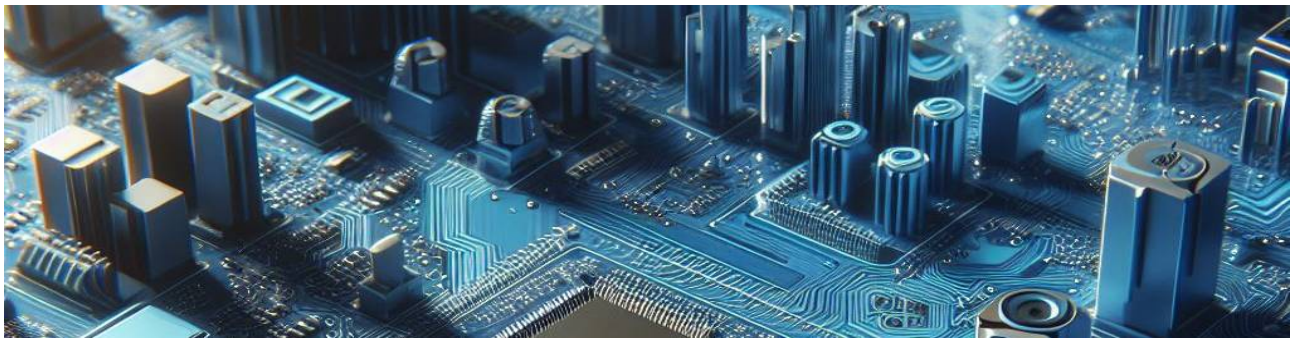
eForth Windows

Manuel de référence

version 1.1 - 21 novembre 2024



Auteur

- Marc PETREMANN petremann@arduino-forth.com

Table des matières

forth

! n addr --

Stocke une valeur entière n à l'adresse addr. n est une valeur 64 bits.

```
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 \ pour ESP32forth.

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.

#s n1 -- n=0

Convertit le reste de n1 en chaîne de caractères dans la chaîne de caractères initiée par <#.

```
: EUROS ( d1 --- str len)
    <#
    # #             \ convert € cents
    [char] , hold   \ add char "," to str buffer
    #s #>           \ convert rest after ","
  ;
15630. EUROS type   \ display 156,30 ok
```

#tib -- n

Nombre de caractères reçus dans le tampon d'entrée du terminal.

' exec: <space>name -- xt

Recherche <name> et laisse son code d'exécution (adresse).

En interprétation, `' xyz EXECUTE` équivaut à `xyz`.

'tib -- addr

Pointeur vers le tampon d'entrée du terminal.

(local) a n --

Mot utilisé pour gérer la création des variables locales.

* n1 n2 -- n3

Multiplication entière de deux nombres.

```
6 3  *   \ push 18 operation 6*3
7 3  *   \ push 21 operation 7*3
-7 3 *   \ push -21
7 -3 *   \ push -21
-7 -3 *  \ push 21
```

*/ n1 n2 n3 -- n4

Multiplie n1 par n2 produisant le résultat intermédiaire à double précision d. Divise d par n3 en donnant le quotient entier n4.

```
5000 1000 4000 */ .    \ display    1250
```

*/MOD n1 n2 n3 -- n4 n5

Multiplie n1 par n2 produisant le résultat intermédiaire à double précision d. Divise d par n3 produisant le reste entier n4 et le quotient entier n5.

```
50000 10 4001 */MOD .    \ display    124 3876
```

+ 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
```

+loop n --

Incrémente l'index de boucle de n.

Marque la fin d'une boucle `n1 0 do ... n2 +loop`.

```
: loopTest
   100 0 do
       i .
   5 +loop
  ;
loopTest  \ display 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95
```

+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 ;
```

.s --

Affiche le contenu de la pile de données, sans action sur le contenu de cette pile.

/ n1 n2 -- n3

Division de deux entiers. Laisse le quotient entier sur la pile.

```
6 3  / .  \ display 2 opération 6/3
7 3  / .  \ display 2 opération 7/3
8 3  / .  \ display 2 opération 8/3
9 3  / .  \ display 3 opération 9/3
```

/mod n1 n2 -- n3 n4

Divise n1 par n2, donnant le reste entier n3 et le quotient entier n4.

```
22 7 /MOD . .    \ display    3 1
```

0< x1 --- fl

Teste si x1 est inférieur à zéro.

0<> n -- fl

Empile -1 si n <> 0

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
```

1+ n -- n+1

Incrémente la valeur située au sommet de la pile.

1- n -- n-1

Décrémente la valeur située au sommet de la pile.

1/F r -- r'

Effectue une opération 1/r.

```
12e 1/F f.  \ display 0.083333  (op: 1/12)
```

2! d addr --

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

2* n -- n*2

Multiplie n par deux.

2/ n -- n/2

Divise n par deux.

n/2 est le résultat du décalage de n d'un bit vers le bit le moins significatif, laissant le bit le plus significatif inchangé.

```
24 2/ .    \ display    12
25 2/ .    \ display    12
26 2/ .    \ display    13
```

2@ addr -- d

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

2drop n1 n2 n3 n4 -- n1 n2

Retire la valeur double précision du sommet de la pile de données.

```
1 2 3 4 2drop   \ leave 1 2 on top of stack
```

2dup n1 n2 -- n1 n2 n1 n2

Duplique la valeur double précision n1 n2.

```
1 2 2dup  \ leave 1 2 1 2 on stack
```

3dup n1 n2 n3 -- n1 n2 n3 n1 n2 n3

Duplique les trois valeurs situées au sommet de la pile de données.

4* n -- n*4

Multiplie n par quatre.

4/ n -- n/4

Divise n par quatre.

: 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" Saturday" ;
: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 ;
```

< n1 n2 -- fl

Laisse fl vrai si n1 < n2

```
4 10 <=   \ leave -1 on stack
4 4  <=   \ leave  0 on stack
4 3  <=   \ leave  0 on stack
```

<# 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
  ;
```

<= n1 n2 -- fl

Laisse fl vrai si n1 <= n2

```
4 10 <=   \ leave -1 on stack
4 4  <=   \ leave -1 on stack
4 3  <=   \ leave  0 on stack
```

<> x1 x2 -- fl

Teste si l'entier simple précision x1 n'est pas égal à x2.

```
5 5 <>      \ push  FALSE on stack
5 4 <>      \ push  TRUE  on stack
```

= n1 n2 -- fl

Laisse fl vrai si n1 = n2

```
4 10 =   \ leave  0 on stack
4 4  =   \ leave -1 on stack
```

> x1 x2 -- fl

Teste si x1 est supérieur à x2.

>= x1 x2 -- fl

Teste si l'entier simple précision x1 est égal à x2.

```
5 5 >=     \ push  FALSE on stack
5 4 >=     \ push  TRUE  on stack
```

>body cfa -- pfa

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

>flags xt -- flags

Convertit l'adresse cfa en adresse des flags.

>in -- addr

Nombre de caractères consommés depuis TIB

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

>link cfa -- cfa2

Convertit l'adresse cfa du mot courant en adresse cfa du mot précédemment défini dans le dictionnaire.

```
' dup >link   \ get cfa from word defined before dup
>name type    \ display "XOR"
```

>link& cfa -- lfa

Transforme l'adresse d'exécution du mot courant en adresse de lien de ce mot. Cette adresse de lien pointe vers le cfa du mot défini avant ce mot.

Utilisé par `>link`

>name cfa -- nfa len

trouve l'adresse du champ de nom d'un mot à partir de son adresse de champ de code cfa.

```
' words       \ push cfa of 'words' on stack
>name         \ convert cfa of 'words' in nfa field followed by len
type          \ display 'words'
```

>name-length cfa -- n

Transforme une adresse cfa en longueur du nom du mot de cette adresse cfa. Mot utilisé par `vlist`

>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.

abort" comp: --

Affiche un message d'erreur et interrompt toute exécution FORTH en cours.

```
: abort-test
    if
        abort" stop program"
    then
    ." continue program"
  ;

0 abort-test   \ display: continue program
1 abort-test   \ display: stop program ERROR
```

abs n -- n'

Renvoie la valeur absolue de n.

```
-7 abs .    \ display 7
```

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
 ;
```

align --

Aligne le pointeur du dictionnaire de la section de données actuelle sur la limite de la cellule.

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.

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**.

argc -- n

Empile le contenu de **'argc**

ARSHIFT x1 u -- x2

Décalage arithmétique à droite de u fois

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"
  ;
```

base -- addr

Variable simple précision déterminant la base numérique courante.

La variable **BASE** contient la valeur 10 (décimal) au démarrage de FORTH.

```
DECIMAL     \ select decimal base
2 BASE !    \ selevt binary base

\ other example
: GN2 \ ( -- 16 10 )
   BASE @ >R HEX BASE @ DECIMAL BASE @ R> BASE !
  ;
```

begin --

Marque le début d'une structure **begin..until**, **begin..again** ou **begin..while..repeat**

```
: endless ( -- )
    0
    begin
        dup . 1+
    again
  ;
```

bg color[0..255] --

Sélectionne la couleur d'affichage en arrière plan. La couleur est dans l'intervalle 0..255 en décimal.

```
: testBG ( -- )
  normal
  256 0 do
    i bg ." X"
  loop ;
```

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 --

Mot défini par `defer`.

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.

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

catch cfa -- fl

Initialise une action à réaliser en cas d'exception déclenchée par `throw`.

cell -- 8

Retourne le nombre d'octets pour un entier 64 bits.

```
cell .    \ display 8
```

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"
```

CLOSE-FILE fileid -- ior

Ferme un fichier ouvert.

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.

constant comp: n -- <name> | exec: -- n

Définition d'une constante.

```
$00000001 constant SDL_INIT_TIMER          \ timer subsystem
$00000010 constant SDL_INIT_AUDIO          \ audio subsystem
$00000020 constant SDL_INIT_VIDEO          \ video subsystem; automatically
initializes the events subsystem
$00000200 constant SDL_INIT_JOYSTICK       \ joystick subsystem; automatically
initializes the events subsystem
```

context -- addr

Pointeur vers le pointeur vers le dernier mot du vocabulaire de contexte

copy from to --

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

cp -- "src" "dst"

Copie le fichier "src" dans "dst".

cr --

Affiche un retour à la ligne suivante.

