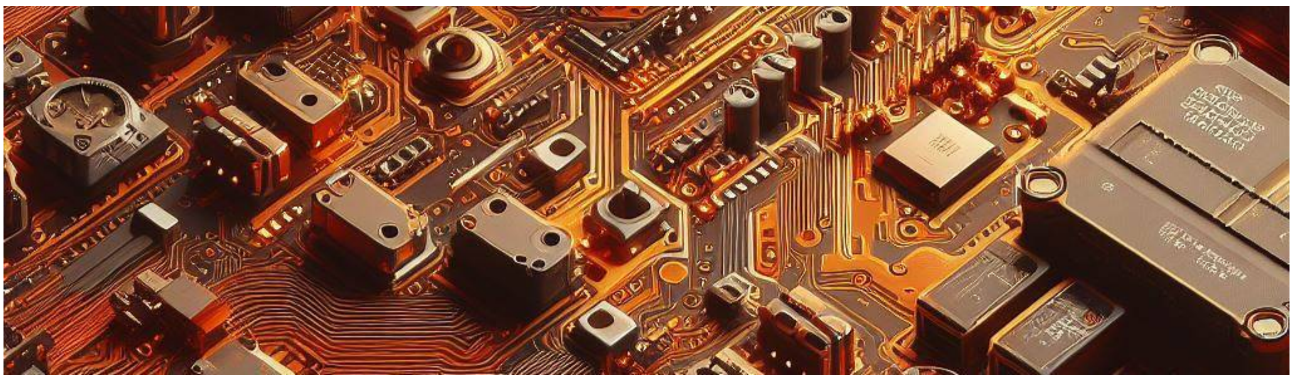
eForth Linux

Reference manual

version 1.0 - 24 novembre 2023



Author

- Marc PETREMANN petremann@arduino-forth.com

Content

FORTH words by usage

arithmetic integer

* (n1 n2 -- n3)
*/ (n1 n2 n3 -- n4)
*/MOD (n1 n2 n3 -- n4 n5)
+ (n1 n2 -- n3)
- (n1 n2 -- n1-n2)
/mod (n1 n2 -- n3 n4)
1+ (n -- n+1)
1- (n -- n-1)
2* (n -- n*2)
2/ (n -- n/2)
4* (n -- n*4)
4/ (n -- n/4)
ARSHIFT (x1 u -- x2)
mod (n1 n2 -- n3)
negate (n -- -n')

arithmetic real

#f+s (r:r)
1/F (r -- r')
F* (r1 r2 -- r3)
F** (r_val r_exp -- r)
F+ (r1 r2 -- r3)
F- (r1 r2 -- r3)
F/ (r1 r2 -- r3)
F0< (r -- fl)
F0= (r -- fl)
F>S (r -- n)
FABS (r1 -- r1')
FATAN2 (r-tan -- r-rad)
fconstant (comp: r -- <name> | exec: -- r)
FCOS (r1 -- r2)
FEXP (ln-r -- r)
FLN (r -- ln-r)
FLOOR (r1 -- r2)
FMAX (r1 r2 -- r1|r2)
FMIN (r1 r2 -- r1|r2)
FNEGATE (r1 -- r1')
FSIN (r1 -- r2)
FSINCOS (r1 -- rcos rsin)
fsqrt (r1 -- r2)
pi (-- r)
S>F (n -- r: r)

block edit list

a (n --)
copy (from to --)
d (n --)
e (n --)
editor (--)
flush (--)
list (n --)
load (n --)
n (--)
open-blocks (addr len --)
p (--)
r (n --)
thru (n1 n2 --)
update (--)
use (-- <name>)
wipe (--)

chars strings

# (n1 -- n2)
#FS (r:r --)
#s (n1 -- n=0)
<# (n --)
extract (n base -- n c)
F>NUMBER? (addr len -- real:r fl)
hold (c --)
r| (comp: -- <string> | exec: addr len)
s" (comp: -- <string> | exec: addr len)
s>z (a n -- z)
str (n -- addr len)
str= (addr1 len1 addr2 len2 -- fl)
z" (comp: -- <string> | exec: -- addr)
z>s (z -- a n)
[char] (comp: -- name | exec: -- xchar)

comparaison logical

0< (x1 --- fl)
0<> (n -- fl)
0= (x -- fl)
< (n1 n2 -- fl)
<= (n1 n2 -- fl)
<> (x1 x2 -- fl)
= (n1 n2 -- fl)
> (x1 x2 -- fl)
>= (x1 x2 -- fl)
f< (r1 r2 -- fl)
f<= (r1 r2 -- fl)
f<> (r1 r2 -- fl)
f= (r1 r2 -- fl)
f> (r1 r2 -- fl)
f>= (r1 r2 -- fl)
invert (x1 -- x2)
max (n1 n2 -- n1|n2)
min (n1 n2 -- n1|n2)
OR (n1 n2 -- n3)
XOR (n1 n2 -- n3)

definition words

: (comp: -- <word> | exec: --)
:noname (-- cfa-addr)
; (--)
constant (comp: n -- <name> | exec: -- n)
CREATE (comp: -- <name> | exec: -- addr)
defer (-- <vec-name>)
DOES> (comp: -- | exec: -- addr)
fvariable (comp: -- <name> | exec: -- addr)
value (comp: n -- <valname> | exec: -- n)
variable (comp: -- <name> | exec: -- addr)
vocabulary (comp: -- <name> | exec: --)

display

. (n --)
." (-- <string>)
.s (--)
? (addr -- c)
cr (--)
emit (x --)
esc (--)
f. (r --)
f.s (--)
ip. (--)
n. (n --)
normal (--)
ok (--)
prompt (--)
see (-- name>)
space (--)
spaces (n --)
type (addr c --)
u. (n --)
vlist (--)
words (--)

