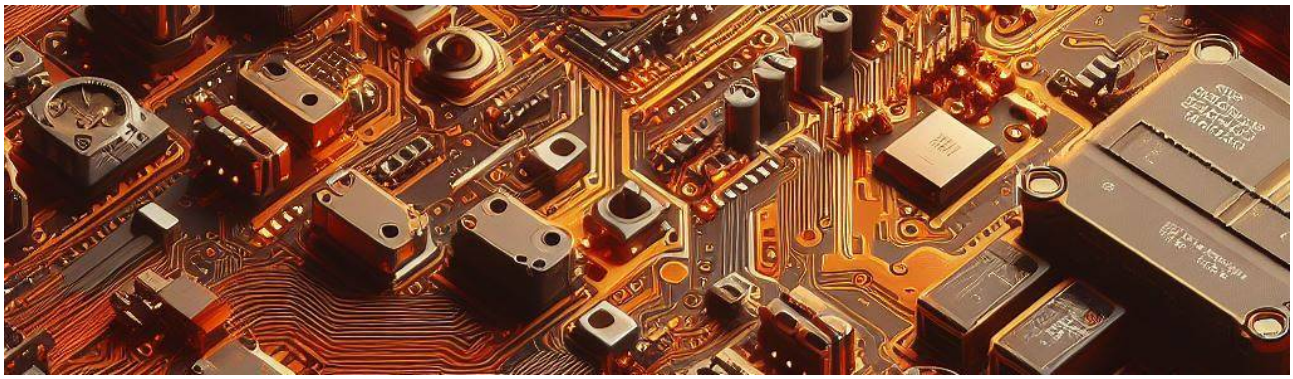
eForth Linux

Manuel de référence

version 1.0 - 24 novembre 2023



Auteur

- Marc PETREMANN petremann@arduino-forth.com

Table des matières

Mots FORTH par utilisation

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 --

Stocke une valeur entière n à l'adresse addr.

```
VARIABLE TEMPERATURE
32 TEMPERATURE !
```

d1 -- d2

Effectue une division modulo la base numérique courante et transforme le reste de la division en chaîne de caractère. Le caractère est déposé dans le tampon définit à l'exécution de <#

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

#! --

Se comporte comme \

Sert d'en-tête de fichier texte pour indiquer au système d'exploitation (de type Unix) que ce fichier n'est pas un fichier binaire mais un script (ensemble de commandes). Sur la même ligne est précisé l'interpréteur permettant d'exécuter ce script.

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

#> n -- addr len

Dépile n. Rend la chaîne de sortie numérique mise en forme sous forme de chaîne de caractères. *addr* et *len* spécifient la chaîne de caractères résultante.

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

#FS r --

Convertit un nombre réel en chaîne de caractères. Utilisé par `f.`

(local) a n --

Mot utilisé pour gérer la création des variables locales.

+ n1 n2 -- n3

Laisse la somme de n1 et n2 sur la pile.

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

+! n addr --

Incrémente le contenu de l'adresse mémoire pointé par addr.

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

+to n --- <valname>

incrémente de n le contenu de *valname*

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

, x --

Ajoute x à la section de données actuelle.

- n1 n2 -- n1-n2

Soustration de deux entiers.

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

-rot n1 n2 n3 -- n3 n1 n2

Rotation inverse de la pile. Action similaire à `rot rot`

. n --

Dépile la valeur au sommet de la pile et l'affiche en tant qu'entier simple précision signé.

```
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>

Le mot ." est utilisable exclusivement dans une définition compilée.

A l'exécution, il affiche le texte compris entre ce mot et le caractère " délimitant la fin de chaîne de caractères.

```
: 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

Teste si l'entier simple précision situé au sommet de la pile est nul.

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

2! d addr --

Stocke la valeur double précision d à l'adresse addr.

2@ addr -- d

Empile la valeur double précision d stockée à l'adresse addr.

