eForth Glossary

Derived from bFORTH by Bill Muench, 1990.

WARNING: Advanced information -- subject to change.

==========

Sort order

!"#$%&'()*+,-./digits:;<=>?@Alpha[\]^_`{|}~

==========

Attributes Capitalized symbols.

C the word may only be used during compilation of a colon definition.
D the word is a defining word.
I the word is IMMEDIATE and will execute during compilation, unless special action is taken.
U a user value.

==========

Conventions

<string> characters in the input stream

a	aligned address
b	byte address
c	character
ca	code address
cy	carry
d	signed double integer
F	logical false
f	flag 0 or non-zero
la	link address
n	signed integer
na	name address
T	logical true
t	flag T or F
u	unsigned integer
ud	unsigned double integer
va	vocabulary address
w	unspecified weighted value

==========

Header: token(ptr) link(la) name(na)

Count-byte and Lexicon bits ioxn nnnn
    i - immediate
    o - compile-only
    x - tag
    n - string length (31 characters MAX)
    Compiler does not set bits in the NAME string
    0 < la na < .. < la na <   va < CONTEXT   @
    0 < FORTH < .. < vl va < vl va < CURRENT CELL+ @

==========

Stack notes

(compile \ run \ child ;Return ;Float ; <input stream>)

(before -- after ;R before -- after ;F before -- after ; <string>)

==========

Glossary

! (w a --)"store"
Store a 16-bit number at aligned address.

!CSP (--) "set c s p"
Save the values of the current stack pointers.

!IO (--) "store i o"
Initialize the serial I/O device.

(d -- d) "number sign"
Convert one digit of a number using the current base. Must be used within <# and #> .

#> (d -- b u) "number sign greater"
Terminate a numeric conversion.

#S (ud -- 0 0) "number sign s"
Convert all digits of a number using the current base.

#TIB (-- a) "number t i b"
The system double variable which holds the size and aligned address of the terminal input buffer.

$" (-- ; <string> \ -- $) I,C "string quote"
Used only within a definition to compile an inline packed string terminated by the " double quote character. At run-time the address of the packed string is pushed on the data stack.

$"| (-- $) C "string quote primitive"
Return the address of a compiled inline packed string. The run-time primitive compiled by " .

$," (-- ; <string>) "string comma quote"
Compile an inline packed character string into the code area, terminated by the " double quote character.

$,n ($ --) "string comma n"
Create a name for a definition using string. Set the code pointer to the next free cell in the code area, no code is compiled. The name is not linked into the dictionary.

$COMPILE ($ --)
Convert a string to a word address. Execute the word in interpreting mode or compile it if in compiling mode.

$INTERPRET ($ --) "string interpret"
At the interactive level, if a word is defined perform its action. If not, try to convert it to a number, if that fails, issue an error message.

' (-- ca ; <string>)"tick"
Return the code address of the word following.

'?KEY (-- a) U "tick question key"
The system input device status vector.

'BOOT (-- a)
Return the address of a system boot-up routine.

'ECHO (-- a) U "tick echo"
The system echo device vector.

'EMIT (-- a) U "tick emit"
The system output device vector.

'EVAL (-- a) "tick eval"
The system interpret/compile vector.

'EXPECT (-- a) U "tick expect"
The system line input vector.

'NUMBER (-- a) "tick number"
The system number conversion vector.

'PROMPT (-- a) U "tick prompt"
The system prompt vector.

'TAP (-- a) U "tick tap"
The input case function vector.

((-- ; <string>) I "paren"
Begin a comment. The comment is terminated by the)
character. May be used inside or outside a definition.

***** (n n -- n) "star"
Multiply two signed numbers. Return a 16-bit signed number.

***/** (n1 n2 n3 -- q) "star slash"
Multiply n1 by n2 producing the 32-bit intermediate product d. Divide d by n3 producing a 16-bit quotient.

***/MOD** (n1 n2 n3 -- r q) "star slash mod"
Multiply n1 by n2 producing the 32-bit intermediate product d. Divide d by n3 producing a 16-bit remainder and a 16-bit quotient.