files words

BIN (mode -- mode')
block (n -- addr)
block-fid (-- n)
block-id (-- n)
cat (-- <path>)
CLOSE-FILE (fileid -- ior)
common-default-use (--)
cp (-- "src" "dst")
CREATE-FILE (a n mode -- fh ior)
DELETE-FILE (a n -- ior)
dump-file (addr len addr2 len2 --)
edit (-- <filename>)
file-exists? (addr len --)
FILE-POSITION (fileid -- ud ior)
FILE-SIZE (fileid -- ud ior)
FLUSH-FILE (fileid –- ior)
include (-- <:name>)
included? (addr len -- f)
ls (-- "path")
mv (-- "src" "dest")
OPEN-FILE (addr n opt -- n)
R/O (-- 0)
R/W (-- 2)
READ-FILE (a n fh -- n ior)
REPOSITION-FILE (ud fileid -- ior)
required (addr len --)
RESIZE-FILE (ud fileid -- ior)
rm (-- "path")
save-buffers (--)
touch (-- "path")
W/O (-- 1)
WRITE-FILE (a n fh -- ior)

loop and branch

+loop (n --)
?do (n1 n2 --)
aft (--)
begin (--)
CASE (--)
else (--)
ENDCASE (--)
ENDOF (--)
for (n --)
if (fl --)
loop (--)
next (--)
OF (n --)
repeat (--)
then (--)
unloop (--)
until (fl --)
while (fl --)
[ELSE] (--)
[IF] (fl --)
[THEN] (--)

memory access

! (n addr --)
2! (n1 n2 addr --)
2@ (addr -- d)
@ (addr -- n)
c! (c addr --)
c@ (addr -- c)
FP@ (-- addr)
m! (val shift mask addr --)
m@ (shift mask addr -- val)
UL@ (addr -- un)
UW@ (addr -- un[2exp0..2exp16-1])

stack manipulation

-rot (n1 n2 n3 -- n3 n1 n2)
2drop (n1 n2 n3 n4 -- n1 n2)
2dup (n1 n2 -- n1 n2 n1 n2)
>r (S: n -- R: n)
?dup (n -- n | n n)
drop (n --)
dup (n -- n n)
FDROP (r1 --)
FDUP (r1 -- r1 r1)
FNIP (r1 r2 -- r2)
FOVER (r1 r2 -- r1 r2 r1)
FSWAP (r1 r2 -- r1 r2)
nip (n1 n2 -- n2)
over (n1 n2 -- n1 n2 n1)
r> (R: n -- S: n)
R@ (-- n)
rdrop (S: -- R: n --)
swap (n1 n2 -- n2 n1)

forth

## ! n addr --

Store n to address.

```
VARIABLE TEMPERATURE
32 TEMPERATURE !
```

# d1 -- d2

Perform a division modulo the current numeric base and transform the rest of the division into a string of characters. The character is dropped in the buffer set to running `<#`

```
: hh ( c -- adr len)
    base @ >r  hex
    <# # # #>
    r> base !
  ;
 3 hh type  \ display 03
26 hh type  \ display 1a
```

#! --

Behaves like `\`

Serves as a text file header to indicate to the operating system (Unix-like) that this file is not a binary file but a script (set of commands). On the same line is specified the interpreter allowing to execute this script.

```
#! /usr/bin/env ueforth
```

#> n -- addr len

Drop n. Make the pictured numeric output string available as a character string. *addr* and *len* specify the resulting character string.

```
\ display address in format: NNNN-NNNN
: DUMPaddr ( n -- )
    <# # # # #  [char] - hold # # # # #>
    type
  ;
```

#FS r --

Converts a real number to a string. Used by `f.`

(local) a n --

Word used to manage the creation of local variables.

+ n1 n2 -- n3

Leave sum of n1 n2 on stack.

```
7 15 +    \ leave 22 on stack
```

+! n addr --

Increments the contents of the memory address pointed to by addr.

```
variable valX
15 valX !
1 valX +!
valX ?   \ display 16
```

+to n --- <valname>

add n to the content of *valname*

```
5 value FINAL-SCORE
1 +to FINAL-SCORE       \ increment content of FINAL-SCORE
FINAL-SCORE  .          \ display 6
```

, x --

Append x to the current data section.

- n1 n2 -- n1-n2

Subtract two integers.

```
6 3  - .  \ display 3
-6 3 - .  \ display -9
```

-rot n1 n2 n3 -- n3 n1 n2

Inverse stack rotation. Same action than `rot rot`

. n --

Remove the value at the top of the stack and display it as a signed single precision integer.

```
1 .                         \ display 1
1 2 .                       \ display 2  leave 1 on stack
1 2 + .                     \ display 3  addition 1 and 2, leave nothing on the
stack
6 3  * .                    \ display 18
```

```
7 3  * 6 3 * + .        \ display 39 operation (7*3)+(6*3)
```

." -- <string>

The word `."` can only be used in a compiled definition.

At runtime, it displays the text between this word and the delimiting **"** character end of string.

```
: TITLE
    ."      GENERAL MENU" CR
    ."      ============" ;
: line1
    ." 1.. Enter datas" ;
: line2
    ." 2.. Display datas" ;
: last-line
    ." F.. end program" ;
: MENU ( ---)
    title cr cr cr
    line1 cr cr
    line2 cr cr
    last-line ;
```

0= x -- fl

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

```
5 0=      \ push  FALSE on stack
0 0=      \ push  TRUE  on stack
```

2! d addr --

Store double precision value in memory address addr.

2@ addr -- d

Leave on stack double precision value d stored at address addr.

: comp: -- <word> | exec: --

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name, called a "colon definition". Enter compilation state and start the current definition.

Subsequent execution of `NOM` performs the execution sequence words compiled in his "colon" definition.

After `: NOM`, the interpreter enters compile mode. All non-immediate words are compiled in the definition, the numbers are compiled in literal form. Only immediate words or placed in square brackets (words `[` and `]`) are executed during compilation to help control it.

