7. The optional Block word set

7.1 Introduction

7.2 Additional terms

block: 1024 characters of data on mass storage, designated by a block number.

block buffer: a block-sized region of data space where a block is made temporarily available for use. The current block buffer is the block buffer most recently accessed by **BLOCK**, **BUFFER**, **LOAD**, **LIST**, or **THRU**.

7.3 Additional usage requirements

7.3.1 Environmental queries

Append table 7.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 7.1 – Environmental Query Strings

String	Value data type	Constant?	Meaning
BLOCK	*flag*	no	block word set present
BLOCK-EXT	*flag*	no	block extensions word set present

7.3.2 Data space

A program may access memory within a valid block buffer.

See: **3.3.3 Data Space**.

7.3.3 Block buffer regions

The address of a block buffer returned by **BLOCK** or **BUFFER** is transient. A call to **BLOCK** or **BUFFER** may render a previously-obtained block-buffer address invalid, as may a call to any word that:

- parses:
- displays characters on the user output device, such as **TYPE** or **EMIT**;
- controls the user output device, such as **CR** or **AT-XY**;
- receives or tests for the presence of characters from the user input device such as **ACCEPT** or **KEY**;
- waits for a condition or event, such as **MS** or **EKEY**;
- manages the block buffers, such as **FLUSH**, **SAVE-BUFFERS**, or **EMPTY-BUFFERS**;
- performs any operation on a file or file-name directory that implies I/O, such as **REFILL** or any word that returns an *ior*;
- implicitly performs I/O, such as text interpreter nesting and un-nesting when files are being used (including un-nesting implied by **THROW**).

If the input source is a block, these restrictions also apply to the address returned by **SOURCE**.

Block buffers are uniquely assigned to blocks.

7.3.4 Parsing

The Block word set implements an alternative input source for the text interpreter. When the input source is a block, **BLK** shall contain the non-zero block number and the input buffer is the 1024-character buffer containing that block.

A block is conventionally displayed as 16 lines of 64 characters.

A program may switch the input source to a block by using **LOAD** or **THRU**. Input sources may be nested using **LOAD** and **EVALUATE** in any order.

A program may reposition the parse area within a block by manipulating **>IN**. More extensive repositioning can be accomplished using **SAVE-INPUT** and **RESTORE-INPUT**.

See: **3.4.1 Parsing**.

7.3.5 Possible action on an ambiguous condition

See: **3.4.4 Possible action on an ambiguous condition**.

– A system with the Block word set may set interpretation state and interpret a block.

7.4 Additional documentation requirements

7.4.1 System documentation

7.4.1.1 Implementation-defined options

– the format used for display by **7.6.2.1770 LIST** (if implemented);
– the length of a line affected by **7.6.2.2535 ** (if implemented).

7.4.1.2 Ambiguous conditions

– Correct block read was not possible;
– I/O exception in block transfer;
– Invalid block number (**7.6.1.0800 BLOCK**, **7.6.1.0820 BUFFER**, **7.6.1.1790 LOAD**);
* A program directly alters the contents of **7.6.1.0790 BLK**;
* No current block buffer for **7.6.1.2400 UPDATE**.

7.4.1.3 Other system documentation

– any restrictions a multiprogramming system places on the use of buffer addresses;
– the number of blocks available for source text and data.

7.4.2 Program documentation

– the number of blocks required by the program.

7.5 Compliance and labeling

7.5.1 ANS Forth systems

The phrase “Providing the Block word set” shall be appended to the label of any Standard System that provides all of the Block word set.

The phrase “Providing *name*(*s*) from the Block Extensions word set” shall be appended to the label of any Standard System that provides portions of the Block Extensions word set.

The phrase “Providing the Block Extensions word set” shall be appended to the label of any Standard System that provides all of the Block and Block Extensions word sets.

7.5.2 ANS Forth programs

The phrase "Requiring the Block word set" shall be appended to the label of Standard Programs that require the system to provide the Block word set.

The phrase "Requiring *name*(*s*) from the Block Extensions word set" shall be appended to the label of Standard Programs that require the system to provide portions of the Block Extensions word set.

The phrase "Requiring the Block Extensions word set" shall be appended to the label of Standard Programs that require the system to provide all of the Block and Block Extensions word sets.

7.6 Glossary

7.6.1 Block words

7.6.1.0790 **BLK** "b-l-k" BLOCK

(-- *a-addr*)

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**.

7.6.1.0800 **BLOCK** BLOCK

(*u* -- *a-addr*)

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 **UPDATE**d, 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*.

7.6.1.0820 **BUFFER** BLOCK

(*u* -- *a-addr*)

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 **UPDATE**d, 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**.

7.6.1.1360 **EVALUATE** BLOCK

Extend the semantics of **6.1.1360 EVALUATE** to include:

Store zero in **BLK**.

7.6.1.1559 **FLUSH** BLOCK

(--)

Perform the function of **SAVE-BUFFERS** then unassign all block buffers.

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 **LOAD**ed.

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

See: **3.4 The Forth text interpreter**.

7.6.1.2180 **SAVE-BUFFERS** BLOCK

(--)

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

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**.

7.6.2 Block extension words

7.6.2.1330 **EMPTY-BUFFERS** BLOCK EXT

(--)

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

See: **7.6.1.0800 BLOCK**.

7.6.2.1770 **LIST** BLOCK EXT

(*u* --)

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

See: **7.6.1.0800 BLOCK**.

7.6.2.2125 **REFILL** BLOCK EXT

(-- *flag*)

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**.

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 **LIST**ed.

7.6.2.2280 **THRU** BLOCK EXT

(*i*x* u_1 u_2 -- *j*x*)

LOAD the mass storage blocks numbered u_1 through u_2 in sequence. Other stack effects are due to the words **LOAD**ed.

**7.6.2.2535 ** "backslash" BLOCK EXT

Extend the semantics of **6.1.2535 ** 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.

See: **6.2.2535 **.

8. The optional Double-Number word set

8.1 Introduction

Sixteen-bit Forth systems often use double-length numbers. However, many Forths on small embedded systems do not, and many users of Forth on systems with a cell size of 32 bits or more find that the use of double-length numbers is much diminished. Therefore, the words that manipulate double-length entities have been placed in this optional word set.

8.2 Additional terms and notation

None.

8.3 Additional usage requirements

8.3.1 Environmental queries

Append table 8.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 8.1 – Environmental Query Strings

String	Value data type	Constant?	Meaning
DOUBLE	*flag*	no	double-number word set present
DOUBLE-EXT	*flag*	no	double-number extensions word set present

8.3.2 Text interpreter input number conversion

When the text interpreter processes a number that is immediately followed by a decimal point and is not found as a definition name, the text interpreter shall convert it to a double-cell number.

For example, entering **`DECIMAL 1234`** leaves the single-cell number **`1234`** on the stack, and entering **`DECIMAL 1234.`** leaves the double-cell number **`1234 0`** on the stack.

See: **3.4.1.3 Text interpreter input number conversion**.

8.4 Additional documentation requirements

8.4.1 System documentation

8.4.1.1 Implementation-defined options

– no additional requirements.

8.4.1.2 Ambiguous conditions

* *d* outside range of *n* in **8.6.1.1140 `D>S`**.

8.4.1.3 Other system documentation

– no additional requirements.

8.4.2 Program documentation

– no additional requirements.

8.5 Compliance and labeling

8.5.1 ANS Forth systems

The phrase "Providing the Double-Number word set" shall be appended to the label of any Standard System that provides all of the Double-Number word set.

The phrase "Providing *name*(*s*) from the Double-Number Extensions word set" shall be appended to the label of any Standard System that provides portions of the Double-Number Extensions word set.

The phrase "Providing the Double-Number Extensions word set" shall be appended to the label of any Standard System that provides all of the Double-Number and Double-Number Extensions word sets.

8.5.2 ANS Forth programs

The phrase "Requiring the Double-Number word set" shall be appended to the label of Standard Programs that require the system to provide the Double-Number word set.

The phrase "Requiring *name*(*s*) from the Double-Number Extensions word set" shall be appended to the label of Standard Programs that require the system to provide portions of the Double-Number Extensions word set.

The phrase "Requiring the Double-Number Extensions word set" shall be appended to the label of Standard Programs that require the system to provide all of the Double-Number and Double-Number Extensions word sets.

8.6 Glossary

8.6.1 Double-Number words

8.6.1.0360 `2CONSTANT` "two-constant" DOUBLE

(x_1 x_2 "<*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: (-- x_1 x_2)

Place cell pair x_1 x_2 on the stack.

See: **3.4.1 Parsing**.

8.6.1.0390 `2LITERAL` "two-literal" DOUBLE

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (x_1 x_2 --)

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

Run-time: (-- x_1 x_2)

Place cell pair x_1 x_2 on the stack.

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`**.

8.6.1.1040 `D+` "d-plus" DOUBLE

($d_1|ud_1$ $d_2|ud_2$ -- $d_3|ud_3$)

Add $d_2|ud_2$ to $d_1|ud_1$, giving the sum $d_3|ud_3$.

