# MANUEL DE REFERENCE
# TOPOCAD

# LE LANGAGE TED

# Syntaxe du langage TED:

(Le langage TED est un langage dérivé du Franglais trés connu en France ;) )

## Présentation:

Un fichier TED (d'extension *".ted"*) est constitué de fonctions ted (au moins une) séparées par la virgule et constitue un programme TED que l'on peut lancer par la commande *Script|Execute un script TED.*

Le point virgule en tête de ligne signifie une ligne commentaire. La ligne commentaire peut être insérée partout dans le programme TED après une commande suivie d'une virgule ou en début de fichier, mais ne peut terminer un fichier.

Les chaines de caractères utilisées par l'application sont entourées ou non de deux guillemets.
Dans le premier cas les espaces sont acceptés ainsi que tout caractère (sauf nul) codé sous la forme "\xNN" , NN étant la valeur hexadécimale du caractère.
Dans le second cas, les espaces avant ou après la chaine sont supprimés et seuls sont acceptés certains caractères.

exemple:
*@setlvar(myvar, coucou me voilou ,STR) enregistrera dans la variable "myvar" la chaine "coucou me voilou"*
ou
*@setlvar(myvar," ceci est \x31 exemple",STR) enregistrera dans la variable "myvar" la chaine " ceci est 1 exemple"*

La casse du nom des fonctions n'a pas d'importance dans un programme TED : vous pouvez écrire *setlvar(myvar,texte,STR)* ou *Setlvar(myvar,texte,STR)* ou *sEtlVaR(myvar,texte,STR)* indifféremment. En règle générale d'ailleurs les comparaisons ne sont pas sensibles à la casse.

Une fonction TED commence toujours par le caractère '@' suivi du nom de la fonction et d'éventuels paramètres entre parenthèses si la fonction attend des paramètres.

exemple:
*@setlvar(variable,0,NUM)*
ou
*@commune*

Chaque fonction TED renvoie une valeur, le programme TED (fichier) renvoie la dernière valeur de la dernière fonction exécutée. Une fonction TED comportant au moins un paramètre doit terminer le fichier (en général le fichier se termine par une fonction @return). Les valeurs renvoyées ont un type (cf plus loin)

Les fonctions peuvent être imbriquées.

exemple:
*@while(@curt,@list(@addvar(stot,@surffc(curt)),@NextWithMask(curt,mask))*

La liste Ted symbolisée par *@list(...)* est une fonction TED composée de plusieurs fonctions TED. Elle doit être composée d'au moins une fonction.

Les fonctions TED peuvent également être portées sur plusieurs lignes pour plus de clarté.

exemple:
```
@while(@curt,
        @list(
            @addvar(stot,@surffc(curt)),
            @NextWithMask(curt,mask)
             )
        ),
```

TED offre la possibilité de stocker des variables par les commandes @local (variable locale créée n'existant pas déjà), @setlvar (variable locale existant ou n'existant pas, qui sera alors créée), @setvar (variable quelconque existant déjà ou n'existant pas), l'appel postérieur à ces variables se fait en nommant ces variables. Les variables ne peuvent donc porter le nom d'une fonction.

exemple:
```
@setlvar(curt,0,FC),                    ;création d'une variable de type FACE nommée CURT
@FirstWithMask(curt,mask),              ;initialisation de la variable
@while(@curt,                           ;appel de la variable
        @list(
            @addvar(stot,@surffc(curt)),
            @NextWithMask(curt,mask)
             )
        ),
```

Les paramètres et variables introduits dans une fonction TED doivent avoir un certain Type.

# Types de Variables:

Il existe 2 principaux types manipulés par TED

- Le type Standard :

  Tout élément de ce type peut être comparé avec un élément du même type, quel que soit le sous−type, en fait ces variables sont stockées ***sous forme de chaines de caractères*** et sont interprétées suivant le sous−type auquel elles appartiennent.
  Il est ainsi possible d'écrire : @setvar(mavar,"coucou",NUM)
  Il sera stocké alors dans "mavar" la valeur "coucou" qui sera interprétée suivant le cas comme une variable numérique (0) ou pas.
  Par exemple à la suite, la commande @equ(@mavar,"coucou") donnera 1 (vrai) car TopoCad verra que l'on veut comparer une variable numérique avec une variable chaine de caractère et fera alors une comparaison des deux chaines.
  Les fonctions TED attendant une variable numérique entière acceptent des variables numériques sous forme d'entier hexa, octal ou décimal : Fournir alors la valeur 010 n'est pas la même valeur que 10 (valeur octale de 8 dans le premier cas)
  De même, le type numérique englobant les flottants et les entiers l'exemple suivant fonctionne

  ```
  @setlvar(couleur1,0x0000FF,NUM),  ;couleur rouge
  @setlvar(couleur2,0x00FF00,NUM),  ;couleur vert
  @setcolor(vpt,@couleur1)
  ```

  car @setcolor s'attend à avoir un entier mais

  ```
  @equ(@couleur1,@couleur2)
  ```

  renvoie 1 (vrai) car @equ s'attend à avoir une valeur numérique quelconque (flottantes décimale) et la conversion de ces deux valeurs donne 0.
  On remedie à cette situation en convertissant explicitement la valeur hexa ou octale en décimale par @numi
  Cette manière de traiter les données facilite le traitement pour le programmeur TED qui n'a pas à se soucier des conversions mais peut présenter des pièges : En effet, examinons le code suivant tout a fait légal :

  ```
  @setlvar(vcou,2e+01,NUM),
  @setcouchetravail(@vcou)
  @addvar(vcou,1),
  @setcouchetravail(@vcou)
  ```

  le code enregistre la valeur 20 (2e+01) dans *vcou* puis indique que la couche de travail doit être la couche 20.
  Cependant, le résultat est tout autre : la fonction @setcouchetravail s'attend à recevoir un entier comme paramètre : elle convertit donc 2e+01 en 2 et le résultat est que la couche 2 devient la couche de travail.
  le second @setcouchetravail, quant à lui fonctionne et positionne la couche de travail sur la couche 21.
  De même si on enregistre dans *vcou* la valeur 1.99, c'est la couche 1 qui devient la couche de travail et non la couche 2.
  Les valeurs flottantes sont enregistrées avec le maximum de décimales (21) puis si la valeur ne peut être stockée sous forme décimale normale (xxx.xxxxxx) compte tenu de la précision requise, elle est stockée sous notation scientifique (x.xxxxxEyy).

  Les sous type sont:

  - ERR: Type Erreur (un texte  indiquant la cause de l'erreur y est associé)
  - STR: Type chaine de caractères
  - NUM: Type numérique (entier ou flottant)
  - ANG: Type Angle
  - BOOL: Type Booléen

- Le type Pointeur

  Les éléments de ce type ne peuvent être comparés que entre éléments de même sous type (sinon la comparaison renverra forcément FALSE). Ils correspondent à des index sur des objets de l'application.

  Les sous types sont:

  - PT : type élément point (visibilité interne au document)
  - LI :type élément liaison (visibilité interne au document)
  - FC: type élément face (visibilité interne au document)
  - EC: type élément écriture (visibilité interne au document)
  - OBJ: type objet (visibilité interne au document)
  - OBS: type observation (visibilité interne au document)
  - MASKCRPT: type masque de création de point
  - MASKCRLI: type masque de création de liaison
  - MASKCRFC: type masque de création de face
  - MASKCREC: type masque de création d'écriture
  - MASKMDPT: type masque de modification de point

- MASKMDLI: type masque de modification de liaison
- MASKMDFC: type masque de modification de face
- MASKMDEC: type masque de modification d'écriture
- MASKCHPT: type masque de recherche de point
- MASKCHLI: type masque de recherche de liaison
- MASKCHFC: type masque de recherche de face
- MASKCHEC: type masque de recherche d'écriture
- RELSEM : type relation sémantique (visibilité interne au document)
- MAPDOC : type document MAP
- PLANVIEW : type vue fenêtre plan

Ces variables pointeurs peuvent être invalidées (NULL), car certaines fonctions peuvent détruire les objets auxquels elles se réfèrent auquel cas elles auront pour valeur NULL et un appel ultérieur à ces variables échouera donc. Les fonctions TED susceptibles d'invalider des variables sont marquées dans le tableaux des fonctions par une double astérisque.

D'autre part certaines variables pointeurs (PT, LI, FC, EC, OBJ, OBS, RELSEM) ne sont utilisables que pour le document auquel elle font référence. Toute tentative d'utilisation de ces variables dans un autre document conduira à un échec.

exemple:

```
; initialise une variable point du document courant
@setlvar(monpt,@getpoint(12654),PT),
; change de document courant
@setdocvue(@autredoc,0),
@out(@monpt),
==> ERREUR, commande ou nom de variable inconnu! la variable monpt n'existe pas dans le document courant.
```

Il existe un opérateur booléen implicite pour chacun de ces types de variables et résultats renvoyés:
Pour les types pointeurs, un pointeur valide donnera TRUE et non valide FALSE. Lors du parcours de pointeurs, la fin de liste donne un pointeur nul et donc FALSE.
Pour les types standard, le sous type ERR donne FALSE, STR,NUM,ANG,BOOL donne FALSE si la conversion numérique de la variable donne 0.

exemple:

```
@setlvar(test1,"1texte",STR),
@setlvar(test2,"0texte",STR),
@setlvar(test3,"vrai",STR),
@setlvar(test4,VRAI,NUM),
@out(@if(@test1,VRAI,FAUX)),    ; écrit VRAI
@out(@if(@test2,VRAI,FAUX)),    ; écrit FAUX
@out(@if(@test3,VRAI,FAUX)),    ; écrit FAUX
@out(@if(@test4,VRAI,FAUX)),    ; écrit FAUX car VRAI est une chaine de caractère (même sans guillemets) convertie en numérique et sa conversion donne 0
```

## Portée des variables:

L'ensemble des variables est stockée dans une pile globale. Il existe trois types de portées pour les variables :

1) Les variables globales utilisateurs : Elles existent tout le long de l'existence de l'instance de TopoCad.exe. Les valeurs de ces dernières sont stockées extérieurement au déroulement d'un script TED et donc peuvent être réutilisées lors d'un autre script. Lorsqu'on lance un script TED, les valeurs d'initialisation de ces variables prennent place dans une variable générale TED placée sur la pile des variables qui sera donc valide durant tout le déroulement de celui−ci. Cette variable TED peut alors être modifiée par *@setvar* mais pas sa valeur d'initialisation, ou bien on peut modifier la variable ainsi que sa valeur d'initialisation par la commande *@setgvar*. Ces variables globales ne peuvent être que de type Standard (leur initialisation est de type STR).

