string of at most +n1 characters. An ambiguous condition exists if +n1 is zero or greater than 32,767. Display graphic characters as they are received. A program that depends on the presence or absence of non-graphic characters in the string has an environmental dependency. The editing functions, if any, that the system performs in order to construct the string are implementation-defined.
Input terminates when an implementation-defined line terminator is received. When input terminates, nothing is appended to the string, and the display is maintained in an implementation-defined way.
+n2 is the length of the string stored at c-addr.

ALIGN (--) CORE

If the data-space pointer is not aligned, reserve enough space to align it.

See: 3.3.3 Data space, 3.3.3.1 Address alignment.

ALIGNED (addr -- a-addr) CORE

a-addr is the first aligned address greater than or equal to addr.

See: 3.3.3.1 Address alignment.

ALLOT

ALLOT (n --) CORE

If n is greater than zero, reserve n address units of data space. If n is less than zero, release |n| address units of data space. If n is zero, leave the data-space pointer unchanged.

If the data-space pointer is aligned and n is a multiple of the size of a cell when ALLOT begins execution, it will remain aligned when ALLOT finishes execution.

If the data-space pointer is character aligned and n is a multiple of the size of a character when ALLOT begins execution, it will remain character aligned when ALLOT finishes execution.

See: 3.3.3 Data space.

AND (x1 x2 -- x3) AND CORE

x3 is the bit-by-bit logical "and" of x1 with x2.

ArithLog

BASE (-- a-addr) BASE CORE

a-addr is the address of a cell containing the current number-conversion radix {{2...36}}

NumConv

BEGIN

BEGIN CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: -- dest)

Put the next location for a transfer of control, dest, onto the control flow stack. Append the run-time semantics given below to the current definition.

Run-time: (--)

Continue execution.

See: 3.2.3.2 Control-flow stack, 6.1.2140 REPEAT, 6.1.2390 UNTIL,

6.1.2430 WHILE.

BL

BL (-- char) "b-l" CORE

char is the character value for a space.

C!

C! (char c-addr --) "c-store" CORE

Store char at c-addr. When character size is smaller than cell size, only the number of low-order bits corresponding to character size are transferred.

See: 3.3.3.1 Address alignment

Memory

C, (char --) "c-comma" CORE

Reserve space for one character in the data space and store char in the space. If the data-space pointer is character aligned when C, begins execution, it will remain character aligned when C, finishes execution. An ambiguous condition exists if the data-space pointer is not character-aligned prior to execution of C,.

See: 3.3.3 Data space, 3.3.3.1 Address alignment.

C@ (c-addr -- char) "c-fetch" CORE

Fetch the character stored at c-addr. When the cell size is greater than character size, the unused high-order bits are all zeroes.

See: 3.3.3.1 Address alignment.

Memory

CELL+ (a-addr1 -- a-addr2) "cell-plus" CORE

Add the size in address units of a cell to a-addr1, giving a-addr2.

See: 3.3.3.1 Address alignment.

Memory

CELLS (n1 -- n2) CELLS CORE

n2 is the size in address units of n1 cells.

Memory

CHAR

&6.1.0895 CHAR "char" CORE

("<spaces>name" -- char)

Skip leading space delimiters. Parse name delimited by a space. Put the value of its first character onto the stack.

See: 3.4.1 Parsing, 6.1.2520 [CHAR].

CHAR+

CHAR+ (c-addr1 -- c-addr2) "char-plus" CORE

Add the size in address units of a character to c-addr1, giving c-addr2.

See: 3.3.3.1 Address alignment.

Memory

CHARS

CHARS (n1 -- n2) "chars" CORE

n2 is the size in address units of n1 characters.

Memory

CONSTANT

CONSTANT (x "<spaces>name" --) CORE

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. name is referred to as a "constant".

name Execution: (-- x)

Place x on the stack.

See: 3.4.1 Parsing.

COUNT

COUNT (c-addr1 -- c-addr2 u) CORE

Return the character string specification for the counted string stored at c-addr1. c-addr2 is the address of the first character after c-addr1. u is the contents of the character at c-addr1, which is the length in characters of the string at c-addr2.

CR

CR (--) "c-r" CORE

Cause subsequent output to appear at the beginning of the next line.

CREATE

CREATE ("<spaces>name" --) CORE

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. If the data-space pointer is not aligned, reserve enough data space to align it. The new data-space pointer defines name's data field. CREATE does not allocate data space in name's data field.

name Execution: (-- a-addr)

a-addr is the address of name's data field. The execution semantics of name may be extended by using DOES>.

See: 3.3.3 Data space, 6.1.1250 DOES>.

DECIMAL

DECIMAL (--) CORE

Set the numeric conversion radix to ten (decimal).

NumConv

DEPTH

(-- +n) DEPTH CORE

+n is the number of single-cell values contained in the data stack before +n was placed on the stack.

DataStack

DO

" DO" CORE

Interpretation: Interpretation semantics for this word are undefined. Place do-sys onto the control-flow stack. Append the run-time semantics given below to the current definition. The semantics are incomplete until resolved by a consumer of do-sys such as LOOP.

Run-time: (n1|u1 n2|u2 --) (R: -- loop-sys)

Set up loop control parameters with index n2|u2 and limit n1|u1. An ambiguous condition exists if n1|u1 and n2|u2 are not both the same type. Anything already on the return stack becomes unavailable until the loop-control parameters are discarded.

See: 3.2.3.2 Control-flow stack, 6.1.0140 +LOOP, 6.1.1800 LOOP.

DOES>

DOES> "does" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: colon-sys1 -- colon-sys2)

Append the run-time semantics below to the current definition. Whether or not the current definition is rendered findable in the dictionary by the compilation of DOES> is implementation defined. Consume colon-sys1 and produce colon-sys2. Append the initiation semantics given below to the current definition.

Run-time: (--) (R: nest-sys1 --)

Replace the execution semantics of the most recent definition, referred to as name, with the name execution semantics given below. Return control to the calling definition specified by nest-sys1. An ambiguous condition exists if name was not defined with CREATE or a user-defined word that calls CREATE.

Initiation: (i*x -- i*x a-addr) (R: -- nest-sys2)

Save implementation-dependent information nest-sys2 about the calling definition. Place name's data field address on the stack. The stack effects i*x represent arguments to name.

name Execution: (i*x -- j*x)

Execute the portion of the definition that begins with the initiation semantics appended by the DOES> which modified name. The stack effects i*x and j*x represent arguments to and results from name, respectively.

See: 6.1.1000 CREATE.

DROP

DROP (x --) CORE

Remove x from the stack.

DataStack

DUP

DUP (x -- x x) "dupe" CORE

Duplicate x.

DataStack

ELSE

ELSE CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: orig1 -- orig2)

Put the location of a new unresolved forward reference orig2 onto the control flow stack. Append the run-time semantics given below to the current definition. The semantics will be incomplete until orig2 is resolved (e.g., by THEN). Resolve the forward reference orig1 using the location following the appended run-time semantics.

Run-time: (--)

Continue execution at the location given by the resolution of orig2.

See: 6.1.1700 IF, 6.1.2270 THEN.

EMIT (x --) EMIT CORE

If x is a graphic character in the implementation-defined character set, display x. The effect of EMIT for all other values of x is implementation-defined.
When passed a character whose character-defining bits have a value between hex 20 and 7E inclusive, the corresponding standard character, specified by 3.1.2.1 Graphic characters, is displayed. Because different output devices can respond differently to control characters, programs that use control characters to perform specific functions have an environmental dependency. Each EMIT deals with only one character.

See: 6.1.2310 TYPE.

ENVIRONMENT? (c-addr u -- false | i*x true)

"environment-query" CORE

c-addr is the address of a character string and u is the string's character count. u may have a value in the range from zero to an implementation-defined maximum which shall not be less than 31. The character string should contain a keyword from 3.2.6 Environmental queries or the optional word sets to be checked for correspondence with an attribute of the present environment. If the system treats the attribute as unknown, the returned flag is false; otherwise, the flag is true and the i*x returned is of the type specified in the table for the attribute queried.

Useless

EVALUATE

EVALUATE (i*x c-addr u -- j*x) CORE

Save the current input source specification. Store minus-one (-1) in SOURCE-ID if it is present. Make the string described by c-addr and u both the input source and input buffer, set >IN to zero, and interpret. When the parse area is empty, restore the prior input source specification. Other stack effects are due to the words EVALUATEd.

Execution

EXECUTE

EXECUTE CORE

(i*x xt -- j*x)

Remove xt from the stack and perform the semantics identified by it. Other stack effects are due to the word EXECUTEd.

See: 6.1.0070 ', 6.1.2510 ['].

Execution

EXIT

EXIT CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (--) (R: nest-sys --)

Return control to the calling definition specified by nest-sys. Before executing EXIT within a do-loop, a program shall discard the loop-control parameters by executing UNLOOP.

See: 3.2.3.3 Return stack, 6.1.2380 UNLOOP.

FILL

FILL (c-addr u char --) CORE

If u is greater than zero, store char in each of u consecutive characters of memory beginning at c-addr.

Group: Memory

FIND

FIND (c-addr -- c-addr 0 | xt 1 | xt -1) CORE

Find the definition named in the counted string at c-addr. If the definition is not found, return c-addr and zero. If the definition is found, return its execution token xt. If the definition is immediate, also return one (1), otherwise also return minus-one (-1). For a given string, the values returned by FIND while compiling may differ from those returned while not compiling.

See: 3.4.2 Finding definition names, A.6.1.0070 ', A.6.1.2510 ['],
A.6.1.2033 POSTPONE, D.6.7 Immediacy.

Group: Execution

FM/MOD

FM/MOD (d1 n1 -- n2 n3) "f-m-slash-mod" CORE

Divide d1 by n1, giving the floored quotient n3 and the remainder n2. Input and output stack arguments are signed. An ambiguous condition exists if n1 is zero or if the quotient lies outside the range of a single-cell signed integer.

See: 3.2.2.1 Integer division, 6.1.2214 SM/REM, 6.1.2370 UM/MOD.

HERE

HERE (-- addr) CORE

addr is the data-space pointer.

See: 3.3.3.2 Contiguous regions.

HOLD

HOLD (char --) CORE

Add char to the beginning of the pictured numeric output string. An ambiguous condition exists if HOLD executes outside of a <# #> delimited number conversion.

NumConv

I

(--n) CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (-- n|u) (R: loop-sys -- loop-sys)

n|u is a copy of the current (innermost) loop index. An ambiguous condition exists if the loop control parameters are unavailable.

IF

IF CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: -- orig)

Put the location of a new unresolved forward reference orig onto the control flow stack. Append the run-time semantics given below to the current definition. The semantics are incomplete until orig is resolved, e.g., by THEN or ELSE.

Run-time: (x --)

If all bits of x are zero, continue execution at the location specified by the resolution of orig.

See: 3.2.3.2 Control flow stack, 6.1.1310 ELSE, 6.1.2270 THEN.

IMMEDIATE

IMMEDIATE (--) CORE

Make the most recent definition an immediate word. An ambiguous condition exists if the most recent definition does not have a name.

See: D.6.7 Immediacy.

INVERT

INVERT (x1 -- x2) CORE

Invert all bits of x1, giving its logical inverse x2.

See: 6.1.1910 NEGATE, 6.1.0270 0=.

ArithLog

J

(--n) CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (-- n|u) (R: loop-sys1 loop-sys2 -- loop-sys1 loop-sys2)

n|u is a copy of the next-outer loop index. An ambiguous condition exists if the loop control parameters of the next-outer loop, loop-sys1, are unavailable.

KEY

(-- char) KEY CORE

Receive one character char, a member of the implementation-defined character set. Keyboard events that do not correspond to such characters are discarded until a valid character is received, and those events are subsequently unavailable.

All standard characters can be received. Characters received by KEY are not displayed.

Any standard character returned by KEY has the numeric value specified in 3.1.2.1 Graphic characters. Programs that require the ability to receive control characters have an environmental dependency.

See: 10.6.2.1307 EKEY, 10.6.1.1755 KEY?.

LEAVE (--) LEAVE CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (--) (R: loop-sys --)

Discard the current loop control parameters. An ambiguous condition exists if they are unavailable. Continue execution immediately following the innermost syntactically enclosing DO ... LOOP or DO ... +LOOP.

See: 3.2.3.3 Return stack, 6.1.0140 +LOOP, 6.1.1800 LOOP.

LITERAL

&6.1.1780 LITERAL CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (x --)

Append the run-time semantics given below to the current definition.

Run-time: (-- x)

Place x on the stack.

LOOP

&6.1.1800 LOOP CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: do-sys --)

Append the run-time semantics given below to the current definition. Resolve the destination of all unresolved occurrences of LEAVE between the location given by do-sys and the next location for a transfer of control, to execute the words following the LOOP.

Run-time: (--) (R: loop-sys1 -- | loop-sys2)

An ambiguous condition exists if the loop control parameters are unavailable. Add one to the loop index. If the loop index is then equal to the loop limit, discard the loop parameters and continue execution immediately following the loop. Otherwise continue execution at the beginning of the loop.