```
: .result ( ---)
    ." Port analys result" cr
    . "pool detectors" cr ;
```

CREATE comp: -- <name> | exec: -- addr

Le mot **CREATE** peut être utilisé seul.

Le mot situé après **CREATE** est créé dans le dictionnaire, ici **DATAS**. L'exécution du mot ainsi créé dépose sur la pile de données l'adresse mémoire de la zone de paramètres. Dans cet exemple, nous avons compilé 4 valeurs 8 bits. Pour les récupérer, il faudra incrémenter l'adresse empilée avec la valeur de décalage de la donnée à récupérer.

```
\ Peripherals accessed by the CPU via 0x3FF40000 ~ 0x3FF7FFFF address space
\ (DPORT address) can also be accessed via 0x60000000 ~ 0x6003FFFF
\ (AHB address). (0x3FF40000 + n) address and (0x60000000 + n)
\ address access the same content, where n = 0 ~ 0x3FFFF.
create uartAhbBase
    $60000000 ,
    $60010000 ,
    $6002E000 ,

: REG_UART_AHB_BASE { idx -- addr }     \ id=[0,1,2]
    uartAhbBase  idx cell *  + @
  ;
```

CREATE-FILE a n mode -- fh ior

Crée un fichier sur le disque, renvoyant un 0 ior en cas de succès et un identifiant de fichier.

current -- cfa

Pointeur vers le pointeur du dernier mot du vocabulaire actuel

```
: test ( -- )
   ." only for test" ;
current @ @ >name type   \ display test
```

DECIMAL --

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

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

default-key -- c

Execute **win-key**.

default-key? -- fl

Execute **win-key?**.

default-type addr len --

Execute **win-type**.

defer -- <vec-name>

Définit un vecteur d'exécution différée.

vec-name exécute le mot dont le code d'exécution est stocké dans l'espace de données de vec-name.

DEFINED? -- <word>

Renvoie une valeur non nulle si le mot est défini.

```
\ other example:
DEFINED? --DAout [if] forget --DAout  [then]
create --DAout
```

definitions --

Rend courant le premier vocabulaire de contexte. Tout mot compilé est chaîné à un vocabulaire de contexte. Initialement, ce vocabulaire est **FORTH**

```
VOCABULARY LOGO         \ create vocabulary LOGO
LOGO DEFINITIONS        \ will set LOGO context vocabulary
: EFFACE
    page ;              \ create word EFFACE in LOGO vocabulary
```

depth -- n

n est le nombre de valeurs de cellule unique contenues dans la pile de données avant que n ne soit placé sur la pile.

```
\ test this after reset:
depth       \ leave 0 on stack
10 32 25
depth       \ leave 3 on stack
```

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
  ;
```

DOES> comp: -- | exec: -- addr

Le mot **CREATE** peut être utilisé dans un nouveau mot de création de mots...

Associé à **DOES>**, on peut définir des mots qui disent comment un mot est créé puis exécuté.

drop n --

Enlève du sommet de la pile de données le nombre entier simple précision qui s'y trouvait.

```
2 5 8  drop   \ leave 2 and 5 on stack
```

dump a n --

Visualise une zone mémoire.

dump-file addr len addr2 len2 --

Transfère le contenu d'une chaîne texte addr len vers le fichier pointé par addr2 len2

Le contenu du fichier */spiffs/autoexec.fs* est automatiquement interprété et/ou compilé au démarrage de ESP32Forth.

Cette fonctionnalité peut être exploitée pour paramétrer l'accès WiFi au démarrage de ESP32Forth en injectant les paramètres d'accès comme ceci:

dup n -- n n

Duplique le nombre entier simple précision situé au sommet de la pile de données.

```
: SQUARE ( n --- nE2)
    DUP * ;
5 SQUARE .    \ display 25
10 SQUARE .   \ display 100
```

echo -- addr

Variable. Contient -1 par défat. Si 0, les commandes ne sont pas affichées.

editor --

Sélectionne le vocabulaire `editor`.

- `l` liste le contenu du bloc courant
- `n` sélectionne le bloc suivant
- `p` sélectionne le bloc précédent
- `wipe` vide le contenu du bloc courant
- `d` efface la ligne n. Le numéro de ligne doit être dans l'intervalle 0..14. Les lignes qui suivent remontent vers le haut.

  Exemple: `3 D` efface le contenu de la ligne 3 et fait remonter le contenu des lignes 4 à 15.

- `e` efface le contenju de la ligne n. Le numéro de ligne doit être dans l'intervalle 0..15. Les autres lignes ne remontent pas.
- `a` insère une ligne n. Le numéro de ligne doit être dans l'intervalle 0..14. Les lignes situées après la ligne insérées redescendent.

  Exemple: `3 A test` insère **test** à la ligne 3 et fait descendre le contenu des lignes 4 à 15.
- `r` remplace le contenu de la ligne n.

  Exemple: `3 R test` remplace le contenu de la ligne 3 par **test**

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 ;
```

emit x --

Si x est un caractère graphique dans le jeu de caractères défini par l'implémentation, affiche x.

L'effet d'`EMIT` pour toutes les autres valeurs de x est défini par l'implémentation.

Lors du passage d'un caractère dont les bits de définition de caractère ont une valeur comprise entre hex 20 et 7E inclus, le caractère standard correspondant s'affiche. Étant donné que différents périphériques de sortie peuvent répondre différemment aux caractères de contrôle, les programmes qui utilisent des caractères de contrôle pour exécuter des fonctions spécifiques ont une dépendance environnementale. Chaque `EMIT` ne traite qu'avec un seul caractère.

```
65 emit    \ display A
66 emit    \ display B
```

empty-buffers --

Vide tous les tampons.

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.

evaluate addr len --

Évalue le contenu d'une chaine de caractères.

```
s" words"
evaluate   \ execute the content of the string, here: words
```

EXECUTE xt --

Exécute le mot pointé par xt.

Prend l'adresse d'exécution xt de la pile de données et exécute ce jeton. Ce mot puissant vous permet d'exécuter n'importe quel jeton qui ne fait pas partie d'une liste de jetons.

exit --

Interrompt l'exécution d'un mot et rend la main au mot appelant.

Utilisation typique: `: X ... test IF ... EXIT THEN ... ;`

En exécution, le mot `EXIT` aura le même effet que le mot `;`

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** r_val r_exp -- r

Elève un réel r_val à la puissance r_exp.

```
2e 3e f** f.    \ display 8.000000
  2e 4e f** f.    \ display 16.000000
 10e 1.5e f** f.  \ display 31.622776
```

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
```

F0< r -- fl

Teste si un nombre réel est inférieur à zéro.

```
5e F0<    \ leave  0 on stack
-3e F0<    \ leave -1 on stack
```

F0= r -- fl

Indique vrai si le réel est nul.

```
3e 3e F- F0= .  \ display -1
```

f< r1 r2 -- fl

fl est vrai si r1 < r2.

```
3.2e 5.25e f<
.  \ display -1
```

f<= r1 r2 -- fl

fl est vrai si r1 <= r2.

```
3.2e 5.25e f<=
.  \ display -1
5.25e 5.25e f<=
.  \ display -1
8.3e 5.25e f<=
.  \ display 0
```

f<> r1 r2 -- fl

fl est vrai si r1 <> r2.

```
3.2e 5.25e f<>
.  \ display -1
5.25e 5.25e f<>
.  \ display 0
```

f= r1 r2 -- fl

fl est vrai si r1 = r2.

```
3.2e 5.25e f=
.  \ display 0
5.25e 5.25e f=
.  \ display -1
```

f> r1 r2 -- fl

fl est vrai si r1 > r2.

```
3.2e 5.25e f>
.  \ display 0
```

f>= r1 r2 -- fl

fl est vrai si r1 >= r2.

```
3.2e 5.25e f>=
.  \ display 0
5.25e 5.25e f>=
.  \ display -1
8.3e 5.25e f>=
.  \ display -1
```

F>S r -- n

Convertit un réel en entier. Laisse sur la pile de données la partie entière si le réel a des parties décimales.

```
3.5e F>S .   \ display 3
```

FABS r1 -- r1'

Délivre la valeur absolue d'un nombre réel.

```
-2e FABS F.  \ display 2.000000
```

FATAN2 r-tan -- r-rad

Calcule l'angle en radian à partir de la tangente.

```
0.5e fatan2 f.    \ display 1.325917
  1e fatan2 f.    \ display 0.785398
```

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
```

FCOS r1 -- r2

Calcule le cosinus d'un angle exprimé en radians.

```
pi 2e f/      \ calc angle 90 deg
FCOS F.       \ display 0.000000
```

fdepth -- n

n est le nombre de réels dans la pile de réels.

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.

FEXP ln-r -- r

Calcule le réel correspondant à e EXP r

```
4.605170e FEXP F.    \ display 100.000018
```

fg color[0..255] --

Sélectionne la couleur d'affichage du texte. La couleur est dans l'intervalle 0..255 en décimal.

```
: testFG ( -- )
  256 0 do
    i fg ." X"
  loop ;
```

file-exists? addr len --

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

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.

FIND addr len -- xt | 0

cherche un mot dans le dictionnaire.

```
32 string t$
s" vlist" t$ $!
t$ find    \ push cfa of VLIST on stack
```

fliteral r:r --

Mot d'exécution immédiate. Compile un nombre réel.

FLN r -- ln-r

Calcule le logarithme naturel d'un nombre réel.

```
100e FLN f.    \ display 4.605170
```

FLOOR r1 -- r2

Arrondi un réel à la valeur entière inférieure.

```
45.67e FLOOR F.   \ display 45.000000
```

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.

FLUSH-FILE fileid –- ior

Essayez de forcer l'écriture de toute information mise en mémoire tampon dans le fichier référencé par fileid vers le stockage de masse. Si l'opération réussit, ior vaut zéro.

FMAX r1 r2 -- r1|r2

Laisse le plus grand réel de r1 ou r2.

```
3e 4e FMAX F.    \ display 4.000000
```

FMIN r1 r2 -- r1|r2

Laisse le plus petit réel de r1 ou r2.

```
3e 4e FMIN F.    \ display 3.000000
```

FNEGATE r1 -- r1'

Inverse le signe d'un nombre réel.