2) Les variables locales : Ces variables se manipulent de la même manière mais ont une durée de vie interne au script (ou fichier) qui les utilise. Elles se créent (une fois) par @local et se modifient par @setvar.... ou se créent ou se modifient par @setlvar.

3) Les variables générales (globales pour une execution TED par un script) : Ces variables une fois créées peuvent être accédées de tout point du script (ou sous−script) lancé par l'utilisateur : Elles sont en fait les variables locales du script lancé au niveau le plus élevé. On les appelle "générales" car il est possible d'y accéder et surtout d'en créer à partir de scripts représentant des sous programmes, bien que ce ne soit pas conseillé pour des raisons de facilité de maintenance : le script de plus haut niveau ne sait pas alors quelles sont les variables à sa portée créées par des scripts de plus bas niveau à moins de consulter tous les scripts de plus bas niveau. La création de telles variables à partir de sous programme n'est pas possible si l'option correspondante de @setdebug est validée

Les variables TED sont manipulées par les fonctions *@setvar*, *@addvar*...etc. et ne peuvent comporter dans leurs noms que des caractères alphanumériques et le symbole "_" (underscore).

Certaines fonctions ont besoin que des variables de noms prédéterminés existent (depuis la version 6) afin de ne pas échouer comme *@getextents* : elles recherchent si une variable de nom désiré existe : si ne la trouve pas renvoie une erreur.

Les noms des variables locales "cachent les mêmes noms des variables à des niveaux plus élevé (et globales)

exemple:

| | |
|---|---|
| programme1.ted: | Etat de la pile des variables en fin d'instruction: |
| *@setlvar(var1,"variable globale1",STR),* | var1 |
| *@local(var2,"variable locale au programme1",STR),* | var1 − var2 |
| *@exec(programme2.ted)* | var1 − var2 'var1(du sous prog) n'existe plus au retour' |
| *@out(@var1),* | var1 − var2 'var1 locale à programme2 a disparue' |
| *@out(@var2)* | |
| programme2.ted: | |
| *@local(var1,"variable locale au programme 2",STR),* | var1 − var2 − var1 |
| *@setvar(var2,"variable globale2",STR),* | var1 − var2 − var1 'var2 a été modifié' |
| *@out(@var1),* | La recherche d'une variable se fait à partir de la fin de la pile jusqu'au début de pile ou jusqu'au point de la pile accepté par la portée lorsque'elle est explicitement définie comme dans @setlvar ou @local qui recherche uniquement dans la portée locale |
| *@out(@var2)* | au retour de programme 2, var1 (du sommet de la pile) est nettoyé |

l'execution de programme1 crée deux variables var1 et var2 puis appelle programme2 qui crée à son tour deux variables de même nom : var1 est une nouvelle variable qui cache la valeur "variable globale1", var2 est recherchée et est trouvée ("variable locale au programme 1" est donc modifiée en "variable globale2" . Il y a donc une pile de 3 variables dont la première est innaccessible à ce point ("variable globale1","variable globale2" , "variable locale au programme 2"). Ces deux dernières sont affichées. Au sortir de programme2, "variable locale au programme 2" disparait car locale (au programme2). Il ne reste que "variable globale1" qui est globale pour programme1 et "variable globale2" qui est locale pour programme1 et qui disparaitrait si programme1 avait été lancée par un autre programme et venait à se terminer.

On voit donc que le fait de déclarer une variable comme locale crée obligatoirement une nouvelle variable si elle n'a pas été créée locale dans le même programme et qu'elle cache toute variable précédente de même nom

Autrement dit, on utilise les variables locales quand on ne veut pas qu'elles interfèrent avec les programmes appelants (donc avec d'autres fichiers), et les variables générales (globales au script) quand on veut transmettre une valeur à l'appelant (bien qu'il soit préférable d'utiliser les paramètres).

## Sous−programmes:

La fonction *@exec(...)* permet d'appeler un autre fichier TED à partir d'un fichier TED. On peut considérer alors ce fichier comme un sous programme. Ce sous programme peut avoir accès à des variables déclarées dans le programme principal en utilisant leur nom, auquel cas ce sous programme ne pourra pas être lancé de manière isolé puisqu'il fera appel à une variable qu'il ne connait pas.

exemple:

*; programme test1.ted*
*@setlvar(var1,"var1 est connue",STR),*
*@exec(test2.ted)*
*; programme test2.ted*
*@setlvar(var2,"var 2 est connue",STR),*
*@out(@var1),*
*@out(@var2)*
*; le lancement du programme test1.ted provoque la sortie suivante :*
*var1 est connue*
*var2 est connue*
*; le lancement du programme test2.ted provoque l'erreur suivante :*
*==> ERREUR=ligne 2 : commande ou nom de variable inconnue*

Il est également possible d'utiliser @exec en fournissant des paramètres :

exemple1: transmission par valeur

```
; programme test1.ted
@setlvar(var1,"valeur de var1",STR),
@exec(test2.ted,@var1)
; programme test2.ted
@param(param1),
@out(@param1),
; le lancement du programme test1.ted provoque la sortie suivante :
valeur de var1
; le lancement du programme test2.ted provoque l'erreur suivante :
==> ERREUR=ligne 1 : nombre de paramètres incorrect
```

exemple2: transmission par référence

```
; programme test1.ted
@setlvar(var1,"valeur de var1",STR),
@exec(test2.ted,@refvar(var1))
@out(@var1)
; programme test2.ted
@param(param1),
@setlvar(param1,"nouvelle valeur",STR),
; le lancement du programme test1.ted provoque la sortie suivante :
nouvelle valeur
; le lancement du programme test2.ted provoque l'erreur suivante :
==> ERREUR=ligne 1 : nombre de paramètres incorrect
```

La fonction @param permet de déclarer les paramètres attendus dans une sous procedure TED. Elle doit être la première instruction d'une sous procédure pour être utilisée. Elle substitue les noms param1, param2.... aux noms des paramètres transmis.

La fonction @refvar fournit dans une procedure @exec une référence au lieu d'une valeur, et n'est utilisable qu'au sein de celle ci.

## Les boucles:

La fonction *@while* effectue la boucle de base et est généralement suivie d'une liste

*exemple:*

```
@while(@curt,                          ;appel de la variable
    @list(
        @addvar(stot,@surffc(curt)),
        @NextWithMask(curt,mask)
        )
    ),
```

La fonction *@for* effectue une boucle incrémentale

*exemple:*

```
@for(@setlvar(@vcou,0,NUM),        ; initialisation
    @inf(@vcou,@nbcouches),    ; condition d'éxécution de la liste de commande
    @list(                         ; effectue une série de commandes pour chaque couche
        @setvar(namevar,@basename,STR),
        @concatvar(namevar,@vcou,0),
        @setlvar(@namevar,@layername(@vcou,""),STR)
        ),
    @next(vcou,-1)                 ; couche suivante
    ),
```

La fonction *@break* provoque une rupture de liste (sortie de la boucle while dans le cas ci dessus, du programme ou sous−programme TED si cette instruction fait partie des instructions composant le fichier TED).

La fonction *@return* provoque une rupture dans l'éxécution d'un fichier TED (sous programme ou programme TED).

La fonction *@exit* provoque une sortie de TED.

La fonction *@error* provoque une sortie de TED avec erreur.

## Contexte d'execution:

Un fichier TED s'execute de manière ***interprétée*** en lançant la commande Script|Execute un script TED.... Il agit à partir du document et de la fenêtre à partir duquel il est exécuté. L'***analyse syntaxique*** du fichier est faite ***avant toute execution de fonction*** du fichier.

Un fichier TED (ou une commande TED) peut cependant être exécuté sans contexte document/vue présent lorsque il est lancé comme initialisation de l'application (par" *topocad −cmd=@exec(c:\topocad\init.ted)*" ou comme Pré−Exécution au chargement d'un document : En effet si l'option de de Pré ou Post Execution est validée par l'application (cf variable de configuration DocAutoExec), deux actions peuvent être entreprises:

- Avant chaque chargement d'un document, un programme TED de pré−execution peut être lancé (par le document) servant éventuellement à initialiser l'application pour le type de document présent : le paramètre PreExec de la section DATA du document représente une ligne de commande TED à pré−executer.
- Après chaque chargement d'une vue plan, un programme TED de post−exécution peut être lancé (par le document) servant à effectuer par exemple des opérations particulières de visualisation pour ce type de document.

Si n'importe quelle fonction peut être lancée en post−éxécution, le modèle document/vue étant présent, cela n'est pas le cas en cas de pré−exécution ou d'éxécution à l'initialisation de l'application (AutoExec) : Seules donc les fonctions de configuration ne nécessitant pas de document/vue seront valides. Dans le cas contraire un message d'erreur s'affiche et le programme TED est arrété (sans arréter cependant le déroulement du chargement du document par exemple).

Apperçu fonctionnel de TED

# Apperçu fonctionnel de TED:

## Fonctions de calcul arithmétiques, logiques et autre:

### fonctions de comparaisons

| | |
|---|---|
| @sup | indique si une valeur est supérieure à une autre |
| @supe | indique si une valeur est supérieure ou égale à une autre |
| @inf | indique si une valeur est inférieure à une autre |
| @infe | indique si une valeur est inférieure ou égale à une autre |
| @equ | indique si une valeur est égale à une autre |

toute valeur de type "standard" comparée à une valeur de type "standard" et sous type "str" (chaine de caractère) est convertie en chaine de caractère et ce sont les chaines de caractères qui sont comparées. Ainsi @sup(2,14) et @sup(2,@num(14)) donne TRUE mais @sup(@num(2),@num(14)) donne FALSE.