8.6.1.1050 `D-` "d-minus" DOUBLE

($d_1|ud_1$ $d_2|ud_2$ -- $d_3|ud_3$)

Subtract $d_2|ud_2$ from $d_1|ud_1$, giving the difference $d_3|ud_3$.

8.6.1.1060 `D.` "d-dot" DOUBLE

(*d* --)

Display *d* in free field format.

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.

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

(*d* -- *flag*)

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

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

(*xd* -- *flag*)

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

8.6.1.1090 D2* "d-two-star" DOUBLE

(xd_1 -- xd_2)

xd_2 is the result of shifting xd_1 one bit toward the most-significant bit, filling the vacated least-significant bit with zero.

8.6.1.1100 D2/ "d-two-slash" DOUBLE

(xd_1 -- xd_2)

xd_2 is the result of shifting xd_1 one bit toward the least-significant bit, leaving the most-significant bit unchanged.

8.6.1.1110 D< "d-less-than" DOUBLE

(d_1 d_2 -- *flag*)

flag is true if and only if d_1 is less than d_2.

8.6.1.1120 D= "d-equals" DOUBLE

(xd_1 xd_2 -- *flag*)

flag is true if and only if xd_1 is bit-for-bit the same as xd_2.

8.6.1.1140 D>S "d-to-s" DOUBLE

(*d* -- *n*)

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

8.6.1.1160 DABS "d-abs" DOUBLE

(*d* -- *ud*)

ud is the absolute value of *d*.

8.6.1.1210 DMAX "d-max" DOUBLE

(d_1 d_2 -- d_3)

d_3 is the greater of d_1 and d_2.

8.6.1.1220 DMIN "d-min" DOUBLE

(d_1 d_2 -- d_3)

d_3 is the lesser of d_1 and d_2.

8.6.1.1230 DNEGATE "d-negate" DOUBLE

(d_1 -- d_2)

d_2 is the negation of d_1.

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

(d_1 n_1 $+n_2$ -- d_2)

Multiply d_1 by n_1 producing the triple-cell intermediate result t. Divide t by $+n_2$ giving the double-cell quotient d_2. An ambiguous condition exists if $+n_2$ is zero or negative, or the quotient lies outside of the range of a double-precision signed integer.

8.6.1.1830 M+ "m-plus" DOUBLE

($d_1|ud_1$ n -- $d_2|ud_2$)

Add n to $d_1|ud_1$, giving the sum $d_2|ud_2$.

8.6.2 Double-Number extension words

8.6.2.0420 2ROT "two-rote" DOUBLE EXT

(x_1 x_2 x_3 x_4 x_5 x_6 -- x_3 x_4 x_5 x_6 x_1 x_2)

Rotate the top three cell pairs on the stack bringing cell pair x_1 x_2 to the top of the stack.

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

(ud_1 ud_2 -- *flag*)

flag is true if and only if ud_1 is less than ud_2.

9. The optional Exception word set

9.1 Introduction

9.2 Additional terms and notation

None.

9.3 Additional usage requirements

9.3.1 **THROW** values

The **THROW** values {-255...-1} shall be used only as assigned by this Standard. The values {-4095...-256} shall be used only as assigned by a system.

If the File-Access or Memory-Allocation word sets are implemented, it is recommended that the non-zero values of *ior* lie within the range of system **THROW** values, as defined above. In an operating-system environment, this can sometimes be accomplished by "biasing" the range of operating-system exception-codes to fall within the **THROW** range.

Programs shall not define values for use with **THROW** in the range {-4095...-1}.

9.3.2 Exception frame

An exception frame is the implementation-dependent set of information recording the current execution state necessary for the proper functioning of **CATCH** and **THROW**. It often includes the depths of the data stack and return stack.

9.3.3 Exception stack

A stack used for the nesting of exception frames by **CATCH** and **THROW**. It may be, but need not be, implemented using the return stack.

9.3.4 Environmental queries

Append table 9.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 9.1 – Environmental query strings

String	Value data type	Constant?	Meaning
EXCEPTION	*flag*	no	Exception word set present
EXCEPTION-EXT	*flag*	no	Exception extensions word set present

9.3.5 Possible actions on an ambiguous condition

A system choosing to execute **THROW** when detecting one of the ambiguous conditions listed in table 9.3.6 shall use the throw code listed there.

See: **3.4.4 Possible actions on an ambiguous condition**.

Table 9.2 – THROW code assignments

Code	Reserved for	Code	Reserved for
-1	**ABORT**	-3	stack overflow
-2	**ABORT"**	-4	stack underflow

-5	return stack overflow	-32	invalid name argument (e.g., **TO xxx**)
-6	return stack underflow	-33	block read exception
-7	do-loops nested too deeply during execution	-34	block write exception
-8	dictionary overflow	-35	invalid block number
-9	invalid memory address	-36	invalid file position
-10	division by zero	-37	file I/O exception
-11	result out of range	-38	non-existent file
-12	argument type mismatch	-39	unexpected end of file
-13	undefined word	-40	invalid **BASE** for floating point conversion
-14	interpreting a compile-only word	-41	loss of precision
-15	invalid **FORGET**	-42	floating-point divide by zero
-16	attempt to use zero-length string as a name	-43	floating-point result out of range
-17	pictured numeric output string overflow	-44	floating-point stack overflow
-18	parsed string overflow	-45	floating-point stack underflow
-19	definition name too long	-46	floating-point invalid argument
-20	write to a read-only location	-47	compilation word list deleted
-21	unsupported operation (e.g., **AT-XY** on a too-dumb terminal)	-48	invalid **POSTPONE**
		-49	search-order overflow
-22	control structure mismatch	-50	search-order underflow
-23	address alignment exception	-51	compilation word list changed
-24	invalid numeric argument	-52	control-flow stack overflow
-25	return stack imbalance	-53	exception stack overflow
-26	loop parameters unavailable	-54	floating-point underflow
-27	invalid recursion	-55	floating-point unidentified fault
-28	user interrupt	-56	**QUIT**
-29	compiler nesting	-57	exception in sending or receiving a character
-30	obsolescent feature		
-31	**>BODY** used on non-**CREATE**d definition	-58	**[IF]**, **[ELSE]**, or **[THEN]** exception

9.3.6 Exception handling

There are several methods of coupling **CATCH** and **THROW** to other procedural nestings. The usual nestings are the execution of definitions, use of the return stack, use of loops, instantiation of locals and nesting of input sources (i.e., with **LOAD**, **EVALUATE**, or **INCLUDE-FILE**).

When a **THROW** returns control to a **CATCH**, the system shall un-nest not only definitions, but also, if present, locals and input source specifications, to return the system to its proper state for continued execution past the **CATCH**.

9.4 Additional documentation requirements

9.4.1 System documentation

9.4.1.1 Implementation-defined options

– Values used in the system by **9.6.1.0875 CATCH** and **9.6.1.2275 THROW** (**9.3.1 THROW values**, **9.3.5 Possible actions on an ambiguous condition**).

9.4.1.2 Ambiguous conditions

– no additional requirements.

9.4.1.3 Other system documentation

– no additional requirements.

9.4.2 Program documentation

– no additional requirements.

9.5 Compliance and labeling

9.5.1 ANS Forth systems

The phrase “Providing the Exception word set” shall be appended to the label of any Standard System that provides all of the Exception word set.

The phrase “Providing *name*(*s*) from the Exception Extensions word set” shall be appended to the label of any Standard System that provides portions of the Exception Extensions word set.

The phrase “Providing the Exception Extensions word set” shall be appended to the label of any Standard System that provides all of the Exception and Exception Extensions word sets.

9.5.2 ANS Forth programs

The phrase “Requiring the Exception word set” shall be appended to the label of Standard Programs that require the system to provide the Exception word set.

The phrase “Requiring *name*(*s*) from the Exception Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the Exception Extensions word set.

The phrase “Requiring the Exception Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the Exception and Exception Extensions word sets.

9.6 Glossary

9.6.1 Exception words

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**.

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).

9.6.2 Exception extension words

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**.

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* x_1 -- | *i*x*) (R: *j*x* -- | *j*x*)

Remove x_1 from the stack. If any bit of x_1 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"**.

10. The optional Facility word set

10.1 Introduction

10.2 Additional terms and notation

None.

10.3 Additional usage requirements

10.3.1 Character types

Programs that use more than seven bits of a character by **10.6.2.1305 EKEY** have an environmental dependency.

See: **3.1.2 Character types**.

10.3.2 Environmental queries

Append table 10.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 10.1 – Environmental query strings

String	Value data type	Constant?	Meaning
FACILITY	*flag*	no	facility word set present
FACILITY-EXT	*flag*	no	facility extensions word set present

10.4 Additional documentation requirements

10.4.1 System documentation

10.4.1.1 Implementation-defined options

- encoding of keyboard events (**10.6.2.1305 EKEY**);
- duration of a system clock tick;
- repeatability to be expected from execution of **10.6.2.1905 MS**.