See: 6.1.1240 DO, 6.1.1680 I, 6.1.1760 LEAVE.

LSHIFT

LSHIFT (x1 u -- x2) "l-shift" CORE

Perform a logical left shift of u bit-places on x1, giving x2. Put zeroes into the least significant bits vacated by the shift. An ambiguous condition exists if u is greater than or equal to the number of bits in a cell.

ArithLog

M*

M* (n1 n2 -- d) "m-star" CORE

d is the signed product of n1 times n2.

ArithLog

MAX

MAX (n1 n2 -- n3) CORE

n3 is the greater of n1 and n2.

ArithLog

MIN

MIN (n1 n2 -- n3) CORE

n3 is the lesser of n1 and n2.

ArithLog

MOD

MOD (n1 n2 -- n3) CORE

Divide n1 by n2, giving the single-cell remainder n3. An ambiguous condition exists if n2 is zero. If n1 and n2 differ in sign, the implementation-defined result returned will be the same as that returned by either the phrase >R S>D R> FM/MOD DROP or the phrase >R S>D R> SM/REM DROP.

See: 3.2.2.1 Integer division.

ArithLog

MOVE

MOVE	(addr1 addr2 u --)	CORE

If u is greater than zero, copy the contents of u consecutive address units at addr1 to the u consecutive address units at addr2. After MOVE completes, the u consecutive address units at addr2 contain exactly what the u consecutive address units at addr1 contained before the move.

See: 17.6.1.0910 CMOVE, 17.6.1.0920 CMOVE>.

Memory

NEGATE

NEGATE	(n1 -- n2)	CORE

Negate n1, giving its arithmetic inverse n2.

See: 6.1.1720 INVERT, 6.1.0270 0=.

ArithLog

OR

OR (x1 x2 -- x3) CORE

x3 is the bit-by-bit inclusive-or of x1 with x2.

ArithLog

OVER

OVER (x1 x2 -- x1 x2 x1) CORE

Place a copy of x1 on top of the stack.

DataStack

POSTPONE

&6.1.2033 POSTPONE CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name. Append the compilation semantics of name to the current definition. An ambiguous condition exists if name is not found.

See: 3.4.1 Parsing.

QUIT

&6.1.2050   QUIT   CORE

(--) (R: i*x --)

Empty the return stack, store zero in SOURCE-ID if it is present, make the user input device the input source, and enter interpretation state. Do not display a message. Repeat the following:

- Accept a line from the input source into the input buffer, set >IN to zero, and interpret.

- Display the implementation-defined system prompt if in interpretation state, all processing has been completed, and no ambiguous condition exists.

See: 3.4 The Forth text interpreter.

R>

&6.1.2060   R>   "r-from"   CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (-- x) (R: x --)

Move x from the return stack to the data stack.

See: 3.2.3.3 Return stack, 6.1.0580 >R, 6.1.2070 R@, 6.2.0340 2>R, 6.2.0410 2R>, 6.2.0415 2R@.

Group: RStack

R@

&6.1.2070   R@   "r-fetch"   CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (-- x) (R: x -- x)

Copy x from the return stack to the data stack.

See: 3.2.3.3 Return stack, 6.1.0580 >R, 6.1.2060 R>, 6.2.0340 2>R, 6.2.0410 2R>, 6.2.0415 2R@.

RStack

RECURSE

&6.1.2120   RECURSE   CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (--)

Append the execution semantics of the current definition to the current definition. An ambiguous condition exists if RECURSE appears in a definition after DOES>.

See: 6.1.1250 DOES>, 6.1.2120 RECURSE.

REPEAT

&6.1.2140 REPEAT CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: orig dest --)

Append the run-time semantics given below to the current definition, resolving the backward reference dest. Resolve the forward reference orig using the location following the appended run-time semantics.

Run-time: (--)

Continue execution at the location given by dest.

See: 6.1.0760 BEGIN, 6.1.2430 WHILE.

ROT

ROT (x1 x2 x3 -- x2 x3 x1) "rote" CORE

Rotate the top three stack entries.

DataStack

RSHIFT

RSHIFT (x1 u -- x2) "r-shift" CORE

Perform a logical right shift of u bit-places on x1, giving x2. Put zeroes into the most significant bits vacated by the shift. An ambiguous condition exists if u is greater than or equal to the number of bits in a cell.

ArithLog

S"

&6.1.2165 S" "s-quote" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by " (double-quote). Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr u)

Return c-addr and u describing a string consisting of the characters ccc. A program shall not alter the returned string.

See: 3.4.1 Parsing, 6.2.0855 C", 11.6.1.2165 S".

S>D

S>D (n -- d) "s-to-d" CORE

Convert the number n to the double-cell number d with the same numerical value.

ArithLog

SIGN

SIGN (n --) CORE

If n is negative, add a minus sign to the beginning of the pictured numeric output string. An ambiguous condition exists if SIGN executes outside of a <# #> delimited number conversion.

NumConv

SM/REM

SM/REM (d1 n1 -- n2 n3) "s-m-slash-rem" CORE

Divide d1 by n1, giving the symmetric quotient n3 and the remainder n2. Input and output stack arguments are signed. An ambiguous condition exists if n1 is zero or if the quotient lies outside the range of a single-cell signed integer.

&See: 3.2.2.1 Integer division, 6.1.1561 FM/MOD, 6.1.2370 UM/MOD.

Group: ArithLog

SOURCE

SOURCE (-- c-addr u) CORE

c-addr is the address of, and u is the number of characters in, the input buffer.

SPACE

SPACE (--) CORE

Display one space.

SPACES

SPACES (n --) CORE

If n is greater than zero, display n spaces.

STATE

STATE (-- a-addr) CORE

a-addr is the address of a cell containing the compilation-state flag. STATE is true when in compilation state, false otherwise. The true value in STATE is non-zero, but is otherwise implementation-defined. Only the following standard words alter the value in STATE: : (colon), ; (semicolon), ABORT, QUIT, :NONAME, [(left-bracket), and] (right-bracket).

Note:
A program shall not directly alter the contents of STATE.

See: 3.4 The Forth text interpreter, 6.1.0450 :, 6.1.0460 ;, 6.1.0670 ABORT, 6.1.2050 QUIT, 6.1.2500 [, 6.1.2540], 6.2.0455 :NONAME, 15.6.2.2250 STATE.

SWAP

SWAP (x1 x2 -- x2 x1) CORE

Exchange the top two stack items.

DataStack

THEN

THEN CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: orig --)

Append the run-time semantics given below to the current definition. Resolve the forward reference orig using the location of the appended run-time semantics.

Run-time: (--)

Continue execution.

See: 6.1.1310 ELSE, 6.1.1700 IF.

TYPE

TYPE (c-addr u --) CORE

If u is greater than zero, display the character string specified by c-addr and u.

When passed a character in a character string whose character-defining bits have a value between hex 20 and 7E inclusive, the corresponding standard character, specified by 3.1.2.1 graphic characters, is displayed. Because different output devices can respond differently to control characters, programs that use control characters to perform specific functions have an environmental dependency.

See: 6.1.1320 EMIT.

U.

U. (u --) "u-dot" CORE

Display u in free field format.

U< "u-less

U< (u1 u2 -- flag) "u-less-than" CORE

flag is true if and only if u1 is less than u2.

See: 6.1.0480 <.

UM*

UM*	(u1 u2 -- ud)	"u-m-star"	CORE

Multiply u1 by u2, giving the unsigned double-cell product ud.

All values and arithmetic are unsigned.

ArithLog

UM/MOD

UM/MOD	(ud u1 -- u2 u3)	"u-m-slash-mod"	CORE

Divide ud by u1, giving the quotient u3 and the remainder u2. All values and arithmetic are unsigned. An ambiguous condition exists if u1 is zero or if the quotient lies outside the range of a single-cell unsigned integer.

See: 3.2.2.1 Integer division, 6.1.1561 FM/MOD, 6.1.2214 SM/REM.

ArithLog

UNLOOP

UNLOOP	CORE

Interpretation: Interpretation semantics for this word are undefined.

Execution: (--) (R: loop-sys --)

Discard the loop-control parameters for the current nesting level. An UNLOOP is required for each nesting level before the definition may be EXITed. An ambiguous condition exists if the loop-control parameters are unavailable. See: 3.2.3.3 Return stack.

UNTIL

&6.1.2390   UNTIL   CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: dest --)

Append the run-time semantics given below to the current definition, resolving the backward reference dest.

Run-time: (x --)

If all bits of x are zero, continue execution at the location specified by dest.

See: 6.1.0760 BEGIN.

VARIABLE

VARIABLE   ("<spaces>name" --)   CORE

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below.

Reserve one cell of data space at an aligned address.

name is referred to as a "variable"

name Execution: (-- a-addr)

a-addr is the address of the reserved cell. A program is responsible or initializing the contents of the reserved cell. See: 3.4.1 Parsing.

WHILE

WHILE CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: dest -- orig dest)

Put the location of a new unresolved forward reference orig onto the control flow stack, under the existing dest. Append the run-time semantics given below to the current definition. The semantics are incomplete until orig and dest are resolved (e.g., by REPEAT).

Run-time: (x --)

If all bits of x are zero, continue execution at the location specified by the resolution of orig.

WORD

&6.1.2450 WORD CORE

(char "<chars>ccc<char>" -- c-addr)

Skip leading delimiters. Parse characters ccc delimited by char. An ambiguous condition exists if the length of the parsed string is greater than the implementation-defined length of a counted string.
c-addr is the address of a transient region containing the parsed word as a counted string. If the parse area was empty or contained no characters other than the delimiter, the resulting string has a zero length. A space, not included in the length, follows the string. A

program may replace characters within the string.

Note: The requirement to follow the string with a space is obsolescent and is included as a concession to existing programs that use CONVERT. A program shall not depend on the existence of the space.

See: 3.3.3.6 Other transient regions, 3.4.1 Parsing.

XOR

XOR	(x1 x2 -- x3)	"x-or"	CORE

x3 is the bit-by-bit exclusive-or of x1 with x2.

ArithLog

[

&6.1.2500	[	"left-bracket"	CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: Perform the execution semantics given below.

Execution: (--)

Enter interpretation state. [is an immediate word.

See: 3.4 The Forth text interpreter, 3.4.5 Compilation, 6.1.2540].

[']

&6.1.2510 ['] "bracket-tick" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name. Append the run-time semantics given below to the current definition.

An ambiguous condition exists if name is not found.

Run-time: (-- xt)

Place name's execution token xt on the stack. The execution token returned by the compiled phrase "['] X " is the same value returned by "' X " outside of compilation state.

See: 3.4.1 Parsing, A.6.1.0070 ', A.6.1.2033 POSTPONE, D.6.7 Immediacy.

Execution

[CHAR]

&6.1.2520 [CHAR] "bracket-char" CORE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition.

Run-time: (-- char)

Place char, the value of the first character of name, on the stack.

See: 3.4.1 Parsing, 6.1.0895 CHAR.

]

]	(--)	"right-bracket"	CORE

Enter compilation state.

See: 3.4 The Forth text interpreter, 3.4.5 Compilation, 6.1.2500 [.

InStream

#TIB

#TIB	(-- a-addr)	"number-t-i-b"	CORE EXT

a-addr is the address of a cell containing the number of characters in the terminal input buffer.

Note: This word is obsolescent and is included as a concession to existing implementations.

.(

&6.2.0200 .("dot-paren" CORE EXT

Compilation: Perform the execution semantics given below.

Execution: ("ccc<paren>" --)

Parse and display ccc delimited by) (right parenthesis). .(is an immediate word.

See: 3.4.1 Parsing, 6.1.0190 .".

.R

&6.2.0210 .R "dot-r" CORE EXT

(n1 n2 --)

Display n1 right aligned in a field n2 characters wide. If the number of characters required to display n1 is greater than n2, all digits are displayed with no leading spaces in a field as wide as necessary.

0<>

0<> (x -- flag) "zero-not-equals" CORE EXT

flag is true if and only if x is not equal to zero.

0>

0> (n -- flag) "zero-greater" CORE EXT

flag is true if and only if n is greater than zero.

2>R

&6.2.0340 2>R "two-to-r" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: (x1 x2 --) (R: -- x1 x2)

Transfer cell pair x1 x2 to the return stack. Semantically equivalent to SWAP >R >R.

See: 3.2.3.3 Return stack, 6.1.0580 >R, 6.1.2060 R>, 6.1.2070 R@, 6.2.0410 2R>, 6.2.0415 2R@.

RStack

2R>

&6.2.0410 2R> "two-r-from" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: (-- x1 x2) (R: x1 x2 --)

Transfer cell pair x1 x2 from the return stack. Semantically equivalent to R> R> SWAP.

See: 3.2.3.3 Return stack, 6.1.0580 >R, 6.1.2060 R>, 6.1.2070 R@, 6.2.0340 2>R, 6.2.0415 2R@.