```
5e FNEGATE f.     \ display -5.000000
-7e FNEGATE f.     \ display  7.000000
```

FNIP r1 r2 -- r2

Supprime second élément sur la pile des réels.

```
2.5e 4.32e
fnip
f.s  \ display: <1> 4.320000
```

for n --

Marque le début d'une boucle `for .. next`

ATTENTION: l'index de boucle sera traité dans l'intervalle [n..0], soit n+1 itérations, ce qui est contraire aux autres versions du langage FORTH implémentant FOR..NEXT (FlashForth).

```
: myLoop ( ---)
    10 for
        r@ . cr  \ display loop index
    next
  ;
```

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.

forth --

Sélectionne le vocabulaire `FORTH` dans l'ordre de recherche des mots pour exécuter ou compiler des mots.

forth-builtins -- cfa

Point d'entrée du vocabulaire `forth`.

FOVER r1 r2 -- r1 r2 r1

Duplique le second réel sur la pile des réels.

```
2.6e 3.4e fover
```

```
f.s  \ display <3> 2.600000 3.400000 2.600000
```

fp0 -- addr

pointe vers le bas de la pile des réels.

FP@ -- addr

Récupère l'adresse du pointeur de pile des réels.

FSIN r1 -- r2

Calcule le sinus d'un angle exprimé en radians.

```
pi 2e f/      \ calc angle 90¨ deg
FSIN F.       \ display 1.000000
```

FSINCOS r1 -- rcos rsin

Calcule le cosinus et le sinus d'un angle exprimé en radians.

```
pi 4e f/
FSINCOS f. f.   \ display 0.707106 0.707106
pi 2e f/
FSINCOS f. f.   \ display 0.000000 1.000000
```

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 de ESP32Forth.

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

fvariable comp: -- <name> | exec: -- addr

Définit une variable de type flottant.

```
fvariable arc
pi 0.5e F*  \ angle 90° in radian   --  PI/2
arc SF!
arc SF@ f.  \ display 1.570796
```

graphics --

sélectionne le vocabulaire `graphics`.

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>.

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/*

included? addr len -- f

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

internalized --

sélectionne le vocabulaire **internalized**.

internals --

Sélectionne le vocabulaire **internals**.

invert x1 -- x2

Complément à un de x1. Agit sur 16 ou 32 bits selon les versions FORTH.

```
1 invert .  \ display -2
```

is --

Assigns the execution code of a word to a vectorized execution word.

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
```

key -- char

Attend l'appui sur une touche. L'appui sur une touche renvoie son code ASCII.

```
key .    \ display 97 if key "a" is active
key .    \ affiche 65 if key "A" is active
```

key? -- fl

Renvoie *vrai* si une touche est appuyée.

```
: keyLoop
    begin
    key? until
  ;
```

L! n addr --

Enregistre une valeur n. n est une valeur 32 bits.

L, n --

Mot non implanté dans eForth Windows.

Stocke une valeur au format 32 bits dans le dictionnaire.

Définition:

```
DEFINED? L, invert [IF]
\ compile 32 bits value in dictionnary
: L,  ( u -- )
    dup c,
    8 rshift dup c,
    8 rshift dup c,
    8 rshift dup c,
    drop
  ;
[THEN]
```

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`.

list n --

Affiche le contenu du bloc n.

literal x --

Compile la valeur x comme valeur litérale.

```
: valueReg ( --- n)
   [ 36 2 * ] literal ;

\ equivalent to:
: valueReg ( --- n)
   72 ;
```

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.

```
: myLoop
  128 32 do
    i emit
  loop ;
```

LSHIFT x1 u -- x2

Décalage vers la gauche de u bits de la valeur x1.

```
8 2 lshift .  \ display 32
```

max n1 n2 -- n1|n2

Laisse le plus grand non signé de u1 et u2.

min n1 n2 -- n1|n2

Laisse min de n1 et n2

mod n1 n2 -- n3

Divise n1 par n2, laisse le reste simple précision n3.

La fonction modulo peut servir à déterminer la divisibilité d'un nombre par un autre.

```
21 7 mod . \ display 0
22 7 mod . \ display 1
23 7 mod . \ display 2
```

```
24 7 mod . \ display 3

: DIV? ( n1 n2 ---)
    OVER OVER MOD CR
    IF
        SWAP . ." is not "
    ELSE
        SWAP . ." is "
    THEN
    ." divisible by " .
 ;
```

ms n --

Attente en millisencondes.

Pour les attentes longues, définir un mot d'attente en secondes.

```
500 ms \ delay for 1/2 second

: seconds ( n --)
    0
    for
        1000 ms
    next
  ;
12 seconds \ delay for 12 seconds
```

ms-ticks -- n

Impulsions système. Une impulsion par milliseconde.

Utile pour mesurer le temps d'exécution d'une définition.

```
: ms-ticks ( -- n )
    GetTickCount ;
```

mv -- "src" "dest"

Renommez le fichier "src" en "dst".

n. n --

Affiche toute valeur n sous sa forme décimale.

negate n -- -n'

Le complément à deux de n.

```
5 negate .   \ display -5
```

next --

Marque la fin d'une boucle `for .. next`

```
: myLoop
  24 for
    r@ .
  next ;
myLoop   \ display: 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

nip n1 n2 -- n2

Enlève n1 de la pile.

nl -- 10

Dépose 10 sur la pile de données.

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
  ;
```

ok --

Affiche la version du langage FORTH.

```
ok
```

```
\ display: uEforth
```

only --

Réinitialise la pile de contexte à un élément, le dictionnaire FORTH

Non standard, car il n'y a pas de vocabulaire ONLY distinct

open-blocks addr len --

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

OPEN-FILE addr n opt -- n

Ouvre un fichier.

opt est une valeur parmi `R/O` ou `R/W` ou `W/O`.

```
s" myFile" r/o open-file
```

OR n1 n2 -- n3

Effectue un OU 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  -1    or  .  \ display 0
0  -1    or  .  \ display -1
-1   0   or  .  \ display -1
-1   -1  or  .  \ display -1
```

order --

Affiche l'ordre de recherche de vocabulaire.

```
windows
order   \ display: windows >> FORTH
```

over n1 n2 -- n1 n2 n1

Place une copie de n1 au sommet de la pile.

```
2 5 OVER  \ duplicate 2 on top of the stack
```

page --

Efface l'écran.

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.

pause --

Passe la main aux autres tâches.

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
```

prompt --

Affiche un texte de disponibilité de l'interpréteur. Affiche par défaut:

ok

r" comp: -- <string> | exec: addr len

Crée une chaîne temporaire terminée par "

R/O -- 0

Constante système. Empile 0.

R/W -- 2

Constante système. Empile 2.

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.

READ-FILE a n fh -- n ior

Lit les données d'un fichier. Le nombre de caractères réellement lus est renvoyé sous la forme u2, et ior est renvoyé 0 pour une lecture réussie.

recognizers --

Sélectionne le vocabulaire `recognizers`.

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
```

remember --

Sauvegarde un instantané dans le fichier par défaut.

Le mot **REMEMBER** vous permet de *geler* le code compilé. Si vous avez compilé une application, exécutez **REMEMBER**. Débranchez la carte ESP32. Rebranchez-là. Vous devriez retrouver votre application.

Utilisez **STARTUP:** pour définir le mot de votre application à exécuter au démarrage.

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.

reset --

Supprime le nom de fichier par défaut.

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.

revive --

Restaure le nom de fichier par défaut.

rm -- "path"

Efface le fichier indiqué.

rot n1 n2 n3 -- n2 n3 n1

Rotation des trois valeurs au sommet de la pile.

rp0 -- addr

pointe vers le bas de la pile de retour de Forth.

RSHIFT x1 u -- x2

Décalage vers la droite de u bits de la valeur x1.

```
64 2 rshift .  \ display 16
```

r| comp: -- <string> | exec: addr len

Crée une chaîne temporaire terminée par |

s" comp: -- <string> | exec: addr len

En interprétation, laisse sur la pile de données la chaine délimitée par "

En compilation, compile la chaine délimitée par "

Lors de l'exécution du mot compilé, restitue l'adresse et la longueur de la chaîne...

```
\ header for DUMP
: headDump
    s" --addr----- 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F"
  ;
headDump        \ push addr len on stack
headDump type   \ display: --addr----- 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

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.

save-buffers --

Sauvegarde tous les tampons.

SCR -- addr

Variable pointant sur le bloc en cours d'édition.

SDL2 --

Sélectionne le vocabulaire **SDL2**.

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.

La précision de calcul sur des nombres réels s'arrête à 6 décimales. Si vous demandez une précision supérieure à 6 sur les décimales des nombres réels, les valeurs affichées au-delà de 6 décimales seront fausses.

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

set-title a n --

Change le titre de la fenêtre de eForth Windows.

SF! r addr --

Stocke un réel préalablement déposé sur la pile des réels à l'adresse mémoire addr.

```
fvariable PRICE
3.25E PRICE SF!
```

sf, r --

Compile un nombre réel.

SF@ addr -- r

Récupère le nombre réel stocké à l'adresse addr, en général une variable définir par `fvariable`.

```
fvariable PRICE
35.25E PRICE SF!
PRICE SF@ F.    \ display: 35.250000
```

sfloat -- 4

Constante. Valeur 4.

sfloat+ addr -- addr+4

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

sfloats n -- n*4

Calcule l'espace nécessaire pour n réels.

SL@ addr -- n

Récupère une valeur 32 bits signée depuis l'adresse addr.

sp0 -- addr

pointe vers le bas de la pile de données de Forth.

SP@ -- addr

Dépose l'adresse du pointeur de pile sur la pile.

```
\ return number cells used on stack
: stackSize ( -- n )
    SP@ SP0 - CELL/
  ;
```

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

startup: -- <name>

Indique le mot qui doit s'exécuter au démarrage de ESP32forth après initialisation de l'environnement général.

state -- fl

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

-1 pour compilateur, 0 pour interpréteur

str n -- addr len

Transforme en chaîne alphanumérique toute valeur n, ce dans la base numérique courante.

```
352 str type    \ display: 352
```

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
```

streams --

Sélectionne le vocabulaire `streams`.

structures --

Sélectionne le vocabulary `structures`.