10.4.1.2 Ambiguous conditions

* **10.6.1.0742 AT-XY** operation can't be performed on user output device.

10.4.1.3 Other system documentation

- no additional requirements.

10.4.2 Program documentation

10.4.2.1 Environmental dependencies

- using more than seven bits of a character in **10.6.2.1305 EKEY**.

10.4.2.2 Other program documentation

- no additional requirements.

10.5 Compliance and labeling

10.5.1 ANS Forth systems

The phrase "Providing the Facility word set" shall be appended to the label of any Standard System that provides all of the Facility word set.

The phrase "Providing *name*(*s*) from the Facility Extensions word set" shall be appended to the label of any Standard System that provides portions of the Facility Extensions word set.

The phrase "Providing the Facility Extensions word set" shall be appended to the label of any Standard System that provides all of the Facility and Facility Extensions word sets.

10.5.2 ANS Forth programs

The phrase "Requiring the Facility word set" shall be appended to the label of Standard Programs that require the system to provide the Facility word set.

The phrase "Requiring *name*(*s*) from the Facility Extensions word set" shall be appended to the label of Standard Programs that require the system to provide portions of the Facility Extensions word set.

The phrase "Requiring the Facility Extensions word set" shall be appended to the label of Standard Programs that require the system to provide all of the Facility and Facility Extensions word sets.

10.6 Glossary

10.6.1 Facility words

10.6.1.0742 **AT-XY** "at-x-y" FACILITY

(u_1 u_2 --)

Perform implementation-dependent steps so that the next character displayed will appear in column u_1, row u_2 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.

10.6.1.1755 **KEY?** "key-question" FACILITY

(-- *flag*)

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.

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.

10.6.2 Facility extension words

10.6.2.1305 EKEY "e-key" FACILITY EXT

(-- *u*)

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

See: **10.6.1.1755 KEY?**, **6.1.1750 KEY**.

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*.

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

(-- *flag*)

If a keyboard event is available, returns *true*. Otherwise returns *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.

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.

10.6.2.1905 MS FACILITY EXT

(*u* --)

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.

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

(-- $+n_1$ $+n_2$ $+n_3$ $+n_4$ $+n_5$ $+n_6$)

Return the current time and date. $+n_1$ is the second {0...59}, $+n_2$ is the minute {0...59}, $+n_3$ is the hour {0...23}, $+n_4$ is the day {1...31} $+n_5$ is the month {1...12}, and $+n_6$ is the year (e.g., 1991).

11. The optional File-Access word set

11.1 Introduction

These words provide access to mass storage in the form of "files" under the following assumptions:

- files are provided by a host operating system;
- file names are represented as character strings;
- the format of file names is determined by the host operating system;
- an open file is identified by a single-cell file identifier (*fileid*);
- file-state information (e.g., position, size) is managed by the host operating system;
- file contents are accessed as a sequence of characters;
- file read operations return an actual transfer count, which can differ from the requested transfer count.

11.2 Additional terms

file-access method: a permissible means of accessing a file, such as "read/write" or "read only".

file position: the character offset from the start of the file.

input file: the file, containing a sequence of lines, that is the input source.

11.3 Additional usage requirements

11.3.1 Data types

Append table 11.1 to table 3.1.

Table 11.1 – Data types

Symbol	Data type	Size on stack
ior	I/O results	1 cell
fam	file access method	1 cell
fileid	file identifiers	1 cell

11.3.1.1 File identifiers

File identifiers are implementation-dependent single-cell values that are passed to file operators to designate specific files. Opening a file assigns a file identifier, which remains valid until closed.

11.3.1.2 I/O results

I/O results are single-cell numbers indicating the result of I/O operations. A value of zero indicates that the I/O operation completed successfully; other values and their meanings are implementation-defined. Reaching the end of a file shall be reported as zero.

An I/O exception in the execution of a File-Access word that can return an I/O result shall not cause a **THROW**; exception indications are returned in the *ior*.

11.3.1.3 File access methods

File access methods are implementation-defined single-cell values.

11.3.1.4 File names

A character string containing the name of the file. The file name may include an implementation-dependent path name. The format of file names is implementation defined.

11.3.2 Blocks in files

If the File-Access word set is implemented, the Block word set shall be implemented.

Blocks may, but need not, reside in files. When they do:

– Block numbers may be mapped to one or more files by implementation-defined means. An ambiguous condition exists if a requested block number is not currently mapped;
– An **UPDATE**d block that came from a file shall be transferred back to the same file.

11.3.3 Environmental queries

Append table 11.2 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 11.2 – Environmental query strings

String	Value data type	Constant?	Meaning
FILE	*flag*	no	file word set present
FILE-EXT	*flag*	no	file extensions word set present

11.3.4 Input source

The File-Access word set creates another input source for the text interpreter. When the input source is a text file, **BLK** shall contain zero, **SOURCE-ID** shall contain the *fileid* of that text file, and the input buffer shall contain one line of the text file.

Input with **INCLUDED**, **INCLUDE-FILE**, **LOAD** and **EVALUATE** shall be nestable in any order to at least eight levels.

A program that uses more than eight levels of input-file nesting has an environmental dependency.

See: **3.3.3.5 Input buffers**, **9. Optional Exception word set**.

11.3.5 Other transient regions

The list of words using memory in transient regions is extended to include **11.6.1.2165 S"**.

See: **3.3.3.6 Other transient regions**.

11.3.6 Parsing

When parsing from a text file using a space delimiter, control characters shall be treated the same as the space character.

Lines of at least 128 characters shall be supported. A program that requires lines of more than 128 characters has an environmental dependency.

A program may reposition the parse area within the input buffer by manipulating the contents of **>IN**. More extensive repositioning can be accomplished using **SAVE-INPUT** and **RESTORE-INPUT**.

See: **3.4.1 Parsing**.

11.4 Additional documentation requirements

11.4.1 System documentation

11.4.1.1 Implementation-defined options

- file access methods used by **11.6.1.0765 BIN**, **11.6.1.1010 CREATE-FILE**, **11.6.1.1970 OPEN-FILE**, **11.6.1.2054 R/O**, **11.6.1.2056 R/W**, and **11.6.1.2425 W/O**;
- file exceptions;
- file line terminator (**11.6.1.2090 READ-LINE**);
- file name format (**11.3.1.4 File names**);
- information returned by **11.6.2.1524 FILE-STATUS**;
- input file state after an exception (**11.6.1.1717 INCLUDE-FILE**, **11.6.1.1718 INCLUDED**);
- *ior* values and meaning (**11.3.1.2 I/O results**);
- maximum depth of file input nesting (**11.3.4 Input source**);
- maximum size of input line (**11.3.6 Parsing**);
- methods for mapping block ranges to files (**11.3.2 Blocks in files**);
- number of string buffers provided (**11.6.1.2165 S"**);
- size of string buffer used by **11.6.1.2165 S"**.

11.4.1.2 Ambiguous conditions

* attempting to position a file outside its boundaries (**11.6.1.2142 REPOSITION-FILE**);
* attempting to read from file positions not yet written (**11.6.1.2080 READ-FILE**, **11.6.1.2090 READ-LINE**);
* *fileid* is invalid (**11.6.1.1717 INCLUDE-FILE**);
* I/O exception reading or closing *fileid* (**11.6.1.1717 INCLUDE-FILE**, **11.6.1.1718 INCLUDED**);
* named file cannot be opened (**11.6.1.1718 INCLUDED**);
* requesting an unmapped block number (**11.3.2 Blocks in files**);
* using **11.6.1.2218 SOURCE-ID** when **7.6.1.0790 BLK** is not zero.

11.4.1.3 Other system documentation

- no additional requirements.

11.4.2 Program documentation

11.4.2.1 Environmental dependencies

- requiring lines longer than 128 characters (**11.3.6 Parsing**);
- using more than eight levels of input-file nesting (**11.3.4 Input source**).

11.4.2.2 Other program documentation

- no additional requirements.

11.5 Compliance and labeling

11.5.1 ANS Forth systems

The phrase “Providing the File Access word set” shall be appended to the label of any Standard System that provides all of the File Access word set.

The phrase "Providing *name*(*s*) from the File Access Extensions word set" shall be appended to the label of any Standard System that provides portions of the File Access Extensions word set.

The phrase "Providing the File Access Extensions word set" shall be appended to the label of any Standard System that provides all of the File Access and File Access Extensions word sets.

11.5.2 ANS Forth programs

The phrase "Requiring the File Access word set" shall be appended to the label of Standard Programs that require the system to provide the File Access word set.

The phrase "Requiring *name*(*s*) from the File Access Extensions word set" shall be appended to the label of Standard Programs that require the system to provide portions of the File Access Extensions word set.

The phrase "Requiring the File Access Extensions word set" shall be appended to the label of Standard Programs that require the system to provide all of the File Access and File Access Extensions word sets.

11.6 Glossary

11.6.1 File Access words

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.