A "colon" definition remains invalid, ie not inscribed in the current vocabulary, as long as the interpreter did not execute ; (semi-colon).

```
: NAME  nomex1 nomex2 ... nomexn ;
NAME  \ execute NAME
```

:noname -- cfa-addr

Define headerless forth code. cfa-addr is the code execution of a definition.

```
:noname s" Saterday" ;
:noname s" Friday" ;
:noname s" Thursday" ;
:noname s" Wednesday" ;
:noname s" Tuesday" ;
:noname s" Monday" ;
:noname s" Sunday" ;

create (ENday) ( --- addr)
        , , , , , , ,

:noname s" Samedi" ;
:noname s" Vendredi" ;
:noname s" Jeudi" ;
:noname s" Mercredi" ;
:noname s" Mardi" ;
:noname s" Lundi" ;
:noname s" Dimanche" ;

create (FRday) ( --- addr)
        , , , , , , ,

defer (day)

: ENdays
    ['] (ENday) is (day) ;

: FRdays
    ['] (FRday) is (day) ;

3 value dayLength
: .day
    (day)
    swap cell *
    + @ execute
    dayLength ?dup if
        min
    then
    type
;
ENdays
0 .day \ display Sun
1 .day \ display Mon
2 .day \ display Tue
FRdays  ok
0 .day \ display Dim
1 .day \ display Lun
2 .day \ display Mar
```

; --

Immediate execution word usually ending the compilation of a "colon" definition.

```
: NAME
    nomex1 nomex2
    nomexn ;
```

<# n --

Marks the start of converting a integer number to a string of characters.

```
\ display address in format: NNNN-NNNN
: DUMPaddr ( n -- )
    <# # # # #  [char] - hold # # # # #>
    type
  ;

\ display byte in format: NN
: DUMPbyte ( c -- )
    <# # # #>
    type
  ;
```

>body cfa -- pfa

pfa is the data-field address corresponding to cfa.

>r S: n -- R: n

Transfers n to the return stack.

This operation must always be balanced with `r>`

```
\ display n in binary format
: b. ( n -- )
    base @ >r
    binary .
    r> base !
  ;
```

? addr -- c

Displays the content of any variable or address.

?do n1 n2 --

Executes a `do loop` or `do +loop` loop if n1 is strictly greater than n2.

```
DECIMAL
: qd ?DO I LOOP ;
   789   789 qd \
 -9876 -9876 qd \
     5     0 qd \  display: 0 1 2 3 4
```

?dup   n -- n | n n

Duplicate n if n is not nul.

@   addr -- n

Retrieves the integer value n stored at address addr.

```
TEMPERATURE @
```

abort   --

Raises an exception and interrupts the execution of the word and returns control to the interpreter.

accept   addr n -- n

Accepts n characters from the keyboard (serial port) and stores them in the memory area pointed to by addr.

```
create myBuffer 100 allot
myBuffer 100 accept     \ on prompt, enter: This is an example
myBuffer swap type      \ display: This is an example
```

afliteral   r:r --

Compiles a real number. Used by `fliteral`

aft   --

Jump to THEN in FOR-AFT-THEN-NEXT loop 1st time through.

```
: test-aft1 ( n -- )
  FOR
    ."  for "      \ first iteration
    AFT
      ."  aft "    \ following iterations
    THEN
    I .           \ all iterations
  NEXT ;
3 test-aft1
\ display for 3 aft 2 aft 1 aft 0
```

again   --

Mark the end on an infinit loop of type `begin ... again`

```
: test ( -- )
    begin
        ." Diamonds are forever" cr
    again
 ;
```

aligned addr1 -- addr2

addr2 is the first aligned address greater than or equal to addr1.

allot n --

Reserve n address units of data space.

also --

Duplicate the vocabulary at the top of the vocabulary stack.

analogRead pin -- n

Analog read from 0-4095.

Use to read analog value. `analogRead` has only one argument which is a pin number of the analog channel you want to use.

```
\ solar cell connected on pin G34
34 constant SOLAR_CELL

: init-solar-cell ( -- )
    SOLAR_CELL input pinMode
  ;

: solar-cell-read ( -- n )
    SOLAR_CELL analogRead
  ;
```

AND n1 n2 --- n3

Execute logic AND.

The words `AND`, `OR`, and `XOR` perform operations binary **bitwise** logic on single-precision integers at the top of the data stack.

```
0   0 and .      \ display 0
0  -1   and .    \ display 0
-1    0 and .    \ display 0
-1    -1   and . \ display -1
```

ansi --

Selects the `ansi` vocabulary.

ARSHIFT x1 u -- x2

Arithmetic right shift of u

asm --

Select the `asm` vocabulary.

assembler --

Alias for `asm`.

Select the `asm` vocabulary.

assert fl --

For tests and asserts.

at-xy x y --

Positions the cursor at the x y coordinates.

```
: menu ( -- )
    page
    10 4 at-xy
      0 bg 7 fg   ."  Your choice, press: " normal
    12 5 at-xy  ." A - accept"
    12 6 at-xy  ." D - deny"
  ;
```

BIN mode -- mode'

Modify a file-access method to include BINARY.

BINARY --

Select binary base.

```
255 BINARY .    \ display 11111111
DECIMAL         \ return to decimal base
```

bl -- 32

Value 32 on stack.

```
\ definition of bl
: bl   ( -- 32 )
    32
  ;
```

blank addr len --

If len is greater than zero, store byte $20 (space) in each of len consecutive characters of memory beginning at addr.

block n -- addr

Get addr 1024 byte for block n.

block-fid -- n

Flag indicating the state of a block file.

block-id -- n

Pointer to a block file.

buffer n - addr

Get a 1024 byte block without regard to old contents.

bye --

Stop eForth.

c! c addr --

Stores an 8-bit c value at address addr.

c, c --

Append c to the current data section.

```
create myDatas
    36 c,   42 c,   24 c,   12 c,
myDatas 1+ c@    \ push 42 on stack
```

c@ addr -- c

Retrieves the 8-bit c value stored at address addr.

```
35 constant PINB  \ adresse registre données PIN de PORT B sur Arduino
PINB c@           \ empile contenu registre pointé par PINB
```

CASE --

```
: day ( n -- addr len )
    CASE
        0 OF s" Sunday"     ENDOF
        1 OF s" Monday"     ENDOF
        2 OF s" Tuesday"    ENDOF
        3 OF s" Wednesday"  ENDOF
        4 OF s" Thursday"   ENDOF
        5 OF s" Friday"     ENDOF
        6 OF s" Saturday"   ENDOF
    ENDCASE
  ;
```

cat -- <path>

Display the file content.

```
cat /spiffs/dumpTool.txt
\ display content of file dumpTool.txt
\ if this file was edited and saved in /spiffs/ file system
```

cell   -- 8

Return number of bytes for integer number.

cell+   n -- n'

Increment `CELL` content.

cell/   n -- n'

Divide `CELL` content.

cells   n -- n'

Multiply `CELL` content.