```
structures
\ Information about the version of SDL in use
struct SDL_version    ( -- 3 )              \ OK 2024-11-06
     i8 field ->version-major
     i8 field ->version-minor
     i8 field ->version-patch
```

SW@ addr -- n

Récupère une valeur 16 bits signée depuis l'adresse addr.

swap n1 n2 -- n2 n1

Echange les valeurs situées au sommet de la pile.

```
2 5 SWAP
.   \ display 2
.   \ display 5
```

task comp: xt dsz rsz -- <name> | exec: -- task

Créer une nouvelle tâche avec taille dsz pour la pile de données et rsz pour la pile de retour.

```
tasks
: hi   begin ." Time is: " ms-ticks . cr 1000 ms again ;
' hi 100 100 task my-counter
my-counter start-task hi
```

tasks --

Sélectionne le vocabulaire `tasks`.

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`.

throw n --

Génère une erreur si n pas égal à zéro.

Si les bits de n ne sont pas nuls, extraie l'exception en tête de la pile d'exceptions, ainsi que tout ce qui se trouve sur la pile de retour au-dessus de ce cadre. Ensuite, restaure la spécification de la source d'entrée utilisée avant le CATCH correspondant et ajuste les profondeurs de toutes les piles définies par cette norme afin qu'elles soient identiques aux profondeurs enregistrées dans le cadre d'exception (i est le même nombre que le i dans les arguments d'entrée au CATCH correspondant), place n au-dessus de la pile de données et transfère le contrôle à un point juste après le CATCH qui a poussé ce cadre d'exception.

```
: could-fail ( -- char )
   KEY DUP [CHAR] Q = IF 1 THROW THEN ;

: do-it ( a b -- c) 2DROP could-fail ;

: try-it ( --)
   1 2 ['] do-it CATCH IF
   ( x1 x2 ) 2DROP ." There was an exception" CR
   ELSE ." The character was " EMIT CR
   THEN
;

: retry-it ( -- )
   BEGIN 1 2 ['] do-it CATCH WHILE
   ( x1 x2) 2DROP ." Exception, keep trying" CR
   REPEAT ( char )
   ." The character was " EMIT CR
;
```

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*

```
0 value MAX_SCORE
120 to MAX_SCORE
```

touch -- "path"

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

type addr c --

Affiche la chaine de caractères sur c octets.

```
: hello ( --- addr c)
s" Hello world" ;
hello type              \ display: Hello world
hello drop 5 type       \ display: Hello
```

u. n --

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

```
1 U.      \ display 1
-1 U.     \ display 18446744073709551615
```

U/MOD u1 u2 -- rem quot

division int/int->int non signée.

UL@ addr -- un

Récupère une valeur non signée 32 bits.

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 son adresse non signée.

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

value comp: n -- <valname> | exec: -- n

Crée un mot de type *value*

valname empile la valeur.

Un mot défini par `value` est semblable à une constante, mais dont la valeur peut être modifiée.

```
12 value APPLES     \ Define APPLES with an initial value of 12
34 to APPLES        \ Change the value of APPLES. to is a parsing word
APPLES              \ puts 34 on the top of the stack
```

variable comp: -- <name> | exec: -- addr

Mot de création. Définit une variable simple précision.

```
variable speed
75 speed !   \ store 75 in speed
```

```
speed @ .    \ display 75
```

visual --

Sélectionne le vocabulaire `visual`.

vlist --

Affiche tous les mots d'un vocabulaire.

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

vocabulary comp: -- <name> | exec: --

Mot de définition d'un nouveau vocabulaire. En 83-STANDARD, les vocabulaires ne sont plus déclarés d'exécution immédiate.

```
\ create new vocabulary FPACK
VOCABULARY FPACK
```

W! n addr --

Stocke une valeur 16 bits à l'adresse addr.

W/O -- 1

Constante système. Empile 1.

while fl --

Marque la partie d'exécution conditionnelle d'une structure `begin..while..repeat`

```
\ logarithmus dualis of n1>0, rounded down to the next integer
: log2 ( +n1 -- n2 )
    2/ 0 begin
        over 0 >
    while
        1+ swap 2/ swap
    repeat
    nip
  ;
  7 log2 .     \ display 2
100 log2 .     \ display 6
```

windows --

sélectionne le vocabulaire `windows`.

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.

WRITE-FILE a n fh -- ior

Écrire un bloc de mémoire dans un fichier.

XOR n1 n2 -- n3

Effectue un OU eXclusif 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 -1 xor .      \ display 0
0 -1  xor .     \ display -1
-1  0 xor .     \ display -1
-1  0  xor .    \ display 0
```

z" comp: -- <string> | exec: -- addr

Compile une chaîne terminée par valeur 0 dans la définition.

ATTENTION: ces chaînes de caractères marquées par `z"` ne sont à exploiter que pour des fonctions spécifiques,.

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>

[char] comp: -- <spaces>name | exec: -- xchar

En compilation, enregistre le code ASCII du caractère indiqué après ce mot.

En exécution, le code xchar est déposé sur la pile de données.

```
: GC1 [CHAR] X     ;
: GC2 [CHAR] HELLO ;
GC1 \ empile 58
GC2 \ empile 48
```

[ELSE] --

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

[IF] fl --

Commence une séquence conditionnelle de type **[IF] ... [ELSE]** ou **[IF] ... [ELSE] ... [THEN]**.

Si l'indicateur est 'TRUE', ne fait rien (et exécute donc les mots suivants normalement). Si l'indicateur est 'FALSE', analyse et supprime les mots de la zone d'analyse, y compris les instances imbriquées de **[IF]**.. **[ELSE]**.. '[THEN]' et **[IF]**.. **[THEN]** jusqu'à l'équilibrage **[ELSE]** ou **[THEN]** a été analysé et supprimé.

```
DEFINED? L, invert [IF]
\ compile 32 bits value in dictionnary
: L,  ( u -- )
    dup c,
    8 rshift dup c,
    8 rshift dup c,
    8 rshift dup c,
    drop
  ;
[THEN]
```

[THEN] --

Termine une séquence conditionnelle de type **[IF] ... [ELSE]** or **[IF] ... [ELSE] ... [THEN]**.

```
DEFINED? mclr [IF]
: mclr  ( mask addr -- )
    dup >r c@ swap invert and r> c!
    ;
[THEN]
```

] --

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

{ -- <names..>

Marque le début de la définition de variables locales. Ces variables locales se comportent comme des pseudo-constantes.

Les variables locales sont une alternative intéressante à la manipulation des données de la pile. Elles rendent le code plus lisible.

```
: summ { n1 n2 }
    n1 n2 + . ;
3 5 summ    \ display 8
```

graphics

color -- n

Définit la couleur. Valeur par défaut: 0

```
\ Pen in red color:
$ff0000 to color    \ $rrggbb
```

CreatePen iStyle cWidth color -- hPen

Crée un stylo logique qui a le style, la largeur et la couleur spécifiés. Le stylet peut ensuite être sélectionné dans un contexte d'appareil et utilisé pour dessiner des lignes et des courbes.

Paramètres:

- **iStyle** style du tracé
- **cWidth** épaisseur du tracé
- **color** couleur du tracé

En retour, on récupère un handle nécessaire pour sélectionner le stylo.

```
0 value HPEN_RED
0 value HPEN_BLUE

: setPens  ( -- )
    PS_SOLID    3 $FF $00 $00 RGB CreatePen to HPEN_RED
    PS_DOT      1 $00 $00 $FF RGB CreatePen to HPEN_BLUE
  ;
```

event -- 0

Constante. Valeur par défaut 0

EXPOSED -- 2

Constante. Valeur 2

FINISHED -- 7

Constante. Valeur 7

g{ --

Préserve la transformation.

height -- 0

Value. Valeur par défaut 0

hwnd -- n

Un objet fenêtre est identifié par une valeur dénommée handle de fenêtre. Et le handle de fenêtre est de type HWND.

Le mot `CreateWindowExA` laisse une valeur qui est stockée dans `hwnd`.

IDLE -- 0

Constante. Valeur 0

key-count -- 256

Constante. Valeur 255

last-char -- 0

Constante. Valeur par défaut 0

last-key -- 0

Constante. Valeur par défaut 0

LEFT-BUTTON -- 255

Constante. Valeur 255

LineTo hdc x y -- fl

Dessine une ligne à partir de la position actuelle jusqu'au point spécifié, sans l'inclure.

Paramètres de `LineTo`:

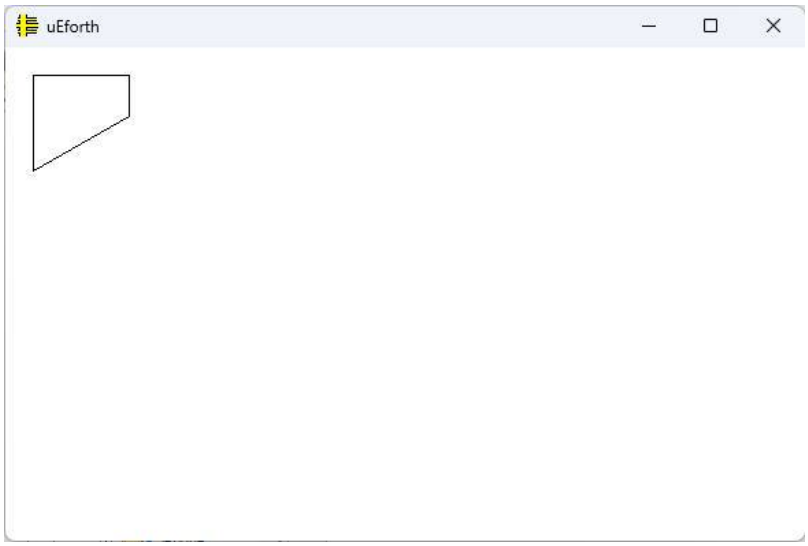
- **hdc** Handle vers un contexte de périphérique
- **x** Spécifie la coordonnée x, en unités logiques, de la nouvelle position
- **y** Spécifie la coordonnée y, en unités logiques, de la nouvelle position

Définition:

```
\ LineTo draws a line from the current position to,
\ but not including, the specified point.
z" LineTo"      3 Gdi32 LineTo ( hdc x y -- fl )

: draw
    hdc 20 20 NULL moveToEx drop
    hdc 90 20 LineTo drop
    hdc 90 50 LineTo drop
    hdc 20 90 LineTo drop
    hdc 20 20 LineTo drop
```

```
;
```



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

moveTo x y --

Déplace de point graphique vers la position x y depuis la position courante.