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

Ignore les délimiteurs d'espace de début. Analyse le nom délimité par un espace. Crée une définition pour le , appelée "définition deux-points". Entre dans l'état de compilation et démarre la définition actuelle.

L'exécution ultérieure de NOM réalise l'enchainement d'exécution des mots compilés dans sa définition "deux-points".

Après : NOM, l'interpréteur entre en mode compilation. Tous les mots non immédiats sont compilés dans la définition, les nombres sont compilés sous forme litérale. Seuls les mots immédiats ou placés entre crochets (mots [et]) sont exécutés pendant la compilation pour permettre de contrôler celle-ci.

Une définition "deux-points" reste invalide, c'est à dire non inscrite dans le vocabulaire courant, tant que l'interpréteur n'a pas exécuté ; (point-virgule).

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

:noname -- cfa-addr

Définit un code FORTH sans en-tête. cfa-addr est l'adresse d'exécution d'une définition.

```
: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
```

; --

Mot d'exécution immédiate terminant habituellement la compilation d'une définition "deux-points".

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

<# n --

Marque le début de la conversion d'un nombre entier en chaîne de caractères.

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

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

>body cfa -- pfa

convertit l'adresse cfa en adresse pfa (Parameter Fieds Address)

>r S: n -- R: n

Transfère n vers la pile de retour.

Cette opération doit toujours être équilibrée avec `r>`

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

? addr -- c

Affiche le contenu d'une variable ou d'une adresse quelconque.

?do n1 n2 --

Exécute une boucle `do loop` ou `do +loop` si n1 est strictement supérieur à 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

Duplique n si n n'est pas nul.

@ addr -- n

Récupère la valeur entière n stockée à l'adresse addr.

```
TEMPERATURE @
```

abort --

Génère une exception et interrompt l'exécution du mot et rend la main à l'interpréteur.

accept addr n -- n

Accepte n caractères depuis le clavier (port série) et les stocke dans la zone mémoire pointée par 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 --

Compile un nombre réel. Utilisé par `fliteral`

aft --

Saute à THEN dans une boucle FOR-AFT-THEN-NEXT lors de la première itération.

```
: 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 --

Marque la fin d'une boucle infinie de type `begin ... again`

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

```
;
```

aligned addr1 -- addr2

addr2 est la première adresse alignée plus grande ou égale à addr1.

allot n --

Réserve n adresses dans l'espace de données.

also --

Duplique le vocabulaire au sommet de la pile des vocabulaires.

analogRead pin -- n

Lecture analogique, intervalle 0-4095.

Utilisé pour lire la valeur analogique. `analogRead` n'a qu'un seul argument qui est un numéro de broche du canal analogique que vous souhaitez utiliser.

```
\ 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

Effectue un ET logique.

Les mots `AND`, `OR` et `XOR` effectuent des opérations logiques binaires **bit à bit** sur les entiers simple précision situés au sommet de la pile de données.

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

ansi --

Sélectionne le vocabulaire `ansi`.

ARSHIFT x1 u -- x2

Décalage arithmétique à droite de u fois

asm --

Sélectionne le vocabulaire `asm`.

assembler --

Alias pour `asm`.

Sélectionne le vocabulaire `asm`.

assert fl --

Pour tests et assertions.

at-xy x y --

Positionne le curseur aux coordonnées x y.

```
: 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'

Modifie une méthode d'accès au fichier pour inclure BINARY.

BINARY --

Sélectionne la base numérique binaire.

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

bl -- 32

Dépose 32 sur la pile de données.

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

blank addr len --

Si len est supérieur à zéro, range un caractère de code $20 (espace) dans toute la zone de longueur len à l'adresse mémoire commençant à addr.

block n -- addr

Récupère l'adresse d'un bloc n de 1024 octets.

block-fid -- n

Flag indiquant l'état d'un fichier de blocs.

block-id -- n

Pointeur vers un fichier de blocs.

buffer n - addr

Obtient un bloc de 1024 octets sans tenir compte de l'ancien contenu.

bye --

Interropt eForth.

c! c addr --

Stocke une valeur 8 bits c à l'adresse addr.

c, c --

Ajoute c à la section de données actuelle.

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

c@ addr -- c

Récupère la valeur 8 bits c stockée à l'adresse addr.

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

CASE --

Marque le début d'une structure **CASE OF ENDOF 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
  ;
```

cat -- <path>

Affiche le contenu du fichier.