RStack

2R@

&6.2.0415 2R@ "two-r-fetch" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: (-- x1 x2) (R: x1 x2 -- x1 x2)

Copy cell pair x1 x2 from the return stack. Semantically equivalent to R> R> 2DUP >R >R SWAP.

See: 3.2.3.3 Return stack, 6.1.0580 >R, 6.1.2060 R>, 6.1.2070 R@, 6.2.0340 2>R, 6.2.0410 2R>.

RStack

:NONAME

&6.2.0455 :NONAME "colon-no-name" CORE EXT

(C: -- colon-sys) (S: -- xt)

Create an execution token xt, enter compilation state and start the current definition, producing colon-sys. Append the initiation

semantics given below to the current definition.
The execution semantics of xt will be determined by the words compiled into the body of the definition. This definition can be executed later by using xt EXECUTE.

If the control-flow stack is implemented using the data stack, colon-sys shall be the topmost item on the data stack.

See 3.2.3.2 Control-flow stack.

Initiation: (i*x -- i*x) (R: -- nest-sys)

Save implementation-dependent information nest-sys about the calling definition. The stack effects i*x represent arguments to xt.

xt Execution: (i*x -- j*x)

Execute the definition specified by xt. The stack effects i*x and j*x represent arguments to and results from xt, respectively.

<>

<> (x1 x2 -- flag) "not-equals" CORE EXT

flag is true if and only if x1 is not bit-for-bit the same as x2.

?DO

?DO "question-do" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: -- do-sys)

Put do-sys onto the control-flow stack. Append the run-time semantics given below to the current definition. The semantics are incomplete until resolved by a consumer of do-sys such as LOOP.

Run-time: (n1|u1 n2|u2 --) (R: -- | loop-sys)

If n1|u1 is equal to n2|u2, continue execution at the location given by the consumer of do-sys. Otherwise set up loop control parameters with index n2|u2 and limit n1|u1 and continue executing immediately following ?DO. Anything already on the return stack becomes unavailable until the loop control parameters are discarded. An ambiguous condition exists if n1|u1 and n2|u2 are not both of the same type.

See: 3.2.3.2 Control-flow stack, 6.1.0140 +LOOP, 6.1.1240 DO, 6.1.1680 I, 6.1.1760 LEAVE, 6.1.1800 LOOP, 6.1.2380 UNLOOP.

AGAIN

AGAIN CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: dest --)

Append the run-time semantics given below to the current definition, resolving the backward reference dest.

Run-time: (--)

Continue execution at the location specified by dest. If no other control flow words are used, any program code after AGAIN will not be executed.

See: 6.1.0760 BEGIN.

C"

C" "c-quote" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by " (double-quote) and append the run-time semantics given below to the current definition.

Run-time: (-- c-addr)

Return c-addr, a counted string consisting of the characters ccc. A program shall not alter the returned string.

See: 3.4.1 Parsing, 6.1.2165 S", 11.6.1.2165 S".

CASE

CASE CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: -- case-sys)

Mark the start of the CASE ... OF ... ENDOF ... ENDCASE structure.

Append the run-time semantics given below to the current definition.

Run-time: (--)

Continue execution.

See: 6.2.1342 ENDCASE, 6.2.1343 ENDOF, 6.2.1950 OF.

COMPILE,

COMPILE,	"compile-comma"	CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: (xt --)

Append the execution semantics of the definition represented by xt to the execution semantics of the current definition.

CONVERT

CONVERT	(ud1 c-addr1 -- ud2 c-addr2)	CORE EXT

ud2 is the result of converting the characters within the text beginning at the first character after c-addr1 into digits, using the number in BASE, and adding each digit to ud1 after multiplying ud1 by the number in BASE. Conversion continues until a character that is not convertible is encountered. c-addr2 is the location of the first unconverted character. An ambiguous condition exists if ud2 overflows.

Note: This word is obsolescent and is included as a concession to existing implementations. Its function is superseded by 6.1.0570 >NUMBER.

See: 3.2.1.2 Digit conversion.

ENDCASE

&6.2.1342 ENDCASE "end-case" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: case-sys --)

Mark the end of the CASE ... OF ... ENDOF ... ENDCASE structure. Use case-sys to resolve the entire structure. Append the run-time semantics given below to the current definition.

Run-time: (x --)

Discard the case selector x and continue execution.

See: 6.2.0873 CASE, 6.2.1343 ENDOF, 6.2.1950 OF.

ENDOF

&6.2.1343 ENDOF "end-of" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: case-sys1 of-sys -- case-sys2)

Mark the end of the OF ... ENDOF part of the CASE structure. The next location for a transfer of control resolves the reference given by of-sys. Append the run-time semantics given below to the current definition. Replace case-sys1 with case-sys2 on the control-flow stack, to be resolved by ENDCASE.

Run-time: (--)

Continue execution at the location specified by the consumer of case-sys2.

See: 6.2.0873 CASE, 6.2.1342 ENDCASE, 6.2.1950 OF.

ERASE

ERASE (addr u --) CORE EXT

If u is greater than zero, clear all bits in each of u consecutive address units of memory beginning at addr .

Memory

EXPECT

EXPECT (c-addr +n --) CORE EXT

Receive a string of at most +n characters. Display graphic characters as they are received. A program that depends on the presence or absence of non-graphic characters in the string has an environmental dependency. The editing functions, if any, that the system performs in order to construct the string of characters are implementation-defined. Input terminates when an implementation-defined line terminator is received or when the string is +n characters long. When input terminates, nothing is appended to the string and the display is maintained in an implementation-defined way.

Store the string at c-addr and its length in SPAN.

Note: This word is obsolescent and is included as a concession to existing implementations. Its function is superseded by 6.1.0695 ACCEPT.

FALSE

FALSE	(-- false)	CORE EXT

Return a false flag.

See: 3.1.3.1 Flags

ArithLog

HEX

&6.2.1660 HEX	(--)	CORE EXT

Set contents of BASE to sixteen.

NumConv

MARKER

&6.2.1850 MARKER CORE EXT

("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below.

name Execution: (--)

Restore all dictionary allocation and search order pointers to the state they had just prior to the definition of name. Remove the definition of name and all subsequent definitions. Restoration of any structures still existing that could refer to deleted definitions or deallocated data space is not necessarily provided. No other contextual information such as numeric base is affected.

See: 3.4.1 Parsing, 15.6.2.1580 FORGET.

NIP

NIP (x1 x2 -- x2) CORE EXT

Drop the first item below the top of stack.

DataStack

OF

OF CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: -- of-sys)

Put of-sys onto the control flow stack. Append the run-time semantics given below to the current definition. The semantics are incomplete until resolved by a consumer of of-sys such as ENDOF.

Run-time: (x1 x2 -- | x1)

If the two values on the stack are not equal, discard the top value and continue execution at the location specified by the consumer of of-sys, e.g., following the next ENDOF. Otherwise, discard both values and continue execution in line.

See: 6.2.0873 CASE, 6.2.1342 ENDCASE, 6.2.1343 ENDOF.

PAD

PAD	(-- c-addr)	CORE EXT

c-addr is the address of a transient region that can be used to hold data for intermediate processing.

See: 3.3.3.6 Other transient regions.

Memory

PARSE

PARSE (char "ccc<char>" -- c-addr u) CORE EXT

Parse ccc delimited by the delimiter char.

c-addr is the address (within the input buffer) and u is the length of the parsed string. If the parse area was empty, the resulting string has a zero length.

See: 3.4.1 Parsing.

PICK

&6.2.2030 PICK CORE EXT

(xu ... x1 x0 u -- xu ... x1 x0 xu)

Remove u. Copy the xu to the top of the stack. An ambiguous condition exists if there are less than u+2 items on the stack before PICK is Executed.

DataStack

QUERY

QUERY (--) CORE EXT

Make the user input device the input source. Receive input into the terminal input buffer, replacing any previous contents. Make the result, whose address is returned by TIB, the input buffer. Set >IN to zero.

Note: This word is obsolescent and is included as a concession to existing implementations.

REFILL

REFILL (-- flag) CORE EXT

Attempt to fill the input buffer from the input source, returning a true flag if successful.
When the input source is the user input device, attempt to receive input into the terminal input buffer. If successful, make the result the input buffer, set >IN to zero, and return true. Receipt of a line containing no characters is considered successful. If there is no input available from the current input source, return false.
When the input source is a string from EVALUATE, return false and perform no other action.

See: 7.6.2.2125 REFILL, 11.6.2.2125 REFILL.

RESTORE-INPUT

RESTORE-INPUT (xn ... x1 n -- flag) CORE EXT

Attempt to restore the input source specification to the state described by x1 through xn. flag is true if the input source specification cannot be so restored.

An ambiguous condition exists if the input source represented by the arguments is not the same as the current input source.
See: A.6.2.2182 SAVE-INPUT.

ROLL

ROLL (xu xu-1 … x0 u -- xu-1 … x0 xu) CORE EXT

Remove u. Rotate u+1 items on the top of the stack. An ambiguous condition exists if there are less than u+2 items on the stack before ROLL is executed.

DataStack

SAVE-INPUT

SAVE-INPUT (-- xn … x1 n) CORE EXT

x1 through xn describe the current state of the input source specification for later use by RESTORE-INPUT.

SOURCE-ID

SOURCE-ID "source-i-d" CORE EXT

(-- 0 | -1)

Identifies the input source as follows:

```
$       SOURCE-ID    Input source
$       -----------------------------------
$              -1     String (via EVALUATE)
$               0     User input device
$       -----------------------------------
```

See: 11.6.1.2218 SOURCE-ID.

SPAN

SPAN (-- a-addr) CORE EXT

a-addr is the address of a cell containing the count of characters stored by the last execution of EXPECT.

Note: This word is obsolescent and is included as a concession to existing implementations.

InStream

TIB

TIB (-- c-addr) "t-i-b" CORE EXT

c-addr is the address of the terminal input buffer.

Note: This word is obsolescent and is included as a concession to existing implementations.

TO

TO CORE EXT

Interpretation: (x "<spaces>name" --)

Skip leading spaces and parse name delimited by a space. Store x in name. An ambiguous condition exists if name was not defined by VALUE.

Compilation: ("<spaces>name" --)

Skip leading spaces and parse name delimited by a space. Append the run-time semantics given below to the current definition. An ambiguous condition exists if name was not defined by VALUE.

Run-time: (x --)

Store x in name.

Note: An ambiguous condition exists if either POSTPONE or [COMPILE] is applied to TO.

See: 6.2.2405 VALUE, 13.6.1.2295 TO.

TRUE

TRUE (-- true) CORE EXT

Return a true flag, a single-cell value with all bits set.

See: 3.1.3.1 Flags.

ArithLog

TUCK

TUCK (x1 x2 -- x2 x1 x2) CORE EXT

Copy the first (top) stack item below the second stack item.

Group: DataStack

U.R

U.R (u n --) "u-dot-r" CORE EXT

Display u right aligned in a field n characters wide. If the number of characters required to display u is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

U>

U> (u1 u2 -- flag) "u-greater-than" CORE EXT

flag is true if and only if u1 is greater than u2.

See: 6.1.0540 >.

UNUSED

UNUSED (-- u) CORE EXT

u is the amount of space remaining in the region addressed by HERE, in address units.

VALUE

&6.2.2405 VALUE CORE EXT

(x "<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below, with an initial value equal to x.

name is referred to as a "value".

name Execution: (-- x)

Place x on the stack. The value of x is that given when name was created, until the phrase x TO name is executed, causing a new value of x to be associated with name.

See: 3.4.1 Parsing.

WITHIN

&6.2.2440 WITHIN CORE EXT

(n1|u1 n2|u2 n3|u3 -- flag)

Perform a comparison of a test value n1|u1 with a lower limit n2|u2 and an upper limit n3|u3, returning true if either (n2|u2 < n3|u3 and (n2|u2 <= n1|u1 and n1|u1 < n3|u3)) or (n2|u2 > n3|u3 and (n2|u2 <= n1|u1 or n1|u1 < n3|u3)) is true, returning false otherwise. An ambiguous condition exists if n1|u1, n2|u2, and n3|u3 are not all the same type.

[COMPILE]

&6.2.2530 [COMPILE] "bracket-compile" CORE EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name. If name has other than default compilation semantics, append them to the current definition; otherwise append the execution semantics of name. An ambiguous condition exists if name is not found.

See: 3.4.1 Parsing.

\

&6.2.2535 \ "backslash" CORE EXT

Compilation: Perform the execution semantics given below.

Execution: ("ccc<eol>"--)

Parse and discard the remainder of the parse area. \ is an immediate word.

See: 7.6.2.2535 \.

Comments

BLK

BLK (-- a-addr) "b-l-k" BLOCK

a-addr is the address of a cell containing zero or the number of the mass-storage block being interpreted. If BLK contains zero, the input source is not a block and can be identified by SOURCE-ID, if SOURCE-ID is available. An ambiguous condition exists if a program directly alters the contents of BLK.

See: 7.3.3 Block buffer regions.