Allows you to position yourself in an array of integers.

```
create table ( -- addr)
    1 ,  5 ,  10 ,  50 ,  100 , 500 ,
\ get values indexed 0 and 3 from table
table 0 cells + @ .   \ display 1
table 3 cells + @ .   \ display 50
```

char   -- <string>

Word used in interpretation only.

Leave the first character of the string following this word.

```
char v .            \ display: 118 (ascii code for "v")
char house .        \ display: 104 - code for "h"
```

cmove   c-addr1 c-addr2 u --

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

code   -- <:name>

Defines a word whose definition is written in assembly language.

```
code my2*
  a1 32 ENTRY,
  a8 a2 0 L32I.N,
  a8 a8 1 SLLI,
  a8 a2 0 S32I.N,
  RETW.N,
end-code
```

copy   from to --

Copy contents of block 'from' to block 'to'

cp   -- "src" "dst"

Copy "src" file to "dst".

DECIMAL   --

Selects the decimal number base. It is the default digital base when FORTH starts.

```
HEX
FF DECIMAL .    \ display 255
```

do   n1 n2 --

Set up loop control parameters with index n2 and limit n1.

```
: testLoop
    256 32 do
        I emit
    loop
  ;
```

dump   a n --

Dump a memory region

This version is not very interesting. Prefer this version:

DUMP tool for ESP32Forth

echo   -- addr

Variable.

else   --

Word of immediate execution and used in compilation only. Mark a alternative in a control structure of the type `IF ... ELSE ... THEN`

At runtime, if the condition on the stack before `IF` is false, there is a break in sequence with a jump following `ELSE`, then resumed in sequence after `THEN`.

```
: TEST ( ---)
    CR ." Press a key " KEY
    DUP 65 122 BETWEEN
    IF
```

```
        CR 3 SPACES ." is a letter "
    ELSE
        DUP 48 57 BETWEEN
        IF
            CR 3 SPACES ." is a digit "
        ELSE
            CR 3 SPACES ." is a special character "
        THEN
    THEN
    DROP ;
```

ENDCASE --

Marks the end of a `CASE OF ENDOF ENDCASE` structure

```
: day ( n -- addr len )
    CASE
        0 OF s" Sunday"     ENDOF
        1 OF s" Monday"     ENDOF
        2 OF s" Tuesday"    ENDOF
        3 OF s" Wednesday"  ENDOF
        4 OF s" Thursday"   ENDOF
        5 OF s" Friday"     ENDOF
        6 OF s" Saturday"   ENDOF
    ENDCASE
  ;
```

ENDOF --

Marks the end of a `OF .. ENDOF` choice in the control structure between `CASE ENDCASE`.

```
: day ( n -- addr len )
    CASE
        0 OF s" Sunday"     ENDOF
        1 OF s" Monday"     ENDOF
        2 OF s" Tuesday"    ENDOF
        3 OF s" Wednesday"  ENDOF
        4 OF s" Thursday"   ENDOF
        5 OF s" Friday"     ENDOF
        6 OF s" Saturday"   ENDOF
    ENDCASE
  ;
```

erase addr len --

If len is greater than zero, store byte $00 in each of len consecutive characters of memory beginning at addr.

extract n base -- n c

Extract the least significant digit of n. Leave on the stack the quotient of n/base and the ASCII character of this digit.

F*   r1 r2 -- r3

Multiplication of two real numbers.

```
1.35e 2.2e F*
F.  \ display 2.969999
```

F+   r1 r2 -- r3

Addition of two real numbers.

```
3.75e 5.21e F+
F.  \ display 8.960000
```

F-   r1 r2 -- r3

Subtraction of two real numbers.

```
10.02e 5.35e F-
F.  \ display 4.670000
```

f.   r --

Displays a real number. The real number must come from the real stack.

```
pi f.    \ display 3.141592
```

f.s   --

Display content of reals stack.

```
2.35e
36.512e
f.s  \ display: <2> 2.350000 36.511996
```

F/   r1 r2 -- r3

Division of two real numbers.

```
22e 7e F/   \ PI approximation
F.          \ display 3.142857
```

fconstant   comp: r -- <name> | exec: -- r

Defines a constant of type real.

```
9.80665e fconstant g    \ gravitation constant on Earth
g f.  \ display 9.806649
```

FDROP r1 --

Drop real r1 from real stack.

FDUP r1 -- r1 r1

Duplicate real r1 from real stack.

file-exists? addr len --

Tests if a file exists. The file is designated by a character string.

```
s" /spiffs/dumpTool.txt" file-exists?
```

FILE-POSITION fileid -- ud ior

Return file position, and return ior=0 on success.

FILE-SIZE fileid -- ud ior

Get size in bytes of an open file as a double number, and return ior=0 on success.

fill addr len c --

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

fliteral r:r --

Immediate execution word. Compiles a real number.

flush --

Save and empty all buffers.

After editing the contents of a block file, running `flush` ensures that changes to the contents of blocks are saved.

forget -- <name>

Searches the dictionary for a name following it. If it is a valid word, trim dictionary below this word. Display an error message if it is not a valid word.

FP@ -- addr

Retrieves the stack pointer address of the reals.

freq chan freq --

sets frequency freq n to channel chan.

Use`ledcWriteTone`

fsqrt   r1 -- r2

Square root of a real number.

```
64e fsqrt
F.          \ display 8.000000
```

FSWAP   r1 r2 -- r1 r2

Reverses the order of the two values on the real stack.

```
3.75e 5.21e FSWAP
F.   \ display 3.750000
F.   \ display 5.210000
```

graphics   --

select `graphics` vocabulary.

handler   -- addr

Ticket for interruptions.

here   -- addr

Leave the current data section dictionary pointer.