Code source:

```
create LPPOINT
    POINT allot

: moveTo ( x y -- )
    hdc -rot LPPOINT Gdi.MoveToEx gdiError
  ;
```

MoveToEx hdc x y LPPOINT -- fl

Met à jour la position actuelle au point spécifié et retourne éventuellement la position précédente.

Paramètres de `MoveToEx`:

- **hdc** Handle vers un contexte de périphérique
- **x** Spécifie la coordonnée x, en unités logiques, de la nouvelle position
- **y** Spécifie la coordonnée y, en unités logiques, de la nouvelle position
- **LPPOINT** Pointeur vers une structure `POINT` qui reçoit la position actuelle précédente. Si ce paramètre est un pointeur `NULL` , la position précédente n'est pas retournée

Définition:

```
\ MoveToEx updates the current position
z" MoveToEx"    4 Gdi32 MoveToEx ( hdc x y LPPOINT -- fl )
```

pixel w h -- a

xxx

PRESSED -- 4

Constante. Valeur 4

PS_DOT -- 2

Constante. Valeur 2.

Le stylet est pointillé. Ce style est valide uniquement lorsque la largeur du stylet est égale ou inférieure à une unité d'appareil.

```
0 value HPEN_BLUE
PS_DOT 1 $00 $00 $FF RGB CreatePen to HPEN_BLUE
```

PS_SOLID -- 0

Constante. Valeur 0.

Le stylet est plein.

```
0 constant PS_SOLID
0 value HPEN_RED
PS_SOLID 1 $FF $00 $00 RGB CreatePen to HPEN_RED
```

RELEASED -- 5

Constante. Valeur 5

RESIZED -- 1

Constante. Valeur 1

RIGHT-BUTTON -- 253

Constante. Valeur 253

screen>g x y -- x' y'

Transforme l'écran en fenêtre d'affichage.

SetTextColor hdc color -- fl

Définit la couleur du texte pour le contexte de périphérique spécifié sur la couleur spécifiée.

Définition:

```
\ Set text color
z" SetTextColor"        2 Gdi32 SetTextColor
```

TYPED -- 6

Constante. Valeur 6

vertical-flip --

Utilise la fenêtre d'affichage de style mathématique.

width -- 0

Value. Valeur par défaut 0

window x y --

Ouvre une nouvelle fenêtre de dimension x y en pixels.

```
graphics
600 400 window
```

}g --

Restaure la transformation.

streams

>stream addr len stream --

Stocke une chaîne de caractères dans un flux.

```
streams
1000 stream myStream
s" this is " myStream >stream
s" a test."  myStream >stream
\ now, myStream content is: "this is a test."
```

ch>stream c stream --

ajoute un caractère c au flux.

```
streams
1000 stream myStream
s" this is" myStream >stream
$0d myStream ch>stream
$0a myStream ch>stream
s" a test" myStream >stream

myStream dup
    0 swap >offset
    swap cell + @
    type
\ display:
\ this is
\ a test.
```

empty? -- fl

Empile -1 si le flux est vide, sinon empile 0.

full? -- fl

Empile -1 si le flux est plein, sinon empile 0.

stream comp: n -- <name> | exec: -- addr

Crée un espace mémoire de n caractères.

```
200 stream input-stream
```

stream# sz -- n

Utilisé par `full?` et `empty?`.

stream>ch addr -- c

Récupère un caractère dans le flux.

structures

field comp: n -- <:name>

Mot de définition d'un nouveau champ dans une 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 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 -- 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**

ptr -- 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`

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.

windows

->biBitCount addr -- addr'

Accesseurs 16 bits pour **BITMAPINFOHEADER**.

->biClrImportant addr -- addr'

Accesseurs 32 bits pour **BITMAPINFOHEADER**.

->biClrUsed addr -- addr'

Accesseurs 32 bits pour **BITMAPINFOHEADER**.

->biCompression addr -- addr'

Accesseurs 32 bits pour **BITMAPINFOHEADER**.

->biHeight addr -- addr'

Accesseurs 32bits pour **BITMAPINFOHEADER**.

->biPlanes addr -- addr'

Accesseurs 16 bits pour **BITMAPINFOHEADER**.

->biSize addr -- addr'

Accesseurs 16 bits pour **BITMAPINFOHEADER**.

->biSizeImage addr -- addr'

Accesseurs 32 bits pour **BITMAPINFOHEADER**.

->biWidth addr -- addr'

Accesseurs 32bits pour **BITMAPINFOHEADER**.

->biXPelsPerMeter addr -- addr'

Accesseurs 32 bits pour **BITMAPINFOHEADER**.

->bmiColors addr -- addr'

Accesseurs pour **BITMAPINFO**. Taille est **RGBQUAD**.

->bmiHeader addr -- addr'

Accesseurs pour **BITMAPINFO**. Taille est **BITMAPINFOHEADER**.

->bottom addr -- addr'

Accesseur pour la structure RECT.

->left addr -- addr'

Accesseur pour la structure RECT.

->rgbBlue addr -- addr'

Accesseurs 8 bits pour **RGBQUAD**.

->rgbGreen addr -- addr'

Accesseurs 8 bits pour **RGBQUAD**.

->rgbRed addr -- addr'

Accesseurs 8 bits pour **RGBQUAD**.

->rgbReserved addr -- addr'

Accesseurs 8 bits pour **RGBQUAD**.

->right addr -- addr'

Accesseur pour la structure RECT.

->top addr -- addr'

Accesseur pour la structure RECT.

->x addr -- addr'

Accesseur pour la structure POINT.

->y addr -- addr'

Accesseur pour la structure POINT.

>biYPelsPerMeter addr -- addr'

Accesseurs 32 bits pour **BITMAPINFOHEADER**.

ANSI_FIXED_FONT -- n

Constante, valeur: $8000000b

ANSI_VAR_FONT -- n

Constante, valeur: $8000000c

BeginPaint hWnd lpPaint -- lpPaint

Prépare la fenêtre spécifiée pour la peinture et remplit une structure `PAINTSTRUCT` avec des informations sur la peinture.

BITMAPINFO -- n

Structure BITMAPINFO.

Liste des accesseurs:

- **->bmiHeader**
- **->bmiColors**

BITMAPINFOHEADER -- n

Liste des accesseurs:

- **->biSize** 16 bits
- **->biWidth** 32 bits
- **->biHeight** 32 bits
- **->biPlanes** 16 bits
- **->biBitCount** 16 bits
- **->biCompression** 32 bits
- **->biSizeImage** 32 bits
- **->biXPelsPerMeter** 32 bits
- **->biYPelsPerMeter** 32 bits
- **->biClrUsed** 32 bits
- **->biClrImportant** 32 bits

BI_RGB -- n

Constante, valeur: 0

BLACK_BRUSH -- n

Constante, valeur: $80000004

Pinceau noir. A utiliser avec `GetStockObject`

BLACK_PEN -- n

Constante, valeur: $80000007

BM_CLICK -- 245

Constante. valeur 245

Utilisé par `WM_>name`

BM_GETCHECK -- 240

Constante. valeur 240

Utilisé par `WM_>name`

Obtient l'état de vérification d'un bouton radio ou d'une case à cocher.

BM_GETIMAGE -- 246

Constante. valeur 246

Utilisé par `WM_>name`

BM_GETSTATE -- 242

Constante. valeur 242

Utilisé par `WM_>name`

Récupère l'état d'un bouton ou d'une case à cocher.

BM_SETCHECK -- 241

Constante. valeur 241

Utilisé par `WM_>name`

Définit l'état de vérification d'un bouton radio ou d'une case à cocher.

BM_SETDONTCLICK -- 248

Constante. valeur 248

Utilisé par `WM_>name`

BM_SETIMAGE -- 247

Constante. valeur 247

Utilisé par `WM_>name`

BM_SETSTYLE -- 244

Constante. valeur 244

Utilisé par `WM_>name`

calls -- addr

Marque un tableau contenant les code exécutables de `call0` à `call15`

CB_ADDSTRING -- 323

Constante. valeur 323

Utilisé par `WM_>name`

Ajoute une chaîne à la zone de liste d'une zone de liste déroulante. Si la combo n'a pas le style CBS_SORT, la chaîne est ajoutée à la fin de la liste. Sinon, la chaîne est insérée dans la liste et la liste est triée.

CB_FINDSTRING -- 332

Constante. valeur 332

Utilisé par `WM_>name`

Recherche dans la zone de liste d'une zone de liste déroulante un élément commençant par les caractères d'une chaîne spécifiée.

CB_FINDSTRINGEXACT -- 344

Constante. valeur 344

Utilisé par `WM_>name`

CB_GETCOMBOBOXINFO -- 356

Constante. valeur 356

Utilisé par `WM_>name`

CB_GETCOUNT -- 326

Constante. valeur 326

Utilisé par `WM_>name`

Obtient le nombre d'éléments dans la zone de liste d'une zone de liste déroulante.

CB_GETCURSEL -- 327

Constante. valeur 327

Utilisé par `WM_>name`

Une application envoie un message CB_GETCURSEL pour récupérer l'index de l'élément actuellement sélectionné, le cas échéant, dans la zone de liste d'une zone de liste déroulante.