11.6.1.0765 BIN FILE

(fam_1 -- fam_2)

Modify the implementation-defined file access method fam_1 to additionally select a "binary", i.e., not line oriented, file access method, giving access method fam_2.

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

11.6.1.0900 CLOSE-FILE FILE

(*fileid* -- *ior*)

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

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.

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.

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.

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.

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 **INCLUDE**d.

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**.

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**.

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.

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**.

11.6.1.2056 R/W "r-w" FILE

(-- *fam*)

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**.

11.6.1.2080 READ-FILE FILE

(*c-addr* u_1 *fileid* -- u_2 *ior*)

Read u_1 consecutive characters to *c-addr* from the current position of the file identified by *fileid*.

If u_1 characters are read without an exception, *ior* is zero and u_2 is equal to u_1.

If the end of the file is reached before u_1 characters are read, *ior* is zero and u_2 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 u_2 is zero.

If an exception occurs, *ior* is the implementation-defined I/O result code, and u_2 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.

11.6.1.2090 **READ-LINE** FILE

(*c-addr* u_1 *fileid* -- u_2 *flag ior*)

Read the next line from the file specified by *fileid* into memory at the address *c-addr*. At most u_1 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 u_2. The line buffer provided by *c-addr* should be at least u_1+2 characters long.

If the operation succeeded, *flag* is true and *ior* is zero. If a line terminator was received before u_1 characters were read, then u_2 is the number of characters, not including the line terminator, actually read ($0 <= u_2 <= u_1$). When $u_1 = u_2$ 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 u_2 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.

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*.

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**.

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**.

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.

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**.

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**.

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**.

11.6.2 File-Access extension words

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.

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.

11.6.2.2125 **REFILL** FILE EXT

(*-- flag*)

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**.

11.6.2.2130 **RENAME-FILE** FILE EXT

($c\text{-}addr_1\ u_1\ c\text{-}addr_2\ u_2$ *-- ior*)

Rename the file named by the character string $c\text{-}addr_1\ u_1$ to the name in the character string $c\text{-}addr_2\ u_2$. *ior* is the implementation-defined I/O result code.

12. The optional Floating-Point word set

12.1 Introduction

12.2 Additional terms and notation

12.2.1 Definition of terms

float-aligned address: the address of a memory location at which a floating-point number can be accessed.

double-float-aligned address: the address of a memory location at which a 64-bit IEEE double-precision number can be accessed.

single-float-aligned address: the address of a memory location at which a 32-bit IEEE single-precision number can be accessed.

IEEE floating-point number: a floating-point number as defined in **ANSI/IEEE Standard 754-1985**.

12.2.2 Notation

12.2.2.1 Numeric notation

The following notation is used to define the syntax of the external representation of floating-point numbers:

- Each component of a floating-point number is defined with a rule consisting of the name of the component (italicized in angle-brackets, e.g., <*sign*>), the characters := and a concatenation of tokens and metacharacters;
- Tokens may be literal characters (in bold face, e.g., **E**) or rule names in angle brackets (e.g., <*digit*>);
- The metacharacter * is used to specify zero or more occurrences of the preceding token (e.g., <*digit*>*);
- Tokens enclosed with [and] are optional (e.g., [<*sign*>]);
- Vertical bars separate choices from a list of tokens enclosed with braces (e.g., { **+** | **-** }).

12.2.2.2 Stack notation

Floating-point stack notation when the floating-point stack is separate from the data stack is:

(F: *before* -- *after*)

12.3 Additional usage requirements

12.3.1 Data types

Append table 12.1 to table 3.1.

Table 12.1 – Data Types

Symbol	Data type	Size on stack
r	floating-point number	implementation-defined
f-addr	float-aligned address	1 cell
sf-addr	single-float-aligned address	1 cell
df-addr	double-float-aligned address	1 cell

12.3.1.1 Addresses

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.

12.3.1.2 Floating-point numbers

The internal representation of a floating-point number, including the format and precision of the significand and the format and range of the exponent, is implementation defined.

Any rounding or truncation of floating-point numbers is implementation defined.

12.3.2 Floating-point operations

"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.

12.3.3 Floating-point stack

A last in, first out list that shall be used by all floating-point operators.

The width of the floating-point stack is implementation-defined. By default the floating-point stack shall be separate from the data and return stacks. A program may determine whether floating-point numbers are kept on the data stack by passing the string "**FLOATING-STACK**" to **ENVIRONMENT?**.

The size of a floating-point stack shall be at least 6 items.

A program that depends on the floating-point stack being larger than six items has an environmental dependency.

12.3.4 Environmental queries

Append table 12.2 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 12.2 – Environmental query strings

String	Value Data type	Constant?	Meaning
FLOATING	*flag*	no	floating-point word set present
FLOATING-EXT	*flag*	no	floating-point extensions word set present
FLOATING-STACK	*n*	yes	If *n* = zero, floating-point numbers are kept on the data stack; otherwise *n* is the maximum depth of the separate floating-point stack.
MAX-FLOAT	*r*	yes	largest usable floating-point number

12.3.5 Address alignment

Since the address returned by a **CREATE**d word is not necessarily aligned for any particular class of floating-point data, a program shall align the address (to be float aligned, single-float aligned, or double-float aligned) before accessing floating-point data at the address.

See: **3.3.3.1 Address Alignment**, **12.3.1.1 Addresses**.

12.3.6 Variables

A program may address memory in data space regions made available by **FVARIABLE**. These regions may be non-contiguous with regions subsequently allocated with **,** (comma) or **ALLOT**.

See: **3.3.3.3 Variables**.

12.3.7 Text interpreter input number conversion

If the Floating-Point word set is present in the dictionary and the current base is **DECIMAL**, the input number-conversion algorithm shall be extended to recognize floating-point numbers in this form:

Convertible string := <*significand*><*exponent*>

<*significand*> := [<*sign*>]<*digits*>[**.**<*digits0*>]
<*exponent*> := **E**[<*sign*>]<*digits0*>
<*sign*> := { **+** | **–** }
<*digits*> := <*digit*><*digits0*>
<*digits0*> := <*digit*>*
<*digit*> := { **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** }

These are examples of valid representations of floating-point numbers in program source:

```
1E   1.E   1.E0   +1.23E-1   -1.23E+1
```

See: **3.4.1.3 Text interpreter input number conversion**, **12.6.1.0558 >FLOAT**.

12.4 Additional documentation requirements

12.4.1 System documentation

12.4.1.1 Implementation-defined options

– format and range of floating-point numbers (**12.3.1 Data types**, **12.6.1.2143 REPRESENT**);
– results of **12.6.1.2143 REPRESENT** when *float* is out of range;
– rounding or truncation of floating-point numbers (**12.3.1.2 Floating-point numbers**);
– size of floating-point stack (**12.3.3 Floating-point stack**);
– width of floating-point stack (**12.3.3 Floating-point stack**).

12.4.1.2 Ambiguous conditions

– **DF@** or **DF!** is used with an address that is not double-float aligned;
– **F@** or **F!** is used with an address that is not float aligned;
– floating point result out of range (**12.6.1.1430 F/**);
– **SF@** or **SF!** is used with an address that is not single-float aligned;
* **BASE** is not decimal (**12.6.1.2143 REPRESENT**, **12.6.2.1427 F.**, **12.6.2.1513 FE.**, **12.6.2.1613 FS.**);
* both arguments equal zero (**12.6.2.1489 FATAN2**);
* cos(r_1) is zero as an argument for **12.6.2.1625 FTAN**;
* *d* can't be precisely represented as *float* in **12.6.1.1130 D>F**;
* dividing by zero (**12.6.1.14340**);

* exponent too big for conversion (**12.6.2.1203 DF!**, **12.6.2.1204 DF@**, **12.6.2.2202 SF!**, **12.6.2.2203 SF@**);
* *float* less than one (**12.6.2.1477 FACOSH**);
* *float* less than or equal to minus-one (**12.6.2.1554 FLNP1**);
* *float* less than or equal to zero (**12.6.2.1553 FLN**, **12.6.2.1557 FLOG**);
* *float* less than zero (**12.6.2.1487 FASINE**, **12.6.2.1618 FSQRT**);
* *float* magnitude greater than one (**12.6.2.1476 FACOS**, **12.6.2.1486 FASIN**, **12.6.2.1491 FATANH**);
* integer part of *float* can't be represented by *d* in **12.6.1.1470 F>D**;
* string larger than pictured-numeric output area (**12.6.2.1427 F.**, **12.6.2.1513 FE.**, **12.6.2.1613 FS.**).

12.4.1.3 Other system documentation

– no additional requirements.

12.4.2 Program documentation

12.4.2.1 Environmental dependencies

– requiring the floating-point stack to be larger than six items (**12.3.3 Floating-point stack**).

12.4.2.2 Other program documentation

– no additional requirements.

12.5 Compliance and labeling

12.5.1 ANS Forth systems

The phrase “Providing the Floating-Point word set” shall be appended to the label of any Standard System that provides all of the Floating-Point word set.