+ (w w -- w) "plus"
Addition.

+! (n a --) "plus store"
Increment the 16-bit value at address by n.

, (w --) "comma"
Compile a 16-bit value into the code area.

- (w w -- w) "minus"
Subtract the top from the second element on the data stack.

-TRAILING (b u -- b u) "dash trailing"
Adjust the count to eliminate any trailing white-space in the string.

. (n --) "dot"
Display the single value, use the current base. If BASE is DECIMAL , display as a signed number.

." (-- ; <string>) I,C "dot quote"
Used only within a definition to compile an inline packed string terminated by the " double quote character. At run-time the string is displayed on the current output device.

."| (--) C "dot quote primitive"
Display a compiled inline packed string to the current output device. This run-time primitive is compiled by ." .

.((-- ; <string>) I "dot paren"
Begin a comment that is displayed to the current output device. May be used inside or outside a definition.

.ID (na --) "dot i d"
Display the packed string at address.

.OK (--) "dot o k"
Display the standard system prompt.

.R (n +n --) "dot r"
Display the single value right-justified in a field of width +n, use the current base.

.S (? -- ?) "dot s"
Display the contents of the data stack.

/ (n n -- q) "slash"
Floored division for 16-bit numbers. Returns only the 16-bit quotient.

/MOD (n n -- r q) "slash mod"
Floored division for 16-bit numbers. 16-bit remainder and 16-bit quotient.

0< (n -- t) "zero less"
Return true if n is less than 0, negative. Comparison is signed. Also used for sign extension.

0= (w -- t) "zero equals"
Return true if w is equal to 0.

2! (d a --) "two store"
Store a 32-bit value at aligned address.

2@ (a -- d) "two fetch"
Return the 32-bit value stored at aligned address.

2DROP (d --) "two drop"
Pop the 32-bit number, or the top two 16-bit numbers, from the data stack.

2DUP (d -- d d) "two dupe"
Duplicate the 32-bit number, or the top two 16-bit numbers, on the data stack.

: (-- ; <string>) D "colon"
Begin a colon definition to be added to the current vocabulary.

; (--) I,C "semicolon"
Terminate a colon definition begun with : .

< (n1 n2 -- t) "less than"
Return true if n1 is less than n2. Comparison is signed.

<# (--) "start number" "less number sign"
Begin a numeric conversion.

= (w w -- t) "equals"
Return true if w1 is equal to w2.

>CHAR (u -- c) "to character"
Convert a value to a printable character. Replace an unprintable character with the _ underscore character.

>IN (-- a) "to in"
The pointer into the input stream.

>NAME (ca -- na, F) "to name"
If possible, convert a code address to a name address. If not possible, return a false flag.

>R (w -- ;R -- w) C "to r"
Pop the top element of the data stack and Push it on the return stack.

? (a --) "question"
Display the single value stored at address, use the current base. If BASE is DECIMAL , display as a signed number.

?branch (f --) "question branch"
Run time rouitne to redirect execution to the address in the next cell if flag is false.

?CSP (--) "question c s p"
Compare the current value of the stack pointers with the saved values. ABORT with an error message if different.

?DUP (w -- w w, 0) "question dupe"
Duplicate the number on top of the data stack only if it is non-zero.

?KEY (-- t) "question key"
Return the status of the current input device.

?RX (-- c T, F) "question r x"
Return a character from the input device and true. Return false only if no character is pending.

?STACK (--) "question stack"
Display an error message if the stack limits have been exceeded.

?UNIQUE ($ -- $) "question unique"
Display a warning massage for a duplicate definition.

@ (a -- w) "fetch"
Return the 16-bit value stored at aligned address.

@EXECUTE (a --) "fetch execute"
Fetch the execution token stored at address and execute it, ie indirect execution. If the value contained in address is zero, do nothing.

ABORT(--)
Reset the data stack and perform the function of QUIT . Note, no message is displayed.