BLOCK

&7.6.1.0800 BLOCK (u -- a-addr) BLOCK

a-addr is the address of the first character of the block buffer assigned to mass-storage block u. An ambiguous condition exists if u is not an available block number.

If block u is already in a block buffer, a-addr is the address of that block buffer.

If block u is not already in memory and there is an unassigned block buffer, transfer block u from mass storage to an unassigned block buffer. a-addr is the address of that block buffer.
If block u is not already in memory and there are no unassigned block buffers, unassign a block buffer. If the block in that buffer has been UPDATEd, transfer the block to mass storage and transfer block u from mass storage into that buffer. a-addr is the address of that block buffer.

At the conclusion of the operation, the block buffer pointed to by a-addr is the current block buffer and is assigned to u.

BUFFER

BUFFER (u -- a-addr) BLOCK

a-addr is the address of the first character of the block buffer assigned to block u. The contents of the block are unspecified. An ambiguous condition exists if u is not an available block number.
If block u is already in a block buffer, a-addr is the address of that block buffer.

If block u is not already in memory and there is an unassigned buffer, a-addr is the address of that block buffer.

If block u is not already in memory and there are no unassigned block buffers, unassign a block buffer. If the block in that buffer has been

UPDATEd, transfer the block to mass storage. a-addr is the address of that block buffer.

At the conclusion of the operation, the block buffer pointed to by a-addr is the current block buffer and is assigned to u.

See: 7.6.1.0800 BLOCK.

EVALUATE

&7.6.1.1360 EVALUATE BLOCK

Extend the semantics of 6.1.1360 EVALUATE to include:
Store zero in BLK.

FLUSH

&7.6.1.1559 FLUSH BLOCK

(--)
Perform the function of SAVE-BUFFERS, then unassign all block buffers.

LOAD

&7.6.1.1790 LOAD BLOCK

(i*x u -- j*x)

Save the current input-source specification. Store u in BLK (thus making block u the input source and setting the input buffer to encompass its contents), set >IN to zero, and interpret. When the parse area is exhausted, restore the prior input source specification. Other stack effects are due to the words LOADed.

An ambiguous condition exists if u is zero or is not a valid block number.

See: 3.4 The Forth text interpreter.

SAVE-BUFFERS

&7.6.1.2180   SAVE-BUFFERS   BLOCK

(--)

Transfer the contents of each UPDATEd block buffer to mass storage. Mark all buffers as unmodified.

UPDATE

&7.6.1.2400   UPDATE   BLOCK

(--)

Mark the current block buffer as modified. An ambiguous condition exists if there is no current block buffer.
UPDATE does not immediately cause I/O.

See: 7.6.1.0800 BLOCK, 7.6.1.0820 BUFFER, 7.6.1.1559 FLUSH, 7.6.1.2180 SAVE-BUFFERS.

EMPTY-BUFFERS

&7.6.2.1330   EMPTY-BUFFERS   BLOCK EXT

(--)

Unassign all block buffers. Do not transfer the contents of any UPDATEd block buffer to mass storage.

See: 7.6.1.0800 BLOCK.

LIST

&7.6.2.1770   LIST   (u --)   BLOCK EXT

Display block u in an implementation-defined format. Store u in SCR.

See: 7.6.1.0800 BLOCK.

REFILL

&7.6.2.2125   REFILL   (-- flag)   BLOCK EXT

Extend the execution semantics of 6.2.2125 REFILL with the following: When the input source is a block, make the next block the input source and current input buffer by adding one to the value of BLK and setting >IN to zero. Return true if the new value of BLK is a valid block number, otherwise false.

See: 6.2.2125 REFILL, 11.6.2.2125 REFILL.

SCR

&7.6.2.2190   SCR   "s-c-r"   BLOCK EXT

(-- a-addr)

a-addr is the address of a cell containing the block number of the block most recently LISTed.

THRU

&7.6.2.2280   THRU   BLOCK EXT

(i*x u1 u2 -- j*x)

LOAD the mass storage blocks numbered u1 through u2 in sequence. Other stack effects are due to the words LOADed.

\

&7.6.2.2535   \   "backslash"   BLOCK EXT

Extend the semantics of 6.2.2535 \ to be:

Compilation: Perform the execution semantics given below.

Execution: ("ccc<eol>"--)

If BLK contains zero, parse and discard the remainder of the parse area; otherwise parse and discard the portion of the parse area corresponding to the remainder of the current line. \ is an immediate word.

2CONSTANT

&8.6.1.0360   2CONSTANT   "two-constant"   DOUBLE

(x1 x2 "<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. name is referred to as a "two-constant".

name Execution: (-- x1 x2)

Place cell pair x1 x2 on the stack.

See: 3.4.1 Parsing.

2LITERAL

&8.6.1.0390   2LITERAL   "two-literal"   DOUBLE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (x1 x2 --)

Append the run-time semantics below to the current definition.

Run-time: (-- x1 x2)

Place cell pair x1 x2 on the stack.

2VARIABLE

&8.6.1.0440   2VARIABLE   "two-variable"   DOUBLE

("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Reserve two consecutive cells of data space.

name is referred to as a "two-variable".
name Execution: (-- a-addr)

a-addr is the address of the first (lowest address) cell of two consecutive cells in data space reserved by 2VARIABLE when it defined name. A program is responsible for initializing the contents.

See: 3.4.1 Parsing, 6.1.2410 VARIABLE.

D+

&8.6.1.1040   D+   "d-plus"   DOUBLE

(d1|ud1 d2|ud2 -- d3|ud3)

Add d2|ud2 to d1|ud1, giving the sum d3|ud3.

Group: ArithLog

D-

&8.6.1.1050   D-   "d-minus"   DOUBLE

(d1|ud1 d2|ud2 -- d3|ud3)

Subtract d2|ud2 from d1|ud1, giving the difference d3|ud3.

ArithLog

D.

&8.6.1.1060 D. "d-dot" DOUBLE

(d --)

Display d in free field format.

D.R

&8.6.1.1070 D.R "d-dot-r" DOUBLE

(d n --)

Display d right aligned in a field n characters wide. If the number of characters required to display d is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

D0<

&8.6.1.1075 D0< "d-zero-less" DOUBLE

(d -- flag)

flag is true if and only if d is less than zero.

D0=

&8.6.1.1080   D0=   "d-zero-equals"   DOUBLE

(xd -- flag)

flag is true if and only if xd is equal to zero.

D2*

&8.6.1.1090   D2*   "d-two-star"   DOUBLE

(xd1 -- xd2)

xd2 is the result of shifting xd1 one bit toward the most-significant bit, filling the vacated least-significant bit with zero.

Group: ArithLog

D2/

&8.6.1.1100   D2/   "d-two-slash"   DOUBLE

(xd1 -- xd2)

xd2 is the result of shifting xd1 one bit toward the least-significant bit, leaving the most-significant bit unchanged.

Group: ArithLog

D<

D<   (d1 d2 -- flag)   "d-less-than"   DOUBLE

flag is true if and only if d1 is less than d2.

D=

&8.6.1.1120 D= "d-equals" DOUBLE

(xd1 xd2 -- flag)

flag is true if and only if xd1 is bit-for-bit the same as xd2.

D>S

&8.6.1.1140 D>S (d -- n) "d-to-s" DOUBLE

n is the equivalent of d. An ambiguous condition exists if d lies outside the range of a signed single-cell number.

DABS

&8.6.1.1160 DABS (d -- ud) "d-abs" DOUBLE

ud is the absolute value of d.

Group: ArithLog

DMAX

DMAX (d1 d2 -- d3) "d-max" DOUBLE

d3 is the greater of d1 and d2.

ArithLog

DMIN

&8.6.1.1220   DMIN   "d-min"   DOUBLE

(d1 d2 -- d3)

d3 is the lesser of d1 and d2.

ArithLog

DNEGATE

&8.6.1.1230   DNEGATE   "d-negate"   DOUBLE

(d1 -- d2)

d2 is the negation of d1.

ArithLog

M*/

&8.6.1.1820   M*/   "m-star-slash"   DOUBLE

(d1 n1 +n2 -- d2)

Multiply d1 by n1 producing the triple-cell intermediate result t. Divide t by +n2 giving the double-cell quotient d2. An ambiguous condition exists if +n2 is zero or negative, or the quotient lies outside of the range of a double-precision signed integer.

ArithLog

M+

&8.6.1.1830 M+ "m-plus" DOUBLE

(d1|ud1 n -- d2|ud2)

Add n to d1|ud1, giving the sum d2|ud2.

ArithLog

2ROT

&8.6.2.0420 2ROT "two-rote" DOUBLE EXT

(x1 x2 x3 x4 x5 x6 -- x3 x4 x5 x6 x1 x2)

Rotate the top three cell pairs on the stack bringing cell pair x1 x2 to the top of the stack.

Group: DataStack

DU<

&8.6.2.1270 DU< "d-u-less" DOUBLE EXT

(ud1 ud2 -- flag)

flag is true if and only if ud1 is less than ud2.

CATCH

&9.6.1.0875   CATCH   EXCEPTION

(i*x xt -- j*x 0 | i*x n)

Push an exception frame on the exception stack and then execute the execution token xt (as with EXECUTE) in such a way that control can be transferred to a point just after CATCH if THROW is executed during the execution of xt.

If the execution of xt completes normally (i.e., the exception frame pushed by this CATCH is not popped by an execution of THROW) pop the exception frame and return zero on top of the data stack, above whatever stack items would have been returned by xt EXECUTE. Otherwise, the remainder of the execution semantics are given by THROW.

THROW

&9.6.1.2275   THROW   EXCEPTION

(k*x n -- k*x | i*x n)

If any bits of n are non-zero, pop the topmost exception frame from the exception stack, along with everything on the return stack above that frame. Then restore the input source specification in use before the corresponding CATCH and adjust the depths of all stacks defined by this Standard so that they are the same as the depths saved in the exception frame (i is the same number as the i in the input arguments to the corresponding CATCH), put n on top of the data stack, and transfer control to a point just after the CATCH that pushed that exception frame.

If the top of the stack is non zero and there is no exception frame on the exception stack, the behavior is as follows:

> If n is minus-one (-1), perform the function of 6.1.0670 ABORT (the version of ABORT in the Core word set), displaying no message.
> If n is minus-two, perform the function of 6.1.0680 ABORT" (the version of ABORT" in the Core word set), displaying the characters ccc associated with the ABORT" that generated the THROW.
> Otherwise, the system may display an implementation-dependent message giving information about the condition associated with the THROW code n. Subsequently, the system shall perform the function of 6.1.0670 ABORT (the version of ABORT in the Core word set).

ABORT

&9.6.2.0670   ABORT   EXCEPTION EXT

Extend the semantics of 6.1.0670 ABORT to be:

(i*x --) (R: j*x --)

Perform the function of -1 THROW .

See: 6.1.0670 ABORT.

ABORT"

&9.6.2.0680   ABORT"   "abort-quote"   EXCEPTION EXT

Extend the semantics of 6.1.0680 ABORT" to be:

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by a " (double-quote). Append the run-time semantics given below to the current definition.

Run-time: (i*x x1 -- | i*x) (R: j*x -- | j*x)

Remove x1 from the stack. If any bit of x1 is not zero, perform the function of -2 THROW, displaying ccc if there is no exception frame on the exception stack.

See: 3.4.1 Parsing, 6.1.0680 ABORT".

AT-XY

AT-XY   (u1 u2 --)   "at-x-y"   FACILITY

Perform implementation-dependent steps so that the next character displayed will appear in column u1, row u2 of the user output device, the upper left corner of which is column zero, row zero. An ambiguous condition exists if the operation cannot be performed on the user output device with the specified parameters.

KEY?

KEY? (-- flag) "key-question" FACILITY

If a character is available, return true. Otherwise, return false. If non-character keyboard events are available before the first valid character, they are discarded and are subsequently unavailable. The character shall be returned by the next execution of KEY.
After KEY? returns with a value of true, subsequent executions of KEY? prior to the execution of KEY or EKEY also return true, without discarding keyboard events.

PAGE

&10.6.1.2005 PAGE FACILITY

(--)

Move to another page for output. Actual function depends on the output device. On a terminal, PAGE clears the screen and resets the cursor position to the upper left corner. On a printer, PAGE performs a form feed.

EKEY

EKEY (-- u) "e-key" FACILITY EXT

Receive one keyboard event u. The encoding of keyboard events is implementation defined.

See: 10.6.1.1755 KEY?, 6.1.1750 KEY.

EKEY>CHAR

&10.6.2.1306 EKEY>CHAR "e-key-to-char" FACILITY EXT

(u -- u false | char true)

If the keyboard event u corresponds to a character in the implementation-defined character set, return that character and true. Otherwise return u and false.

EKEY?

&10.6.2.1307 EKEY? "e-key-question" FACILITY EXT

(-- flag)

If a keyboard event is available, return true. Otherwise return false. The event shall be returned by the next execution of EKEY. After EKEY? returns with a value of true, subsequent executions of EKEY? prior to the execution of KEY, KEY? or EKEY also return true, referring to the same event.

