

Appendix A

Glossary of Standard Forth words used in this article.

'

(-- xt) <spaces>name<space>

Apostrophe (pronounced: "tic") Skip leading space delimiters. Parse name delimited by a space. Find name and return its execution address.

-

(n1 n2 -- n3)

subtract n2 from n1.

!

(x addr --)

Exclamation mark (pronounced "store") Store x at addr

(

Parentheses are used for comments. The opening parenthesis is a Forth word which must be enclosed by blanks on either side. The action of this word is to skip all text until a closing parenthesis is found. The closing parenthesis is a delimiter. Some versions of Forth only allow comments to extend for one line.

*

(n1 n2 -- n3)

multiply n1 by n2

*/

(n1 n2 n3 -- n4)

Multiply n1 by n2 producing the intermediate double-cell result d. Divide d by n3 giving the single-cell quotient n4.

,

(x --)

Store x in the dictionary in next available cell

.

(n --)

(pronounced "dot") Display n as a signed number followed by a space.

."

follow by text to "

This word will print all text following it up to a delimiting double quote ("). This word can only be used during compilation (it actually saves the text to be printed in the dictionary).

.(follow by text to)
This word will print all text following it up to a delimiting closing parenthesis. It does not store the text but prints it immediately. It can be used interpretively, outside of definitions, to print text when a file is loading.

.ID (nfa --)
print the name at the given name field address

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

.S (--)
Print the contents of the stack without removing elements.

/MOD (n1 n2 -- r q)
Divide n1 by n2, giving the single-cell remainder r and the single-cell quotient q. If n1 and n2 differ in sign, the result will depend on implementation.

? (addr --)
This is a shorthand for @. (print the single cell number at the given address)

?DO (n1 n2 --)
This starts a loop (like DO) but will not enter the loop if n1=n2.

?DUP (x -- 0 | x x)
Duplicate x if it is not zero.

@ (addr -- x)
(pronounced "fetch") x is the single-cell value stored at addr

+ (n1 n2 -- n3)
add n2 to n1

+! (n addr --)
Add n to the single-cell number stored at addr

<

(n1 n2 -- t/f)

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

<>

(x1 x2 -- t/f)

True if and only if x1 is not the same, bit for bit, as x2

=

(x1 x2 -- t/f)

True if and only if x1 is the same, bit for bit, as x2

>

(n1 n2 -- t/f)

The flag is true if and only if n1 is bigger than n2

>=

(n1 n2 -- t/f)

The flag is true if and only if n1 is bigger than or equal to n2

>R

(x --)

Move x to the return stack

0<

(n -- t/f)

True if and only if n is less than zero

0<>

(n -- t/f)

True if and only if n is not equal to zero

0=

(n -- t/f)

True if and only if n is equal to zero

0>

(n -- t/f)

True if and only if n is greater than zero

1-

(n1 -- n2)

Subtract one from n1 giving the difference n2

1+

(n1 -- n2)

Add one to n1 giving the sum n2

2!

(x1 x2 addr --)

Store the cell pair at addr with x2 at addr and x1 at the next cell

2*

(n1 -- n2)
Multiply n1 by 2 giving the product n2

2/

(n1 -- n2)
Divide n1 by 2 giving the quotient n2

2@

(addr -- x1 x2)
Fetch the cell pair stored at addr. x2 is stored at addr and x1 and the next consecutive cell.

2DROP

(x1 x2 --)
Drop the cell pair x1 x2 from the stack

2DUP

(x1 x2 -- x1 x2 x1 x2)
Duplicate the cell pair x1 x2

2SWAP

(x1 x2 x3 x4 -- x3 x4 x1 x2)
Exchange the two top cell pairs

3DUP

(x1 x2 x3 -- x1 x2 x3 x1 x2 x3)
Repeat the top cell triple

ABORT

(--)
Empty the stacks returning control to the outer interpreter

ABORT"

(x1 --)
Abort" <message>" if any bit of x1 is non-zero this will display the message and perform ABORT

ACCEPT

(addr n1 -- n2)
0 < n1 <=32767. Get at most n1 characters from the keyboard and put them as consecutive bytes starting at addr. The input terminates when the enter key is pressed. n2 is the number of characters received.

ALIGN

(--)
Some systems require the dictionary pointer to be aligned to specific addresses. This command aligns the pointer to the next legal address.

ALLOT

(n --)

Move the dictionary pointer n units. If n is positive this reserves space in the dictionary. If n is negative it releases space.