The dictionary pointer is incremented as the words are compiled and variables and data tables are defined.

```
here u.       \ display 1073709120
: null ;
here u.       \ display 1073709144
```

HEX   --

Selects the hexadecimal digital base.

```
255 HEX .   \ display FF
DECIMAL     \ return to decimal base
```

hld   -- addr

Pointer to text buffer for number output.

hold   c --

Inserts the ASCII code of an ASCII character into the character string initiated by `<#`.

i   -- n

n is a copy of the current loop index.

```
: mySingleLoop ( -- )
    cr
    10 0 do
        i .
    loop
  ;
mySingleLoop
\ display 0 1 2 3 4 5 6 7 8 9
```

if fl --

The word **IF** is executed immediately.

IF marks the start of a control structure for type **IF..THEN** or **IF..ELSE..THEN**.

```
: WEATHER? ( fl ---)
    IF
        ." Nice weather "
    ELSE
        ." Bad weather "
    THEN ;
1 WEATHER?    \ display: Nice weather
0 WEATHER?    \ display: Bad weather
```

immediate --

Make the most recent definition an immediate word.

Sets the compile-only lexicon bit in the name field of the new word just compiled. When the interpreter encounters a word with this bit set, it will not execute this word, but spit out an error message. This bit prevents structure words to be executed accidentally outside of a compound word.

include -- <:name>

Loads the contents of a file designated by <name>.

The word **include** can only be used from the terminal.

To load the contents of a file from another file, use the word **included**.

```
include /spiffs/dumpTool.txt
\ load content of dump.txt

\ to include a file from an other file, use included
s" /spiffs/dumpTool.txt" included
```

included addr len --

Loads the contents of a file from the SPIFFS filesystem, designated by a character string.

The word **included** can be used in a FORTH listing stored in the SPIFFS file system.

For this reason, the filename to load should always be preceded by */spiffs/*

```
s" /spiffs/dumpTool.txt" included
```

included? addr len -- f

Tests whether the file named in the character string has already been compiled.

internalized --

select `internalized` vocabulary.

j -- n

n is a copy of the next-outer loop index.

```
: myDoubleLoop ( -- )
    cr
    10 0 do
        cr
        10 0 do
            i 1+  j 1+  * .
        loop
    loop
  ;
myDoubleLoop
\ display:
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

k -- n

n is a copy of the next-next-outer loop index.

```
: myTripleLoop ( -- )
    cr
    5 0 do
        cr
        5 0 do
            cr
            5 0 do
                i 1+  j 1+  k 1+  * * .
            loop
        loop
    loop
  ;
myTripleLoop
```

latestxt -- xt

Stacks the execution code (cfa) address of the last compiled word.

```
: txtxtx  ;
latest
>name type  \ display txtxtx
```

leave --

Prematurely terminates the action of a `do..loop` loop.

```
256 string LoRaRX
s" +RCV=55,27,this is a transmission test,-36,40" LoRaRX $!

: scan$ ( char addr len -- )
    0 do
        2dup i + c@ = if
            i cr .
            leave
        then
    loop
    2drop
  ;


char , LoRaRX scan$
```

list n --

Displays the contents of block n.

load n --

Evaluate a block.

`load` preceded by the number of the block you want to execute and/or compile the content. To compile the content of our block 0, we will execute `0 load`

loop --

Add one to the loop index. If the loop index is then equal to the loop limit, discard the loop parameters and continue execution immediately following the loop. Otherwise continue execution at the beginning of the loop.

```
: ascii-chars ( -- )
    128 32 do
        i emit
    loop
  ;
```

ls -- "path"

Displays the contents of a file path.

```
ls /spiffs/  \ display:
dump.txt
```

mv -- "src" "dest"

Rename "src" file to "dst".

normal --

Disables selected colors for display.

OCTAL --

Selects the octal digital base.

```
255 OCTAL .      \ display 377
DECIMAL          \ return to decimal base
```

OF n --

Marks a `OF .. ENDOF` choice in the control structure between `CASE ENDCASE`

If the tested value is equal to the one preceding `OF`, the part of code located between `OF ENDOF` will be executed.

```
: day ( n -- addr len )
    CASE
        0 OF s" Sunday"     ENDOF
        1 OF s" Monday"     ENDOF
        2 OF s" Tuesday"    ENDOF
        3 OF s" Wednesday"  ENDOF
        4 OF s" Thursday"   ENDOF
        5 OF s" Friday"     ENDOF
        6 OF s" Saturday"   ENDOF
    ENDCASE
  ;
```

open-blocks addr len --

Open a block file. The default blocks file is *blocks.fb*

order --

Print the vocabulary search order.

```
Serial
order    \ display Serial
```

PARSE c "string" -- addr count

Parse the next word in the input stream, terminating on character c. Leave the address and character count of word. If the parse area was empty then count=0.

pi -- r

PI constant.

```
pi
F.          \ display 3.141592
\ perimeter of a circle, for r = 5.2   ---   P = 2 π R
5.2e 2e F*  pi  F*
F.          \ display 32.672560
```

precision -- n

Pseudo constant determining the display precision of real numbers.

Initial value 6.

If we reduce the display precision of real numbers below 6, the calculations will be when even performed with precision to 6 decimal places.

```
precision . \ display 6
pi f.       \ display 3.141592
4 set-precision
precision . \ display 4
pi f.       \ display 3.1415
```

PSRAM? -- -1|0

Stacks -1 if PSRAM memory is available.

r> R: n -- S: n

Transfers n from the return stack.

This operation must always be balanced with >r

```
\ display n in binary format
: b. ( n -- )
    base @ >r
    binary .
    r> base !
  ;
```

R@ -- n

Copies the contents of the top of the return stack onto the data stack.

rdrop S: -- R: n --

Discard top item of return stack.

recurse --

Append the execution semantics of the current definition to the current definition.

The usual example is the coding of the factorial function.

```
: FACTORIAL ( +n1 -- +n2)
   DUP 2 < IF DROP 1 EXIT THEN
   DUP 1- RECURSE *
;
```

remaining -- n

Indicates the remaining space for your definitions.