CB_GETDROPPEDCONTROLRECT -- 338

Constante. valeur 338

Utilisé par `WM_>name`

CB_GETDROPPEDSTATE -- 343

Constante. valeur 343

Utilisé par `WM_>name`

CB_GETDROPPEDWIDTH -- 351

Constante. valeur 351

Utilisé par `WM_>name`

CB_GETEDITSEL -- 320

Constante. valeur 320

Utilisé par `WM_>name`

CB_GETEXTENDEDUI -- 342

Constante. valeur 342

Utilisé par `WM_>name`

CB_GETHORIZONTALEXTENT -- 349

Constante. valeur 349

Utilisé par `WM_>name`

CB_GETITEMDATA -- 336

Constante. valeur 336

Utilisé par `WM_>name`

CB_GETITEMHEIGHT -- 340

Constante. valeur 340

Utilisé par `WM_>name`

CB_GETLBTEXT -- 328

Constante. valeur 328

Utilisé par `WM_>name`

CB_GETLBTEXTLEN -- 329

Constante. valeur 329

Utilisé par `WM_>name`

CB_GETLOCALE -- 346

Constante. valeur 346

Utilisé par `WM_>name`

CB_GETTOPINDEX -- 347

Constante. valeur 347

Utilisé par `WM_>name`

CB_INITSTORAGE -- 353

Constante. valeur 353

Utilisé par `WM_>name`

CB_INSERTSTRING -- 330

Constante. valeur 330

Utilisé par `WM_>name`

CB_LIMITTEXT -- 321

Constante. valeur 321

Utilisé par `WM_>name`

CB_MSGMAX -- 357

Constante. valeur 357

Utilisé par `WM_>name`

CB_MULTIPLEADDSTRING -- 355

Constante. valeur 355

Utilisé par `WM_>name`

CB_RESETCONTENT -- 331

Constante. valeur 331

Utilisé par `WM_>name`

CB_SELECTSTRING -- 333

Constante. valeur 333

Utilisé par `WM_>name`

CB_SETCURSEL -- 334

Constante. valeur 334

Utilisé par `WM_>name`

CB_SETDROPPEDWIDTH -- 352

Constante. valeur 352

Utilisé par `WM_>name`

CB_SETEDITSEL -- 322

Constante. valeur 322

Utilisé par `WM_>name`

CB_SETEXTENDEDUI -- 341

Constante. valeur 341

Utilisé par `WM_>name`

CB_SETHORIZONTALEXTENT -- 350

Constante. valeur 350

Utilisé par `WM_>name`

CB_SETITEMDATA -- 337

Constante. valeur 337

Utilisé par `WM_>name`

CB_SETITEMHEIGHT -- 339

Constante. valeur 339

Utilisé par `WM_>name`

CB_SETLOCALE -- 345

Constante. valeur 345

Utilisé par `WM_>name`

CB_SETTOPINDEX -- 348

Constante. valeur 348

Utilisé par `WM_>name`

CB_SHOWDROPDOWN -- 335

Constante. valeur 335

Utilisé par `WM_>name`

COLOR_WINDOW -- 5

Constante. valeur 5.

CommandLineToArgvW lpCmdLine *pNumArgs -- LPWSTR

Analyse une chaîne de ligne de commande Unicode et retourne un tableau de pointeurs vers les arguments de ligne de commande, ainsi qu'un nombre de ces arguments, d'une manière similaire aux valeurs argv et argc d'exécution C standard.

console-started -- 0

Valeur initialisée à zéro.

Utilisée par `init-console`

CreateSolidBrush param -- null|brush

Crée un pinceau logique qui a la couleur unie spécifiée.

```
255 192 0 RGB CreateSolidBrush constant orange
0 255 0   RGB CreateSolidBrush constant green
```

CreateWindowExA 12params -- 0|HWND

Permet d'effectuer la création d'une sous-fenêtre ou une fenêtre surgissante (PopUp).

Paramètres:

- **dwExStyle** permet d'indiquer le style de fenêtre étendue lors de sa création.
- **lpClassName** permet d'indiquer la chaîne de caractères ou le nom de l'atome de classe crée par un appel précédent à la fonction RegisterClassA ou RegisterClassExA.
- **lpWindowName** permet d'indiquer le nom de la fenêtre. Si le style de fenêtre à spécifier une barre de titre, le titre de la fenêtre pointer par le paramètre lpWindowName est affiché dans la barre de titre.
- **dwStyle** permet d'indiquer le style de la fenêtre à créer.

- **x** permet d'indiquer la position horizontal initiale de la fenêtre.
- **y** permet d'indiquer la position verticale initiale de la fenêtre.
- **nWidth** permet d'indiquer la largeur, dans l'unité de périphérique, de la fenêtre.
- **nHeight** permet d'indiquer la hauteur, dans l'unité de périphérique, de la fenêtre.
- **hWndParent** permet d'indiquer l'identificateur de gestionnaire Handle de la fenêtre parente ou du propriétaire de la fenêtre de la fenêtre à créer. Ce paramètre est optionnel dans le cas de fenêtre surgissante.
- **hMenu** permet d'indiquer l'identificateur du gestionnaire vers un menu ou spécifie la fenêtre enfant, indépendamment du style de fenêtre.
- **hInstance** permet d'indiquer l'identificateur du gestionnaire Handle de l'instance de module associé avec la fenêtre.
- **lpParam** permet d'indiquer un pointeur vers une valeur a passer vers la fenêtre par la structure CREATESTRUCT (membre de lpCreateParams) pointé par le paramètre lParam du message WM_CREATE.

Si la fonction réussit, la valeur de retour est un handle pour la nouvelle fenêtre.

Si la fonction échoue, la valeur de retour est NULL. Pour obtenir des informations détaillées sur l'erreur, appelez GetLastError.

DC_BRUSH -- n

Constante, valeur: $80000012

Pinceau de couleur unie. La couleur par défaut est le blanc.

DC_PEN -- n

Constante, valeur: $80000013

DefaultInstance -- $400000

Constante, valeur $400000.

DEFAULT_GUI_FONT -- n

Constante, valeur: $80000011

DEFAULT_PALETTE -- n

Constante, valeur: $8000000f

DEVICE_DEFAULT_PALETTE -- n

Constante, valeur: $8000000e

DIB_RGB_COLORS -- 0

Constante. valeur 0.

DISABLE_NEWLINE_AUTO_RETURN -- n

Constante. Valeur $0008

DKGRAY_BRUSH -- n

Constante, valeur: $80000003

Pinceau gris foncé.

dll comp: zStr -- <:name>

Crée un ticket d'accès à une librairie Windows.

```
z" Kernel32.dll" dll Kernel32
```

EM_CHARFROMPOS -- 215

Constante. valeur 215

Utilisé par `WM_>name`

EM_EMPTYUNDOBUFFER -- 205

Constante. valeur 205

Utilisé par `WM_>name`

EM_FMTLINES -- 200

Constante. valeur 200

Utilisé par `WM_>name`

EM_GETFIRSTVISIBLELINE -- 206

Constante. valeur 206

Utilisé par `WM_>name`

EM_GETIMESTATUS -- 217

Constante. valeur 217

Utilisé par `WM_>name`

EM_GETLIMITTEXT -- 213

Constante. valeur 213

Utilisé par `WM_>name`

EM_GETMARGINS -- 212

Constante. valeur 212

Utilisé par `WM_>name`

EM_GETPASSWORDCHAR -- 210

Constante. valeur 210

Utilisé par `WM_>name`

EM_GETWORDBREAKPROC -- 209

Constante. valeur 209

Utilisé par `WM_>name`

EM_LINEFROMCHAR -- 201

Constante. valeur 201

Utilisé par `WM_>name`

EM_POSFROMCHAR -- 214

Constante. valeur 214

Utilisé par `WM_>name`

EM_SETIMESTATUS -- 216

Constante. valeur 216

Utilisé par `WM_>name`

EM_SETMARGINS -- 211

Constante. valeur 211

Utilisé par `WM_>name`

EM_SETPASSWORDCHAR -- 204

Constante. valeur 204

Utilisé par `WM_>name`

EM_SETREADONLY -- 207

Constante. valeur 207

Utilisé par `WM_>name`

EM_SETTABSTOPS -- 203

Constante. valeur 203

Utilisé par `WM_>name`

EM_SETWORDBREAK -- 202

Constante. valeur 202

Utilisé par `WM_>name`

EM_SETWORDBREAKPROC -- 209

Constante. valeur 209

Utilisé par `WM_>name`

EM_UNDO -- 199

Constante. valeur 199

Utilisé par `WM_>name`

ENABLE_INSERT_MODE -- n

Constante, valeur: $0020

ENABLE_PROCESSED_INPUT -- n

Constante, valeur: $0001

ExitProcess uExitCode --

Code de sortie pour le processus et tous les threads.

FillRect hDC *lprc hbr -- fl

Remplit un rectangle à l'aide du pinceau spécifié. Inclut les bordures gauche et supérieure, mais exclut les bordures droite et inférieure du rectangle.

Paramètres:

- **hDC** Handle pour le contexte de l'appareil.
- **lprc** Pointeur vers une structure RECT qui contient les coordonnées logiques du rectangle à remplir.
- **hbr** Poignée du pinceau utilisée pour remplir le rectangle.

gdi32 zstr n --

Mot défini par `dll`.

`Gdi32` permet de créer les mots liés à la librairie **Gdi32.dll**.

```
z" DeleteObject" 1 Gdi32 DeleteObject
```

GetCommandLineW -- str

Récupère la chaîne de ligne de commande pour le processus actuel.

GetDC hWnd -- hdc

Récupère un handle dans un contexte d'appareil (DC) pour la zone cliente d'une fenêtre spécifiée ou pour l'ensemble de l'écran. Vous pouvez utiliser le handle retourné dans les fonctions GDI suivantes pour dessiner dans le contrôleur de domaine.

GetLastError -- err

Récupère la valeur du dernier code d'erreur du thread appelant. Le dernier code d'erreur est conservé pour chaque thread. Plusieurs threads n'écrasent pas le dernier code d'erreur de l'autre.

GetMessageA lpMsg hWnd wMsgFilterMin wMsgFilterMax -- fl

Récupère un message à partir de la file d'attente de messages du thread appelant.

Paramètres:

- **lpMsg** Pointeur vers une structure MSG qui reçoit les informations de message de la file d'attente de messages du thread
- **hWnd** Handle de la fenêtre dont les messages doivent être récupérés. La fenêtre doit appartenir au thread actif.
- **wMsgFilterMin** Valeur entière de la valeur de message la plus basse à récupérer.
- **wMsgFilterMax** Valeur entière de la valeur de message la plus élevée à récupérer.<:li>

GetModuleHandleA lpModuleName -- HMODULE

Récupère un handle de module pour le module spécifié. Le module doit avoir été chargé par le processus appelant.

GetProcessHeap -- handle

Récupère un handle vers le tas par défaut du processus appelant. Ce handle peut ensuite être utilisé dans les appels ultérieurs aux fonctions de tas.