ABORT" (-- ; <string> \ f --) I,C "abort quote"
Used only within a definition to compile an inline packed string terminated the " double quote character. At run-time, if the flag is false, execute the sequence of words following the string. Otherwise, the string is displayed on the current output ce, execution is then passed to an error handling routine.

abort" (f --) C "abort quote primitive"
The run-time primitive compiled by ABORT" .

ABS (n -- +n)
Return the absolute value of n.

accept (b u -- b u)
Receive a line of u characters maximum to the an input buffer at byte address. Terminate input if a carriage return is received. Return the actual count of received characters. Perform any currently defined keyboard macros. Use the current input device.

AFT (a -- a a \ --) I,C
Used within a loop structure to unconditional skip a portion of code the first time thru the loop. AFT compiles the machine unconditional branch instruction and leaves an address to be resolved by THEN .

AGAIN (a -- \ --) I,C
Terminate an infinite loop structure. AGAIN compiles an unconditional branch instruction, and uses the address left by BEGIN to resolve this backward branch.

AHEAD (-- a \ --) I,C
Mark the beginning of a forward branching, unconditional branch structure. AHEAD compiles the machine unconditional branch instruction and leaves an address to be resolved by THEN .

ALIGNED (b -- a)
Convert a byte address to a word aligned address.

ALLOT(n --)
Adjust the code area pointer by n.

AND (w w -- w)
A bitwise logical AND.

BASE (-- a) U
The system variable which holds the current numeric conversion radix.

BEGIN (-- a \ --) I,C
Mark the beginning of an indefinite loop structure. Leave an address to be resolved by UNTIL, or WHILE and REPEAT .

BL (-- c) "b l"
Push the value of a space, the blank character, on the data stack.

branch (--) C
Run time routine to redirect execution to the address in the next cell.

BYE (--)
Exit Forth and return to the underlying environment or DOS.

C! (v b --) "c store"
Store an byte value at byte address.

C@ (b -- v) "c fetch"
Return the byte value stored at byte address.

CALL, (a --) C "call comma"
Assemble a 4 byte subroutine call to the designated address.

CATCH (ca -- err#/0)
Setup a local error frame and execute the word referenced by the execution token ca. Return a non-zero error number or zero for no error.

CELL+ (a1 -- a2) "cell plus"
Add cell size in bytes to address a1.

CELL- (a1 -- a2) "cell minus"
Subtract cell size in bytes to address a1.

CELLS (n1 -- n2)
Multiply n1 by the cell size in byte.

CHAR (-- c ; <string>)
Return the value of the first character in <string>. If used within a defintion, use the phrase [CHAR <string>] LITERAL.

CHARS (+n c --)
Display +n of character to the current output device.

CMOVE (b1 b2 u --) "c move"
Move u byte values from byte address b1 to b2, proceeding from lower to higher memory. Overwrite occurs if b1<b2<b1+u .

COLD (--)
Completely re-initialize the system, but does not re-load the system from ROM.

COMPILE (--) C
Used only within a definition. At run-time the word following COMPILE is not executed, but its code address is copied into the code area.

CONSOLE (--)
Initialize the vectored input and output devices to a terminal.

CONTEXT (-- a)
The variable used to specify the dictionary search order.

COUNT ($ -- b +n)
Return the byte address and byte count of a packed string.

CP (-- a) "c p"
The pointer to the next available dictionary location in code space. Since code and names are seperated, the traditional DP, dictionary pointer, had to be split into CP, code pointer, and NP, name pointer.

CR (--) "carriage return"
Position the cursor at the beginning of the next line of the current output device.

CREATE (-- ; <string> \ -- a) D
Build a named definition. At run-time the address pointing to next availabe code space is pushed on the data stack.

CSP (-- a) "c s p"
The system variable which holds the current stack pointer. Used for error checking.

CURRENT (-- a)
The variable used to specify the vocabulary in which new definitions are compiled.

DECIMAL (--)
Set decimal as the current BASE . Base 10.