EMIT?

&10.6.2.1325 EMIT? "emit-question" FACILITY EXT

(-- flag)

flag is true if the user output device is ready to accept data and the execution of EMIT in place of EMIT? would not have suffered an indefinite delay. If the device status is indeterminate, flag is true.

MS

MS (u --) FACILITY EXT

Wait at least u milliseconds.

Note: The actual length and variability of the time period depends upon the implementation-defined resolution of the system clock and upon other system and computer characteristics beyond the scope of this Standard.

TIME&DATE

&10.6.2.2292 TIME&DATE "time-and-date" FACILITY EXT

(-- +n1 +n2 +n3 +n4 +n5 +n6)

Return the current time and date. +n1 is the second {0...59}, +n2 is the minute {0...59}, +n3 is the hour {0...23}, +n4 is the day {1...31} +n5 is the month {1...12}, and +n6 is the year (e.g., 1991).

(

&11.6.1.0080 ("paren" FILE

("ccc<paren>" --)

Extend the semantics of 6.1.0080 (to include:

When parsing from a text file, if the end of the parse area is reached before a right parenthesis is found, refill the input buffer from the next line of the file, set >IN to zero, and resume parsing, repeating this process until either a right parenthesis is found or the end of the file is reached.

BIN

&11.6.1.0765   BIN   (fam1 -- fam2)   FILE

Modify the implementation-defined file access method fam1 to additionally select a "binary", i.e., not line oriented, file access method, giving access method fam2.

See: 11.6.1.2054 R/O, 11.6.1.2056 R/W, 11.6.1.2425 W/O.

CLOSE-FILE

CLOSE-FILE   (fileid -- ior)   FILE

Close the file identified by fileid. ior is the implementation-defined I/O result code.

CREATE-FILE

&11.6.1.1010   CREATE-FILE   FILE

(c-addr u fam -- fileid ior)

Create the file named in the character string specified by c-addr and u, and open it with file access method fam. The meaning of values of fam is implementation defined. If a file with the same name already exists, recreate it as an empty file.

If the file was successfully created and opened, ior is zero, fileid is its identifier, and the file has been positioned to the start of the file.

Otherwise, ior is the implementation-defined I/O result code and fileid is undefined.

DELETE-FILE

&11.6.1.1190 DELETE-FILE FILE

(c-addr u -- ior)

Delete the file named in the character string specified by c-addr u.

ior is the implementation-defined I/O result code.

FILE-POSITION

&11.6.1.1520 FILE-POSITION FILE

(fileid -- ud ior)

ud is the current file position for the file identified by fileid. ior is the implementation-defined I/O result code. ud is undefined if ior is non-zero.

FILE-SIZE

&11.6.1.1522 FILE-SIZE FILE

(fileid -- ud ior)

ud is the size, in characters, of the file identified by fileid. ior is the implementation-defined I/O result code. This operation does not affect the value returned by FILE-POSITION. ud is undefined if ior is non-zero.

INCLUDE-FILE

&11.6.1.1717 INCLUDE-FILE FILE

(i*x fileid -- j*x)

Remove fileid from the stack. Save the current input source specification, including the current value of SOURCE-ID. Store fileid in SOURCE-ID. Make the file specified by fileid the input source. Store zero in BLK. Other stack effects are due to the words INCLUDEd. Repeat until end of file: read a line from the file, fill the input buffer from the contents of that line, set >IN to zero, and interpret. Text interpretation begins at the file position where the next file read would occur.

When the end of the file is reached, close the file and restore the input source specification to its saved value.

An ambiguous condition exists if fileid is invalid, if there is an I/O exception reading fileid, or if an I/O exception occurs while closing fileid. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation-defined.

See: 11.3.4 Input source.

INCLUDED

&11.6.1.1718   INCLUDED   FILE

(i*x c-addr u -- j*x)

Remove c-addr u from the stack. Save the current input source specification, including the current value of SOURCE-ID. Open the file specified by c-addr u, store the resulting fileid in SOURCE-ID, and make it the input source. Store zero in BLK. Other stack effects are due to the words included.

Repeat until end of file: read a line from the file, fill the input buffer from the contents of that line, set >IN to zero, and interpret. Text interpretation begins at the file position where the next file read would occur.

When the end of the file is reached, close the file and restore the input source specification to its saved value.

An ambiguous condition exists if the named file can not be opened, if an I/O exception occurs reading the file, or if an I/O exception occurs while closing the file. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation-defined.

See: 11.6.1.1717 INCLUDE-FILE.

OPEN-FILE

&11.6.1.1970 OPEN-FILE FILE

(c-addr u fam -- fileid ior)

Open the file named in the character string specified by c-addr u, with file access method indicated by fam. The meaning of values of fam is implementation defined.

If the file is successfully opened, ior is zero, fileid is its identifier, and the file has been positioned to the start of the file. Otherwise, ior is the implementation-defined I/O result code and fileid is undefined.

R/O

&11.6.1.2054 R/O "r-o" FILE

(-- fam)

fam is the implementation-defined value for selecting the "read only" file access method.

See: 11.6.1.1010 CREATE-FILE, 11.6.1.1970 OPEN-FILE.

R/W

&11.6.1.2056 R/W (-- fam) "r-w" FILE

fam is the implementation-defined value for selecting the "read/write" file access method.

See: 11.6.1.1010 CREATE-FILE, 11.6.1.1970 OPEN-FILE.

READ-FILE

&11.6.1.2080 READ-FILE FILE

(c-addr u1 fileid -- u2 ior)

Read u1 consecutive characters to c-addr from the current position of the file identified by fileid.

If u1 characters are read without an exception, ior is zero and u2 is equal to u1.

If the end of the file is reached before u1 characters are read, ior is zero and u2 is the number of characters actually read.
If the operation is initiated when the value returned by FILE-POSITION is equal to the value returned by FILE-SIZE for the file identified by fileid, ior is zero and u2 is zero.

If an exception occurs, ior is the implementation-defined I/O result code, and u2 is the number of characters transferred to c-addr without an exception.

An ambiguous condition exists if the operation is initiated when the value returned by FILE-POSITION is greater than the value returned by FILE-SIZE for the file identified by fileid, or if the requested operation attempts to read portions of the file not written.

At the conclusion of the operation, FILE-POSITION returns the next file position after the last character read.

READ-LINE

&11.6.1.2090 READ-LINE FILE

(c-addr u1 fileid -- u2 flag ior)

Read the next line from the file specified by fileid into memory at the address c-addr. At most u1 characters are read. Up to two implementation-defined line-terminating characters may be read into memory at the end of the line, but are not included in the count u2. The line buffer provided by c-addr should be at least u1+2 characters long.

If the operation succeeded, flag is true and ior is zero. If a line terminator was received before u1 characters were read, then u2 is the number of characters, not including the line terminator, actually read (0 <= u2 <= u1). When u1 = u2, the line terminator has yet to be reached.

If the operation is initiated when the value returned by FILE-POSITION is equal to the value returned by FILE-SIZE for the file identified by fileid, flag is false, ior is zero, and u2 is zero. If ior is non-zero, an exception occurred during the operation and ior is the implementation-defined I/O result code.

An ambiguous condition exists if the operation is initiated when the value returned by FILE-POSITION is greater than the value returned by FILE-SIZE for the file identified by fileid, or if the requested operation attempts to read portions of the file not written.
At the conclusion of the operation, FILE-POSITION returns the next file position after the last character read.

Request for clarification:

11.6.1.2090 READ-LINE

(c-addr u1 fileid -- u2 flag ior)

Q: What is the meaning of "flag"? The rationale suggests that it is an end-of-file flag (or not-end-of-file from the behavior documented in the definition), but this is not clear from the definition.

A: In the following excerpt from the definition of READ-LINE , the word "if" means "If and only if".

If the operation is initiated when the value returned by FILE-POSITION is equal to the value returned by FILE-SIZE for the file identified by fileid, flag is false ...

Q: What happens if the end of the file is encountered while reading a line? The definition states that "If the operation succeeded, flag is true and ior is zero", but success is not defined. I assume that success is either that u1 characters were read or that a line terminator was encountered.

A: "success" means that one of the following situations occurred:

1.Some non-terminator characters were read into the buffer or

2.A line with a valid terminator sequence, but no characters other than the terminator sequence, was read. In this case, u2=0, flag=true, and ior=zero.

Specifically, if the last line in the file is non-empty, but has no terminator, an attempt to read that line will "succeed", returning the number of characters thus read, and flag will be true. The next read, assuming that no intervening REPOSITION-FILE occurs, will return u2=0, flag=false, ior=false.

Here is complete list of return value combinations and their meanings:

```
>
> u2    flag     ior      Meaning
> --    ----     ---      -------
> X     X        nonzero Something bad and unexpected happened
>                (end-of-file is not "unexpected")
>
> 0     false   zero      End-of-file; no characters were read
>
> 0     true     zero     A blank line was read
```

```
>
> 0<u2<u1 true       zero    The entire line was read
>
> u1     true     zero    A partial line was read; the rest would
>                 not fit in the buffer, and can be acquired
>                 by additional calls to READ-LINE.
```

Suggested resolution: flag is true unless the end-of-file is encountered, in which case it is false. ior does not reflect whether the end-of-file was encountered or not.

REPOSITION-FILE

&11.6.1.2142   REPOSITION-FILE   FILE

(ud fileid -- ior)

Reposition the file identified by fileid to ud. ior is the implementation-defined I/O result code. An ambiguous condition exists if the file is positioned outside the file boundaries.
At the conclusion of the operation, FILE-POSITION returns the value ud.

RESIZE-FILE

&11.6.1.2147   RESIZE-FILE   FILE

(ud fileid -- ior)

Set the size of the file identified by fileid to ud. ior is the implementation-defined I/O result code.

If the resultant file is larger than the file before the operation, the portion of the file added as a result of the operation might not have been written.

At the conclusion of the operation, FILE-SIZE returns the value ud and FILE-POSITION returns an unspecified value.

See: 11.6.1.2080 READ-FILE, 11.6.1.2090 READ-LINE.

S"

&11.6.1.2165   S"   "s-quote"   FILE

Extend the semantics of 6.1.2165 S" to be:

Interpretation: ("ccc<quote>" -- c-addr u)

Parse ccc delimited by " (double quote). Store the resulting string c-addr u at a temporary location. The maximum length of the temporary buffer is implementation-dependent but shall be no less than 80 characters. Subsequent uses of S" may overwrite the temporary buffer. At least one such buffer shall be provided.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by " (double quote). Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr u)

Return c-addr and u that describe a string consisting of the characters ccc.

See: 3.4.1 Parsing, 6.2.0855 C", 6.1.2165 S", 11.3.5 Other transient regions.

SOURCE-ID

&11.6.1.2218   SOURCE-ID   "source-i-d"   FILE

(-- 0 | -1 | fileid)

Extend 6.2.2218 SOURCE-ID to include text-file input as follows:

$ SOURCE-ID Input source
$ -------------------------------
$ fileid   Text file "fileid"
$ -1   String (via EVALUATE)
$ 0   User input device
$ -------------------------------

An ambiguous condition exists if SOURCE-ID is used when BLK contains a non-zero value.

W/O

&11.6.1.2425   W/O   "w-o"   FILE

(-- fam)

fam is the implementation-defined value for selecting the "write only" file access method.

See: 11.6.1.1010 CREATE-FILE, 11.6.1.1970 OPEN-FILE.

WRITE-FILE

&11.6.1.2480 WRITE-FILE FILE

(c-addr u fileid -- ior)

Write u characters from c-addr to the file identified by fileid starting at its current position. ior is the implementation-defined I/O result code.

At the conclusion of the operation, FILE-POSITION returns the next file position after the last character written to the file, and FILE-SIZE returns a value greater than or equal to the value returned by FILE-POSITION.

See: 11.6.1.2080 READ-FILE, 11.6.1.2090 READ-LINE.

WRITE-LINE

&11.6.1.2485 WRITE-LINE FILE

(c-addr u fileid -- ior)

Write u characters from c-addr followed by the implementation-dependent line terminator to the file identified by fileid starting at its current position. ior is the implementation-defined I/O result code.

At the conclusion of the operation, FILE-POSITION returns the next file position after the last character written to the file, and FILE-SIZE returns a value greater than or equal to the value returned by FILE-POSITION.

See: 11.6.1.2080 READ-FILE, 11.6.1.2090 READ-LINE.

FILE-STATUS

&11.6.2.1524 FILE-STATUS FILE EXT

(c-addr u -- x ior)

Return the status of the file identified by the character string c-addr u. If the file exists, ior is zero; otherwise ior is the implementation-defined I/O result code. x contains implementation-defined information about the file.

FLUSH-FILE

&11.6.2.1560 FLUSH-FILE FILE EXT

(fileid -- ior)

Attempt to force any buffered information written to the file referred to by fileid to be written to mass storage, and the size information for the file to be recorded in the storage directory if changed. If the operation is successful, ior is zero. Otherwise, it is an implementation-defined I/O result code.