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

cell -- 8

Retourne le nombre d'octets pour un nombre entier.

cell+ n -- n'

Incrémente contenu de **CELL**.

cell/ n -- n'

Divise contenu de **CELL**.

cells n -- n'

Multiplie contenu de **CELL**.

Permet de se positionner dans un tableau d'entiers.

```
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>

Mot utilisable en interprétation seulement.

Empile le premier caractère de la chaîne qui suit ce mot.

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

cmove c-addr1 c-addr2 u --

Si u est supérieur à zéro, copier u caractères consécutifs de l'espace de données commençant à c-addr1 vers celui commençant à c-addr2, en procédant caractère par caractère des adresses inférieures aux adresses supérieures.

code -- <:name>

Définit un mot dont la définition est écrite en assembleur.

```
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 --

Copie le contenu du bloc 'from' vers le bloc 'to'

cp -- "src" "dst"

Copie le fichier "src" dans "dst".

DECIMAL --

Sélectionne la base numérique décimale. C'est la base numérique par défaut au démarrage de FORTH.

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

do n1 n2 --

Configure les paramètres de contrôle de boucle avec l'index n2 et la limite n1.

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

dump a n --

Visualise une zone mémoire.

Cette version est peu intéressante. Préférez cette version:

DUMP tool for ESP32Forth

echo -- addr

Variable.

else --

Mot d'exécution immédiate et utilisé en compilation seulement. Marque une alternative dans une structure de contrôle du type `IF ... ELSE ... 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 --