### opérateurs logiques

| | |
|---|---|
| @not | fournit la négation d'une valeur booléenne |
| @or | ou logique |
| @and | et logique |

### opérateurs binaires

| | |
|---|---|
| @binnot | fournit le complément à 2 |
| @binor | ou binaire |
| @binand | et binaire |

### calculs arithmétiques

| | |
|---|---|
| @add | addition |
| @addvar | |
| @sub | soustraction |
| @subvar | |
| @mul | multiplication |
| @mulvar | |
| @div | division |
| @divvar | |
| @neg | fournit la négation d'une valeur arithmétique |
| @negvar | |
| @abs | valeur absolue |
| @absvar | |
| @sin | sinus |
| @cos | cosinus |
| @inv | inverse |
| @invvar | |
| @roundval | arrondi |
| @sqrt | racine carrée |
| @sqrtvar | |
| @modulo | fonction modulo : reste de la division entière de 2 nombres entiers |

### calculs géodésiques

| | |
|---|---|
| @projfwd | coordonnées en projection à partir de longitude, latitude |
| @projinv | l'inverse |
| @projtransform | changement de système de coordonnées |
| @projdatumtransform | changement de système géodésique à partir de longitude latitude |
| @projgeodetic2geocentric | transforme des coordonnées géodésiques en géocentriques |
| @projgeocentric2geodetic | transforme des coordonnées géocentriques en géodésiques |
| @projgeocentric2wgs | transforme des coordonnées géocentriques d'un système vers le système pivot WGS84 |
| @projwgs2geocentric | transforme des coordonnées du système pivot WGS84 vers un système de coordonnées géocentriques |
| @chgtsystemept | changement de système de coordonnées |
| @chgtsysteme | |

| | |
|---|---|
| @projapplygridshift | application d'une grille de changement de système de coordonnées |
| @projgetgridshift | recherche les valeurs (interpollées) d'une grille |
| @projgeod | détermination de l'azimuth inverse et des long/lat d'un point à partir d'un point initial connu en long/lat, d'un azimuth et d'une distance orthodromique. |
| @projinvgeod | détermination des azimuths aller et retour et de la distance orthodromique à partir de deux points connus en long/lat. |

Les latitudes et longitudes étant traitées en tant que valeurs exprimées en radian, les fonctions suivantes permettent de convertir ces valeurs dans les deux sens en format Degré, Miniutes, Secondes intelligible.

| | |
|---|---|
| @dms2r | traduit une chaine DMS en valeur numérique radian |
| @r2dms | inversement |
| @setr2dms | fixe le format de sortie des chaines DMS |
| @getproj | fournit la définition d'un système de coordonnées à partir du code Edigeo du système |

### calculs relatifs aux graphes

| | |
|---|---|
| @deleltdata | supprime une donnée d'un élément |
| @seteltdata | fixe une donnée d'un élément |
| @geteltdata | recherche et fournit la donnée d'un élément |
| @dijkstra | calcul du plus court chemin par algorithme de dijkstra |
| @bellmann | algorithme de bellmann |
| @graphshortestpath | calcul d'un chemin par algorithme de dijkstra |
| @graphnearest | recherche le plus proche point dans un graphe |
| @graphinsert | insertion d'un point extérieur au graphe dans un graphe |
| @graphselect | sélectionne un graphe à partir d'une liaison |

### calculs divers

| | |
|---|---|
| @surf | surface |
| @surffc | |
| @surfobj | |
| @dist | distance, longueur, périmètre |
| @distentreobj | |
| @distentreelt | |
| @disteltobj | |
| @distobj | |
| @transfpt | transformations sur les coordonnées |
| @calculetransf | calcule une transformation d'après les correspondances existantes |
| @calculeinterp | calcule une interpolation d'après les correspondances existantes |
| @angle | calcul d'angle |
| @bissectrice | calcul d'angle bissectrice entre deux angles |

## Fonctions de traitement des chaines de caractères et conversions:

| | |
|---|---|
| @concat | concaténation de chaines |
| @concatvar | |
| @casestr | casse d'une chaine de caractère |
| @len | longueur d'une chaine de carcatère |
| @contains | indique si une chaine est contenue dans une autre |
| @replacestr | remplacement de chaine dans une chaine |
| @format | création d'une chaine de caractère suivant un certain format |
| @extract | extraction d'une partie de chaine à partir d'une chaine |
| @extractword | |
| @substr | |
| @rightstr | |
| @leftstr | |
| @c_str | conversion d'une donnée quelconque en chaine de caractère |
| @num | conversion d'une chaine en valeur numérique |
| @numi | conversion d'une chaine hexa ou octal en valeur numérique |
| @compidu | compose une chaine de caractère d'identifiant de base de donnée |
| @decompidu | décompose un identifiant de la base de donnée en différentes chaines de caractères |
| @regexmatch | teste d'une expresssion régulière |

## Manipulation de la base de donnée:

la base de donnée de TopoCad est un ensemble de tables DBF qui peuvent être organisées entre elles. Pour l'application, celles ci renferment les propriétés des objets. Il est également possible de faire appel à des bases externes DBF.

| | |
|---|---|
| @dbopen | ouverture d'une table |
| @dbclose | fermeture d'une table |
| @str2base | conversion code d'une base de donnée en indice de cette base |
| @adddatabase | ajout d'une table de donnée |
| @deldatabase | suppression d'une table de donnée |
| @dbsettobegin | se positionne en début de table |
| @dbsettoend | se positionne en fin de table |
| @dbnext | enregistrement suivant |
| @dbprevious | enregistrement précédent |
| @dbseek | recherche d'un enregistrement |
| @getdbprop | acquisition d'une propriété parmi les bases de l'application |
| @getdbpropwithid | |
| @setdbprop | écriture d'une propriété |
| @setdbpropwithid | |
| @adddbprop | |
| @adddbpropwithid | |
| @dbbrowse | ouverture d'une boite de dialogue de navigation dans la table |
| @propdlg | boite de dialogue d'edition des propriétés d'un objet identifiable |
| @dbsetfield | écriture d'un champ |
| @dbgetfield | acquisition d'un champ |
| @dbclass2dbstand | conversion d'une base de classe en base standard |
| @dbstand2dbclass | conversion d'une base standard en base de classe |
| @dbsetdefault | initialisation des champs d'un enregistrement avec les valeurs par défaut |
| @adddbrule | ajout d'une règle de saisie de la base de donnée |
| @dbcreateindex | création ou régénération d'un index sur une base de données |
| @dbpack | compactage de la base de données |

## Presse papiers:

| | |
|---|---|
| @paste | coller |
| @copy | copier |

## Impression:

l'impression se fait avec le format de page courant.

| | |
|---|---|
| @print | impression d'un élément ou objet ou de la fenetre courante |
| @printat | impression centrée sur un point (coordonnées) |
| @multiprintbegin | debut de séquence d'impression multiple |
| @multiprintend | fin de séquence d'impression multiple |
| @multiprintimage | séquence d'impression d'image dans une impression multiple |
| @multiprintmap | séquence d'impression de plan dans une impression multiple |
| @multiprintmapat | séquence d'impression de plan centré sur des coordonnées dans une impression multiple |

## Entrées−Sorties sur fichier:

| | |
|---|---|
| @export | export suivant différents formats |

| | |
|---|---|
| @import | import suivant différent formats |
| @importcarnet | import de carnets électroniques suivant différent formats |
| @out | sortie dans le fichiers des erreurs ERRORxxx.LOG |
| @getline | extrait une ligne d'un fichier |
| @popline | extrait la ligne suivante d' un fichier |
| @pushline | ajoutte une ligne à un fichier |

## Manipulation des documents et vues de l'application:

| | |
|---|---|
| @loaddoc | charge un document MAP qui devient document courant |
| @loadtxt | charge un document TXT qui devient document courant |
| @savedoc | sauvegarde le document courant |
| @getvue | cherche une vue d'un document |
| @getdoc | cherche un document |
| @getdocname | cherche le nom du fichier associé à un document |
| @newdoc | nouveau document |
| @closeview | ferme une vue |
| @setdocvue | établit la vue/le document courant |

## Manipulation de la configuration:

### Formats de page:

| | |
|---|---|
| @setformatpage | fixe le format de page courant |
| @cadreformatpage2bk | passe un cadre du format de page à l'arrière plan |
| @nbcadreformatpage | fournit le nombre de cadres du format de page |
| @setcadreformatpage | modifie un cadre du format de page |
| @delcadreformatpage | supprime un cadre du format de page |
| @addcadreformatpage | ajoutte un cadre du format de page |
| @loadformatpage | charge un format de page |

### Formes des points:

| | |
|---|---|
| @nbformespoint | les formes des points |
| @addformepoint | |
| @delformepoint | |
| @setformepoint | |

### Formes des liaisons:

| | |
|---|---|
| @nbformesliaison | les formes des liaisons |
| @addformeliaison | |
| @delformeliaison | |
| @setformeliaison | |

### Formes des faces:

| | |
|---|---|
| @nbformesface | les formes de faces |
| @addformeface | |
| @delformeface | |
| @setformeface | |

### Motifs:

| | |
|---|---|
| @nbpatterns | les motifs |
| @addpattern | |
| @delpattern | |
| @setpattern | |

### Les classes ou type d'objet:

| | |
|---|---|
| @nbclasses | les classes ou types d'objets |
| @gettobj | |
| @settobj | |
| @getprid | |
| @setprid | |
| @getinit | |
| @setinit | |
| @getidalgo | |
| @setidalgo | |
| @getidparm | |
| @setidparm | |

### Les types de relation sémantique:

| | |
|---|---|
| @nbtrel | nombre de types de relations sémantiques |
| @gettrel | recherche ou modifie une donnée d'un type de relation sémantique |
| @settrel | |
| @currenttrelsem | type de relation sémantique courante de la fenètre |
| @setcurrenttrelsem | |

### Les masques:

| | |
|---|---|
| @getmask | les masques du modèle |
| @setmask | |
| @getmaskno | |
| @setmaskno | |
| @getmask2str | |
| @getmaskno2str | |
| @loadmasques | |

### Les options de configuration:

| | |
|---|---|
| @loadmodele | le modèle complet |
| @tableload | les tables de conversions |
| @getdata | recherche les différentes valeurs de configuration |
| @setdata | modifie une valeur de configuration |
| @getapicce | lecture d'un code état Apic de la configuration Topocad |