The phrase “Providing *name*(*s*) from the Floating-Point Extensions word set” shall be appended to the label of any Standard System that provides portions of the Floating-Point Extensions word set.

The phrase “Providing the Floating-Point Extensions word set” shall be appended to the label of any Standard System that provides all of the Floating-Point and Floating-Point Extensions word sets.

12.5.2 ANS Forth programs

The phrase “Requiring the Floating-Point word set” shall be appended to the label of Standard Programs that require the system to provide the Floating-Point word set.

The phrase “Requiring *name*(*s*) from the Floating-Point Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the Floating-Point Extensions word set.

The phrase “Requiring the Floating-Point Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the Floating-Point and Floating-Point Extensions word sets.

12.6 Glossary

12.6.1 Floating-Point words

12.6.1.0558 >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** }

12.6.1.1130 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.

12.6.1.1400 F! "f-store" FLOATING

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

Store *r* at *f-addr*.

12.6.1.1410 F* "f-star" FLOATING

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

Multiply r_1 by r_2 giving r_3.

12.6.1.1420 F+ "f-plus" FLOATING

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

Add r_1 to r_2 giving the sum r_3.

12.6.1.1425 F- "f-minus" FLOATING

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

Subtract r_2 from r_1, giving r_3.

12.6.1.1430 **F/** "f-slash" FLOATING

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

Divide r_1 by r_2, giving the quotient r_3. An ambiguous condition exists if r_2 is zero, or the quotient lies outside of the range of a floating-point number.

12.6.1.1440 **F0<** "f-zero-less-than" FLOATING

(-- *flag*) (F: *r* --) or (*r* -- *flag*)

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

12.6.1.1450 **F0=** "f-zero-equals" FLOATING

(-- *flag*) (F: *r* --) or (*r* -- *flag*)

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

12.6.1.1460 **F<** "f-less-than" FLOATING

(-- *flag*) (F: r_1 r_2 --) or (r_1 r_2 -- *flag*)

flag is true if and only if r_1 is less than r_2.

12.6.1.1470 **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.

12.6.1.1472 **F@** "f-fetch" FLOATING

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

r is the value stored at *f-addr*.

12.6.1.1479 **FALIGN** "f-align" FLOATING

(--)

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

12.6.1.1483 **FALIGNED** "f-aligned" FLOATING

(*addr* -- *f-addr*)

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

12.6.1.1492 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.

See: **3.4.1 Parsing**.

12.6.1.1497 FDEPTH "f-depth" FLOATING

(-- +*n*)

+*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.

12.6.1.1500 FDROP "f-drop" FLOATING

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

Remove *r* from the floating-point stack.

12.6.1.1510 FDUP "f-dupe" FLOATING

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

Duplicate *r*.

12.6.1.1552 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.

12.6.1.1555 FLOAT+ "float-plus" FLOATING

(*f-addr*$_1$ -- *f-addr*$_2$)

Add the size in address units of a floating-point number to *f-addr*$_1$, giving *f-addr*$_2$.

12.6.1.1556 FLOATS FLOATING

(n_1 -- n_2)

n_2 is the size in address units of n_1 floating-point numbers.

12.6.1.1558 FLOOR FLOATING

(F: r_1 -- r_2) or (r_1 -- r_2)

Round r_1 to an integral value using the "round toward negative infinity" rule, giving r_2.

12.6.1.1562 FMAX "f-max" FLOATING

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

r_3 is the greater of r_1 and r_2.

12.6.1.1565 FMIN "f-min" FLOATING

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

r_3 is the lesser of r_1 and r_2.

12.6.1.1567 FNEGATE "f-negate" FLOATING

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the negation of r_1.

12.6.1.1600 FOVER "f-over" FLOATING

(F: r_1 r_2 -- r_1 r_2 r_1) or (r_1 r_2 -- r_1 r_2 r_1)

Place a copy of r_1 on top of the floating-point stack.

12.6.1.1610 FROT "f-rote" FLOATING

(F: r_1 r_2 r_3 -- r_2 r_3 r_1) or (r_1 r_2 r_3 -- r_2 r_3 r_1)

Rotate the top three floating-point stack entries.

12.6.1.1612 FROUND "f-round" FLOATING

(F: r_1 -- r_2) or (r_1 -- r_2)

Round r_1 to an integral value using the "round to nearest" rule, giving r_2.

See: **12.3.2 Floating-point operations**.

12.6.1.1620 FSWAP "f-swap" FLOATING

(F: r_1 r_2 -- r_2 r_1) or (r_1 r_2 -- r_2 r_1)

Exchange the top two floating-point stack items.

12.6.1.1630 **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.

See: **3.4.1 Parsing**.

12.6.1.2143 **REPRESENT** FLOATING

(*c-addr u -- n* $flag_1$ $flag_2$) (F: *r* --) or (*r c-addr u -- n* $flag_1$ $flag_2$)

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 $flag_1$ and "valid result" as $flag_2$. 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 $flag_2$ is true then *r* was in the implementation-defined range of floating-point numbers. If $flag_1$ is true then *r* is negative.

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

When $flag_2$ is false, *n* and $flag_1$ are implementation defined, as are the contents of *c-addr*. Under these circumstances, the string at *c-addr* shall consist of graphic characters.

See: **3.2.1.2 Digit conversion**, **6.1.0750 BASE**, **12.3.2 Floating-point operations**.

12.6.2 Floating-Point extension words

12.6.2.1203 **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.

See: **12.3.1.1 Addresses**, **12.3.2 Floating-point operations**.

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.

See: **2.3.1.1 Addresses**, **12.3.2 Floating-point operations**.

12.6.2.1205 DFALIGN "d-f-align" FLOATING EXT

(--)

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

See: **12.3.1.1 Addresses**.

12.6.2.1207 DFALIGNED "d-f-aligned" FLOATING EXT

(*addr* -- *df-addr*)

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

See: **12.3.1.1 Addresses**.

12.6.2.1208 DFLOAT+ "d-float-plus" FLOATING EXT

($df\text{-}addr_1$ -- $df\text{-}addr_2$)

Add the size in address units of a 64-bit IEEE double-precision number to $df\text{-}addr_1$, giving $df\text{-}addr_2$.

See: **12.3.1.1 Addresses**.

12.6.2.1209 DFLOATS "d-floats" FLOATING EXT

(n_1 -- n_2)

n_2 is the size in address units of n_1 64-bit IEEE double-precision numbers.

12.6.2.1415 F** "f-star-star" FLOATING EXT

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

Raise r_1 to the power r_2, giving the product r_3.

12.6.2.1427 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.

See: **12.6.1.0558 >FLOAT**.

12.6.2.1474 FABS "f-abs" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the absolute value of r_1.

12.6.2.1476 FACOS "f-a-cos" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the principal radian angle whose cosine is r_1. An ambiguous condition exists if $|r_1|$ is greater than 1.

12.6.2.1477 FACOSH "f-a-cosh" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the floating-point value whose hyperbolic cosine is r_1. An ambiguous condition exists if r_1 is less than one.

12.6.2.1484 FALOG "f-a-log" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

Raise ten to the power r_1, giving r_2.

12.6.2.1486 FASIN "f-a-sine" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the principal radian angle whose sine is r_1. An ambiguous condition exists if $|r_1|$ is greater than one.

12.6.2.1487 FASINH "f-a-cinch" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the floating-point value whose hyperbolic sine is r_1. An ambiguous condition exists if r_1 is less than zero.

12.6.2.1488 FATAN "f-a-tan" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the principal radian angle whose tangent is r_1.

12.6.2.1489 FATAN2 "f-a-tan-two" FLOATING EXT

(F: r_1 r_2 -- r_3) or (r_1 r_2 -- r_3)

r_3 is the radian angle whose tangent is r_1/r_2. An ambiguous condition exists if r_1 and r_2 are zero.

12.6.2.1491 FATANH "f-a-tan-h" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the floating-point value whose hyperbolic tangent is r_1. An ambiguous condition exists if r_1 is outside the range of -1E0 to 1E0.

12.6.2.1493 FCOS "f-cos" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the cosine of the radian angle r_1.

12.6.2.1494 FCOSH "f-cosh" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the hyperbolic cosine of r_1.

12.6.2.1513 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 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.

See: **6.1.0750 BASE**, **12.3.2 Floating-point operations**, **12.6.1.2143 REPRESENT**.

12.6.2.1515 FEXP "f-e-x-p" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

Raise e to the power r_1, giving r_2.

12.6.2.1516 FEXPM1 "f-e-x-p-m-one" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

Raise *e* to the power r_1 and subtract one, giving r_2.

12.6.2.1553 FLN "f-l-n" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the natural logarithm of r_1. An ambiguous condition exists if r_1 is less than or equal to zero.

12.6.2.1554 FLNP1 "f-l-n-p-one" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the natural logarithm of the quantity r_1 plus one. An ambiguous condition exists if r_1 is less than or equal to negative one.