REFILL

&11.6.2.2125   REFILL   (-- flag)   FILE EXT

Extend the execution semantics of 6.2.2125 REFILL with the following: When the input source is a text file, attempt to read the next line from the text-input file. If successful, make the result the current input buffer, set >IN to zero, and return true. Otherwise return false.

See: 6.2.2125 REFILL, 7.6.2.2125 REFILL.

RENAME-FILE

&11.6.2.2130   RENAME-FILE   FILE EXT

(c-addr1 u1 c-addr2 u2 -- ior)

Rename the file named by the character string c-addr1 u1 to the name in the character string c-addr2 u2. ior is the implementation-defined I/O result code.

(LOCAL)

&13.6.1.0086   (LOCAL)   "paren-local-paren"   LOCAL

Interpretation: Interpretation semantics for this word are undefined.

Execution: (c-addr u --)

When executed during compilation, (LOCAL) passes a message to the system that has one of two meanings. If u is non-zero, the message identifies a new local whose definition name is given by the string of characters identified by c-addr u. If u is zero, the message is "last local" and c-addr has no significance.

The result of executing (LOCAL) during compilation of a definition is to create a set of named local identifiers, each of which is a definition name, that only have execution semantics within the scope of that definition's source.

local Execution: (-- x)

Push the local's value, x, onto the stack. The local's value is initialized as described in 13.3.3 Processing locals and may be changed by preceding the local's name with TO. An ambiguous condition exists when local is executed while in interpretation state.

Note: This word does not have special compilation semantics in the usual sense because it provides access to a system capability for use by other user-defined words that do have them. However, the locals facility as a whole and the sequence of messages passed defines specific usage rules with semantic implications that are described in detail in section 13.3.3 Processing locals.

Note: This word is not intended for direct use in a definition to declare that definition's locals. It is instead used by system or user compiling words. These compiling words in turn define their own syntax, and may be used directly in definitions to declare locals. In this context, the syntax for (LOCAL) is defined in terms of a sequence of compile-time messages and is described in detail in section 13.3.3 Processing locals.

Note: The Locals word set modifies the syntax and semantics of 6.2.2295 TO as defined in the Core Extensions word set.

See: 3.4 The Forth text interpreter.

TO

&13.6.1.2295 TO LOCAL

Extend the semantics of 6.2.2295 TO to be:

Interpretation: (x "<spaces>name" --)

Skip leading spaces and parse name delimited by a space. Store x in name. An ambiguous condition exists if name was not defined by VALUE.

Compilation: ("<spaces>name" --)

Skip leading spaces and parse name delimited by a space. Append the run-time semantics given below to the current definition. An ambiguous condition exists if name was not defined by either VALUE or (LOCAL).

Run-time: (x --)

Store x in name.

Note: An ambiguous condition exists if either POSTPONE or [COMPILE] is applied to TO.

See: 3.4.1 Parsing, 6.2.2295 TO, 6.2.2405 VALUE, 13.6.1.0086 (LOCAL).

LOCALS|

&13.6.2.1795 LOCALS| "locals-bar" LOCAL EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("<spaces>name1" "<spaces>name2" ... "<spaces>namen" "|" --)

Create up to eight local identifiers by repeatedly skipping leading spaces, parsing name, and executing 13.6.1.0086 (LOCAL). The list of locals to be defined is terminated by |. Append the run-time semantics given below to the current definition.

Run-time: (xn ... x2 x1 --)

Initialize up to eight local identifiers as described in 13.6.1.0086 (LOCAL), each of which takes as its initial value the top stack item, removing it from the stack. Identifier name1 is initialized with x1, identifier name2 with x2, etc. When invoked, each local will return its value. The value of a local may be changed using 13.6.1.2295 TO.

ALLOCATE

&14.6.1.0707 ALLOCATE MEMORY

(u -- a-addr ior)

Allocate u address units of contiguous data space. The data-space pointer is unaffected by this operation. The initial content of the allocated space is undefined.

If the allocation succeeds, a-addr is the aligned starting address of the allocated space and ior is zero.

If the operation fails, a-addr does not represent a valid address and ior is the implementation-defined I/O result code.

See: 6.1.1650 HERE, 14.6.1.1605 FREE, 14.6.1.2145 RESIZE.

FREE

&14.6.1.1605 FREE (a-addr -- ior) MEMORY

Return the contiguous region of data space indicated by a-addr to the system for later allocation. a-addr shall indicate a region of data space that was previously obtained by ALLOCATE or RESIZE. The data-space pointer is unaffected by this operation.

If the operation succeeds, ior is zero. If the operation fails, ior is the implementation-defined I/O result code.

See: 6.1.1650 HERE, 14.6.1.0707 ALLOCATE, 14.6.1.2145 RESIZE.

RESIZE

&14.6.1.2145 RESIZE MEMORY

(a-addr1 u -- a-addr2 ior)

Change the allocation of the contiguous data space starting at the address a-addr1, previously allocated by ALLOCATE or RESIZE, to u address units. u may be either larger or smaller than the current size of the region. The data-space pointer is unaffected by this operation. If the operation succeeds, a-addr2 is the aligned starting address of u address units of allocated memory and ior is zero. a-addr2 may be, but need not be, the same as a-addr1. If they are not the same, the values contained in the region at a-addr1 are copied to a-addr2, up to the minimum size of either of the two regions. If they are the same, the values contained in the region are preserved to the minimum of u or the original size. If a-addr2 is not the same as a-addr1, the region of memory at a-addr1 is returned to the system according to the operation of FREE.

If the operation fails, a-addr2 equals a-addr1, the region of memory at a-addr1 is unaffected, and ior is the implementation-defined I/O result code.

See: 6.1.1650 HERE, 14.6.1.0707 ALLOCATE, 14.6.1.1605 FREE.

.S

&15.6.1.0220   .S   "dot-s"   TOOLS

(--)

Copy and display the values currently on the data stack. The format of the display is implementation-dependent.

.S may be implemented using pictured numeric output words. Consequently, its use may corrupt the transient region identified by #>.

See: 3.3.3.6 Other transient regions.

?

&15.6.1.0600   ?   "question"   TOOLS

(a-addr --)

Display the value stored at a-addr.

? may be implemented using pictured numeric output words. Consequently, its use may corrupt the transient region identified by #>.

See: 3.3.3.6 Other transient regions.

DUMP

&15.6.1.1280   DUMP   TOOLS

(addr u --)

Display the contents of u consecutive addresses starting at addr. The format of the display is implementation dependent.

DUMP may be implemented using pictured numeric output words. Consequently, its use may corrupt the transient region identified by #>.

See: 3.3.3.6 Other Transient Regions.

SEE

&15.6.1.2194 SEE TOOLS

("<spaces>name" --)

Display a human-readable representation of the named word's definition.

The source of the representation (object-code decompilation, source block, etc.) and the particular form of the display is implementation defined.
SEE may be implemented using pictured numeric output words.
Consequently, its use may corrupt the transient region identified by #>.

See: 3.3.3.6 Other transient regions.

WORDS

&15.6.1.2465 WORDS TOOLS

(--)

List the definition names in the first word list of the search order.
The format of the display is implementation-dependent.
WORDS may be implemented using pictured numeric output words.
Consequently, its use may corrupt the transient region identified by #>.

See: 3.3.3.6 Other Transient Regions.

;CODE

&15.6.2.0470 ;CODE "semicolon-code" TOOLS EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: colon-sys --)

Append the run-time semantics below to the current definition. End the current definition, allow it to be found in the dictionary, and enter interpretation state, consuming colon-sys.
Subsequent characters in the parse area typically represent source code in a programming language, usually some form of assembly language. Those characters are processed in an implementation-defined manner, generating the corresponding machine code. The process continues, refilling the input buffer as needed, until an implementation-defined ending sequence is processed.

Run-time: (--) (R: nest-sys --)

Replace the execution semantics of the most recent definition with the name execution semantics given below. Return control to the calling definition specified by nest-sys. An ambiguous condition exists if the most recent definition was not defined with CREATE or a user-defined word that calls CREATE.

name Execution: (i*x -- j*x)
Perform the machine code sequence that was generated following ;CODE.
See: 6.1.1250 DOES>.

AHEAD

&15.6.2.0702 AHEAD TOOLS EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (C: -- orig)

Put the location of a new unresolved forward reference orig onto the control flow stack. Append the run-time semantics given below to the current definition. The semantics are incomplete until orig is resolved (e.g., by THEN).

Run-time: (--)

Continue execution at the location specified by the resolution of orig.

ASSEMBLER

&15.6.2.0740 ASSEMBLER TOOLS EXT

(--)

Replace the first word list in the search order with the ASSEMBLER word list.

See: 16. The optional Search-Order word set.

***g: StartEnd

BYE

&15.6.2.0830 BYE TOOLS EXT

(--)

Return control to the host operating system, if any.

CODE

&15.6.2.0930 CODE TOOLS EXT

("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name, called a "code definition", with the execution semantics defined below.

Subsequent characters in the parse area typically represent source code in a programming language, usually some form of assembly language. Those characters are processed in an implementation-defined manner, generating the corresponding machine code. The process continues, refilling the input buffer as needed, until an implementation-defined ending sequence is processed.

name Execution: (i*x -- j*x)

Execute the machine code sequence that was generated following CODE.
See: 3.4.1 Parsing.

CS-PICK

&15.6.2.1015 CS-PICK "c-s-pick" TOOLS EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: (C: destu ... orig0|dest0 -- destu ... orig0|dest0 destu)

(S: u --)

Remove u. Copy destu to the top of the control-flow stack. An ambiguous condition exists if there are less than u+1 items, each of which shall be an orig or dest, on the control-flow stack before CS-PICK is executed.

If the control-flow stack is implemented using the data stack, u shall be the topmost item on the data stack.

CS-ROLL

&15.6.2.1020 CS-ROLL "c-s-roll" TOOLS EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: (C: origu|destu origu-1|destu-1 ... orig0|dest0 -- origu-1|destu-1 ... orig0|dest0 origu|destu)

(S: u --)

Remove u. Rotate u+1 elements on top of the control-flow stack so that origu|destu is on top of the control-flow stack. An ambiguous condition

exists if there are less than u+1 items, each of which shall be an orig or dest, on the control-flow stack before CS-ROLL is executed.
If the control-flow stack is implemented using the data stack, u shall be the topmost item on the data stack.

EDITOR

&15.6.2.1300 EDITOR TOOLS EXT

(--)

Replace the first word list in the search order with the EDITOR word list.

See: 16. The Optional Search-Order Word Set.

FORGET

&15.6.2.1580 FORGET TOOLS EXT

("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name, then delete name from the dictionary along with all words added to the dictionary after name. An ambiguous condition exists if name cannot be found.

If the Search-Order word set is present, FORGET searches the compilation word list. An ambiguous condition exists if the compilation word list is deleted.

An ambiguous condition exists if FORGET removes a word required for correct execution.

Note: This word is obsolescent and is included as a concession to existing implementations.

See: 3.4.1 Parsing.

STATE

&15.6.2.2250   STATE   TOOLS EXT

(-- a-addr)

Extend the semantics of 6.1.2250 STATE to allow ;CODE to change the value in STATE. A program shall not directly alter the contents of STATE.

See: 3.4 The Forth text interpreter, 6.1.0450 :, 6.1.0460 ;, 6.1.0670 ABORT, 6.1.2050 QUIT, 6.1.2250 STATE, 6.1.2500 [, 6.1.2540], 6.2.0455 :NONAME, 15.6.2.0470 ;CODE.

[ELSE]

&15.6.2.2531   [ELSE]   "bracket-else"   TOOLS EXT

Compilation: Perform the execution semantics given below.

Execution: ("<spaces>name ... " --)

Skipping leading spaces, parse and discard space-delimited words from the parse area, including nested occurrences of [IF] ... [THEN] and [IF] ... [ELSE] ... [THEN], until the word [THEN] has been parsed and discarded. If the parse area becomes exhausted, it is refilled as with REFILL. [ELSE] is an immediate word.

See: 3.4.1 Parsing.

[IF]

&15.6.2.2532 [IF] "bracket-if" TOOLS EXT

Compilation: Perform the execution semantics given below.

Execution: (flag | flag "<spaces>name ... " --)

If flag is true, do nothing. Otherwise, skipping leading spaces, parse and discard space-delimited words from the parse area, including nested occurrences of [IF] ... [THEN] and [IF] ... [ELSE] ... [THEN], until either the word [ELSE] or the word [THEN] has been parsed and discarded. If the parse area becomes exhausted, it is refilled as with REFILL. [IF] is an immediate word.

An ambiguous condition exists if [IF] is POSTPONEd, or if the end of the input buffer is reached and cannot be refilled before the terminating [ELSE] or [THEN] is parsed.

See: 3.4.1 Parsing.

[THEN]

&15.6.2.2533 [THEN] "bracket-then" TOOLS EXT

Compilation: Perform the execution semantics given below.

Execution: (--)

Does nothing. [THEN] is an immediate word.