AND

(x1 x2 -- x3)

x3 is the bit by bit logical and of x1 with x2

ASCII

follow by string (-- c)

c is the ASCII code of the first character in the string

AT-XY

(-- x y)

Return the current screen coordinates of the cursor. The upper left corner is at (0,0). x is measured to the right, y is measured downward

BEGIN

Initiate BEGIN .. UNTIL or BEGIN .. WHILE .. REPEAT control structures

BETWEEN

(n a b -- t/f)

true if and only if a <= n <= b

BL

(-- n)

ASCII code for the space character (blank)

BODY>

(pfa -- cfa)

Move from the address of the body of a word to the execution address (code field address)

C!

(c addr --)

Store byte c at addr

C"

(-- addr)

this is followed by a string ending with delimiter ". The string is stored as a counted string at addr

C,

(c --)

Store the byte c in the next available dictionary location and increase the dictionary pointer

C@

(addr -- c)

Fetch the single byte, c, located at addr

CELL

(-- n)

n is the size, in bytes, of a cell

CELL+

(addr1 -- addr2)

add the size of a cell to addr1 to get addr2

CELLS

(n1 -- n2)

n2 is the size of n1 cells in bytes

CMOVE

(addr1 addr2 u --)

Copy u consecutive characters from the block of memory starting at addr1 to the block of memory starting at addr2. The copying is character-by-character starting from the lower address to the higher.

COMPARE

(addr1 u1 addr2 u2 -- n)

Compare the string1 (addr1 with length u1 to string2 (addr2 with length u2). Imagine the shorter string to be padded with blanks to match the length of the longer. If string1 comes before string2, then n is -1, if the strings are equal then n=0, if string1 comes after string2 then n is 1.

COUNT

(cs-addr -- addr2 u)

cs-address is the address of a counted string. addr2 is the address of the characters and u is the count of characters.

CR

cause subsequent output to be at the start of the next line

CREATE

follow by spaces and a name

skip leading spaces and parse name up to a space delimiter. Make a dictionary entry for the name. CREATE xxx will make a new word xxx whose action will be to return the address of its body

DECR

(addr --)

Decrement the integer stored at addr

DEPTH

(-- n)

n is the number of single cell values that were in the stack before n was placed on it

DO

(lim start --)

used in DO ... LOOP control structure. This compiles a word which sets up limits for a loop. start is the starting index, the loop continues when the current index crosses the boundary between lim-1 and lim (the test occurs at LOOP)

DOES>

Used with CREATE to make a defining word for words with a family behavior. : <name> ... CREATE <words> DOES> <words> ; When this word is evoked it will make a new dictionary entry from the following word; the words between CREATE and DOES> describe what happens when the new word is compiled; The words after DOES> describe the action. Example: : CONS CREATE , DOES> @ ; defines a word which acts exactly like CONSTANT.

DROP

(x --)

remove x from the stack

DUP

(x -- x x)

duplicate x

ELSE

Used in IF .. ELSE .. THEN construct. A IF B ELSE C THEN (where A,B,C are collections of words and A leaves a number on the stack) will execute B if the number on the stack is non-zero, C if it is zero (in either case execution continues after THEN)

EMIT

(c --)

c is the ASCII code of a character (printable or control). EMIT displays the character

ERASE

(addr u --)

set to zero u consecutive bytes starting at addr

EVALUATE

(addr u --)

the string of u characters starting at addr is treated as Forth input.

Example: S" 3 DUP + ." EVALUATE will print 6

EXECUTE

(cfa --)

Remove the code address of a word from the stack and execute the word. At the keyboard this does the same as typing the word -- but this can be used to implement jump tables

EXIT

This immediately returns execution to the word from which the current word was called. Before exiting from a DO..LOOP the word UNLOOP must be used before EXIT to discard the loop parameters

FALSE

(-- 0)

In Forth 0 is taken as FALSE and any non-zero value as TRUE. This word is used to make clear that a boolean is at issue

FILL

(addr u char --)

Store char in each of u consecutive memory locations beginning at addr

HERE

(-- addr)

addr is the position of the dictionary pointer

I

(-- n)

used within a DO .. LOOP construct, n is the current value of the loop index

IF

used in IF .. THEN and IF .. ELSE .. THEN this marks the location of a forward branch

INCLUDE

INCLUDE <filename>

read and interpret a file consisting of Forth code. This makes the file the input stream

INCLUDED

(addr u --)

include the file whose filename at addr is u characters long

INCR

(addr --)

Increment the integer stored at addr

J

Used in a nested loop construct DO .. DO .. LOOP .. LOOP. Within the inner loop this is the current value of the index of the next outermost loop

KEY

(-- char)

Wait for a key press and return the corresponding character

KEY?