12.6.2.1557 FLOG "f-log" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the base-ten logarithm of r_1. An ambiguous condition exists if r_1 is less than or equal to zero.

12.6.2.1613 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.

See: **6.1.0750 BASE**, **12.3.2 Floating-point operations**, **12.6.1.2143 REPRESENT**.

12.6.2.1614 FSIN "f-sine" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the sine of the radian angle r_1.

12.6.2.1616 FSINCOS "f-sine-cos" FLOATING EXT

(F: r_1 -- r_2 r_3) or (r_1 -- r_2 r_3)

r_2 is the sine of the radian angle r_1. r_3 is the cosine of the radian angle r_1.

12.6.2.1617 FSINH "f-cinch" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the hyperbolic sine of r_1.

12.6.2.1618 FSQRT "f-square-root" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the square root of r_1. An ambiguous condition exists if r_1 is less than zero.

12.6.2.1625 FTAN "f-tan" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the tangent of the radian angle r_1. An ambiguous condition exists if cos(r_1) is zero.

12.6.2.1626 FTANH "f-tan-h" FLOATING EXT

(F: r_1 -- r_2) or (r_1 -- r_2)

r_2 is the hyperbolic tangent of r_1.

12.6.2.1640 F~ "f-proximate" FLOATING EXT

(-- *flag*) (F: r_1 r_2 r_3 --) or (r_1 r_2 r_3 -- *flag*)

If r_3 is positive, *flag* is true if the absolute value of (r_1 minus r_2) is less than r_3.

If r_3 is zero, *flag* is true if the implementation-dependent encoding of r_1 and r_2 are exactly identical (positive and negative zero are unequal if they have distinct encodings).

If r_3 is negative, *flag* is true if the absolute value of (r_1 minus r_2) is less than the absolute value of r_3 times the sum of the absolute values of r_1 and r_2.

12.6.2.2035 PRECISION FLOATING EXT

(-- *u*)

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

12.6.2.2200 SET-PRECISION FLOATING EXT

(*u* --)

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

12.6.2.2202 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.

See: **12.3.1.1 Addresses**, **12.3.2 Floating-point operations**.

12.6.2.2203 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.

See: **12.3.1.1 Addresses**, **12.3.2 Floating-point operations**.

12.6.2.2204 SFALIGN "s-f-align" FLOATING EXT

(--)

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

See: **12.3.1.1 Addresses**.

12.6.2.2206 SFALIGNED "s-f-aligned" FLOATING EXT

(*addr* -- *sf-addr*)

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

See: **12.3.1.1 Addresses**.

12.6.2.2207 SFLOAT+ "s-float-plus" FLOATING EXT

($sf\text{-}addr_1$ -- $sf\text{-}addr_2$)

Add the size in address units of a 32-bit IEEE single-precision number to $sf\text{-}addr_1$, giving $sf\text{-}addr_2$.

See: **12.3.1.1 Addresses**.

12.6.2.2208 SFLOATS “s-floats” FLOATING EXT

(n_1 -- n_2)

n_2 is the size in address units of n_1 32-bit IEEE single-precision numbers.

See: **12.3.1.1 Addresses**.

13. The optional Locals word set

13.1 Introduction

See: **Annex A.13 The Locals Word Set**.

13.2 Additional terms and notation

None.

13.3 Additional usage requirements

13.3.1 Locals

A local is a data object whose execution semantics shall return its value, whose scope shall be limited to the definition in which it is declared, and whose use in a definition shall not preclude reentrancy or recursion.

13.3.2 Environmental queries

Append table 13.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 13.1 – Environmental query strings

String	Value data type	Constant?	Meaning
#LOCALS	*n*	yes	maximum number of local variables in a definition
LOCALS	*flag*	no	locals word set present
LOCALS-EXT	*flag*	no	locals extensions word set present

13.3.3 Processing locals

To support the locals word set, a system shall provide a mechanism to receive the messages defined by **(LOCAL)** and respond as described here.

During the compilation of a definition after **:** (colon), **:NONAME**, or **DOES>**, a program may begin sending local identifier messages to the system. The process shall begin when the first message is sent. The process shall end when the "last local" message is sent. The system shall keep track of the names, order, and number of identifiers contained in the complete sequence.

The system, upon receipt of a sequence of local-identifier messages, shall take the following actions at compile time:

1) Create temporary dictionary entries for each of the identifiers passed to **(LOCAL)**, such that each identifier will behave as a *local*. These temporary dictionary entries shall vanish at the end of the definition, denoted by **;** (semicolon), **;CODE**, or **DOES>**. The system need not maintain these identifiers in the same way it does other dictionary entries as long as they can be found by normal dictionary searching processes. Furthermore, if the Search-Order word set is present, local identifiers shall always be searched before any of the word lists in any definable search order, and none of the Search-Order words shall change the locals' privileged position in the search order. Local identifiers may reside in mass storage.
2) For each identifier passed to **(LOCAL)**, the system shall generate an appropriate code sequence that does the following at execution time:

a) Allocate a storage resource adequate to contain the value of a local. The storage shall be allocated in a way that does not preclude re-entrancy or recursion in the definition using the local.

b) Initialize the value using the top item on the data stack. If more than one local is declared, the top item on the stack shall be moved into the first local identified, the next item shall be moved into the second, and so on.

The storage resource may be the return stack or may be implemented in other ways, such as in registers. The storage resource shall not be the data stack. Use of locals shall not restrict use of the data stack before or after the point of declaration.

3) Arrange that any of the legitimate methods of terminating execution of a definition, specifically **;** (semicolon), **;CODE**, **DOES>** or **EXIT**, will release the storage resource allocated for the locals, if any, declared in that definition. **ABORT** shall release all local storage resources, and **CATCH** / **THROW** (if implemented) shall release such resources for all definitions whose execution is being terminated.

4) Separate sets of locals may be declared in defining words before **DOES>** for use by the defining word, and after **DOES>** for use by the word defined.

A system implementing the Locals word set shall support the declaration of at least eight locals in a definition.

Immediate words in a program may use **(LOCAL)** to implement syntaxes for local declarations with the following restrictions:

– A program shall not compile any executable code into the current definition between the time **(LOCAL)** is executed to identify the first local for that definition and the time of sending the single required “last local” message;

– The position in program source at which the sequence of **(LOCAL)** messages is sent, referred to here as the point at which locals are declared, shall not lie within the scope of any control structure;

– Locals shall not be declared until values previously placed on the return stack within the definition have been removed;

– After a definition’s locals have been declared, a program may place data on the return stack. However, if this is done, locals shall not be accessed until those values have been removed from the return stack;

– Words that return execution tokens, such as **'** (tick), **[']**, or **FIND**, shall not be used with local names;

– A program that declares more than eight locals in a single definition has an environmental dependency;

– Locals may be accessed or updated within control structures, including do-loops;

– Local names shall not be referenced by **POSTPONE** and **[COMPILE]**.

See: **3.4 The Forth text interpreter**.

13.4 Additional documentation requirements

13.4.1 System documentation

13.4.1.1 Implementation-defined options

– maximum number of locals in a definition (**13.3.3 Processing locals**, **13.6.2.1795 LOCALS|**).

13.4.1.2 Ambiguous conditions

* executing a named *local* while in interpretation state (**13.6.1.0086 (LOCAL)**);

* *name* not defined by **VALUE** or **LOCAL** (**13.6.1.2295 TO**).

13.4.1.3 Other system documentation

– no additional requirements.

13.4.2 Program documentation

13.4.2.1 Environmental dependencies

– declaring more than eight locals in a single definition (**13.3.3 Processing locals**).

13.4.2.2 Other program documentation

– no additional requirements.

13.5 Compliance and labeling

13.5.1 ANS Forth systems

The phrase “Providing the Locals word set” shall be appended to the label of any Standard System that provides all of the Locals word set.

The phrase “Providing *name*(*s*) from the Locals Extensions word set” shall be appended to the label of any Standard System that provides portions of the Locals Extensions word set.

The phrase “Providing the Locals Extensions word set” shall be appended to the label of any Standard System that provides all of the Locals and Locals Extensions word sets.

13.5.2 ANS Forth programs

The phrase “Requiring the Locals word set” shall be appended to the label of Standard Programs that require the system to provide the Locals word set.

The phrase “Requiring *name*(*s*) from the Locals Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the Locals Extensions word set.

The phrase “Requiring the Locals Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the Locals and Locals Extensions word sets.

13.6 Glossary

13.6.1 Locals words

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**.

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)**.

13.6.2 Locals extension words

13.6.2.1795 **LOCALS|** "locals-bar" LOCAL EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation: ("$name_1$" "$name_2$" ... "$name_n$" "**|**" --)

Create up to eight local identifiers as described in **(LOCAL)**. The list of locals to be defined is terminated by **|**. Append the run-time semantics given below to the current definition.