DEFINITIONS

&16.6.1.1180 DEFINITIONS SEARCH

(--)

Make the compilation word list the same as the first word list in the search order. Specifies that the names of subsequent definitions will be placed in the compilation word list. Subsequent changes in the search order will not affect the compilation word list.

See: 16.3.3 Finding Definition Names.

FIND

&16.6.1.1550 FIND SEARCH

Extend the semantics of 6.1.1550 FIND to be:

(c-addr -- c-addr 0 | xt 1 | xt -1)

Find the definition named in the counted string at c-addr. If the definition is not found after searching all the word lists in the search order, return c-addr and zero. If the definition is found, return xt.

If the definition is immediate, also return one (1); otherwise also return minus-one (-1). For a given string, the values returned by FIND while compiling may differ from those returned while not compiling.

See: 3.4.2 Finding definition names, 6.1.0070 ', 6.1.1550 FIND, 6.1.2033 POSTPONE, 6.1.2510 ['], D.6.7 Immediacy.

FORTH-WORDLIST

&16.6.1.1595   FORTH-WORDLIST   SEARCH

(-- wid)

Return wid, the identifier of the word list that includes all standard words provided by the implementation. This word list is initially the compilation word list and is part of the initial search order.

GET-CURRENT

&16.6.1.1643   GET-CURRENT   SEARCH

(-- wid)

Return wid, the identifier of the compilation word list.

GET-ORDER

&16.6.1.1647 GET-ORDER SEARCH

(-- widn ... wid1 n)

Returns the number of word lists n in the search order and the word list identifiers widn ... wid1 identifying these word lists. wid1 identifies the word list that is searched first, and widn the word list that is searched last. The search order is unaffected.

SEARCH-WORDLIST

&16.6.1.2192 SEARCH-WORDLIST SEARCH

(c-addr u wid -- 0 | xt 1 | xt -1)

Find the definition identified by the string c-addr u in the word list identified by wid. If the definition is not found, return zero. If the definition is found, return its execution token xt and one (1) if the definition is immediate, minus-one (-1) otherwise.

Group: Execution

SET-CURRENT

&16.6.1.2195 SET-CURRENT SEARCH

(wid --)

Set the compilation word list to the word list identified by wid.

SET-ORDER

&16.6.1.2197 SET-ORDER SEARCH

(widn ... wid1 n --)

Set the search order to the word lists identified by widn ... wid1. Subsequently, word list wid1 will be searched first, and word list widn searched last. If n is zero, empty the search order. If n is minus one, set the search order to the implementation-defined minimum search order. The minimum search order shall include the words FORTH-WORDLIST and SET-ORDER. A system shall allow n to be at least eight.

WORDLIST

&16.6.1.2460 WORDLIST SEARCH

(-- wid)

Create a new empty word list, returning its word list identifier wid. The new word list may be returned from a pool of preallocated word lists or may be dynamically allocated in data space. A system shall allow the creation of at least 8 new word lists in addition to any provided as part of the system.

ALSO

&16.6.2.0715 ALSO SEARCH EXT

(--)

Transform the search order consisting of widn, ... wid2, wid1 (where wid1 is searched first) into widn, ... wid2, wid1, wid1. An ambiguous condition exists if there are too many word lists in the search order.

FORTH

&16.6.2.1590 FORTH SEARCH EXT

(--)

Transform the search order consisting of widn, ... wid2, wid1 (where wid1 is searched first) into widn, ... wid2, widFORTH-WORDLIST.

ONLY

&16.6.2.1965 ONLY SEARCH EXT

(--)

Set the search order to the implementation-defined minimum search order. The minimum search order shall include the words FORTH-WORDLIST and SET-ORDER.

ORDER

&16.6.2.1985 ORDER (--) SEARCH EXT

Display the word lists in the search order in their search order sequence, from first searched to last searched. Also display the word list into which new definitions will be placed. The display format is implementation dependent.

ORDER may be implemented using pictured numeric output words. Consequently, its use may corrupt the transient region identified by #>.

See: 3.3.3.6 Other Transient Regions.

*** PREVIOUS

&16.6.2.2037 PREVIOUS (--) SEARCH EXT

Transform the search order consisting of widn, ... wid2, wid1 (where wid1 is searched first) into widn, ... wid2. An ambiguous condition exists if the search order was empty before PREVIOUS was executed.

*** -TRAILING

&17.6.1.0170 -TRAILING "dash-trailing" STRING

(c-addr u1 -- c-addr u2)

If u1 is greater than zero, u2 is equal to u1 less the number of spaces at the end of the character string specified by c-addr u1. If u1 is zero or the entire string consists of spaces, u2 is zero.

/STRING

&17.6.1.0245 /STRING "slash-string" STRING

(c-addr1 u1 n -- c-addr2 u2)

Adjust the character string at c-addr1 by n characters. The resulting character string, specified by c-addr2 u2, begins at c-addr1 plus n characters and is u1 minus n characters long.

BLANK

&17.6.1.0780 BLANK (c-addr u --) STRING

If u is greater than zero, store the character value for space in u consecutive character positions beginning at c-addr.

CMOVE

CMOVE (c-addr1 c-addr2 u --) "c-move" STRING

If u is greater than zero, copy u consecutive characters from the data space starting at c-addr1 to that starting at c-addr2, proceeding character-by-character from lower addresses to higher addresses.

Contrast with: 17.6.1.0920 CMOVE>.

CMOVE>

&17.6.1.0920   CMOVE>   "c-move-up"   STRING

(c-addr1 c-addr2 u --)

If u is greater than zero, copy u consecutive characters from the data space starting at c-addr1 to that starting at c-addr2, proceeding character-by-character from higher addresses to lower addresses.

Contrast with: 17.6.1.0910 CMOVE.

COMPARE

&17.6.1.0935   COMPARE   STRING

(c-addr1 u1 c-addr2 u2 -- n)

Compare the string specified by c-addr1 u1 to the string specified by c-addr2 u2. The strings are compared, beginning at the given addresses, character by character, up to the length of the shorter string or until a difference is found. If the two strings are identical, n is zero. If the two strings are identical up to the length of the shorter string, n is minus-one (-1) if u1 is less than u2 and one (1) otherwise. If the two strings are not identical up to the length of the shorter string, n is minus-one (-1) if the first non-matching character in the string specified by c-addr1 u1 has a lesser numeric value than the corresponding character in the string specified by c-addr2 u2 and one (1) otherwise.

SEARCH

&17.6.1.2191 SEARCH STRING

(c-addr1 u1 c-addr2 u2 -- c-addr3 u3 flag)

Search the string specified by c-addr1 u1 for the string specified by c-addr2 u2. If flag is true, a match was found at c-addr3 with u3 characters remaining. If flag is false there was no match and c-addr3 is c-addr1 and u3 is u1.

SLITERAL

&17.6.1.2212 SLITERAL STRING

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (c-addr1 u --)

Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr2 u)

Return c-addr2 u describing a string consisting of the characters specified by c-addr1 u during compilation. A program shall not alter the returned string.

>FLOAT

to-float FLOATING

(c-addr u -- true | false) (F: -- r |)
or (c-addr u -- r true | false)

An attempt is made to convert the string specified by c-addr and u to internal floating-point representation. If the string represents a valid floating-point number in the syntax below, its value r and true are returned. If the string does not represent a valid floating-point number only false is returned.

A string of blanks should be treated as a special case representing zero.

The syntax of a convertible string := <significand>[<exponent>]

```
<significand> := [<sign>]{<digits>[.<digits0>] |
.<digits> }
<exponent>    := <marker><digits0>
<marker>      := {<e-form> | <sign-form>}
<e-form>      := <e-char>[<sign-form>]
<sign-form>   := { + | - }
<e-char>      := { D | d | E | e }
```

>FLOAT enables programs to read floating-point data in legible ASCII format. It accepts a much broader syntax than does the text interpreter since the latter defines rules for composing source programs whereas >FLOAT defines rules for accepting data. >FLOAT is defined as broadly as is feasible to permit input of data from ANS Forth systems as well

as other widely used standard programming environments.

This is a synthesis of common FORTRAN practice. Embedded spaces are explicitly forbidden in much scientific usage, as are other field separators such as comma or slash.

While >FLOAT is not required to treat a string of blanks as zero, this behavior is strongly encouraged, since a future version of ANS Forth may include such a requirement.

D>F

d-to-f FLOATING

(d --) (F: -- r)
or (d -- r)

r is the floating-point equivalent of d. An ambiguous condition exists if d cannot be precisely represented as a floating-point value.

DF!

d-f-store FLOATING EXT

(df-addr --) (F: r --)
or (r df-addr --)

Store the floating-point number r as a 64-bit IEEE double-precision number at df-addr. If the significand of the internal representation of r has more precision than the IEEE double-precision format, it will be rounded using the round to nearest rule. An ambiguous condition exists if the exponent of r is too large to be accommodated in IEEE double-precision format.

DF@

12.6.2.1204 DF@

d-f-fetch FLOATING EXT

(df-addr --) (F: -- r)

or (df-addr -- r)

Fetch the 64-bit IEEE double-precision number stored at df-addr to the floating-point stack as r in the internal representation.
If the IEEE double-precision significand has more precision than the internal representation it will be rounded to the internal representation using the round to nearest rule. An ambiguous condition exists if the exponent of the IEEE double-precision representation is too large to be accommodated by the internal representation.

*** FROUND

f-round FLOATING

(F: r1 -- r2)

or (r1 -- r2)

Round r1 to an integral value using the round to nearest rule, giving r2.

*** DF!

*** DF@

*** FROUND

*** FLOOR

Round to nearest means round the result of a floating-point operation to the representable value nearest the result. If the two nearest representable values are equally near the result, the one having zero as its least significant bit shall be delivered.
Round toward negative infinity means round the result of a floating-point operation to the representable value nearest to and no greater than the result

*** DFALIGN

d-f-align FLOATING EXT

(--)

If the data-space pointer is not double-float aligned, reserve enough data space to make it so.

*** DFALIGNED

d-f-aligned FLOATING EXT

(addr -- df-addr)

df-addr is the first double-float-aligned address greater than or equal to addr.

DFLOAT+

d-float-plus (df-addr1 -- df-addr2) FLOATING EXT

Add the size in address units of a 64-bit IEEE double-precision number to df-addr1, giving df-addr2

*** DFALIGN

*** DFALIGNED

*** **DFLOAT+**

The set of float-aligned addresses is an implementation-defined subset of the set of aligned addresses. Adding the size of a floating-point number to a float-aligned address shall produce a float-aligned address.

The set of double-float-aligned addresses is an implementation-defined subset of the set of aligned addresses. Adding the size of a 64-bit IEEE double-precision floating-point number to a double-float-aligned address shall produce a double-float-aligned address.

The set of single-float-aligned addresses is an implementation-defined subset of the set of aligned addresses. Adding the size of a 32-bit IEEE single-precision floating-point number to a single-float-aligned address shall produce a single-float-aligned address.

DFLOATS

D-floats (n1 -- n2) FLOATING EXT

n2 is the size in address units of n1 64-bit IEEE double-precision numbers.

F!

f-store FLOATING

(f-addr --) (F: r --)
or (r f-addr --)

Store r at f-addr.

F*

f-star FLOATING

(F: r1 r2 -- r3)
or (r1 r2 -- r3)

Multiply r1 by r2 giving r3.

F+

f-plus FLOATING

(F: r1 r2 -- r3)
or (r1 r2 -- r3)

Add r1 to r2 giving the sum r3.

F-

f-minus FLOATING

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

Subtract r2 from r1, giving r3.

F/

f-slash FLOATING

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

Divide r1 by r2, giving the quotient r3. An ambiguous condition exists if r2 is zero, or the quotient lies outside of the range of a floating-point number.

F0<

f-zero-less-than FLOATING

(-- flag) (F: r --)

or (r -- flag)

flag is true if and only if r is less than zero.

F0=

F-zero-equals FLOATING

(-- flag) (F: r --)

or (r -- flag)

flag is true if and only if r is equal to zero.

F<

F-less-than FLOATING

(-- flag) (F: r1 r2 --)

or (r1 r2 -- flag)

flag is true

if and only if r1 is less than

F**

F-star-star FLOATING EXT

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

Raise r1 to the power r2, giving the product r3.

F.

f-dot FLOATING EXT

(--) (F: r --)

or (r --)

Display, with a trailing space, the top number on the floating-point stack using fixed-point notation:

[-] <digits>.<digits0>

An ambiguous condition exists if the value of BASE is not (decimal) ten or if the character string representation exceeds the size of the pictured numeric output string buffer.

For example, 1E3 F. displays 1000. .

F>D

f-to-d FLOATING

(-- d) (F: r --)

or (r -- d)

d is the double-cell signed-integer equivalent of the integer portion of r. The fractional portion of r is discarded.
An ambiguous condition exists if the integer portion of r cannot be precisely represented as a double-cell signed integer.

F@

f-fetch FLOATING

(f-addr --) (F: -- r)

or (f-addr -- r)

r is the value stored at f-addr.