## Manipulation du document:

### Les données intrinsèques du document:

| | |
|---|---|
| @orior | orientation d'origine |
| @setorior | |
| @echorigin | échelle d'origine |
| @setechorigin | |
| @echelle | échelle d'édition |
| @setechelle | |
| @codeinsee | code INSEE |
| @setcodeinsee | |
| @commune | nom commune |
| @setcommune | |
| @section | section |
| @setsection | |
| @feuille | feuille ou subdivision de section |
| @setfeuille | |
| @zonegeo | système de coordonnées |
| @setzonegeo | |
| @copl | mode de confection du plan |
| @setcopl | |
| @qupl | qualité du plan |
| @setqupl | |

| | |
|---|---|
| @modeincorp | mode d'incorporation au SIG |
| @setmodeincorp | |
| @dateincorp | date d'incorporation au SIG 'interruption par abandon ou variable non appropriée ou variable inconnue |
| @setdateincorp | |
| @dateedi | date d'édition |
| @setdateedi | |
| @dateredi | date de réédition |
| @setdateredi | |
| @rem | remarque du document |
| @setrem | |

## Parcours des données graphiques du document éléments, objets, relations, observations:

| | |
|---|---|
| @first | pour parcourir les éléments du document |
| @next | |
| @nbeltforelt | |
| @firstwithobj | pour parcourir les éléments d'un objet |
| @nextwithobj | |
| @nbeltforobj | |
| @firstwithelt | pour parcourir les objets d'un élément |
| @nextwithelt | |
| @nbobjforelt | |
| @firstwithmask | pour parcourir les éléments répondant à un masque de recherche |
| @nextwithmask | |
| @firstobjet | pour parcourir les objets du document |
| @nextobjet | |
| @getobjetwithid | recherche d'un objet par son identifiant |
| @getobjet | recherche d'un objet par son numéro unique |
| @getelementat | recherche d'un ou plusieurs éléments à une position donnée ou à proximité |
| @getobjetat | recherche d'un ou plusieurs objets à une position donnée ou à proximité |
| @firstrelsem | pour parcourir les relations sémantiques |
| @nextrelsem | |
| @nbrelsem | |
| @getrelsem | |
| @getsrcerelsem | et rechercher les objets ou éléments source ou destination de ces relations |
| @getdestrelsem | |
| @getsrceli | pour rechercher les points source et destination d'une liaison |
| @getdestli | |
| @inversesensli | inverse le sens d'une liaison |
| @getsens | donne le sens d'une relation entre liaison et face |
| @firstobservation | fournit la première observation du document (eventuellement de numero donné) |
| @nextobservation | fournit l'observation suivante du document (eventuellement de numero donné) |

## Manipulation des données graphiques du document (éléments, objets, couches...):

| | |
|---|---|
| @nbcouches | nombre de couches du document |
| @getselectcou | sélection d'une couche |
| @selectcou | |
| @clearselectcou | |
| @reservesel | teste ou réserve un niveau de sélection |
| @layer | donne l'indice d'une couche d'après son nom |
| @layername | fournit ou fixe le nom d'une couche d'indice donné |
| @layercolor | fournit la couleur d'une couche |
| @getlayerinit | propriétés de la couche |

| | |
|---|---|
| @layerinit | |
| @layeradd | ajoutte une couche |
| @getidu | donne l'identifiant d'un objet |
| @getcouche | donne la couche d'un élément, objet, relation, masque... |
| @delcouche | supprime une couche |
| @getforme | forme d'un élément |
| @setforme | |
| @getcolor | couleur d'un élément, objet, masque... |
| @setcolor | |
| @getepaisseur | epaisseur d'un élément, objet, masque... |
| @setepaisseur | |
| @getatt | attributs d'un élément, objet, masque... |
| @clearatt | |
| @setatt | |
| @getclasse | classe d'un élément, objet, masque... |
| @setclasse | |
| @getlabel | donne l'étiquette d'un élément ou objet |
| @setlabel | |
| @getnum | recherche le numéro d'un  point ou objet |
| @getx | recherche ou attribue des coordonnées |
| @setx | |
| @gety | |
| @sety | |
| @getz | |
| @setz | |
| @z | |
| @getselect | sélection des élément, objet,masque... |
| @select | |
| @clearselect | |
| @razselect | |
| @textobj | concaténation des écritures d'un objet (séparées par un espace) |
| @gettext | fournit ou fixe le texte de l'écriture |
| @settext | |
| @getcentroide | centroide d'un élément, objet |
| @appartienta | informations sur les appartenances, inclusions, dépendances |
| @estinclusdans | |
| @dependde | |
| @getpoint | fournit le point de numéro donné |
| @getextents | fournit l'étendu d'un élément ou objet |
| @getbmpname | fournit ou fixe le nom du fichier raster d'une couche |
| @setbmpname | |
| @getbmp2tp | fournit la matrice de passage des coordonnées du raster vers coordonnées terrain ou papier, informations sur le bitmap (largeur, hauteur, dpi...) |
| @settp2bmp | |
| @setbmp2tp | |
| @scr2tp | conversion coordonnées écran en terrain et vice versa |
| @loadtrf | transformation courante de la fenêtre |
| @savetrf | |
| @gettrf | |
| @settrf | |
| @deleltobj | suppression d'élément ou d'objet |
| @delelttoobj | supprimer l'appartenance d'un élément à un objet |
| @deltextobj | supprimer l'appartenance des écritures d'un objet à cet objet |
| @addelttoobj | ajoutter un élément à un objet |
| @delete | effacer, supprimer du graphisme (éléments, objets) |
| @mask2obj | modifie les éléments d'un objet par un masque de modification |
| @modifie | modifier un objet ou élément par un masque de modification |
| @appliquemaskcla | modification en bloc par masques |
| @clearallsm | supprimer tous les signes de mitoyenneté |
| @ajusteptsurfc | fonctions d'ajustement et de positionnement d'éléments par rapport à d'autres éléments |
| @ajusteec | |
| @ajustept | |

## controle des données graphiques:

| | |
|---|---|
| @diffpt | comparaison de points |
| @checkmodele | test de conformité aux règles de graphisme imposées par le modèle |
| @texthorsface | détecte les écritures en dehors de leurs faces |
| @compareelt | comparaison d'éléments |
| @diffelt | |
| @compareobj | comparaisons d'objets |
| @objintegrity | recalcule la classe et couche ... des objets et teste leur intégrité |
| @controleid | controle la validité des identifiants |
| @objsansclasse | détecte les incohérences des classes entre les éléments et leurs objets |
| @classesansobj | |
| @seldblfaces | détecte les faces quasi semblables |

## manipulation des relations sémantiques:

| | |
|---|---|
| @makeallrelsem | création automatique des relations sémantiques pouvant se déduire |
| @delinvalrelsem | suppression de relation sémantiques invalides |
| @makerelsem | création automatique de relations sémantiques |
| @delallrelsem | suppressions de relations sémantiques |
| @addrelsem | ajout d'une relation sémantique |
| @delrelsem | supprimer une relation sémantique |
| @gettyperelsem | fournir le type de relation sémantique |
| @delselrelsem | supprimer des relations sémantiques |
| @selcrossrel | sélection de relations croisées |
| @nbrelsem | nombre de relations |
| @getrelsem | recherche une relation |
| @getsrcerelsem | recherche les objets ou éléments source ou destination de ces relations |
| @getdestrelsem | |

## modification des symboles:

| | |
|---|---|
| @makerattacht | création de signes de mitoyenneté en fonction des liaiosns et faces environnantes |
| @fixdec | calcule et fixe le posiitionnement droite ou gauche des déports d'écriture |
| @createsmwithpt | création des signes de mitoyenneté à partir de points |
| @makedeport | création de déports d'écriture à partir de polylignes |
| @createdeport | |
| @selliwithsm | détection de polyligne en fonction des signes de mitoyenneté |
| @createsmwithli | création de signe de mitoyenneté à partir d'une liaison |

## modification des observations:

| | |
|---|---|
| @observationcalcule | calcule une observation |
| @observationconcerne | indique si un point est composante d'une observation |
| @observationtype | fournit le type d'une observation |
| @addobservation | ajoutte une observation |
| @delobservation | supprime une observation |
| @selptobservation | sélectionne le ou les points d'observation |
| @getptobservation | fournit le point d'une observation |
| @getnumobservation | fournit le numero d'une observation |

## modification des autres données graphiques:

| | |
|---|---|
| @decompose | décompose une face complexe en faces simples |
| @makefacesatrou | reconstittion de faces à trous |

| | |
|---|---|
| @selextremity | sélection des extrémités de polylignes |
| @selext | propagation de sélection |
| @selectextclass | sélections d'éléments d'une classe donnée |
| @createfcwithaddfc | union de faces dans un milieu topologique |
| @creatept | création de point |
| @majdispocouches | réorganisation des couches |
| @majdiff | mise à jour d'une couche par une autre |
| @makeobjlbl | création d'objets à partir des étiquettes des éléments |
| @createobjwlbl | |
| @makeobjext | création d'objets à partir des données extra des éléments |
| @createobjwext | |
| @selop | opérations sur les sélections |
| @createobjwselelt | création d'objet à partir de la sélection |
| @createtextobj | création d'une écriture et attribution à un objet |
| @echangecouches | échange de couches |
| @decoupecouche | decoupe de couche |
| @ajustecouche | ajustement de couche par rapport aux autres couches |
| @transfertcouche | transfert de couche |
| @zapclasse | supprime tous les éléments d'une classe en respectant la hiérarchie |
| @fusiondblfaces | fusionne les faces quasi identiques |
| @hierarchise | hiérarchise le graphisme |
| @etiqueter | étiquetter des objets |
| @crois2faces | création des faces en fonction de croisillons |
| @createfcwithfc | opérations arithmétiques sur les faces dans un milieu topologique |
| @crois2obj | création de couples de liaisons en objet |
| @createec | création d'une écriture |
| @createautoobj | création automatique des objets |
| @conformemodele | mise en conformité automatique au modèle |
| @faces2crois | création des croisillons à partir de faces |
| @selneighbours | sélectionne les éléments voisins |
| @purgepts | suppression de points inutiles |
| @purgelis | purge de liaisons inutiles |
| @selectwithmask | sélection à partir de masque de recherche |
| @createliwithpt | création de liaisons à partir de points |
| @fusiondblobj | fusion d'objets identiques |
| @distrib | copie et distribution d'objets et élements à travers plusieurs couches |
| @createtext | création d'une écriture relative à un objet |
| @maj | opération en bloc de mise à jour |
| @forwards | amene un élément en avant plan |
| @backwards | amene un élément en arrière plan |
| @ordonne | ordonne les éléments d'une couche suivant un ordre établi par l'étiquette ou les données extra de l'élément |