Marque la fin d'une structure **CASE OF ENDOF 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
  ;
```

ENDOF --

Marque la fin d'un choix **OF .. ENDOF** dans la structure de contrôle entre **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 --

Si len est supérieur à zéro, range un caractère de code $00 dans toute la zone de longueur len à l'adresse mémoire commençant à addr.

extract n base -- n c

Extrait le digit de poids faible de n. Laisse sur la pile le quotient de n/base et le caractère ASCII de ce digit.

F* r1 r2 -- r3

Multiplication de deux nombres réels.

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

F+ r1 r2 -- r3

Addition de deux nombres réels.

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

F- r1 r2 -- r3

Soustraction de deux nombres réels.

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

f. r --

Affiche un nombre réel. Le nombre réel doit venir de la pile des réels.

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

f.s --

Affiche le contenu de la pile des réels.

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

F/ r1 r2 -- r3

Division de deux nombres réels.

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

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

Définit une constante de type réel.

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

FDROP r1 --

Enlève le nombre réel r1 du sommet de la pile des réels.

FDUP r1 -- r1 r1

Duplique le nombre réel r1 du sommet de la pile des réels.

file-exists? addr len --

Teste si un fichier existe. Le fichier est désigné par une chaîne de caractères.

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

FILE-POSITION fileid -- ud ior

Renvoie la position du fichier et renvoie ior=0 en cas de succès

FILE-SIZE fileid -- ud ior

Récupère la taille en octets d'un fichier ouvert sous la forme d'un nombre double et renvoie ior=0 en cas de succès.

fill addr len c --

Si len est supérieur à zéro, range c dans toute la zone de longueur len à l'adresse mémoire commençant à addr.

fliteral r:r --

Mot d'exécution immédiate. Compile un nombre réel.

flush --

Enregistre et vide tous les tampons.

Après édition du contenu d'un fichier bloc, exécutez `flush` garantit que les modification du contenu des blocs sont sauvegardées.

forget -- <name>

Cherche dans le dictionnaire le mot qui suit. Si c'est un mot valide, supprime tous les mots définis jusqu'à ce mot. Affiche un message d'erreur si ce n'est pas un mot valide.

FP@ -- addr

Récupère l'adresse du pointeur de pile des réels.

freq chan freq --

définit la fréquence freq sur le canal chan.

Utilise `ledcWriteTone`

fsqrt r1 -- r2

Racine carrée d'un nombre réel.

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

FSWAP r1 r2 -- r1 r2

Inverse l'ordre des deux valeurs sur la pile des réels.

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

graphics --

sélectionne le vocabulaire `graphics`.

handler -- addr

Ticket pour les interruptions.

here -- addr

Restitue l'adresse courante du pointeur de dictionnaire.

Le pointeur de dictionnaire s'incrémente au fur et à mesure de la compilation de mots et définition des variables et tableaux de données.

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

HEX --

Sélectionne la base numérique hexadécimale.

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

hld -- addr

Pointeur vers le tampon de texte pour la sortie numérique.

hold c --

Insère le code ASCII d'un caractère ASCII dans la chaîne de caractères initiée par `<#`.

i -- n

n est une copie de l'index de boucle actuel.

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

if fl --

Le mot **IF** est d'exécution immédiate.

IF marque le début d'une structure de contrôle de type **IF..THEN** ou **IF..ELSE..THEN**.

Lors de l'exécution, la partie de définition située entre **IF** et **THEN** ou entre **IF** et **ELSE** est exécutée si le flag booléen situé au sommet de la pile de données est vrai (f<>0).

Dans le cas contraire, si le flag booléen est faux (f=0), c'est la partie de définition située entre **ELSE** et **THEN** qui sera exécutée. S'il n'y a pas de **ELSE**, l'exécution se poursuit après **THEN**.

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

immediate --

Rend la définition la plus récente comme mot immédiat.

Définit le bit de lexique de compilation uniquement dans le champ de nom du nouveau mot compilé. Lorsque l'interpréteur rencontre un mot avec ce bit défini, il ne l'exécutera pas, mais transmet un message d'erreur. Ce bit empêche l'exécution des mots de structure en dehors d'une définition de mot.

include -- <:name>

Charge le contenu d'un fichier désigné par <name>.

Le mot **include** n'est utilisable que depuis le terminal.

Pour charger le contenu d'un fichier depuis un autre fichier, utiliser le mot **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 --

Charge le contenu d'un fichier depuis le système de fichiers SPIFFS, désigné par une chaîne de caractères.

Le mot **included** peut être utilisé dans un listing FORTH stocké dans le système de fichiers SPIFFS.

Pour cette raison, le nom de fichier à charger doit toujours être précédé de */spiffs/*

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

included? addr len -- f

Teste si le fichier désigné dans la chaîne de caractères a déjà été compilé.

internalized --

sélectionne le vocabulaire **internalized**.

j -- n

n est une copie de l'index de boucle externe suivant.

```
: 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 est la copie en 3ème niveau dans une boucle do do..loop.

```
: 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

Empile l'adresse du code d'exécution (cfa) du dernier mot compilé.

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

leave --

Termine prématurément l'action d'une boucle `do..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 --

Affiche le contenu du bloc n.

load n --

Charge et interprète le contenu d'un bloc.

`load` précédé du numéro du bloc que vous souhaitez exécuter et/ou compiler le contenu. Pour compiler le contenu de notre bloc 0, nous allons exécuter `0 load`

loop --

Ajoute un à l'index de la boucle. Si l'index de boucle est alors égal à la limite de boucle, supprime les paramètres de boucle et poursuit l'exécution immédiatement après la boucle. Sinon, continue l'exécution au début de la boucle.

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

ls -- "path"

Affiche le contenu d'un chemin de fichiers.

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

mv -- "src" "dest"

Renommez le fichier "src" en "dst".

normal --

Désactive les couleurs sélectionnées pour l'affichage.

OCTAL --

Sélectionne la base numérique octale.

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

OF n --

Marque un choix **OF .. ENDOF** dans la structure de contrôle entre **CASE ENDCASE**

Si la valeur testée est égale à celle qui précède **OF**, la partie de code située entre **OF ENDOF** sera exécutée.