GetRect LPRECT -- left top right bottom

Obtenir les coordonnées du rectangle spécifié.

Définition:

```
: GetRect  ( LPRECT -- left top right bottom )
    >r
    r@ ->left   SL@
    r@ ->top    SL@
    r@ ->right  SL@
    r@ ->bottom SL@
    r> drop
  ;
```

GetStockObject i --

Récupère une poignée dans l'un des stylets, pinceaux, polices ou palettes de stock.

Paramètres

- **i** Type d'objet stock

Ce paramètre peut prendre les valeurs suivantes:

- **BLACK_BRUSH** Pinceau noir.
- **DKGRAY_BRUSH** Pinceau gris foncé.
- **DC_BRUSH** Pinceau de couleur unie. La couleur par défaut est le blanc.
- **GRAY_BRUSH** Pinceau gris.
- **HOLLOW_BRUSH** Pinceau creux (équivalent à NULL_BRUSH).
- **LTGRAY_BRUSH** Pinceau gris clair.
- **NULL_BRUSH** Pinceau Null (équivalent à HOLLOW_BRUSH).
- **WHITE_BRUSH** Pinceau blanc.
- **BLACK_PEN** Stylet noir.
- **DC_PEN** Couleur de stylet unie. La couleur par défaut est le noir.
- **NULL_PEN** Stylet Null. Le stylet null ne dessine rien.
- **WHITE_PEN** Stylet blanc.
- **ANSI_FIXED_FONT** Police système windows à pas fixe (monospace).
- **ANSI_VAR_FONT** Police système à pas variable (espace proportionnel) Windows.
- **DEVICE_DEFAULT_FONT** Police dépendante de l'appareil.

- **DEFAULT_GUI_FONT** Police par défaut pour les objets d'interface utilisateur tels que les menus et les boîtes de dialogue. Il n'est pas recommandé d'utiliser DEFAULT_GUI_FONT ou SYSTEM_FONT pour obtenir la police utilisée par les boîtes de dialogue et les fenêtres. La police par défaut est Tahoma.
- **OEM_FIXED_FONT** Police à pas fixe (monospace) dépendante du fabricant d'équipement d'origine (OEM).
- **SYSTEM_FONT** Police système. Par défaut, le système utilise la police système pour dessiner des menus, des contrôles de boîte de dialogue et du texte. Il n'est pas recommandé d'utiliser DEFAULT_GUI_FONT ou SYSTEM_FONT pour obtenir la police utilisée par les boîtes de dialogue et les fenêtres. La police système par défaut est Tahoma.
- **SYSTEM_FIXED_FONT** Police système à pas fixe (monospace). Cet objet stock est fourni uniquement à des fins de compatibilité avec les versions 16 bits de Windows antérieures à 3.0.
- **DEFAULT_PALETTE** Palette par défaut. Cette palette se compose des couleurs statiques de la palette système.

GetTickCount -- ms

Récupère le nombre de millisecondes écoulées depuis le démarrage du système, jusqu'à 49,7 jours.

IDI_MAIN_ICON -- 1001

Constante, valeur 1001.

init-console --

Initialise la console Windows.

Kernel32 --

Mot défini par **dll**.

Permet ensuite d'accéder aux fonctions de **Kernel32.dll**

LoadLibraryA dllname-z -- module

La richesse des DLL Windows et des fonctionnalités du système est accessible via l'interface de chargement dynamique.

Un handle vers une bibliothèque est obtenu avec **LOADLIBRARYA**, puis les symboles individuels sont accessibles avec **GETPROCADDRESS**

LTGRAY_BRUSH -- $80000001

Pinceau gris.

MALLOC_CAP_32BIT -- 2

Constante. Valeur 2

MALLOC_CAP_8BIT -- 4

Constante. Valeur 4

MALLOC_CAP_DMA -- 8

Constante. Valeur 8

MALLOC_CAP_EXEC -- 1

Constante. Valeur 1

MB_ABORTRETRYIGNORE -- 2

Constant. Value 2.

MB_CANCELTRYCONTINUE -- 6

Constant. Value 6.

MB_OK -- 0

Constant. Value 0. Utilisé par `MessageBoxA`.

La boîte de message contient un bouton d'envoi: OK. Il s'agit de la valeur par défaut.

MB_OKCANCEL -- 1

Constant. Value 1.

MB_RETRYCANCEL -- 5

Constant. Value 5.

MB_YESNO -- 4

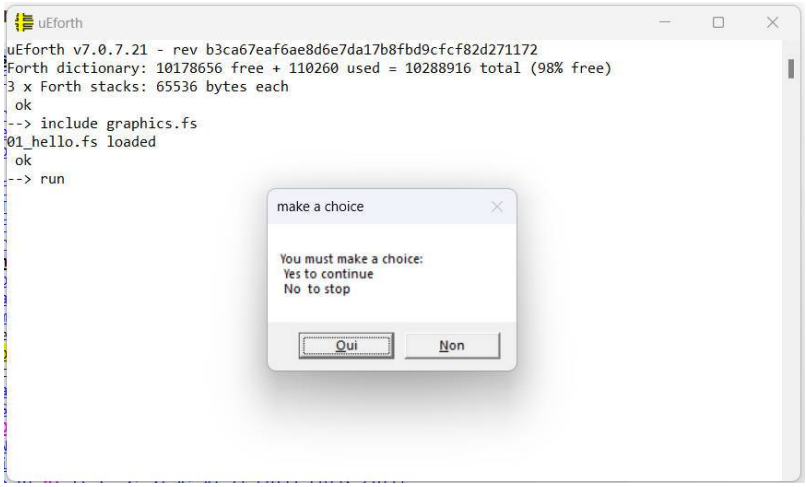
Constant. Value 4. Utilisé par `MessageBoxA`.

La boîte de message contient deux boutons pousseurs : Oui et Non.

```
z" Will you continue?" constant lpText
z" make a choice"      constant lpCaption

: MSGbox  ( -- )
    NULL lpText lpCaption MB_YESNO MessageBoxA
    ?dup if
```

```
        cr ." You have pressed: "
        case
            6 of ." Yes"      endof
            7 of ." No"       endof
        endcase
    then
;
```



MB_YESNOCANCEL -- 3

Constant. Value 3.

MessageBoxA hWnd lpText lbCaption uType -- 0|val

Affiche une boîte de dialogue modale qui contient une icône système, un ensemble de boutons et un bref message spécifique à l'application, tel que des informations de status ou d'erreur. La boîte de message retourne une valeur entière qui indique le bouton sur lequel l'utilisateur a cliqué.

Si la fonction échoue, la valeur de retour est égale à zéro.

`MessageBoxA` prend des chaînes de caractères ANSI (American Standard Code for Information Interchange). L'ANSI est un encodage de caractères à un octet, ce qui signifie qu'il peut représenter un nombre limité de caractères, principalement les caractères de l'alphabet latin. Il est donc moins adapté pour gérer des textes dans d'autres langues, notamment celles utilisant des caractères accentués ou des alphabets non latins.

NULL -- 0

Équivalent à 0.

NULL_BRUSH -- n

Constante. Valeur $80000005

PAINTSTRUCT -- n

Structure.

Liste des accesseurs:

- **->hdc** 64 bits
- **->fErase** 32 bits
- **->rcPaint** taille **RECT**
- **->fRestore** 32 bits
- **->fIncUpdate** 32 bits
- **->rgbReserved** 32 octets

POINT -- n

Structure POINT.

Liste des accesseurs:

- **->x** 32 bits
- **->y** 32 bits

RECT -- n

Structure.

Liste des accesseurs:

- **->left**
- **->top**
- **->right**
- **->bottom**

RGB r g b -- n

Assemble trois couleurs **r g b**, valeurs 8 bits en une seule couleur.

```
255 192 0 RGB CreateSolidBrush constant orange
```

RGBQUAD -- n

structure RGBQUAD

Liste des accesseurs:

- **->rgbBlue** 8 bits

- **->rgbGreen** 8 bits
- **->rgbRed** 8 bits
- **->rgbReserved** 8 bits

SBM_ENABLE_ARROWS -- 228

Constante. valeur 228

Utilisé par `WM_>name`

SBM_GETPOS -- 225

Constante. valeur 225

Utilisé par `WM_>name`

SBM_GETRANGE -- 227

Constante. valeur 227

Utilisé par `WM_>name`

SBM_GETSCROLLBARINFO -- 235

Constante. valeur 235

Utilisé par `WM_>name`

SBM_GETSCROLLINFO -- 234

Constante. valeur 234

Utilisé par `WM_>name`

SBM_SETPOS -- 224

Constante. valeur 224

Utilisé par `WM_>name`

SBM_SETRANGE -- 226

Constante. valeur 226

Utilisé par `WM_>name`

SBM_SETRANGEREDRAW -- 230

Constante. valeur 230

Utilisé par `WM_>name`

SBM_SETSCROLLINFO -- 233

Constante. valeur 233

Utilisé par `WM_>name`

SetForegroundWindow hWnd -- fl

Place le thread qui a créé la fenêtre spécifiée au premier plan et active la fenêtre.

SetRect LPRECT xLeft yTop xRight yBottom -- fl

Définit les coordonnées du rectangle spécifié. Cela revient à affecter les arguments gauche, supérieur, droit et inférieur aux membres appropriés de la structure RECT

Définition:

```
\ sets the coordinates of the specified rectangle
z" SetRect"     5 User32 SetRect

\ Example:
create zone  RECT allot
zone 10 10 80 50 SetRect
```

SetupCtrlBreakHandler --

Utilisation interne à Windows.

Shell32 zstr n --

Mot défini par dll

Permet ensuite d'accéder aux fonctions de Shell32.dll

```
z" CommandLineToArgvW" 2 Shell32 CommandLineToArgvW
```

ShowWindow hWnd nCmdShow -- fl

Définit l'état d'affichage de la fenêtre spécifiée.