Run-time: (x_n ... x_2 x_1 --)

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 $name_1$ is initialized with x_1, identifier $name_2$ with x_2, etc. When invoked, each local will return its value. The value of a local may be changed using **13.6.1.2295 TO**.

14. The optional Memory-Allocation word set

14.1 Introduction

14.2 Additional terms and notation

None.

14.3 Additional usage requirements

14.3.1 I/O Results data type

I/O results are single-cell numbers indicating the result of I/O operations. A value of zero indicates that the I/O operation completed successfully; other values and their meanings are implementation-defined.

Append table 14.1 to table 3.1.

Table 14.1 – Data types

Symbol	Data type	Size on stack
ior	I/O results	1 cell

14.3.2 Environmental queries

Append table 14.2 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 14.2 – Environmental query strings

String	Value data type	Constant?	Meaning
MEMORY-ALLOC	*flag*	no	memory-allocation word set present
MEMORY-ALLOC-EXT	*flag*	no	memory-allocation extensions word set present

14.3.3 Allocated regions

A program may address memory in data space regions made available by **ALLOCATE** or **RESIZE** and not yet released by **FREE**.

See: **3.3.3 Data space**.

14.4 Additional documentation requirements

14.4.1 System documentation

14.4.1.1 Implementation-defined options

– values and meaning of *ior* (**14.3.1 I/O Results data type**, **14.6.1.0707 ALLOCATE**, **14.6.1.1605 FREE**, **14.6.1.2145 RESIZE**).

14.4.1.2 Ambiguous conditions

– no additional requirements.

14.4.1.3 Other system documentation

– no additional requirements.

14.4.2 Program documentation

– no additional requirements.

14.5 Compliance and labeling

14.5.1 ANS Forth systems

The phrase “Providing the Memory-Allocation word set” shall be appended to the label of any Standard System that provides all of the Memory-Allocation word set.

The phrase “Providing *name*(*s*) from the Memory-Allocation Extensions word set” shall be appended to the label of any Standard System that provides portions of the Memory-Allocation Extensions word set.

The phrase “Providing the Memory-Allocation Extensions word set” shall be appended to the label of any Standard System that provides all of the Memory-Allocation and Memory-Allocation Extensions word sets.

14.5.2 ANS Forth programs

The phrase “Requiring the Memory-Allocation word set” shall be appended to the label of Standard Programs that require the system to provide the Memory-Allocation word set.

The phrase “Requiring *name*(*s*) from the Memory-Allocation Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the Memory-Allocation Extensions word set.

The phrase “Requiring the Memory-Allocation Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the Memory-Allocation and Memory-Allocation Extensions word sets.

14.6 Glossary

14.6.1 Memory-Allocation words

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`**.

14.6.1.1605 **FREE** MEMORY

(*a-addr* -- *ior*)

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**.

14.6.1.2145 **RESIZE** MEMORY

($a\text{-}addr_1$ *u* -- $a\text{-}addr_2$ *ior*)

Change the allocation of the contiguous data space starting at the address $a\text{-}addr_1$, 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, $a\text{-}addr_2$ is the aligned starting address of *u* address units of allocated memory and *ior* is zero. $a\text{-}addr_2$ may be, but need not be, the same as $a\text{-}addr_1$. If they are not the same, the values contained in the region at $a\text{-}addr_1$ are copied to $a\text{-}addr_2$, 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 $a\text{-}addr_2$ is not the same as $a\text{-}addr_1$, the region of memory at $a\text{-}addr_1$ is returned to the system according to the operation of **FREE**.

If the operation fails, $a\text{-}addr_2$ equals $a\text{-}addr_1$, the region of memory at $a\text{-}addr_1$ 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**.

14.6.2 Memory-Allocation extension words

None

15. The optional Programming-Tools word set

15.1 Introduction

This word set extension contains words most often used during the development of applications.

15.2 Additional terms and notation

None.

15.3 Additional usage requirements

15.3.1 Environmental queries

Append table 15.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 15.1 – Environmental query strings

String	Value data type	Constant?	Meaning
TOOLS	*flag*	no	programming-tools word set present
TOOLS-EXT	*flag*	no	programming-tools extensions word set present

15.3.2 The Forth dictionary

A program using the words **CODE** or **;CODE** associated with assembler code has an environmental dependency on that particular instruction set and assembler notation.

Programs using the words **EDITOR** or **ASSEMBLER** require the Search Order word set or an equivalent implementation-defined capability.

See: **3.3 The Forth dictionary**.

15.4 Additional documentation requirements

15.4.1 System documentation

15.4.1.1 Implementation-defined options

- ending sequence for input following **15.6.2.0470 ;CODE** and **15.6.2.0930 CODE**;
- manner of processing input following **15.6.2.0470 ;CODE** and **15.6.2.0930 CODE**;
- search-order capability for **15.6.2.1300 EDITOR** and **15.6.2.0740 ASSEMBLER** (**15.3.3 The Forth dictionary**);
- source and format of display by **15.6.1.2194 SEE**.

15.4.1.2 Ambiguous conditions

* deleting the compilation word-list (**15.6.2.1580 FORGET**);
* fewer than *u*+1 items on control-flow stack (**15.6.2.1015**, **15.6.2.1020**);
* *name* can't be found (**15.6.2.1580 FORGET**);
* *name* not defined via **6.1.1000 CREATE** (**15.6.2.0470 ;CODE**);
* **6.1.2033 POSTPONE** applied to **15.6.2.2532 [IF]**;

* reaching the end of the input source before matching **15.6.2.2531 [ELSE]** or **15.6.2.2533 [THEN]** (**15.6.2.2532 [IF]**);
* removing a needed definition (**15.6.2.1580 FORGET**).

15.4.1.3 Other system documentation

– no additional requirements.

15.4.2 Program documentation

15.4.2.1 Environmental dependencies

– using the words **15.6.2.0470 ;CODE** or **15.6.2.0930 CODE**.

15.4.2.2 Other program documentation

– no additional requirements.

15.5 Compliance and labeling

15.5.1 ANS Forth systems

The phrase “Providing the Programming-Tools word set” shall be appended to the label of any Standard System that provides all of the Programming-Tools word set.

The phrase “Providing *name*(*s*) from the Programming-Tools Extensions word set” shall be appended to the label of any Standard System that provides portions of the Programming-Tools Extensions word set.

The phrase “Providing the Programming-Tools Extensions word set” shall be appended to the label of any Standard System that provides all of the Programming-Tools and Programming-Tools Extensions word sets.

15.5.2 ANS Forth programs

The phrase “Requiring the Programming-Tools word set” shall be appended to the label of Standard Programs that require the system to provide the Programming-Tools word set.

The phrase “Requiring *name*(*s*) from the Programming-Tools Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the Programming-Tools Extensions word set.

The phrase “Requiring the Programming-Tools Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the Programming-Tools and Programming-Tools Extensions word sets.

15.6 Glossary

15.6.1 Programming-Tools words

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**.

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**.

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**.

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**.

15.6.2 Programming-Tools extension words

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*.

Process subsequent characters in the parse area in an implementation-defined manner, thus generating corresponding machine code. Those characters typically represent source code in a programming language, usually some form of assembly language. 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 recen 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>**.

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*.

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**.

15.6.2.0830 **BYE** TOOLS EXT

(--)

Return control to the host operating system, if any.

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.

Process subsequent characters in the parse area in an implementation-defined manner, thus generating corresponding machine code. Those characters typically represent source code in a programming language, usually some form of assembly language. 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**.

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

Interpretation: Interpretation semantics for this word are undefined.

Execution: (C: $dest_u$ ... $orig_0|dest_0$ -- $dest_u$ ... $orig_0|dest_0$ $dest_u$)
(S: *u* --)

Remove *u*. Copy $dest_u$ 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.

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

Interpretation: Interpretation semantics for this word are undefined.

Execution: (C: $orig_u|dest_u\ orig_{u-1}|dest_{u-1}$ ... $orig_0|dest_0$ -- $orig_{u-1}|dest_{u-1}$ ... $orig_0|dest_0\ orig_u|dest_u$)
(S: *u* --)

Remove *u*. Rotate *u*+1 elements on top of the control-flow stack so that $orig_u|dest_u$ 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.

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**.

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**.

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**.

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**.

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 **POSTPONE**d, 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**.

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

Compilation: Perform the execution semantics given below.

Execution: (--)

Does nothing. **[THEN]** is an immediate word.

16. The optional Search-Order word set

16.1 Introduction

16.2 Additional terms and notation

compilation word list: the word list into which new definition names are placed.

search order: a list of word lists specifying the order in which the dictionary will be searched.

16.3 Additional usage requirements

16.3.1 Data types

Word list identifiers are implementation-dependent single-cell values that identify word lists.

Append table 16.1 to table 3.1.

Table 16.1 – Data types

Symbol	Data type	Size on stack
wid	word list identifiers	1 cell

See: **3.1 Data types**, **3.4.2 Finding definition names**, **3.4 The Forth text interpreter**.