```
: 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 --

Ouvre un fichier de blocs. Le fichier de blocs par défaut est *blocks.fb*

order --

Affiche l'ordre de recherche de vocabulaire.

```
Serial
order   \ display Serial
```

PARSE c "string" -- addr count

Analyse le mot suivant dans le flux d'entrée, se terminant au caractère c. Laissez l'adresse et le nombre de caractères du mot. Si la zone d'analyse était vide, alors count=0.

pi -- r

Constante PI.

```
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 constante déterminant la précision d'affichage des nombres réels.

Valeur initiale 6.

Si on réduit la précision d'affichage des nombres réels en dessous de 6, les calculs seront quand même réalisés avec une précision à 6 décimales.

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

PSRAM? -- -1|0

Empile -1 si la mémoire PSRAM est disponible.

r> R: n -- S: n

Transfère n depuis la pile de retour.

Cette opération doit toujours être équilibrée avec >r

```
\ display n in binary format
: b. ( n -- )
```

```
    base @ >r
    binary .
    r> base !
  ;
```

R@ -- n

Copie sur la pile de données le contenu du sommet de la pile de retour.

rdrop S: -- R: n --

Jete l'élément supérieur de la pile de retour.

recurse --

Ajoute un lien d'exécution correspondant à la définition actuelle.

L'exemple habituel est le codage de la fonction factorielle.

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

remaining -- n

Indique l'espace restant pour vos définitions.

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

repeat --

Achève une boucle indéfinie `begin.. while.. repeat`

REPOSITION-FILE ud fileid -- ior

Définir la position du fichier et renvoyer ior=0 en cas de succès

required addr len --

Charge le contenu du fichier désigné dans la chaîne de caractères s'il n'a pas déjà été chargé.

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

RESIZE-FILE ud fileid -- ior

Définit la taille du fichier par ud, un nombre double non signé. Après avoir utilisé `RESIZE-FILE`, le résultat renvoyé par `FILE-POSITION` peut être invalide

restore -- <:name>

Restaure un instantané à partir d'un fichier.

rm -- "path"

Efface le fichier indiqué.

rot n1 n2 n3 -- n2 n3 n1

Rotation des trois valeurs au sommet de la pile.

S>F n -- r: r

Convertit un nombre entier en nombre réel et transfère ce réel sur la pile des réels.

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

s>z a n -- z

Convertir une chaîne addr len en chaine terminée par zéro.

save -- <:name>

Enregistre un instantané du dictionnaire actuel dans un fichier.

see -- name>

Décompile une définition FORTH.

```
see include
: include  bl PARSE included ;

see space
: space  bl emit ;
```

set-precision n --

Modifie la précision d'affichage des nombres Réels.

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

sf, r --

Compile un nombre réel.

SFLOAT -- 4

Constante. Valeur 4.

sfloat+ addr -- addr+4

Incrémente une adresse mémoire de la longueur d'un réel.

space --

Affiche un caractère espace.

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

spaces n --

Affiche n fois le caractère espace.

Défini depuis la version 7.071

state -- fl

Etat de compilation. L'état ne peut être modifié que par `[` et `]`.

-1 pour compilateur, 0 pour interpréteur

str= addr1 len1 addr2 len2 -- fl

Compare deux chaînes de caractères. Empile vrai si elles sont identiques.

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

then --

Mot d'exécution immédiate utilisé en compilation seulement. Marque la fin d'une structure de contrôle de type `IF..THEN` ou `IF..ELSE..THEN`.

thru n1 n2 --

Charge le contenu d'un fichier de blocs, du bloc n1 au bloc n2.

tib -- addr

renvoie l'adresse du tampon d'entrée du terminal où la chaîne de texte d'entrée est conservée.

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

to n --- <valname>

`to` affecte une nouvelle valeur à *valname*

tone chan freq --

définit la fréquence freq sur le canal chan.

Utilise `ledcWriteTone`

touch -- "path"

Créez un chemin de fichier "path" s'il n'existe pas.

UL@ addr -- un

Récupère une valeur non signée.

unloop --

Arrête une action do..loop. Utiliser `unloop` avant `exit` seulement dans une structure do..loop.

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

until fl --

Ferme une structure `begin.. until`.

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

update --

Utilisé pour l'édition de blocs. Force le bloc courant à l'état modifié.

use -- <name>

Utilise "name" comme fichier de blocs.