```
remaining .        \ display 76652
: t ;
remaining .        \ display 76632
```

repeat --

End a indefinite loop `begin.. while.. repeat`

REPOSITION-FILE ud fileid -- ior

Set file position, and return ior=0 on success

required addr len --

Loads the contents of the file named in the character string if it has not already been loaded.

```
s" /spiffs/dumpTool.txt" required
```

RESIZE-FILE ud fileid -- ior

Set the size of the file to ud, an unsigned double number. After using `RESIZE-FILE`, the result returned by `FILE-POSITION` may be invalid

restore -- <:name>

Restore a snapshot from a file.

rm -- "path"

Delete the file designed in file path.

rot n1 n2 n3 -- n2 n3 n1

Rotate three values on top of stack.

S>F n -- r: r

Converts an integer to a real number and transfers this real to the stack of reals.

```
35 S>F
F.   \ display 35.000000
```

s>z   a n -- z

Convert a counted string string to null terminated (copies string to heap)

save   -- <:name>

Saves a snapshot of the current dictionary to a file.

see   -- name>

Decompile a FORTH definition.

```
see include
: include  bl PARSE included ;

see space
: space  bl emit ;
```

set-precision   n --

Changes the display precision of Real numbers.

```
pi f.    \ display 3.141592
2 set-precision
pi f.    \ display 3.14
```

sf,   r --

Compile a real number.

SFLOAT   -- 4

Constant. value 4.

sfloat+   addr -- addr+4

Increments a memory address by the length of a real.

space   --

Display one space.

```
\ definition of space
: space  ( -- )
    bl emit
  ;
```

spaces   n --

Displays the space character n times.

Defined since version 7.071

state   -- fl

Compilation state. State can only be changed by `[` and `]`.

-1 for compiling, 0 for interpreting

str=   addr1 len1 addr2 len2 -- fl

Compare two strings. Leave true if they are identical.

```
s" 123"   s" 124"
str = .        \ display 0
s" 156"   s" 156"
str= .         \ display -1
```

then   --

Immediate execution word used in compilation only. Mark the end a control structure of type `IF..THEN` or `IF..ELSE..THEN` .

thru   n1 n2 --

Loads the contents of a block file, from block n1 to block n2.

tib   -- addr

returns the address of the the terminal input buffer where input text string is held.

```
tib >in @ type
\ display:
tib >in @
```

to   n --- <valname>

`to` assign new value to *valname*

tone   chan freq --

sets frequency freq n to channel chan.

Use `ledcWriteTone`

touch   -- "path"

Create "path" file if it doesn't exist.

UL@ addr -- un

Retrieve a unsigned value.

unloop --

Stop a do..loop action. Using `unloop` before `exit` only in a do..loop structure.

```
: example ( -- )
    100 0 do
        cr i .
        key bl = if
            unloop exit
        then
    loop
  ;
```

until fl --

End of `begin.. until` structure.

```
: myTestLoop ( -- )
    begin
        key dup .
        [char] A =
    until
  ;
myTestLoop \ end loop if key A pressed
```

update --

Used for block editing. Forces the current block to the modified state.

use -- <name>

Use "name" as the blockfile.

```
USE /spiffs/foo
```

used -- n

Specifies the space taken up by user definitions. This includes already defined words from the FORTH dictionary.

UW@ addr -- un[2exp0..2exp16-1]

Extracts the least significant 16 bits part of a memory zone pointed by addr.

```
variable valX
hex 10204080 valX !
valX UW@ .      \ display 4080
valX 2 + UW@ .  \ display 1020
```

vlist --

Display all words from a vocabulary.

```
graphics vlist  \ display content of Serial vocabulary
```

words --

List the definition names in the first word list of the search order. The format of the display is implementation-dependent.

x11 --

`x11` vocabulary

z>s z -- a n

Convert a null terminated string to a counted string.

[--

Enter interpretation state. `[` is an immediate word.

```
\ source for [
: [
    0 state !
    ; immediate
```

['] comp: -- <name> | exec: -- addr

Use in compilation only. Immediate execution.

Compile the cfa of <name>

```
serial \ Select Serial vocabulary

: serial2-type ( a n -- )
    Serial2.write drop ;

: typeToLoRa ( -- )
    0 echo !    \ disable display echo from terminal
    ['] serial2-type is type
  ;

: typeToTerm ( -- )
    ['] default-type is type
    -1 echo !   \ enable display echo from terminal
  ;
```

[ELSE] --

Mark a part of conditional sequence in `[IF] ... [ELSE] ... [THEN]`.

] --

Return to compilation.] is an immediate word.

With FlashForth, the words [and] allow you to use assembly code, subject to first compiling an assembler.

```
\ Load constant $1234 to top of stack
: a-number ( -- 1234 )
    dup                     \ Make space for new TOS value
    [ R24 $34 ldi, ]
    [ R25 $12 ldi, ]
;
```

graphics

Words defined in `graphics` vocabulary.

```
window heart vertical-flip viewport scale translate }g g{ screen>g box
color pressed? pixel height width event last-char last-key mouse-y mouse-x
RIGHT-BUTTON MIDDLE-BUTTON LEFT-BUTTON FINISHED TYPED RELEASED PRESSED
MOTION EXPOSED RESIZED IDLE internals
```

event -- 0

Constant. Default Value 0

EXPOSED -- 2

Constant. Value 2

FINISHED -- 7

Constant. Value 7

height -- 0

Value. Default Value 0

IDLE -- 0

Constant. Value 0

last-char -- 0

Constant. Default Value 0

last-key -- 0

Constant. Default Value 0

LEFT-BUTTON -- 255

Constant. Value 255

MIDDLE-BUTTON -- 254

Constant. Value 254

MOTION -- 3

Constant. Value 3

mouse-x -- 0

Constant. Default Value 0

mouse-y -- 0

Constant. Default Value 0

pixel w h --

Draws a pixel in position w h

PRESSED -- 4

Constant. Value 4

RELEASED -- 5

Constant. Value 5

RESIZED -- 1

Constant. Value 1

RIGHT-BUTTON -- 253

Constant. Value 253

TYPED -- 6

Constant. Value 6

width -- 0

Value. Default Value 0

Graphics → internals

Mots définis dans le vocabulaire `graphics → internals`

```
raw-heart heart-ratio heart-initialize cmax! cmin! heart-end heart-start
heart-size heart-steps heart-f raw-box g> >g gp gstack hline ty tx sy sx
```

Graphics → internals

Words defined in `graphics → internals` vocabulary.