## topologie

| | |
|---|---|
| @cardinality | controle de la cardinalité des objets et sélection |
| @cardinalityrel | controle de la cardinalité des relations sémantiques et sélection |
| @topologiecou | effectue la topologie de 1° niveau d'une couche |
| @topologiefc | sélectionne les faces qui s'intersectent |
| @puzzle | traitement de coincidences de faces les unes envers les autres |
| @puzzleobj | |
| @puzzlerel | |
| @topologie | |
| @topologieobj | |
| @topologierco | topologie par les relations de construction |
| @selproxptpt | sélectionne les points proches de points |
| @concatenept | fusion des points proches de points |
| @selmultli | sélectionne les liaisons pouvant être fusionnées |
| @concateneli | fusions des liaisons |
| @selproxptli | sélectionne les points à proximité de liaisons |
| @concatptli | fusion des points proches de liaisons |
| @topologieli | sélection de liaisons qui s'intersectent |
| @fusionlili | création de points aux intersection des liaisons |
| @cassertopologie | détruire la topologie entre deux éléments ou objets |

| | |
|---|---|
| @maillage | détection et sélection d'un réseau de liaisons topologique |
| @createfcwithli | création des faces à partir d'un réseau de liaisons dans un milieu topologique |
| @maillage2faces | |

## interface homme−machine

| | |
|---|---|
| @mdimove | déplacer une fenêtre |
| @fileopensavedlg | boite de dialogue de chargement ou sauvegarde de fichier |
| @hinttext | affichage d'un message dans la barre de statut |
| @getwindow | infos concernant la fenêtre en cours |
| @msgbox | boite de dialogue de message et choix (en milieu d'écran et messages longs) |
| @msgdlg | boite de dialogue de message et choix (en marge et messages courts) |
| @propdlg | boite de dialogue d'edition des propriétés d'un objet identifiable |
| @displayat | affichage à un point précis |
| @display | affichage d'un élément ou objet |
| @setvision | fixe la vision ou fournit la vision courante |
| @setmodestate | mode courant |
| @selecttravail | niveau courant de sélection |
| @setselecttravail | |
| @classetravail | classe courante |
| @setclassetravail | |
| @couchetravail | couche de travail |
| @setcouchetravail | |
| @recentrage | fonctions de zoom |
| @zoom | |
| @setmodeprogram | programme le mode utilisateur |
| @mdiorg | |
| @getvar | boite de dialogue de saisie de donnée |
| @getchoice | boite de dialogue de saisie d'un choix |
| @nbscriptcmd | nombre de scripts enregistrés |
| @razscriptcmd | efface les scripts |
| @addscriptcmd | ajoutte un script |
| @scriptcmdstate | donne ou modifie l'état d'une commande (coché/non coché ou bouton enfoncé/non enfoncé) |
| @beep | émet un avertissement sonore ou manipule l'indicateur MSG de la barre de statut |

## programmation TED

| | |
|---|---|
| @exit | sortie de TED |
| @vartype | fournit le type de la variable |
| @dumpvar | dump des variable courantes dans ERRORxxx.LOG |
| @while | boucle 'tant que' |
| @error | retourne une erreur |
| @list | liste de fonctions TED |
| @return | renvoie une valeur (sortie du programme/fichier TED) |
| @foreach | itération sur des noms de fichiers ou autre |
| @break | rupture de séquence |
| @exec | exécution d'un programme TED |
| @param | transmission des paramètres dans une sous procédure TED |
| @refvar | fournit une référence de variable en transmission d'une procédure @exec |
| @run | exécution d'un programme externe |
| @if | condition 'si' |
| @for | boucle 'for' |
| @setdebug | mode débogage |
| @setvar | enregistrement variable existante |
| @setlvar | enregistrement variable locale |
| @local | création initialisation variable locale |
| @setgvar | enregistrement initialisation de variable globale |
| @global | création initialisation d'une variable globale |
| @ivar | accès indirect à une variable |

| | |
|---|---|
| @exist | indique si un fichier ou répertoire existe |
| @date | date courante |
| @defined | indique si une variable est définie |
| @getenv | fournit une variable d'environnement |
| @setenv | modifie une variable d'environnement |

Syntaxe du langage

# Liste des fonctions TED classées par ordre alphabétique:

@abs
@absvar
@add
@addcadreformatpage
@adddatabase
@adddbrule
@adddbprop
@adddbpropwithid
@addelttoobj
@addformeface
@addformeliaison
@addformepoint
@addobservation
@addpattern
@addrelsem
@addscriptcmd
@addvar
@ajustecouche
@ajusteec
@ajustept
@ajusteptsurfc
@and
@angle
@appartienta
@appliquemaskcla
@backwards
@bellman
@beep
@binand
@binnot
@binor
@bissectrice
@break
@c_str
@cadreformatpage2bk
@calculeinterp
@calculetransf
@cardinality
@cardinalityrel
@casestr
@cassertopologie
@checkmodele
@chgtsysteme
@chgtsystemept
@classesansobj
@classetravail
@clearallsm
@clearatt
@clearselect
@clearselectcou
@closeview
@codeinsee
@commune
@compareelt
@compareobj
@compidu
@concat
@concateneli
@concatenept
@concatptli
@concatvar
@conformemodele
@contains
@controleid
@copl

@copy
@cos
@couchetravail
@createautoobj
@createdeport
@createec
@createfcwithaddfc
@createfcwithfc
@createfcwithli
@createliwithpt
@createobjwext
@createobjwlbl
@createobjwselelt
@creatept
@createsmwithli
@createsmwithpt
@createtext
@createtextobj
@crois2faces
@crois2obj
@currenttrelsem
@date
@dateedi
@dateincorp
@dateredi
@dbbrowse
@dbclass2dbstand
@dbclose
@dbcreateindex
@dbgetfield
@dbnext
@dbopen
@dbpack
@dbprevious
@dbseek
@dbsetdefault
@dbsetfield
@dbsettobegin
@dbsettoend
@dbstand2dbclass
@decompidu
@decompose
@decoupecouche
@defined
@delallrelsem
@delcadreformatpage
@delcouche
@deldatabase
@deleltdata
@deleltobj
@delelttoobj
@delete
@delformeface
@delformeliaison
@delformepoint
@delinvalrelsem
@delobservation
@delpattern
@delrelsem
@delselrelsem
@deltextobj
@dependde
@deselectall
@diffelt
@diffpt
@dijkstra
@display

@displayat
@dist
@disteltobj
@distentreelt
@distentreobj
@distobj
@distrib
@div
@divvar
@dms2r
@dumpvar
@echangecouches
@echelle
@echorigin
@equ
@error
@estinclusdans
@etiqueter
@exec
@exist
@exit
@export
@extract
@extractword
@faces2crois
@feuille
@fileopensavedlg
@first
@firstobjet
@firstobservation
@firstrelsem
@firstwithelt
@firstwithmask
@firstwithobj
@fixdec
@for
@foreach
@format
@forwards
@fusiondblfaces
@fusiondblobj
@fusionlili
@getapicce
@getatt
@getbmp2tp
@getbmpname
@getcentroide
@getchoice
@getclasse
@getcolor
@getcouche
@getdata
@getdbprop
@getdbpropwithid
@getdestli
@getdestrelsem
@getdoc
@getdocname
@getelementat
@geteltdata
@getenv
@getepaisseur
@getextents
@getforme
@getidalgo
@getidparm
@getidu

@maj
@majdiff
@majdispocouches
@makeallrelsem
@makedeport
@makefacesatrou
@makeobjext
@makeobjlbl
@makerattacht
@makerelsem
@mask2obj
@mdimove
@mdiorg
@modeincorp
@modifie
@modulo
@msgbox
@msgdlg
@mul
@multiprintbegin
@multiprintend
@multiprintimage
@multiprintmap
@multiprintmapat
@mulvar
@nbcadreformatpage
@nbclasses
@nbcouches
@nbeltforelt
@nbeltforobj
@nbformesface
@nbformesliaison
@nbformespoint
@nbobjforelt
@nbpatterns
@nbrelsem
@nbscriptcmd
@nbtrel
@neg
@negvar
@newdoc
@next
@nextobjet
@nextobservation
@nextrelsem
@nextwithelt
@nextwithmask
@nextwithobj
@not
@num
@numi
@objintegrity
@objsansclasse
@observationcalcule
@observationconcerne
@observationtype
@or
@ordonne
@orior
@out
@param
@paste
@popline
@print
@printat
@projapplygridshift
@projdatumtransform

@projfwd
@projgeocentric2geodetic
@projgeocentric2wgs
@projgeod
@projgeodetic2geocentric
@projgetgridshift
@projinv
@projinvgeod
@projtransform
@projwgs2geocentric
@propdlg
@purgelis
@purgepts
@pushline
@puzzle
@puzzleobj
@puzzlerel
@qupl
@r2dms
@razscriptcmd
@razselect
@recentrage
@refvar
@regexmatch
@rem
@replacestr
@reservesel
@return
@rightstr
@roundval
@run
@savedoc
@savetrf
@scr2tp
@scriptcmdstate
@section
@seldblfaces
@selcrossrel
@select
@selectcou
@selectextclass
@selecttravail
@selectwithmask
@selext
@selextremity
@selliwithsm
@selmultli
@selneighbours
@selop
@selproxptli
@selproxptpt
@selptobservation
@setatt
@setbmp2tp
@setbmpname
@setcadreformatpage
@setclasse
@setclassetravail
@setcodeinsee
@setcolor
@setcommune
@setcopl
@setcouchetravail
@setcurrenttrelsem
@setdata
@setdateedi
@setdateincorp