DEPTH (-- n)
The number of elements on the data stack, does not include n .

DIGIT (u -- c)
Convert a single digit number to its character value.

DIGIT? (c base -- v t) "digit question"
Try to convert a character to binary digit. Return true if the digit is valid for the current base.

dm+ (a u -- p a+) "d m plus"
Display the 16-bit values starting at the aligned address a.

DNEGATE (d -- -d) "d negate"
Return the two's complement a double number. Change the sign of a double number.

do$ (-- $) C "do string"
Return the address of a compiled inline packed string.

doLIST (a --) C "do list"
The run-time routine which executes the list in a colon definition pointed to by a.

doLIT (-- n) C "do literal"
Return the in-line literal compiled by LITERAL.

doUSER (--) C "do user variable"
The run-time action of user variables.

doVAR (-- a) C "do variable"
The run-time action of variable.

DROP (w --)
Remove the top element on the data stack.

DUMP (a u --)
Display the HEX and character values starting at aligned address a, for count u.

DUP (w -- w w) "dupe"
Duplicate the top element on the data stack.

ELSE (-- \ a -- a) I,C
Used within a conditional branch structure. ELSE resolves a forward conditional branch compiled by IF . ELSE then compiles an unconditional branch instruction, leaving an address to be resolve by THEN .

EMIT (w --)
Output a character to the current output device.

EVAL (--)
Interpret or compile the tokens from the input stream.

EXECUTE (w -- ;R -- w)
Execute the word definition indicated by the execution token w.

EXIT (-- ;R w --)
Compile a subroutine return.

EXPECT (b u --)
Receive a line of u characters maximum to the an input buffer at byte address. Terminate input if a carriage return is received. The count of received characters is saved in the variable SPAN. Use the current input device. This word is vectored.

EXTRACT (d base -- d' c)
Used incrementally to convert each digit in a number to its character value.

FILE (--)
Specify system input from a file using pace handshake. File input is not echoed, all output messages are displayed.

FILL (b u v --)
Fill an area at byte address of length u using the byte value v.

find ($ va -- ca f, $ F) "find primitive"
Given a string and a dictionary entry thread, search for a name match. If found, return the code address and a true flag. If not found, the string address and a false flag.

FOR (u --) I,C
Begin a down-counting loop. Repeat the loop till NEXT u+1 times from u to 0.

FORTH(--)
Make the default system vocabulary FORTH the context vocabulary.

HAND (--)
Specify system input from keyboard, no handshake. All input is echoed.

HANDLER (-- a)
The current error handler frame pointer.

HERE (-- a)
Push the address of the next free cell in the code area on the data stack.

HEX (--)
Set hexadecimal as the current BASE . Base 16.

hi (--)
Display the sign-on message.

HLD (-- a) "h l d"
The pointer to a formatted numeric output string.

HOLD (c --)
Insert the character in the formatted numeric output string.

I/O (-- a) "i slash o"
An array used by CONSOLE to initialize the system input and output vectors. The vector order is 'KEY? 'KEY and 'EMIT.

IF (-- a \ f --) I,C
Mark the beginning of a forward branching, conditional branch structure. IF compiles the machine conditional branch instruction and leaves an address to be resolved by THEN or ELSE .

IMMEDIATE (--)
Mark the most recently created dictionary entry as a word which will execute during compilation.

INVERT (w -- w)
Bitwise logical invert. Equivalent to -1 XOR. The one's complement.

KEY (-- c)
Return a character from the current input device. If no key is ready, wait until one is available.

kTAP (b b b c -- b b b) "k tap"
The 'tap routine used for file input.

LAST (-- a)
The pointer to the name of the most recently created dictionary entry.

LITERAL (w -- \ -- w) I
Compile a number as an inline value.

M* (n n -- d) "m star"
Multiply two signed numbers. Return a 32-bit signed number.

M/MOD (d n -- r q) "m slash mod"
Floored division of a 32-bit number divided by a 16-bit number. Return a 16-bit quotient and a 16-bit remainder.