```
USE /spiffs/foo
```

used -- n

Indique l'espace pris par les définitions utilisateur. Ceci inclue les mots déjà définis du dictionnaire FORTH.

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

Extrait la partie poids faible 16 bits d'une zone mémoire pointée par addr.

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

vlist --

Affiche tous les mots d'un vocabulaire.

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

words --

Répertorie les noms de définition dans la première liste de mots de l'ordre de recherche. Le format de l'affichage dépend de l'implémentation.

z>s z -- a n

Convertit une chaîne terminée par zéro en chaine addr len.

[--

Entre en mode interprétation. [est un mot d'exécution immédiate.

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

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

Utilisable en compilation seulement. Exécution immédiate.

Compile le cfa de <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] --

Marque la partie de code d'une séquence **[IF] ... [ELSE] ... [THEN]**.

] --

Retour en mode compilation. **]** est un mot immédiat.

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

graphics

Mots définis dans le vocabulaire `graphics`

```
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

Constante. Valeur par défaut 0

EXPOSED -- 2

Constante. Valeur 2

FINISHED -- 7

Constante. Valeur 7

height -- 0

Value. Valeur par défaut 0

IDLE -- 0

Constante. Valeur 0

last-char -- 0

Constante. Valeur par défaut 0

last-key -- 0

Constante. Valeur par défaut 0

LEFT-BUTTON -- 255

Constante. Valeur 255

MIDDLE-BUTTON -- 254

Constante. Valeur 254

MOTION -- 3

Constante. Valeur 3

mouse-x -- 0

Constante. Valeur par défaut 0

mouse-y -- 0

Constante. Valeur par défaut 0

pixel w h --

Trace un pixel en position w h

PRESSED -- 4

Constante. Valeur 4

RELEASED -- 5

Constante. Valeur 5

RESIZED -- 1

Constante. Valeur 1

RIGHT-BUTTON -- 253

Constante. Valeur 253

TYPED -- 6

Constante. Valeur 6

width -- 0

Value. Valeur par défaut 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

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
key-state! key-state key-count backbuffer
```

Internals

Mots définis dans le vocabulaire `internals`

```
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

Mots définis dans le vocabulaire `internals → internalized`

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

posix

Mots définis dans le vocabulaire **posix**

```
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 --

Définit un pointeur vers une librairie externe.

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

sockets

Mots définis dans le vocabulaire **sockets**

```
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

Constante. Valeur 2

bind sock addr addrlen -- 0/err

Lie un nom à un socket.

structures

Mots définis dans le vocabulaire `structures`

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

i16 -- 2

Pseudo constante définie par **typer**. En exécution, dépose la taille du type de données et met une copie de cette taille dans la variable **last-align**

i32 -- 4

Pseudo constante définie par **typer**. En exécution, dépose la taille du type de données et met une copie de cette taille dans la variable **last-align**

i64 -- 8

Pseudo constante définie par **typer**. En exécution, dépose la taille du type de données et met une copie de cette taille dans la variable **last-align**

i8 -- 1

Pseudo constante définie par **typer**. En exécution, dépose la taille du type de données et met une copie de cette taille dans la variable **last-align**

last-struct -- addr

Variable pointant sur la dernière structure définie.

long -- 8

Pseudo constante définie par **typer**. En exécution, dépose la taille du type de données et met une copie de cette taille dans la variable **last-align**

ptr -- 8

Pseudo constante définie par **typer**. En exécution, dépose la taille du type de données et met une copie de cette taille dans la variable **last-align**

struct comp: -- <:name>

Mot de définition de structures.

```
also structures
struct esp_partition_t
```

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

Mot de définition pour `i8 i16 i32 i64 ptr long`

tasks

Mots définis dans le vocabulaire **tasks**

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

.tasks --

Affiche la liste des tâches actives.