@setdateredi
@setdbprop
@setdbpropwithid
@setdebug
@setdocvue
@setechelle
@setechorigin
@seteltdata
@setenv
@setepaisseur
@setfeuille
@setformatpage
@setforme
@setformeface
@setformeliaison
@setformepoint
@setgvar
@setidalgo
@setidparm
@setinit
@setlabel
@setlvar
@setmask
@setmaskno
@setmodeincorp
@setmodeprogram
@setmodestate
@setorior
@setpattern
@setprid
@setqupl
@setr2dms
@setrem
@setsection
@setselecttravail
@settext
@settobj
@settp2bmp
@settrel
@settrf
@setvar
@setvision
@setx
@sety
@setz
@setzonegeo
@sin
@sqrt
@sqrtvar
@str2base
@sub
@substr
@subvar
@sup
@supe
@surf
@surffc
@surfobj
@tableload
@texthorsface
@textobj
@topologie
@topologiecou
@topologiefc
@topologieli
@topologieobj
@topologierco

@transfertcouche
@transfpt
@vartype
@while
@z
@zapclasse
@zonegeo
@zoom

# Liste des fonctions créatrices de variables

setvar(*)
setgvar(*)
setlvar
local
global

(*) suivant option choisie

# Liste des fonctions dont les paramètres sont non exécutés

for
foreach
if
setmask
setmaskno
while

# Liste des fonctions invalidant les variables pointeurs;

clearallsm
concateneli
concatenept
concatptli
createdeport
createsmwithpt
createsmwithli
createtextobj
delcouche
delete
fusiondblfaces
fusiondblobj
fusionlili
maj

# @abs

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val | STANDARD | valeur |

## Action:

|nomvar|
valeur absolue

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| |val| | NUM | résultat |

## Exemples:

| | |
|---|---|
| @abs(@mavar) | fournit <|mavar|> |

# @absvar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |

## Action:

[nomvar] <== [|nomvar|]
valeur absolue de la variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| [|nomvar|] | NUM | résultat (valeur de la variable après avoir pris sa valeur absolue) |

## Exemples:

| | |
|---|---|
| @absvar(mavar) | fournit <|mavar|> |

# @add

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | valeur 1 |
| val2 | STANDARD | valeur 2 |

## Action:

val1+val2
Attention: comportement particulier si les deux valeurs sont de type angulaire : 390gv+20gv = 10gv

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [val1+val2] | NUM | renvoie la somme |

## Exemples:

@add(@mavar,5.3)

# @addcadreformatpage

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice du cadre dans le format de page (un indice de 0 indique qu'on veut le cadre à l'avant plan, le dernier indice indique l'arrière plan)<br>*ex: si 4 cadres existent déjà, donner l'indice 4 indique que l'on veut mettre le nouveau cadre à l'arrière plan. Il y aura alors 5 cadres d'indice 0 à 4 inclus.* |
| nom | STANDARD | nom du cadre. Ce nom peut être composé de texte formattés (ex: "Section @section" indiquera d'inscrire "Section AB" dans le cadre, cf "textes formattés" pour plus d'infos) |
| x0 | STANDARD | coordonnée X du point supérieur gauche du cadre |
| y0 | STANDARD | coordonnée Y du point supérieur gauche du cadre |
| x1 | STANDARD | coordonnée X du point inférieur droite du cadre |
| y1 | STANDARD | coordonnée Y du point  inférieur droite du cadre |

## Action:

ajoutte un cadre de format de page au format de page courant. (ce format de  page modifié n'est pas sauvegardé dans la configuration de TopoCad).
L'indice fournit peut être quelconque car il est ajusté pour être valide (à 0 si l'indice fourni est négatif et en un indice indiquant l'arrière plan soit le plus fort indice en cas d'indice supérieur au maximum autorisé)
ATTENTION: Le nom ne doit pas être "@plan".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <nb> | NUM | indice du cadre inséré |

## Exemples:

@addcadreformatpage(0,"DUPONT Maurice",0,0,500,20)  ajoutte un cadre d'entète à la mise en page courante

# @adddatabase

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | Type de base de donnée<br>0=Standard<br>1=De Classe<br>2=Auxiliaire<br>3=Externe |
| classe | STANDARD | classe concernée par la base (si type=1) sinon 0 |
| nom | STANDARD | nom du type de la base (ex: Permis ou Parcelles) |
| prefixe | STANDARD | préfixe d'identifiant (4 caractères en majuscules) |

## Action:

Ajoutte une base de donnée à la table des bases pouvant être gérées par l'application. Si une base non standard de même nom/type/classe/prefixe existe, renvoie l'indice de cette base. Il est ensuite néccessaire d'ouvrir cette base de donnée par @dbopen pour pouvoir l'utiliser.

La base standard est automatiquement créée à l'initialisation de l'application (même si elle n'est pas ouverte) et ne peut être créée ou supprimée par l'utilisateur (une tentative de créer une base standard à la table des bases de données renvoie une erreur).

ATTENTION: le nom et le préfixe en cas de base de classe DOIT correspondre aux données fournies dans le Type d'Objet dans la section OBJET de topocad.ini. Les bases de classes sont automatiquement créées à l'initialisation de l'application (ainsi que les autres), fixer l'IdAlgo l'IdParm et le prefixe d'identifiant doit donc être complété par un ajout de la base de donnée par @adddatabase. Il en est de même si l'on charge un nouveau modèle fixant ces paramètres pour une nouvelle base de classe.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'ajouter la base de donnée | ERR | si erreur |
| N | NUM | index de la base créée ou trouvée |

## Exemples:

| | |
|---|---|
| @adddatabase(2,0,"Permis",PERM) | ajoutte une base auxiliaire de permis à l'application |
| @loadmodele(w:\communes\geofla.ini),<br>@adddatabase(1,25,COMMUNE,COMG),<br>@adddatabase(1,26,DEPARTEMENT,DEPT),<br>@adddatabase(1,27,REGION,REGN),<br>@dbopen(C25,w:\communes\geofla_commune.dbf),<br>@dbopen(C26,w:\communes\geofla_departement.dbf),<br>@loaddoc(w:\communes\geofla.map,1), | charge un nouveau modele définissant 3 nouveaux types d'objets pouvant gérer des données et donc fixant leur méthode de calcul de leur identifiant et leur préfixe puis ajoutte ces trois types de tables à la base de donnée de l'application avant de pouvoir ouvrir ces dernières. Les noms et prefixe ont les mêmes valeurs que celles fournies par le modèle. |

# @adddbprop

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom de la variable contenant l'objet |
| obj | ou | ou |
| | OBJ | objet dont il faut affecter la propriété |
| nomprop | STANDARD | nom de la propriété ou créer |
| value | STANDARD | valeur de la propriété |

## Action:

modifie ou affecte une propriété à l'objet "obj". Une base standard ou de ce type d'objet doit être ouverte et l'objet doit être en mesure de calculer son identifiant afin de se connecter à la base de donnée de TopoCad qui sera donc modifiée à l'issue de l'opération.

lorsqu'on ajoutte une propriété à une base de classe, le nom de la propriété étant un nom de champ de cette base, si le champ n'existe pas déjà, TopoCad rajouttera un champ à cette base de classe ce qui peut s'avérer plus lent que prévu. Si *NoAutoAddField* est à 1 alors cette opération n'est pas réalisée et la fonction renvoie une erreur.

Cette fonction gère les multipropriétés (la capacité de la base standard d'avoir plusieurs propriétés de même nom et de différentes valeurs)
Sur les bases de classe elle se comporte comme @setdbprop , ces dernières ne gérant pas les multipropriétés (fichiers DBF à index à clef unique)
Sur les bases standard, elle ajoutte la propriété si la valeur n'existe pas déjà.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de calculer l'identifiant de l'objet | ERR | si l'objet ne peut calculer l'identifiant |
| Impossible d'écrire la propriété de l'objet | ERR | si problème dans l'écriture de la propriété (base non ouverte ou non accesisible...) |
| OK | STR | propriété traduite et écrite |

## Exemples:

| | |
|---|---|
| @adddbprop(obj,SURFACE,@surfobj(@obj)) | calcule la surface de l'objet "obj" et écrit une propriété pour cet objet de nom "SURFACE" contenant le résultat |

# @adddbpropwithid

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| idu | STANDARD | identifiant complet de l'objet dont il faut modifier la propriété (avec préfixe) |
| nomprop | STANDARD | nom de la propriété à modifier ( modifier ou créer pour une base standard) |
| value | STANDARD | valeur de la propriété |

## Action:

modifie ou affecte une propriété à l'objet d'identifiant "idu". Une base standard ou de ce type d'objet doit être ouverte.

lorsqu'on ajoutte une propriété à une base de classe, le nom de la propriété étant un nom de champ de cette base, si le champ n'existe pas déjà, TopoCad rajouttera un champ à cette base de classe ce qui peut s'avérer plus lent que prévu. Si *NoAutoAddField* est à 1 alors cette opération n'est pas réalisée et la fonction renvoie une erreur.

Cette fonction gère les multipropriétés (la capacité de la base standard d'avoir plusieurs propriétés de même nom et de différentes valeurs)
Sur les bases de classe elle se comporte comme @setdbprop , ces dernières ne gérant pas les multipropriétés (fichiers DBF à index à clef unique)
Sur les bases standard, elle ajoutte la propriété si la valeur n'existe pas déjà.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'écrire la propriété de l'objet d'IDU xxxx | ERR | si problème dans l'écriture de la propriété (base non ouverte ou non accesisible...) |
| OK | STR | propriété traduite et écrite |

## Exemples:

| | |
|---|---|
| @adddbpropwithid(@idu,SECTION,@section) | ajoutte le nom de la section comme propriété de l'objet dont l'identifiant est dans la variable "idu" |
| ; ajouts de 2 propriétés de valeurs différentes<br>@adddbpropwithid(@idparc,FILLE,"0030000A1125")<br>@adddbpropwithid(@idparc,FILLE,"0030000A1126") | ajout de 2 propriétés "fille" d'une parcelle de valeurs différentes |
| ; ajout de 2 propriétés de valeurs identiques<br>@adddbpropwithid(@idparc,FILLE,"0030000A1125")<br>@adddbpropwithid(@idparc,FILLE,"")<br>@dbsetfield(0,VALUE,"0030000A1125") | ajout de 2 propriétés "fille" d'une parcelle ayant la même valeur 0030000A1125 |

# @adddbrule

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| rule | STANDARD | règle de saisie suivie de ses paramètres |

## Action:

Ajoutte une règle de saisie de propriété dans l'application. Si une règle de saisie de même nom existe déjà pour le même type d'objet, la règle ne sera pas ajoutée et aucune erreur n'est renvoyée (considère que l'opération s'est déroulée correctement)