MAX (n n -- n)
Leave the greater of the two signed values.

MIN (n n -- n)
Leave the smaller of the two signed values.

MOD (n n -- r)
Floored division for 16-bit numbers. Returns only the 16-bit remainder.

NAME> (na -- ca) "name to code"
Convert a name address to a code address.

NAME?($ -- ca f, $ F) "name question"
Given a string, search for a name match. If found, return the code address and a true flag. If not found, the string address and a false flag.

NEGATE (n -- -n)
Equivalent to 0 SWAP-. The two's complement of a number. Change the sign of a number.

NEXT (a -- \ -- ;R u -- [u-1]) I,C
Terminate a down-counting loop structure. NEXT compiles the machine loop instruction, pointing to the address left by FOR . The loop count is held on the return stack. Looping continues until the count is equal to zero.

next (--) C
Run-time routine to terminate a down-counting FOR-NEXT loop. See NEXT.

NP (-- a) "n p"
The pointer to the next available dictionary location in name space.

NUF? (-- t) "nuf question"
Continue until paused or terminated by user. Any key will pause, while paused any key except 'enter' will restart and return false, enter will return true.

NULL$ (-- $) "null string"
The address of a string with a zero count.

NUMBER? ($ -- d T, $ F) "number question"
Try to convert a packed string to binary number. If possible return the number and a true. Otherwise, return the string address and false. A leading '$' for hexidecimal, a leading '-' for negative, and/or the decimal point for a double number.

OR (w w -- w)
A bitwise logical OR.

OVER (w1 w2 -- w1 w2 w1)
Copy the second element to the top of the data stack.

OVERT(--)
Used by ; to link a successfully defined word into the search order.

PACE (--)
Send the file transfer handshake character.

PACK$ (b u $ -- $) "pack string"
Move and convert the string at byte address with byte count to a packed string at address $.

PAD (-- a)
Short for scratch pad. Address of a temporary buffer.

PARSE (c -- b u ; <string>)
Scan the current input stream for the given character as a delimiter. Return the beginning byte address and count of the delimited string.

parse (b u c -- b u delta ; <string>)
Scan string for the given character as a delimiter. Return the beginning byte address and count of the delimited string. Delta is the beginning to current offset.

PICK (+n -- w)
Copy the +nth data stack value to the top of the data stack.

PRESET (--) C
Clear the data stack, the return stack and initialize system.

QUERY (--)
Receive a line to the current input buffer from current input device.

QUIT (--)
Clear the return stack, set interpret state, and return control to the current command line interpreter.

R> (-- w ;R w --) C "r from"
Pop the top element of the return stack and Push it on the data stack.

R@ (-- w ;R w -- w) "r fetch"
Copy the top element of the return stack and Push it on the data stack.

RECURSE (-- \ -- ;R -- a) I,C
Used only within the word currently being defined to allow self reference. Recursion.

REPEAT (a a -- \ --) I,C
Terminate an indefinite loop structure. REPEAT compiles an unconditional branch instruction, and uses the address left by WHILE to resolve this backward branch.

ROT (w1 w2 w3 -- w2 w3 w1)"rote"
Rotate the top three elements on the data stack. Third element to top and all other shifted down.

RP! (a --) C "r p store"
Set the return stack pointer to address.

RP0 (-- a) "r p zero"
Return the bottom address of the return stack pointer.

RP@ (-- a) "r p fetch"
Return the address of the return stack pointer.

SAME? (a1 a2 u -- a1' f \ -0+) "same question"
Compare the two strings, return the beginning address of the first string and a truth flag.

SIGN (n --)
Display a minus sign if n is negative. Must be used within <# and #> .

SP! (a --) "s p store"
Set the data stack pointer to address.

SP0 (-- a) "s p zero"
Return the bottom address of the data stack pointer.

SP@ (-- a) "s p fetch"
Return the address of the data stack pointer.

SPACE (--)
Display one space, blank character, to the current output device.

SPACES (+n --)
Display +n spaces, blank characters, to the current output device.