*** FABS f-abs FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the absolute value of r1.

*** FACOS

f-a-cos FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the principal radian angle whose cosine is r1.

An ambiguous condition exists if |r1| is greater than one.

FACOSH

F-a-cosh FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the floating-point value whose hyperbolic cosine is r1.

An ambiguous condition exists if r1 is less than one.

FALOG

f-a-log FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

Raise ten to the power r1, giving r2.

FASIN

F-a-sine FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the principal radian angle whose sine is r1.

An ambiguous condition exists if |r1| is greater than one.

FASINH

F-a-cinch FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the floating-point value whose hyperbolic sine is r1.

An ambiguous condition exists if r1 is less than zero.

*** FATAN

f-a-tan FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the principal radian angle whose tangent is r1.

FATAN2

f-a-tan-two FLOATING EXT

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

r3 is the radian angle whose tangent is r1/r2.
An ambiguous condition exists if r1 and r2 are zero.

FSINCOS and FATAN2 are a complementary pair of operators which convert angles to 2-vectors and vice-versa. They are essential to most geometric and physical applications since they correctly and unambiguously handle this conversion in all cases except null vectors, even when the tangent of the angle would be infinite.

FSINCOS returns a Cartesian unit vector in the direction of the given angle, measured counter-clockwise from the positive X-axis.
The order of results on the stack, namely y underneath x, permits the 2-vector data type to be additionally viewed and used as a ratio approximating the tangent of the angle. Thus the phrase FSINCOS F/ is functionally equivalent to FTAN, but is useful over only a limited and discontinuous range of angles, whereas FSINCOS

and FATAN2 are useful for all angles. This ordering has been found convenient for nearly two decades, and has the added benefit of being easy to remember. A corollary to this observation is that vectors in general should appear on the stack in this order.

The argument order for FATAN2 is the same, converting a vector in the conventional representation to a scalar angle. Thus, for all angles, FSINCOS FATAN2 is an identity within the accuracy of the arithmetic and the argument range of FSINCOS. Note that while FSINCOS always returns a valid unit vector, FATAN2 will accept any non-null vector. An ambiguous condition exists if the vector argument to FATAN2 has zero magnitude.

FATANH

f-a-tan-h FLOATING EXT

(F: r1 -- r2)
or (r1 -- r2)

r2 is the floating-point value whose hyperbolic tangent is r1.
An ambiguous condition exists if r1 is outside the range of -1E0 to 1E0.

FCOS

f-cos FLOATING EXT

(F: r1 -- r2)
or (r1 -- r2)

r2 is the cosine of the radian angle r1.

FCOSH

f-cosh FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the hyperbolic cosine of r1.

FALIGN

(--) f-align FLOATING

If the data-space pointer is not float aligned, reserve enough data space to make it so.

FALIGNED

f-aligned (addr -- f-addr) FLOATING

f-addr is the first float-aligned address greater than or equal to addr.

FCONSTANT

f-constant FLOATING

("<spaces>name" --) (F: r --)

or (r "<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space.

Create a definition for name with the execution semantics defined below.

name is referred to as an f-constant.

name Execution: (--) (F: -- r)

or (-- r)

Place r on the floating-point stack.

Typical use: r FCONSTANT name

FDEPTH

F-depth (-- +n) FLOATING

+n is the number of values contained on the default separate floating-point stack. If floating-point numbers are kept on the data stack, +n is the current number of possible floating-point values contained on the data stack.

FDROP

f-drop FLOATING

(F: r --)

or (r --)

Remove r from the floating-point stack.

FDUP

F-dupe FLOATING

(F: r -- r r)

or (r -- r r)

Duplicate r.

FLITERAL

f-literal FLOATING

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (F: r --)

or (r --)

Append the run-time semantics given below to the current definition.

Run-time: (F: -- r)

or (-- r)

Place r on the floating-point stack.

See: A.12.6.1.1552 FLITERAL

FLOAT+ (f-addr1 -- f-addr2) “FLOAT-plus” FLOATING

Add the size in address units of a floating-point number to f-addr1, giving f-addr2.

FLOATS (n1 -- n2) FLOATING

n2 is the size in address units of n1 floating-point numbers.

FLOOR (F: r1 -- r2) FLOATING

or (r1 -- r2)

Round r1 to an integral value using the round toward negative infinity rule, giving r2.

FMAX

F-max FLOATING

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

r3 is the greater of r1 and r2.

FMIN

F-min FLOATING

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

r3 is the lesser of r1 and r2.

FNEGATE

f-negate FLOATING

(F: r1 -- r2)

or (r1 -- r2)

r2 is the negation of r1.

FOVER

f-over FLOATING

(F: r1 r2 -- r1 r2 r1)

or (r1 r2 -- r1 r2 r1)

Place a copy of r1 on top of the floating-point stack.

FROT

f-rote FLOATING

(F: r1 r2 r3 -- r2 r3 r1)

or (r1 r2 r3 -- r2 r3 r1)

Rotate the top three floating-point stack entries.

FSWAP

F-swap FLOATING

(F: r1 r2 -- r2 r1)

or (r1 r2 -- r2 r1)

Exchange the top two floating-point stack items.

FVARIABLE

f-variable FLOATING

("<spaces>name" --)

Skip leading space delimiters. Parse name delimited by a space.

Create a definition for name with the execution semantics

defined below. Reserve 1 FLOATS address units of data space at a float-aligned address.

name is referred to as an f-variable.

name Execution: (-- f-addr)

f-addr is the address of the data space reserved by FVARIABLE when it created name. A program is responsible for initializing the contents of the reserved space.

Typical use: FVARIABLE name

REPRESENT

FLOATING

(c-addr u -- n flag1 flag2) (F: r --)
or (r c-addr u -- n flag1 flag2)

At c-addr, place the character-string external representation of the significand of the floating-point number r. Return the decimal-base exponent as n, the sign as flag1 and valid result as flag2.
The character string shall consist of the u most significant digits of the significand represented as a decimal fraction with the implied decimal point to the left of the first digit, and the first digit zero only if all digits are zero. The significand is rounded to u digits following the round to nearest rule; n is adjusted, if necessary, to correspond to the rounded magnitude of the significand. If flag2 is true then r was in the implementation-defined range of floating-point numbers.
If flag1 is true then r is negative.

An ambiguous condition exists if the value of BASE is not decimal ten.

When flag2 is false, n and flag1 are implementation defined, as are the contents of c-addr. Under these circumstances, the string at c-addr shall consist of graphic characters.

This word provides a primitive for floating-point display. Some floating-point formats, including those specified by IEEE-754, allow representations of numbers outside of an implementation-defined range. These include plus and minus infinities, denormalized numbers, and others. In these cases we expect that REPRESENT will usually be implemented to return appropriate character strings, such as +infinity or nan, possibly truncated.

FE.

F-e-dot FLOATING EXT

(--) (F: r --)
or (r --)

Display, with a trailing space, the top number on the floating-point stack using engineering notation, where the significand is greater than or equal to 1.0 and less than 1000.0 and the decimal exponent is a multiple of three.

An ambiguous condition exists if the value of BASE is not (decimal) ten or if the character string representation exceeds the size of the pictured numeric output string buffer.

FEXP

f-e-x-p FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

Raise e to the power r1, giving r2.

FEXPM1

f-e-x-p-m-one FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

Raise e to the power r1 and subtract one, giving r2.

This function allows accurate computation when its arguments are close to zero, and provides a useful base for the standard exponential functions. Hyperbolic functions such as cosh(x) can be efficiently and accurately implemented by using FEXPM1; accuracy is lost in this function for small values of x if the word FEXP is used.

An important application of this word is in finance; say a loan is repaid at 15% per year; what is the daily rate? On a computer with single precision (six decimal digit) accuracy:

1. Using FLN and FEXP:

FLN of 1.15 = 0.139762, divide by 365 = 3.82910E-4, form the exponent using FEXP = 1.00038, and subtract one (1) and convert to percentage = 0.038%.

Thus we only have two digit accuracy.

2. Using FLNP1 and FEXPM1:

FLNP1 of 0.15 = 0.139762, (this is the same value as in the first example, although with the argument closer to zero it may not be so) divide by 365 = 3.82910E-4, form the exponent and subtract one (1) using FEXPM1 = 3.82983E-4, and convert to percentage = 0.0382983%.

This is full six digit accuracy.

The presence of this word allows the hyperbolic functions to be computed with usable accuracy. For example, the hyperbolic sine can be defined as:

```
: FSINH  ( r1 -- r2 )
        FEXPM1  FDUP  FDUP 1.0E0 F+  F/  F+  2.0E0 F/ ;
```

FLN

f-l-n FLOATING EXT

```
        ( F: r1 -- r2 )
        or ( r1 -- r2 )
```

r2 is the natural logarithm of r1. An ambiguous condition exists if r1 is less than or equal to zero.

FLNP1

f-l-n-p-one FLOATING EXT

```
( F: r1 -- r2 )
or ( r1 -- r2 )
```

r2 is the natural logarithm of the quantity r1 plus one. An ambiguous condition exists if r1 is less than or equal to negative one.

This function allows accurate compilation when its arguments are close to zero, and provides a useful base for the standard logarithmic functions. For example, FLN can be implemented as:

```
: FLN   1.0E0 F-  FLNP1 ;
```

FLOG

f-log FLOATING EXT

```
( F: r1 -- r2 )
or ( r1 -- r2 )
```

r2 is the base-ten logarithm of r1. An ambiguous condition exists if r1 is less than or equal to zero.

FMIN

f-min FLOATING

(F: r1 r2 -- r3)

or (r1 r2 -- r3)

r3 is the lesser of r1 and r2.

FS.

f-s-dot FLOATING EXT

(--) (F: r --)

or (r --)

Display, with a trailing space, the top number on the floating-point stack in scientific notation:

<significand><exponent>

where:

<significand> := [-]<digit>.<digits0>

<exponent> := E[-]<digits>

An ambiguous condition exists if the value of BASE is not (decimal) ten or if the character string representation exceeds the size of the pictured numeric output string buffer

FSIN

f-sine FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the sine of the radian angle r1.

FSINCOS

f-sine-cos FLOATING EXT

(F: r1 -- r2 r3)

or (r1 -- r2 r3)

r2 is the sine of the radian angle r1. r3 is the cosine of the radian angle r1.

FSINH

f-cinch FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the hyperbolic sine of r1.

FSQRT

f-square-root FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the square root of r1. An ambiguous condition exists if r1 is less than zero.

FTAN

f-tan FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the tangent of the radian angle r1. An ambiguous condition exists if cos(r1) is zero.

FTANH

f-tan-h FLOATING EXT

(F: r1 -- r2)

or (r1 -- r2)

r2 is the hyperbolic tangent of r1.

F~

f-proximate FLOATING EXT

(-- flag) (F: r1 r2 r3 --)

or (r1 r2 r3 -- flag)

If r3 is positive, flag is true if the absolute value of (r1 minus r2) is less than r3.

If r3 is zero, flag is true if the implementation-dependent encoding of r1 and r2 are exactly identical (positive and negative zero are unequal if they have distinct encodings).

If r3 is negative, flag is true if the absolute value of (r1 minus r2) is less than the absolute value of r3 times the sum of the absolute values of r1 and r2.

This provides the three types of floating point equality in common use -- close in absolute terms, exact equality as represented, and relatively close.

PRECISION (-- u) FLOATING EXT

Return the number of significant digits currently used by F., FE., or FS. as u.

SET-PRECISION (u --) FLOATING EXT

Set the number of significant digits currently used by F., FE., or FS. to u.

SF!

s-f-store FLOATING EXT

(sf-addr --) (F: r --)
or (r sf-addr --)

Store the floating-point number r as a 32-bit IEEE single-precision number at sf-addr. If the significand of the internal representation of r has more precision than the IEEE single-precision format, it will be rounded using the round to nearest rule. An ambiguous condition exists if the exponent of r is too large to be accommodated by the IEEE single-precision format.

SF@

s-f-fetch FLOATING EXT

(sf-addr --) (F: -- r)

or (sf-addr -- r)

Fetch the 32-bit IEEE single-precision number stored at sf-addr to the floating-point stack as r in the internal representation. If the IEEE single-precision significand has more precision than the internal representation, it will be rounded to the internal representation using the round to nearest rule. An ambiguous condition exists if the exponent of the IEEE single-precision representation is too large to be accommodated by the internal representation.

SFALIGN

s-f-align (--) FLOATING EXT

If the data-space pointer is not single-float aligned, reserve enough data space to make it so.

SFALIGNED

s-f-aligned (addr -- sf-addr) FLOATING EXT

sf-addr is the first single-float-aligned address greater than or equal to addr.

SFLOAT+

s-float-plus (sf-addr1 -- sf-addr2) FLOATING EXT

Add the size in address units of a 32-bit IEEE single-precision number to sf-addr1, giving sf-addr2.

*** SFLOATS

s-floats (n1 -- n2) FLOATING EXT

n2 is the size in address units of n1 32-bit IEEE single-precision numbers.