La règle de saisie existante peut cependant être modifiée dans sa valeur par défaut, son lien avec une base auxiliaire ou ses drapeaux (DBRULE_READONLY)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| OK | STR | règle ajoutée |

## Exemples:

@adddbrule(5 4 0 0 "numero_parcelle" "0" 2 0 "0" 0 "9999"), ajoutte la règle de saisie sur la propriété *numero_parcelle* concernant les objets de classe *parcelle* à l'application
La règle se décompose comme suit:
5 = classe (ici parcelle)
4 = type de règle (ici entier borné)

- 0 = DBRULE_UNDEF
- 1 = DBRULE_STRING = chaine quelconque
- 2 = DBRULE_NUMERO = numéro de point
- 3 = DBRULE_INT = entier
- 4 = DBRULE_INTRANGE = entier borné (2 paramètres suivant indiquent les bornes)
- 5 = DBRULE_REEL = réel (2 décimales par défaut)
- 6 = DBRULE_COORD = coordonnées
- 7 = DBRULE_ANGLE = angle (ex "254.2154gv")
- 8 = DBRULE_DISTANCE = distance
- 9 = DBRULE_REELPOS = réel positif
- 10 = DBRULE_REELRANGE = réel borné
- 11 = DBRULE_DATE = date
- 12 = DBRULE_TIME = heure
- 13 = DBRULE_ENUM = valeurs d'enum ou chaines imposés
- 14 = DBRULE_ENUMLIBRE = ensemble de chaines proposées et libre saisie
- 15 = DBRULE_FILEEXEC = nom de fichier à exécuter
- 16 = DBRULE_FILTER = filtre
- 17 = DBRULE_PICTURE = masque de saisie (EnumVal= AutoFill, EnumString= pict)
- 18 = DBRULE_TIMESTAMP = date et heure

0 = drapeaux (0x100 = propriété en lecture seule)
0 = lien

- 0 = pas de lien
- 1 à ... = indice de la table auxiliaire

"numero_parcelle" = nom de la propriété (ex: nom du champ en cas de base de classe)

"0" = valeur par défaut pour la propriété (fixe egalement la largeur du champ en cas d'ajout de champs d'une base de classe)
2 = nombre de paramètres suivants

Chaque paramètre est un couple (entier + chaine)

0 "0" = borne inférieure (donnée par la chaine)
0 "9999" = borne supérieure (donnée par la chaine)

# @addelttoobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obj | OBJ | objet auquel il faut ajoutter l'élément |
| elt | PT<br>LI<br>FC<br>EC | élement à ajoutter à l' objet |

## Action:

Ajoutte l'élément à l'objet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | l'élément a été ajouté à l'objet |

## Exemples:

| | |
|---|---|
| @addelttoobj(@obj,@elt) | ajoutte l'élément à l'objet |

# @addformeface

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| chaine forme | STANDARD | chaine de description de la forme |

## Action:

ajoutte la forme de face décrite par sa chaine de définition. La nouvelle forme est ajoutée en queue du tableau des formes de face et prend donc l'indice immédiatement supérieur au dernier indice existant.
l'indice fourni dans la chaine de decsription est automatiquement rectifié si besoin est.
la chaine de description de la forme est la même que celle fournie dans le fichier TopoCad.ini mais les guillemets sont remplacés par les caractères "\x22".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <ind> | NUM | indice de la forme de face ajoutée |

## Exemples:

| | |
|---|---|
| @addformeface("0 \x22Mon Cimetière\x22 R 0 0 0") | ajoutte une nouvelle forme de face "Mon cimetiere" de type raster et référençant le premier motif de l'application. |

# @addformeliaison

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| chaine forme | STANDARD | chaine de description de la forme |

## Action:

ajoutte la forme de liaison décrite par sa chaine de définition. La nouvelle forme est ajoutée en queue du tableau des formes de liaison et prend donc l'indice immédiatement supérieur au dernier indice existant.
l'indice fourni dans la chaine de decsription est automatiquement rectifiée si besoin est.
la chaine de description de la forme est la même que celle fournie dans le fichier TopoCad.ini mais les guillemets sont remplacés par les caractères "\x22".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <ind> | NUM | indice de la forme de liaison ajoutée |

## Exemples:

| | |
|---|---|
| @addformeliaison("0 \x22Mes Pointillés\x22 0 Trait 3 Vide 7 0") | ajoutte une nouvelle forme de liaison "Mes Pointillés" |

# @addformepoint

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| chaine forme | STANDARD | chaine de description de la forme |

## Action:

ajoutte la forme de point décrite par sa chaine de définition. La nouvelle forme est ajoutée en queue du tableau des formes de point et prend donc l'indice immédiatement supérieur au dernier indice existant.
l'indice fourni dans la chaine de decsription est automatiquement rectifiée si besoin est.
la chaine de description de la forme est la même que celle fournie dans le fichier TopoCad.ini mais les guillemets sont remplacés par les caractères "\x22".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <ind> | NUM | indice de la forme de point ajouté |

## Exemples:

| | |
|---|---|
| @addformepoint("3 \x22MaBorne\x22 \x22BO\x22 PenC BrushE EllipseE −50 0 −50 0 50 1 50 1 0") | ajoutte une nouvelle forme de point "MaBorne" |

# @addobservation

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | STD | observation sous forme texte |

## Action:

ajoutte une observation au document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | erreur de syntaxe dans le texte fournissant l'observation |
| obs | OBS | observation ajoutée ou NULL si n'a pu être ajoutée |

## Exemples:

| | |
|---|---|
| @addobservation("PtAligne 37765,A=37754,B=37752,AX=−3.560,") | renvoie le type d'observation de "monobs" |

# @addpattern

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| fichier | STANDARD | nom du fichier BMP constituant le motif sans chemin (ex: "CIMETISR.BMP" |

## Action:

ajoutte le motif de fichier BMP fourni. Le nouveau motif est ajouté en queue du tableau des motifs et prend donc l'indice immédiatement supérieur au dernier indice existant.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| XXXX.BMP : fichier bitmap attendu inexistant! | ERR | si le fichier BMP est inexistant dans le répertoire de l'application |
| <ind> | STR | indice de la forme du motif ajouté (de 0 à ...) |

## Exemples:

| | |
|---|---|
| @addpattern("CIMETISR.BMP") | ajoutte un nouveau motif |

# @addrelsem

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | STANDARD<br>PT<br>LI<br>FC<br>EC<br>OBJ | variable indiquant la source ou directement source de la relation |
| dest | STANDARD<br>PT<br>LI<br>FC<br>EC<br>OBJ | variable indiquant la destination ou directement destination de la relation |
| type | STANDARD | type de relation (de 0 à NbTRel−1) |

## Action:

ajoutte une relation sémantiques entre une source et une destination de type donné.
un type non désiré pour la source ou la destination, c'est à dire ne correspondant pas pour ce type de relation, renvoie une erreur

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <rel> | RELSEM | 0 si la relation existait déjà<br>la relation sémantique, si celle ci a été ajoutée |

## Exemples:

| | |
|---|---|
| @addrelsem(@pt1,@pt2,0) | ajoutte une relation entre le point 1 et le point 2 de type "correspondance" |
| @addrelsem(pt1,pt2,0) | idem |
| @addrelsem(@obj1,@obj2,6) | ajoutte une relation de type 6=relation numvoie => parcelle entre l'objet obj1 et l'objet obj2. Si obj1 n'est pas un objet "numero de voirie" ou si l'objet obj2 n'est pas un objet "parcelle", alors la fonction renvoie une erreur. |

# @addscriptcmd

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom | STANDARD | nom de la commande |
| tedfile | STANDARD | nom du fichier script TED (chemin complet) |
| bmpfile | STANDARD | nom du fichier BMP (chemin complet) |
| etat | STANDARD | état de la commande script (de 0 à ...) |
| nbetats | STANDARD | nombre d'états possibles ou −1 si commande de mise en mode programmé |

## Action:

Cette commande ajoutte une commande à l'application dans le menu *Script*

*nom* indique le nom de la commande s'ajoutant au menu "script".
Si *nom* est vide, alors l'application considère qu'il s'agit d'un séparateur.

Si *bmpfile* est vide, alors rien n'est inséré dans la boite à outils.
Si *bmpfile* est "Separator" (et *nom* est vide), alors un icone séparateur est inséré dans la boite à outils.
Si *bmpfile* est fourni, alors le fichier BMP désigné est utilisé pour dessiner le bouton.

Si *nbetats* est supérieur à 1, alors la commande de menu ne pourra être cochée et le bouton ne pourra être à l'état enfoncé et dans ce cas, l'application recherche et charge les fichiers de nom *bmpfile0.bmp, bmpfile1.bmp, bmpfile2.bmp....bmpfile10.bmp*...etc jusqu'au nombre *nbetats* de fichiers atteint. Le bouton avec l'image d'indice *etat* sera affiché (l'indice pouvant prendre les valeurs de *0* à *nbetat−1*).

Si *nbetats* est inférieur ou égal à 1, alors la commande de menu et le bouton seront respectivement cochée et enfoncé suivant la valeur de *etat* qui peut prendre les valeurs 0 (non coché/non enfoncé) ou 1 (coché/enfoncé)

Si nbetats est −1, il s'agit alors d'une commande de mise en mode programmé (ON ou OFF) et le fichier unique est bmpfile.bmp. Dans ce cas @setmodeprogram doit avoir fixé au préalable le nom du mode programmé par @setmodeprogram(8,"NomMode") au préalable afin que la commande @addscriptcmd enregistre et sache quel mode programmé surveiller. Ainsi, si plusieurs mode programmés coexistent, @scriptcmdstate changera l'état (à OFF) des autres commandes de modes programmés.

L'exécution par le menu ou la boite à outils aura pour effet de lancer le programme TED désigné par *tedfile*.

Les commandes sont ajouttées les unes derrière les autres tant au niveau du menu que des boutons.
Il n'est pas possible d'enregistrer plus de 200 commandes en tout pour une instance de l'application.