```
raw-heart heart-ratio heart-initialize cmax! cmin! heart-end heart-start
heart-size heart-steps heart-f raw-box g> >g gp gstack hline ty tx sy sx
key-state! key-state key-count backbuffer
```

Internals

Words defined in `internals` vocabulary.

```
ca! errno CALLCODE CALL0 CALL1 CALL2 CALL3 CALL4 CALL5 CALL6 CALL7 CALL8
CALL9 CALL10 CALL11 CALL12 CALL13 CALL14 CALL15 DOFLIT S>FLOAT? fill32
'heap 'context 'latestxt 'notfound 'heap-start 'heap-size 'stack-cells
'boot 'boot-size 'tib 'argc 'argv 'runner 'throw-handler NOP BRANCH 0BRANCH
DONEXT DOLIT DOSET DOCOL DOCON DOVAR DOCREATE DODOES ALITERAL LONG-SIZE
S>NUMBER? 'SYS YIELD EVALUATE1 'builtins internals-builtins autoexec boot-set-title
e' @line grow-blocks use?! common-default-use block-data block-dirty clobber
clobber-line include+ path-join included-files raw-included include-file
sourcedirname sourcefilename! sourcefilename sourcefilename# sourcefilename&
starts../ starts./ dirname ends/ default-remember-filename remember-filename
restore-name save-name forth-wordlist setup-saving-base 'cold park-forth
park-heap saving-base crtype cremit cases (+to) (to) --? }? ?room scope-create
do-local scope-clear scope-exit local-op scope-depth local+! local! local@
<>locals locals-here locals-area locals-gap locals-capacity ?ins. ins.
vins. onlines line-pos line-width size-all size-vocabulary vocs. voc. voclist
voclist-from see-all >vocnext see-vocabulary nonvoc? see-xt ?see-flags
see-loop see-one indent+! icr see. indent mem= ARGS_MARK -TAB +TAB NONAMED
BUILTIN_FORK SMUDGE IMMEDIATE_MARK dump-line ca@ cell-shift cell-base cell-mask
#f+s internalized BUILTIN_MARK zplace $place free. boot-prompt raw-ok [SKIP]'
[SKIP] ?stack sp-limit input-limit tib-setup raw.s $@ digit parse-quote
leaving, leaving )leaving leaving( value-bind evaluate&fill evaluate-buffer
arrow ?arrow. ?echo input-buffer immediate? eat-till-cr wascr *emit *key
notfound last-vocabulary voc-stack-end xt-transfer xt-hide xt-find& scope
```

internals → internalized

Words defined in `internals → internalized` vocabulary.

```
flags'or! LEAVE LOOP +LOOP ?DO DO NEXT FOR AFT REPEAT WHILE ELSE IF THEN
AHEAD UNTIL AGAIN BEGIN cleave
```

posix

Words defined in `posix` vocabulary.

```
FNDELAY F_SETFL fcntl CLOCK_BOOTTIME_ALARM CLOCK_REALTIME_ALARM CLOCK_BOOTTIME
CLOCK_MONOTONIC_COARSE CLOCK_REALTIME_COARSE CLOCK_MONOTONIC_RAW
CLOCK_THREAD_CPUTIME_ID
CLOCK_PROCESS_CPUTIME_ID CLOCK_MONOTONIC CLOCK_REALTIME timespec clock_gettime
0777 SIGPIPE SIGBUS SIGKILL SIGINT SIGHUP SIG_IGN SIG_DFL EPIPE EAGAIN
d0=ior d0<ior 0=ior 0<ior stdin-key stdout-write O_NONBLOCK O_APPEND O_TRUNC
O_CREAT O_RDWR O_WRONLY O_RDONLY MAP_ANONYMOUS MAP_FIXED MAP_PRIVATE PROT_EXEC
PROT_WRITE PROT_READ PROT_NONE SEEK_END SEEK_CUR SEEK_SET stderr stdout
stdin errno .d_name .d_type readdir closedir opendir getwd rmdir mkdir
chdir signal usleep realloc sysfree malloc rename unlink mprotect munmap
mmap waitpid wait fork sysexit fsync ftruncate lseek write read close creat
open sign-extend shared-library sysfunc sofunc calls dlopen 'dlopen RTLD_NOW
RTLD_LAZY
```

shared-library   comp: z-string --

Defines a pointer to an external library.

```
z" libX11.so" shared-library xlib
```

sockets

Words defined in `sockets` vocabulary.

```
sockaccept ip. ip# ->h_addr ->addr! ->addr@ ->port! ->port@ sockaddr l,
s, bs, SO_REUSEADDR SOL_SOCKET sizeof(sockaddr_in) AF_INET SOCK_RAW SOCK_DGRAM
SOCK_STREAM gethostbyname recvmsg recvfrom recv sendmsg sendto send setsockopt
poll sockaccept connect listen bind socket
```

AF_INET -- 2

Constant. value 2.

bind sock addr addrlen -- 0/err

Bind a name to a socket.

structures

Words defined in `structures` vocabulary.

```
field struct-align align-by struct last-struct long ptr i64 i32 i16 i8
typer last-align
```

field   comp: n -- <:name>

Definition word for a new field in a structure.

```
also structures
struct esp_partition_t
  ( Work around changing struct layout )
  esp_partition_t_size 40 >= [IF]
    ptr field p>gap
  [THEN]
  ptr field p>type
  ptr field p>subtype
  ptr field p>address
  ptr field p>size
  ptr field p>label
```

i16   -- 2

Pseudo constant defined by `typer`. At runtime, drops the size of the datatype and puts a copy of that size in the `last-align` variable

i32   -- 4

Pseudo constant defined by `typer`. At runtime, drops the size of the datatype and puts a copy of that size in the `last-align` variable

i64   -- 8

Pseudo constant defined by `typer`. At runtime, drops the size of the datatype and puts a copy of that size in the `last-align` variable

i8   -- 1

Pseudo constant defined by `typer`. At runtime, drops the size of the datatype and puts a copy of that size in the `last-align` variable

last-struct   -- addr

Variable pointing to the last defined structure.

long -- 8

Pseudo constant defined by `typer`. At runtime, drops the size of the datatype and puts a copy of that size in the `last-align` variable

ptr -- 8

Pseudo constant defined by `typer`. At runtime, drops the size of the datatype and puts a copy of that size in the `last-align` variable

struct comp: -- <:name>

Definition word for structures.