(-- t/f)

return TRUE if and only if a key has been pressed since the last operation that consumes key presses. This indicates that a key character is in the buffer but does not clear the buffer

LEAVE

Inside a DO .. LOOP construct this causes an immediate exit from the loop

LOOP

Completes a DO .. LOOP construction. This resolves the forward branch left by DO and compiles a word which tests the loop parameters for the exit condition

LSHIFT

(x1 u -- x2)

perform a logical left shift of u places of the bits of x1 to give x2. The vacant bits are replaced by 0

MAX

(n1 n2 -- n3)

n3 is the larger of n1 and n2

MIN

(n1 n2 -- n3)

n3 is the smaller of n1 and n2

MOD

(a b -- r)

a = bq + r in floored division r has the same sign as b, in centered division it has the same sign as a.

MOVE

(addr1 addr2 u --)

Move u bytes from addr1 to addr2. After the move the u consecutive units starting at addr1 are exactly what was after addr1 before the move

NEGATE

(n - n')

n' is -n

NIP

(x1 x2 -- x2)

Drop the element below the top of the stack

NOT

(x1 -- x2)

This is not in the ANSI standard because two meanings have been available. In some versions of Forth, NOT exchanges TRUE and FALSE (equivalent to 0=) in others it inverts the bits of x1 (if TRUE is represented by a number with all bits set, the latter does the former)

NUMBER

(cstr -- d)

Convert the counted string cstr to a double precision number

NUMBER?

(addr u -- d t/f)

Attempt to convert the string at addr of u characters to a number. TRUE if and only if the conversion is successful. d is the result, as double precision

OFF

(var --)

Set the value of variable var to zero (FALSE)

ON

(var --)

Set the value of variable var to -1 (TRUE)

OR

(x1 x2 -- x3)

x3 is bitwise inclusive or of x1 with x2

OVER

(x1 x2 -- x1 x2 x1)

Copy the second element on the stack to the top

PAGE

clear the display, return cursor to home position

PARSE

(char -- c-addr u) <text>

Read the following input text until a delimiter char is found. Store the text in the input buffer. Return c-addr, the address where the text is stored, and u the length of the parsed string.

PICK

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

Remove u. Copy the xu to the top of the stack.

PLACE

(addr len dest --)

convert the string of len characters at addr to a counted string placed at dest

POSTPONE

POSTPONE <name>

used in compilation of an immediate word, this does not compile the action of <name> into the current word, but causes the current word to compile the action of <name>

QUERY

Gets input from the keyboard up to <ret> and places it into the terminal input buffer, resetting the input buffer pointer

R@

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

Copy the top of the return stack to the parameter stack

R>

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

Remove the element on top of the return stack and put it on the parameter stack

RECURSE

When used in definition this compiles the action of the current definition (a self-reference)

REPEAT

Used in BEGIN .. WHILE .. REPEAT. Control is sent back to point after BEGIN

ROT

(a b c -- b c a)

Rotate third stack element to top

S"

s" <text>" (-- addr u)

Parse the text up to delimiting ", store the text, return the address and count of the stored string

SPACE

(--)

Display one space

SPACES

(n --)

Display n spaces

STATE

(-- addr)

addr is the address of a cell which contains the compilation state flag

SWAP

(a b -- b a)

Swap top two elements on stack

TAB

(n --)

Advance cursor to next column which is a multiple of n. (non-standard word)

THEN

Terminate IF..THEN construct, this sets target of conditional branch instruction compiled by IF

TO

<n> TO <value>

IF X is a VALUE datatype, 3 TO X makes 3 the value of X

TRUE

(-- true)

Puts the value of a true flag on the stack (the integer with all bits set = -1 for most versions of Forth)

TUCK

(a b -- b a b)

tuck the top stack element under the element next below

TYPE

(addr cnt --)

Print the string of cnt bytes starting at addr

U<

(u1 u2 -- t/f)

Interpret the top two stack elements as unsigned integers. Returns true if and only if u1 < u2

U>

(u1 u2 -- t/f)

Interpret the top two stack elements as unsigned integers. Returns true if and only if u1 > u2

UM/MOD

(ud u1 -- ur uq)

Divide the double precision number ud by u1 giving the remainder ur and quotient uq. All values and arithmetic are unsigned

UNTIL

Terminate BEGIN -- UNTIL construct. This compiles a conditional branch back to the place marked by BEGIN

UPC

(c -- c')

c' is the character c converted to upper case (non-standard)

UPPER

(addr len --)

convert the string of len characters at addr to upper case (non-standard)

WHILE

Use in BEGIN .. WHILE .. REPEAT to compile a conditional branch to place marked just after REPEAT if false

WITHIN

(n a b -- t/f)
TRUE if $a \leq n < b$

WORD

<delim> WORD <text> -- addr
Skip leading delimiters. Parse the text up to delim. Store as a counted string in a transient area and return the address addr. Returns a string of length zero if no delim is found

XOR

(x1 x2 -- x3)
x3 is the bit by bit exclusive or of x1 and x2

Backslash is used for comments. It is a Forth word whose action is to discard all characters to the end of the line.