SPAN (-- a)
The system variable which holds the count of characters input by EXPECT.

str (d -- b u) "s t r"???
Convert number to a string in current BASE . Signed if DECIMAL .

SWAP (w1 w2 -- w2 w1)
Exchange the top two elements on the data stack.

TAP (b b b c -- b b b')
Echo and store the keystroke, and update the cursor position.

temp (-- a) U
Return address of a user variable for temporary storage.

THEN (a -- \ --) I,C
Terminate a conditional branch structure. Resolves a forward branch compiled by IF, ELSE, AHEAD, or AFT.

THROW (err# -- err#)
Reset the state of the system to the current local error frame, and update the error flag.

TIB (-- a) "t i b"
Address of the terminal input buffer.

TOKEN (-- $; <string>)
Scan the current input stream for a blank delimited word. Move the word to the end of the names area as a packed string. Return the address of the packed string.

TX! (c --) "t x store"
Send a character to the output device. Primitive of EMIT.

TYPE (b u --)
Output u characters of the string at b address to the current output device.

U. (u --) "u dot"
Display the unsigned single value, use the current base.

U.R (u +n --) "u dot r"
Display the unsigned single value right-justified in a field of width +n, use the current base.

U< (u1 u2 -- t) "u less"
Return true if u1 is less than u2. Comparison is unsigned.

UM* (u u -- ud) "u m star"
Multiply two unsigned 16-bit numbers. Return an unsigned 32-bit number.

UM+ (u u -- ud) "u m plus"
Add two unsigned numbers and return a 32-bit sum.

UM/MOD (ud u -- ur uq) "u m slash mod"
Unsigned division of a 32-bit number divided by a 16-bit number. Return an unsigned 16-bit quotient and an unsigned 16-bit remainder.

UNTIL (a -- \ f --) I,C
Terminate an indefinite loop structure. Condition testing is done after executing the code within the loop. UNTIL compiles the machine conditional branch instruction, and uses the address left by BEGIN to resolve this backward branch.

UP (-- a) "u p"
Return the address of the current user area.

USER (u -- ; <string> \ -- a) D
Build a named user variable with an offset from the current user base. At run-time the address of the variable is pushed on the data stack.

VARIABLE (-- ; <string> \ -- a) D
Build a named variable. At run-time the address of the variable is pushed on the data stack.

VER (-- n)
Return the version code. Major revision is in the high byte and minor release in the low byte.

VOCABS (-- a)
Return the address of the first vocabulary FORTH in the vocabulary area.

WHILE (a -- a a \ f --) I,C
Used within an indefinite loop structure. Condition testing is done before executing the code within the loop. WHILE compiles the machine conditional branch instruction and leaves an address to be resolved by REPEAT .

WITHIN (u lo hi -- t)
Return true if lo <= u < hi. Comparison is unsigned and circular.

WORD (c -- $; <string>)
Scan the current input stream for the string delimited by 'c'. Return the address of the packed string.

WORDS (--)
Display the words in the CONTEXT vocabulary. Display continues until paused or terminated by user.

XIO (a1 a2 a3 --) "x i o"
Revector 'prompt, 'echo and 'tap to the code addresses on the stack.

XOR (w w -- w)
Bitwise logical Exclusive OR.

[(--) I "left bracket"
Begin interpreting text from the input stream. Change from compiling to interpreting.

[COMPILE] (-- ; <string> \ --) I "bracket compile"
Used only within a definition to force the compilation of the following IMMEDIATE word.

\ (-- ; <string>) I "backslash"
Begin a comment. The comment is terminated by the system end-of-line character. May be used inside or outside a definition.

] (--) "right bracket"
Change from interpreting to compiling.

^H (b b b -- b b b' ; <backspace>) "control h"
A keyboard macro to delete characters from the current input stream. No action is taken if the beginning of the input stream is reached.

_TYPE (b u --) "printable type"
Display the the string starting at the byte address b, for count u. Substitute _ the underscore character for unprintable characters.