```
also structures
struct esp_partition_t
```

typer comp: n1 n2 -- <name> | exec: -- n

Definition word for `i8` `i16` `i32` `i64` `ptr` `long`

tasks

Words defined in `tasks` vocabulary.

```
main-task .tasks task-list
```

.tasks   --

Display list active tasks.

```
.tasks  \ display:
main-task
```

main-task   -- task

Main task. Leave pointer task on stack

task-list   -- addr

Variable. Point to tasks list.

x11

Words defined in `x11` vocabulary.

```
GenericEvent MappingNotify ClientMessage ColormapNotify SelectionNotify
SelectionRequest SelectionClear PropertyNotify CirculateRequest CirculateNotify
ResizeRequest GravityNotify ConfigureRequest ConfigureNotify ReparentNotify
MapRequest MapNotify UnmapNotify DestroyNotify CreateNotify VisibilityNotify
NoExpose GraphicsExpose Expose KeymapNotify FocusOut FocusIn LeaveNotify
EnterNotify MotionNotify ButtonRelease ButtonPress KeyRelease KeyPress
xevent# OwnerGrabButtonMask ColormapChangeMask PropertyChangeMask FocusChangeMask
SubstructureRedirectMask SubstructureNotifyMask ResizeRedirectMask StructureNotifyMask
VisibilityChangeMask ExposureMask KeymapStateMask ButtonMotionMask Button5MotionMask
Button4MotionMask Button3MotionMask Button2MotionMask Button1MotionMask
PointerMotionHintMask PointerMotionMask LeaveWindowMask EnterWindowMask
ButtonReleaseMask ButtonPressMask KeyReleaseMask KeyPressMask xmask NoEventMask
xexposure xconfigure xmotion xkey xbutton xany bool time win xevent-size
NULL ZPixmap XYPixmap XYBitmap XFillRectangle XSetBackground XSetForeground
XDrawString XSelectInput XPutImage XNextEvent XMapWindow XLookupString
XFlush XDestroyImage XDefaultVisual XDefaultDepth XCreateSimpleWindow XCreateImage
XCreateGC XCheckMaskEvent XRootWindow XDefaultScreen XDefaultColormap XScreenOfDisplay
XDisplayOfScreen XWhitePixel XBlackPixel XOpenDisplay xlib
```

x11

Button1MotionMask -- 256

Constant. Value 256

Button2MotionMask -- 512

Constant. Value 512

Button3MotionMask -- 1024

Constant. Value 1024

Button4MotionMask -- 2048

Constant. Value 2048

Button5MotionMask -- 4096

Constant. Value 4096

ButtonMotionMask -- 8192

Constant. Value 8192

ButtonPressMask -- 4

Constant. Value 4

ButtonReleaseMask **-- 8**

Constant. Value 8

ColormapChangeMask **-- 8388608**

Constant. Value 8388608

EnterWindowMask **-- 16**

Constant. Value 16

ExposureMask **-- 32768**

Constant. Value 32768

FocusChangeMask **-- 2097152**

Constant. Value 2097152

KeymapStateMask **-- 16384**

Constant. Value 16384

KeyPressMask **-- 1**

Constant. Value 1

KeyReleaseMask **-- 2**

Constant. Value 2

LeaveWindowMask **-- 32**

Constant. Value 32

NULL **-- 0**

Constant. Value 0

OwnerGrabButtonMask **-- 16777216**

Constant. Value 16777216

PointerMotionHintMask **-- 128**

Constant. Value 128

PointerMotionMask **-- 64**

Constant. Value 64

PropertyChangeMask -- 4194304

Constant. Value 4194304

ResizeRedirectMask -- 262144

Constant. Value 262144

StructureNotifyMask -- 131072

Constant. Value 131072

SubstructureNotifyMask -- 524288

Constant. Value 524288

SubstructureRedirectMask -- 1048576

Constant. Value 1048576

VisibilityChangeMask -- 65536

Constant. Value 65536

xany --

Vocabulary `xany`

XBlackPixel a n -- n

Get the black pixel value.

xbutton --

`xbutton` vocabulary.

`xbutton` is a subvocabulary of the `x11` vocabulary

xconfigure --

`xconfigure` vocabulary.

`xconfigure` is a subvocabulary of the `x11` vocabulary

XDisplayOfScreen a -- a

Get display structure of specified Screen structure.

xevent-size -- 256

Constant. value 0

```
\ definition:
```

```
32 cells constant xevent-size
```

xexposure --

`xexposure` vocabulary.

`xexposure` is a subvocabulary of the `x11` vocabulary

xkey --

`xkey` vocabulary.

`xkey` is a subvocabulary of the `x11` vocabulary

xlib --

Pointer linking to the x11 library.

```
\ refer to X11 documentation
\ here, definition for: XOpenDisplay
z" XOpenDisplay" 1 xlib XOpenDisplay ( a -- a )
```

xmotion --

`xmotion` vocabulary.

`xmotion` is a subvocabulary of the `x11` vocabulary

XOpenDisplay a -- a

Specifies the connection to the X server.

Specifies the hardware display name, which determines the display and communications domain to be used. On a POSIX-conformant system, if the display_name is NULL, it defaults to the value of the DISPLAY environment variable.

```
0 value myDisplay
NULL XOpenDisplay to myDisplay
```

XScreenOfDisplay a n -- a

Get display structure of specified Screen structure.

XWhitePixel a n -- n

Returns the white pixel value for the specified screen

x11 → xany

Words defined in `x11 → xany` vocabulary.

```
->window ->display ->send_event ->serial ->type XAnyEvent
```