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

main-task -- task

Tâche principale. Empile pointeur task

task-list -- addr

Variable pointant vers la liste des tâches.

x11

Mots définis dans le vocabulaire `x11`

```
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
```

Button1MotionMask -- 256

Constante. valeur 256

Button2MotionMask -- 512

Constante. valeur 512

Button3MotionMask -- 1024

Constante. valeur 1024

Button4MotionMask -- 2048

Constante. valeur 2048

Button5MotionMask -- 4096

Constante. valeur 4096

ButtonMotionMask -- 8192

Constante. valeur 8192

ButtonPressMask -- 4

Constante. valeur 4

ButtonReleaseMask -- 8

Constante. valeur 8

ColormapChangeMask -- 8388608

Constante. valeur 8388608

EnterWindowMask -- 16

Constante. valeur 16

ExposureMask -- 32768

Constante. valeur 32768

FocusChangeMask -- 2097152

Constante. valeur 2097152

KeymapStateMask -- 16384

Constante. valeur 16384

KeyPressMask -- 1

Constante. valeur 1

KeyReleaseMask -- 2

Constante. valeur 2

LeaveWindowMask -- 32

Constante. valeur 32

NULL -- 0

Constante. valeur 0

OwnerGrabButtonMask -- 16777216

Constante. valeur 16777216

PointerMotionHintMask -- 128

Constante. valeur 128

PointerMotionMask -- 64

Constante. valeur 64

PropertyChangeMask -- 4194304

Constante. valeur 4194304

ResizeRedirectMask -- 262144

Constante. valeur 262144

StructureNotifyMask -- 131072

Constante. valeur 131072

SubstructureNotifyMask -- 524288

Constante. valeur 524288

SubstructureRedirectMask -- 1048576

Constante. valeur 1048576

VisibilityChangeMask -- 65536

Constante. valeur 65536

xany --

Vocabulaire `xany`

XBlackPixel a n -- n

Récupère la valeur du pixel noir.

xbutton --

Vocabulaire `xbutton`.

`xbutton` est un sous-vocabulaire du vocabulaire `x11`

xconfigure --

Vocabulaire `xconfigure`.

`xconfigure` est un sous-vocabulaire du vocabulaire `x11`

XDisplayOfScreen a -- a

Récupère la structure d'affichage de la structure d'écran spécifiée.

xevent-size -- 256

Constante. valeur 0

```
\ definition:
32 cells constant xevent-size
```

xexposure --

Vocabulaire **xexposure**.

xexposure est un sous-vocabulaire du vocabulaire **x11**

xkey --

Vocabulaire **xkey**.

xkey est un sous-vocabulaire du vocabulaire **x11**

xlib --

Pointeur faisant le lien avec la librairie x11.

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

xmotion --

Vocabulaire **xmotion**.

xmotion est un sous-vocabulaire du vocabulaire **x11**

XOpenDisplay a -- a

Spécifie la connexion au serveur X.

Spécifie le nom d'affichage du matériel, qui détermine le domaine d'affichage et de communication à utiliser. Sur un système conforme à POSIX, si display_name est NULL, la valeur par défaut est la valeur de la variable d'environnement DISPLAY.

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

XScreenOfDisplay a n -- a

Récupère la structure d’affichage de la structure d’écran spécifiée.

XWhitePixel a n -- n

Récupère la valeur du pixel blanc de l'écran courant.

x11 → xany

Mots définis dans le vocabulaire `x11 → xany`

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