16.3.2 Environmental queries

Append table 16.2 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 16.2 – Environmental query strings

String	Value data type	Constant?	Meaning
SEARCH-ORDER	*flag*	no	search-order word set present
SEARCH-ORDER-EXT	*flag*	no	search-order extensions word set present
WORDLISTS	*n*	yes	maximum number of word lists usable in the search order

16.3.3 Finding definition names

When searching a word list for a definition name, the system shall search each word list from its last definition to its first. The search may encompass only a single word list, as with **SEARCH-WORDLIST**, or all the word lists in the search order, as with the text interpreter and **FIND**.

Changing the search order shall only affect the subsequent finding of definition names in the dictionary.

A system with the Search-Order word set shall allow at least eight word lists in the search order.

An ambiguous condition exists if a program changes the compilation word list during the compilation of a definition or before modification of the behavior of the most recently compiled definition with **;CODE**, **DOES>**, or **IMMEDIATE**.

A program that requires more than eight word lists in the search order has an environmental dependency.

See: **3.4.2 Finding definition names**

16.3.4 Contiguous regions

The regions of data space produced by the operations described in **3.3.3.2 Contiguous regions** may be non-contiguous if **WORDLIST** is executed between allocations.

16.4 Additional documentation requirements

16.4.1 System documentation

16.4.1.1 Implementation-defined options

– maximum number of word lists in the search order (**16.3.3 Finding definition names**, **16.6.1.2197 SET-ORDER**);
– minimum search order (**16.6.1.2197 SET-ORDER**, **16.6.2.1965 ONLY**).

16.4.1.2 Ambiguous conditions

* changing the compilation word list (**16.3.3 Finding definition names**);
* search order empty (**16.6.2.2037 PREVIOUS**);
* too many word lists in search order (**16.6.2.0715 ALSO**).

16.4.1.3 Other system documentation

– no additional requirements.

16.4.2 Program documentation

16.4.2.1 Environmental dependencies

– requiring more than eight word-lists in the search order (**16.3.3 Finding definition names**).

16.4.2.2 Other program documentation

– no additional requirements.

16.5 Compliance and labeling

16.5.1 ANS Forth systems

The phrase “Providing the Search-Order word set” shall be appended to the label of any Standard System that provides all of the Search-Order word set.

The phrase “Providing *name*(*s*) from the Search-Order Extensions word set” shall be appended to the label of any Standard System that provides portions of the Search-Order Extensions word set.

The phrase “Providing the Search-Order Extensions word set” shall be appended to the label of any Standard System that provides all of the Search-Order and Search-Order Extensions word sets.

16.5.2 ANS Forth programs

The phrase “Requiring the Search-Order word set” shall be appended to the label of Standard Programs that require the system to provide the Search-Order word set.

The phrase “Requiring *name*(*s*) from the Search-Order Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the Search-Order Extensions word set.

The phrase “Requiring the Search-Order Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the Search-Order and Search-Order Extensions word sets.

16.6 Glossary

16.6.1 Search-Order words

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**.

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**.

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.

16.6.1.1643 GET-CURRENT SEARCH

(-- *wid*)

Return *wid*, the identifier of the compilation word list.

16.6.1.1647 GET-ORDER SEARCH

(-- wid_n ... wid_1 *n*)

Returns the number of word lists *n* in the search order and the word list identifiers wid_n ... wid_1 identifying these word lists. wid_1 identifies the word list that is searched first, and wid_n the word list that is searched last. The search order is unaffected.

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.

16.6.1.2195 SET-CURRENT SEARCH

(*wid* --)

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

16.6.1.2197 SET-ORDER SEARCH

(wid_n ... wid_1 *n* --)

Set the search order to the word lists identified by wid_n ... wid_1. Subsequently, word list wid_1 will be searched first, with word list wid_n 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.

16.6.1.2460 WORDLIST SEARCH

(-- *wid*)

Creates 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.

16.6.2 Search-Order extension words

16.6.2.0715 ALSO SEARCH EXT

(--)

Transform the search order consisting of wid_n, ... wid_2, wid_1 (where wid_1 is searched first) into wid_n, ... wid_2, wid_1, wid_1. An ambiguous condition exists if there are too many word lists in the search order.

16.6.2.1590 FORTH SEARCH EXT

(--)

Transform the search order consisting of wid_n, ... wid_2, wid_1 (where wid_1 is searched first) into wid_n, ... wid_2, $wid_{\text{FORTH-WORDLIST}}$.

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**.

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**.

16.6.2.2037 **PREVIOUS** SEARCH EXT

(--)

Transform the search order consisting of wid_n, ... wid_2, wid_1 (where wid_1 is searched first) into wid_n, ... wid_2. An ambiguous condition exists if the search order was empty before **PREVIOUS** was executed.

17. The optional String word set

17.1 Introduction

17.2 Additional terms and notation

None.

17.3 Additional usage requirements

Append table 17.1 to table 3.5.

See: **3.2.6 Environmental queries**.

Table 17.1 – Environmental query strings

String	Value data type	Constant?	Meaning
STRING	*flag*	no	string word set present
STRING-EXT	*flag*	no	string extensions word set present

17.4 Additional documentation requirements

None.

17.5 Compliance and labeling

17.5.1 ANS Forth systems

The phrase “Providing the String word set” shall be appended to the label of any Standard System that provides all of the String word set.

The phrase “Providing *name*(*s*) from the String Extensions word set” shall be appended to the label of any Standard System that provides portions of the String Extensions word set.

The phrase “Providing the String Extensions word set” shall be appended to the label of any Standard System that provides all of the String and String Extensions word sets.

17.5.2 ANS Forth programs

The phrase “Requiring the String word set” shall be appended to the label of Standard Programs that require the system to provide the String word set.

The phrase “Requiring *name*(*s*) from the String Extensions word set” shall be appended to the label of Standard Programs that require the system to provide portions of the String Extensions word set.

The phrase “Requiring the String Extensions word set” shall be appended to the label of Standard Programs that require the system to provide all of the String and String Extensions word sets.

17.6 Glossary

17.6.1 String words

17.6.1.0170 -TRAILING "dash-trailing" STRING

(*c-addr* u_1 -- *c-addr* u_2)

If u_1 is greater than zero, u_2 is equal to u_1 less the number of spaces at the end of the character string specified by *c-addr* u_1. If u_1 is zero or the entire string consists of spaces, u_2 is zero.

17.6.1.0245 /STRING "slash-string" STRING

(*c-addr*$_1$ u_1 *n* -- *c-addr*$_2$ u_2)

Adjust the character string at *c-addr*$_1$ by *n* characters. The resulting character string, specified by *c-addr*$_2$ u_2, begins at *c-addr*$_1$ plus *n* characters and is u_1 minus *n* characters long.

17.6.1.0780 BLANK STRING

(*c-addr u* --)

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

17.6.1.0910 CMOVE "c-move" STRING

(*c-addr*$_1$ *c-addr*$_2$ *u* --)

If *u* is greater than zero, copy *u* consecutive characters from the data space starting at *c-addr*$_1$ to that starting at *c-addr*$_2$, proceeding character-by-character from lower addresses to higher addresses.

Contrast with: **17.6.1.0920 CMOVE>**.

17.6.1.0920 CMOVE> "c-move-up" STRING

(*c-addr*$_1$ *c-addr*$_2$ *u* --)

If *u* is greater than zero, copy *u* consecutive characters from the data space starting at *c-addr*$_1$ to that starting at *c-addr*$_2$, proceeding character-by-character from higher addresses to lower addresses.

Contrast with: **17.6.1.0910 CMOVE**.

17.6.1.0935 COMPARE STRING

(*c-addr*$_1$ *u*$_1$ *c-addr*$_2$ *u*$_2$ -- *n*)

Compare the string specified by *c-addr*$_1$ *u*$_1$ to the string specified by *c-addr*$_2$ *u*$_2$. 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 *u*$_1$ is less than *u*$_2$ 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-addr*$_1$ *u*$_1$ has a lesser numeric value than the corresponding character in the string specified by *c-addr*$_2$ *u*$_2$ and one (1) otherwise.

17.6.1.2191 SEARCH STRING

(*c-addr*$_1$ *u*$_1$ *c-addr*$_2$ *u*$_2$ -- *c-addr*$_3$ *u*$_3$ *flag*)

Search the string specified by *c-addr*$_1$ *u*$_1$ for the string specified by *c-addr*$_2$ *u*$_2$. If *flag* is true, a match was found at *c-addr*$_3$ with *u*$_3$ characters remaining. If *flag* is false there was no match and *c-addr*$_3$ is *c-addr*$_1$ and *u*$_3$ is *u*$_1$.

17.6.1.2212 SLITERAL STRING

Interpretation: Interpretation semantics for this word are undefined.

Compilation: (*c-addr*$_1$ *u* --)

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

Run-time: (-- *c-addr*$_2$ *u*)

Return *c-addr*$_2$ *u* describing a string consisting of the characters specified by *c-addr*$_1$ *u* during compilation. A program shall not alter the returned string.

17.6.2 String extension words

None