Paramètres:

- **hWnd** Handle de la fenêtre.
- **nCmdShow** Contrôle la façon dont la fenêtre doit être affichée. Ce paramètre est ignoré la première fois qu'une application appelle `ShowWindow`, si le programme qui a lancé l'application fournit une structure STARTUPINFO . Sinon, la première fois que `ShowWindow` est appelé, la valeur doit être la valeur obtenue par la fonction `WinMain` dans son paramètre nCmdShow

Sleep ms --

Suspend l'exécution du thread actif jusqu'à ce que l'intervalle de délai d'attente s'écoule.

```
: ms ( n -- )
    Sleep ;
```

SRCCOPY -- $00cc0020

Constante. valeur $00cc0020.

stdin -- 0

Valeur initialisée à zéro.

Utilisée par `init-console`

stdout -- 0

Valeur initialisée à zéro.

Utilisée par `init-console`

User32 zstr n --

Création de mots en lien avec la librairie User32.dll.

```
z" MessageBoxA" 4 User32 MessageBoxA
```

WaitForSingleObject hHandle Ms --

Attend que l'objet spécifié soit dans l'état signalé ou que l'intervalle de délai d'attente s'écoule.

Si la fonction réussit, la valeur de retour indique l'événement qui a provoqué le retour de la fonction. Il peut s'agir de l'une des valeurs suivantes.

wargc -- addr

Mémorise l'action de `GetCommandLineW`

wargv -- addr

Mémorise l'action de `CommandLineToArgvW`

WHITE_BRUSH -- $80000000

Pinceau blanc.

win-type addr len --

Affiche une chaine vers une console windows

WINDCLASSA -- n

Structure.

Liste des accesseurs de cette structure:

- **->style** 16 bits
- **->lpfnWndProc** pointeur
- **->cbClsExtra** 32 bits
- **->cbWndExtra** 32 bits
- **->hInstance** pointeur
- **->hIcon** pointeur
- **->hCursor** pointeur
- **->hbrBackground** pointeur
- **->lpszMenuName** pointeur
- **->lpszClassName** pointeur

WindowProcShim --

Utilisation interne à Windows.

windows-builtins -- n

Point d'entrée du vocabulaire `windows`

WM_>name msg -- a n

Extrait l'adresse en longueur de l'en-tête correspondant au message windows compris entre `WM_PENWINLAST` et `WM_NULL`

WM_ACTIVATE -- 6

Constante. valeur 6

Utilisé par `WM_>name`

WM_AFXFIRST -- 864

Constant. valeur 864.

Utilisée par `WM_>name`

WM_AFXLAST -- 896

Constant. valeur 895.

Utilisée par `WM_>name`

WM_APPCOMMAND -- 793

Constant. valeur 793.

Utilisée par `WM_>name`

WM_CHANGECBCHAIN -- 781

Constante. valeur 781

WM_CHAR -- 258

empile 258.

Utilisée par `WM_>name`

WM_CLEAR -- 771

Constante. valeur 771

Utilisé par `WM_>name`

WM_COPY -- 769

Constante. valeur 769

Utilisé par `WM_>name`

WM_CREATE -- 1

empile 1.

WM_CUT -- 768

Constante. valeur 768

Utilisé par `WM_>name`

WM_DEADCHAR -- 259

empile 259.

WM_DESTROY -- 2

Constante. valeur 2

Utilisé par `WM_>name`

WM_DESTROYCLIPBOARD -- 775

Constante. valeur 775

Utilisé par `WM_>name`

WM_DRAWCLIPBOARD -- 776

Constante. valeur 776

Utilisé par `WM_>name`

WM_ENABLE -- 10

Constante. valeur 10

Utilisé par `WM_>name`

WM_ENTERIDLE -- 289

Constante. valeur 289

Utilisé par `WM_>name`

WM_GETTEXT -- 13

Constante. valeur 13

Utilisé par `WM_>name`

WM_GLOBALRCCHANGE -- 899

Constant. valeur 899.

Utilisée par `WM_>name`

WM_HANDHELDFIRST -- 856

Constant. valeur 856.

Utilisée par `WM_>name`

WM_HANDHELDLAST -- 863

Constant. valeur 863.

Utilisée par `WM_>name`

WM_HEDITCTL -- 901

Constante. Valeur 901

Utilisée par `WM_>name`

WM_HOOKRCRESULT -- 898

Constant. valeur 898.

Utilisée par `WM_>name`

WM_HOTKEY -- 786

Constante. valeur 786

WM_HSCROLL -- 276

Constante. valeur 276

Utilisé par `WM_>name`

WM_HSCROLLCLIPBOARD -- 782

Constante. valeur 782

WM_IMEKEYDOWN -- 656

Constante. valeur 656

Utilisé par `WM_>name`

WM_IMEKEYUP -- 657

Constante. valeur 657

Utilisé par `WM_>name`

WM_IME_CHAR -- 646

Constante. valeur 646

Utilisé par `WM_>name`

WM_IME_COMPOSITIONFULL -- 644

Constante. valeur 644

Utilisé par `WM_>name`

WM_IME_CONTROL -- 643

Constante. valeur 643

Utilisé par `WM_>name`

WM_IME_KEYDOWN -- 656

Constante. valeur 656

Utilisé par `WM_>name`

WM_IME_KEYUP -- 657

Constante. valeur 657

Utilisé par `WM_>name`

WM_IME_NOTIFY -- 642

Constante. valeur 642

Utilisé par `WM_>name`

WM_IME_REPORT -- 640

Constante. valeur 640

Utilisé par `WM_>name`

WM_IME_REQUEST -- 648

Constante. valeur 648

Utilisé par `WM_>name`

WM_IME_SELECT -- 645

Constante. valeur 645

Utilisé par `WM_>name`

WM_IME_SETCONTEXT -- 641

Constante. valeur 641

Utilisé par `WM_>name`

WM_INITMENU -- 278

Constante. valeur 278

Utilisé par `WM_>name`

WM_INITMENUPOPUP -- 279

Constante. valeur 279

Utilisé par `WM_>name`

WM_INPUT -- 255

Constante. valeur 255

Utilisé par `WM_>name`

WM_KEYDOWN -- 256

Constante. Valeur 256

Utilisée par `WM_>name`

WM_KEYUP -- 257

Constante. Valeur 257

Utilisée par `WM_>name`

WM_KILLFOCUS -- 0

Constante. Valeur 0

WM_LBUTTONDBLCLK -- 515

Constante. valeur 515

Utilisé par `WM_>name`

WM_LBUTTONDOWN -- 513

Constante. valeur 513

Utilisé par `WM_>name`

WM_LBUTTONUP -- 514

Constante. valeur 514

Utilisé par `WM_>name`

WM_MBUTTONDBLCLK -- 521

Constante. valeur 521

Utilisé par `WM_>name`

WM_MBUTTONDOWN -- 519

Constante. valeur 519

Utilisé par `WM_>name`

WM_MENUCHAR -- 288

Constante. valeur 288

Utilisé par `WM_>name`

WM_MENUSELECT -- 287

Constante. valeur 287

Utilisé par `WM_>name`

WM_MOUSEFIRST -- 512

Constante. valeur 512

Utilisé par `WM_>name`

WM_MOUSEHOVER -- 673

Constante. valeur 673

Utilisé par `WM_>name`

WM_MOUSELAST -- 521

Constante. valeur 521

Utilisé par `WM_>name`

WM_MOUSELEAVE -- 675

Constante. valeur 675

Utilisé par `WM_>name`

WM_MOUSEMOVE -- 512

Constante. valeur 512

Utilisé par `WM_>name`

WM_MOVE -- 3

Constante. Valeur 3

Utilisée par `WM_>name`

WM_NCMOUSEHOVER -- 672

Constante. valeur 672

Utilisé par `WM_>name`

WM_NCMOUSELEAVE -- 674

Constante. valeur 674

Utilisé par `WM_>name`

WM_NULL -- 0

Constante. Valeur 0

WM_PAINTCLIPBOARD -- 777

Constante. valeur 777

Utilisé par `WM_>name`

WM_PALETTECHANGED -- 785

Constante. valeur 785

WM_PALETTEISCHANGING -- 784

Constante. valeur 784

WM_PASTE -- 770

Constante. valeur 770

Utilisé par `WM_>name`

WM_PENCTL -- 901

Constante. Valeur 901

Utilisée par `WM_>name`

WM_PENEVENT -- 904

Constante. Valeur 904

Utilisée par `WM_>name`

WM_PENMISC -- 902

Constante. Valeur 902

Utilisée par `WM_>name`

WM_PENMISCINFO -- 899

Constant. valeur 899.

Utilisée par `WM_>name`

WM_PENWINFIRST -- 896

Constant. valeur 896.

Utilisée par `WM_>name`

WM_PENWINLAST -- 911

Constante. Valeur 911

Utilisée par `WM_>name`

Valeur utilisée pour définir une limite supérieure pour les messages de style Pen Windows (PenWin).

WM_PRINTCLIENT -- 792

Constante. valeur 792

WM_QUERYNEWPALETTE -- 783

Constante. valeur 783

WM_RBUTTONDBLCLK -- 518

Constante. valeur 518

Utilisé par `WM_>name`

WM_RBUTTONDOWN -- 516

Constante. valeur 516

Utilisé par `WM_>name`

WM_RBUTTONUP -- 517

Constante. valeur 517

Utilisé par `WM_>name`

WM_RCRESULT -- 898

Constant. valeur 897.

Utilisée par `WM_>name`

WM_RENDERALLFORMATS -- 774

Constante. valeur 774

Utilisé par `WM_>name`

WM_RENDERFORMAT -- 774

Constante. valeur 773

Utilisé par `WM_>name`

WM_SETFOCUS -- 7

Constante. Valeur 7

WM_SETREDRAW -- 11

Constante. Valeur 11

WM_SETTEXT -- 12

Constante. valeur 12

Utilisé par `WM_>name`

WM_SIZE -- 5

Constante. valeur 5

Utilisé par `WM_>name`

WM_SKB -- 900

Constante. Valeur 900

Utilisée par `WM_>name`

WM_SYSDEADCHAR -- 258

empile 263.

WM_SYSTIMER -- 280

Constante. valeur 280

Utilisé par `WM_>name`

WM_UNDO -- 772

Constante. valeur 772

Utilisé par `WM_>name`

WM_VSCROLL -- 277

Constante. valeur 277

Utilisé par `WM_>name`

WM_VSCROLLCLIPBOARD -- 778

Constante. valeur 778

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)