Les fichiers BMP fournis doivent comporter 16 couleurs et avoir une taille de 20x20 pixels (afin d'avoir un résultat satisfaisant mais d'autres dimensions et profondeur sont possibles)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Mode programmé non établi! | ERR | L'utilisateur tente d'insérer une commande script renvoyant à un mode programmé qui n'a pas encore été initialisé par @setmodeprogram(8,"monmode"). |
| <ind> | NUM | renvoie l'indice de la commande de script (de 0 à 199) ou −1 si il existe déjà 200 commandes de script enregistrées et qu'il est alors impossible d'en rajouter |

## Exemples:

| | |
|---|---|
| @addscriptcmd("Topologie de la couche","c:\topocad\ted\topologie de la couche de travail.ted","c:\topocad\ted\topologie de la couche de travail.bmp", 0,0) | ajoutte une commande au menu script et un bouton de commande |
| @addscriptcmd("Controles edigeo","c:\topocad\ted\controles edigeo.ted","",0,0) | ajoutte une commande au menu script uniquement |
| @addscriptcmd("Controles edigeo","c:\topocad\ted\controles edigeo.ted","",1,0) | ajoutte une commande cochée au menu script uniquement |
| @addscriptcmd("Controles edigeo","c:\topocad\ted\controles edigeo.ted", "c:\topocad\bmp\controles edigeo.bmp", 1,0) | ajoutte une commande cochée au menu script et un bouton de commande enfoncé |

| | |
|---|---|
| @addscriptcmd("Controles","c:\topocad\ted\controles.ted","c:\topocad\bmp\controle",1,3) | ajoutte une commande non cochée au menu et un bouton pouvant être représenté par 3 images BMP fournies par controle0.bmp, controle1.bmp et controle2.bmp. L'image s'affichant par défaut est controle1.bmp |

# @addvar

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable |
| value | STANDARD | valeur à ajoutter à la valeur de la variable |

## Action:

[nomvar] <== [nomvar]+value
Attention: comportement particulier si les deux valeurs sont de type angulaire : 390gv+20gv = 10gv

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| impossible d'ajoutter à une variable pointeur | ERR | si la variable n'est pas standard |
| [nomvar] | NUM | renvoie la variable après y avoir ajoutté la valeur |

## Exemples:

@addvar(mavar,5.3)

# @ajustecouche

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche à ajuster (de 0 à Nb_Couches_document−1) |
| virtuel | STANDARD | indique si l'ajustement est virtuel (1 ou autre) ou non (0) |
| vect_rast | STANDARD | indique en cas d'ajustement non virtuel ce qui est ajusté<br>0x01 = raster<br>0x10 = vectoriel<br>0x11 = raster + vectoriel |

## Action:

ajustement réel :
les coordonnées des éléments de la couche et le raster de la couche sont transformés par la transformation en cours de la fenêtre (agit sur le vectoriel ou/et le raster suivant "vect_rast")
ajustement virtuel:
le positionnement de la couche par rapport aux autres couches est transformé par la transformation en cours de la fenêtre. On agit sur les coordonnées écran et donc une transformation avec un offset de +10,+10 déplace la couche en bas à droite par rapport aux autres couches, mais les coordonnées des éléments et le référencement du raster n'est nullement modifié. Dans ce cas le paramètre "vect_rast" n'a aucun effet (mais doit être valide)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| OK | STR | le positionnement a eu lieu |

## Exemples:

| | |
|---|---|
| @settrf(2,0,0,2,0,0),<br>@ajustecouche(3,0,0x10) | multiplie l'échelle du bitmap par 2 |

# @ajusteec

*Nb paramètres*
9

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| selsrce | STANDARD | niveau de sélection des écritures à modifier (de 0 à 31) |
| seldest | STANDARD | niveau de sélection des éléments directeurs de l'ajustement (de 0 à 31)<br>cette entrée n'est pas utilisée si l'ajustement sur l'objet est choisie (flag 8 est ON) |
| telt | STANDARD | Type d'élément sur lequel faire l'ajustement. peut être :<br>1=Points<br>2=Liaisons<br>4=Ecritures<br>16=Faces |
| epsilon | STANDARD | écart maxi que peut faire l'écriture pour se repositionner (au dela, l'écriture n'est pas modifiée) |
| dx | STANDARD | offset en X à opérer (relatif au système de coord courant ou celui défini par la liaison) |
| dy | STANDARD | offset en Y à opérer |
| or | STANDARD | offset en angle à opérer |
| flags | STANDARD | indicateurs binaires pour le type d'ajustement à faire (ajouter valeurs pour obtenir l'effet désiré)<br>1 = Offset en X utilisé<br>2 = Offset en Y utilisé (3=offset en X et Y)<br>4 = Offset en orientation utilisé<br>8 = Ajustement sur l'objet<br>16 = Ajuste le Déport d'écriture (flèche) plutot que l'écriture quand il existe<br>32 = Ajustement type présentation (le plus horizontal possible pour le sens donné)<br>64 = Ajustement au dessus de la direction donnée<br>128 = Ajustement au dessous de la direction donnée<br>256= Ajustement à l'extérieur des faces<br>512= Ajustement à l'intérieur des faces |

## Action:

repositionne l'écriture ou la flèche de rattachement d'écriture (déport d'écriture) et la réoriente par rapport à un élément (point,liaison,face,écriture) suivant les critères donnés.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N écritures/déports rectifiés | STR | N = Nombre d'écritures ou déports rectifiés |

## Exemples:

| | |
|---|---|
| @ajusteec(0,0,0,2,30,0,2,0,43) | ajuste les écritures sélectionnées au niveau 0 sur les liaisons de l'objet dont elles font partie en se positionnant dans le sens le plus lisible au milieu et au dessus à 2 metre de la liaison et orientée suivant la liaison. Si l'écriture et distante de plus de 30 mètres de la liaison, alors l'écriture n'est pas repositionnée et est sélectionnée au niveau 1. |

# @ajustept

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) des points à ajuster sur la destination |
| selsrce | STANDARD | niveau de sélection des points à ajuster (de 0 à 31) |
| couchedest | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) des liaisons sur lesquelles ajuster les points sources |
| seldest | STANDARD | niveau de sélection des liaisons destinations des points ajustés (de 0 à 31) |
| eps | STANDARD | distance maximale entre le point et la liaison au dela de laquelle l'ajustement ne peut être réalisé (tolérance), autrement dit : déplacement maximal du point |
| flags | STANDARD | indicateurs binaires pour le type d'ajustement à faire (ajouter valeurs pour obtenir l'effet désiré)<br>0x01 = AJUSTEPT_INSERE = insertion du point dans la liaison désiré (sinon un simple déplacement est effectué). Est sans effet si les couches source et destination sont différentes (pas d'insertion).<br>0x02 = AJUSTEPT_PROLONGE = prolongement désiré (lorsque le point est à une extrémité), sinon le point se projette perpendiculairement sur la liaison<br>0x04 = AJUSTEPT_NODONLYSEL = le caractère d'extrémité du point à déplacer est considéré par rapport aux seules liaisons sélectionnées au niveau 'selsrce' sinon le point est considéré comme une extrémité que s'il concoure à une unique liaison. N'a d'effet que si AJUSTEPT_PROLONGE est positionné. |

## Action:

repositionne les points sélectionnés au  niveau 'selsrce' sur les liaisons sélectionnées au niveau 'seldest' si le déplacement n'excède pas 'eps' (tolérance). Le type d'ajustement est donné par 'flags'.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <N> | NUM | N = Nombre de points sélectionnés n'ayant pu être ajustés |

## Exemples:

| | |
|---|---|
| @ajustept(0,1,0,2,0.01,1) | ajuste les points sélectionnés au niveau 1 sur les liaisons sélectionnées au niveau 2 si ceux−là sont à moins d'un cm de la liaison et insère ce point dans la liaison. |

# @ajusteptsurfc

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) des points à ajuster sur la destination |
| selsrce | STANDARD | niveau de sélection des points à ajuster (de 0 à 31) |
| fcdest | FC | face sur laquelle positionner les points |
| eps | STANDARD | distance maximale entre le point et la liaison au dela de laquelle l'ajustement ne peut être réalisé (tolérance), autrement dit : déplacement maximal du point |
| flags | STANDARD | indicateurs binaires pour le type d'ajustement à faire (ajouter valeurs pour obtenir l'effet désiré)<br>0x01 = AJUSTEPT_INSERE = insertion du point dans la liaison désiré (sinon un simple déplacement est effectué). Est sans effet si la couche source et la couche de la face destination sont différentes (pas d'insertion).<br>0x02 = AJUSTEPT_PROLONGE = prolongement désiré (lorsque le point est à une extrémité), sinon le point se projette perpendiculairement sur la liaison de la face<br>0x04 = AJUSTEPT_NODONLYSEL = le caractère d'extrémité du point à déplacer est considéré par rapport aux seules liaisons sélectionnées au niveau 'selsrce' sinon le point est considéré comme une extrémité que s'il concoure à une unique liaison. N'a d'effet que si AJUSTEPT_PROLONGE est positionné. |

## Action:

repositionne les points sélectionnés au  niveau 'selsrce' sur les liaisons de la face 'fcdest' si le déplacement n'excède pas 'eps' (tolérance). Le type d'ajustement est donné par 'flags'.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <N> | NUM | N = Nombre de points sélectionnés n'ayant pu être ajustés |

## Exemples:

| | |
|---|---|
| @ajusteptsurfc(0,1,@vfc,0.01,1) | ajuste les points sélectionnés au niveau 1 sur les liaisons de la face 'vfc' si ceux−là sont à moins d'un cm de la face et insère ce point dans la liaison de la face. |

# @and

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | première valeur |
| val2 | STANDARD | deuxieme valeur |

## Action:

val1 ET val2

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| val1 ET val2 | BOOL | renvoie le résultat de (val1 ET val2) |

## Exemples:

| | |
|---|---|
| @and(@var1,@var2) | renvoie [var1] ET [var2], var1 et var2 étant des noms de variables de type STANDARD |
| @and(2 liaisons sélectionnées,@var2) | la première valeur (STANDARD) est considérée vrai car la conversion donne 2 et non 0 qui donc est considéré VRAI. Le résultat sera donc équivalent à la valeur de la variable "var2" (TRUE ou FALSE/VRAI OU FAUX) |

# @angle

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| num1 | STANDARD PT | numéro du point 1 ou point 1 |
| num2 | STANDARD PT | numéro du point 2 ou point 2 |
| num3 | STANDARD PT | numéro du point 3 ou point 3 |

## Action:

renvoie l'angle que fait num2−>num3 par rapport à num2−>num1
si num3==num1 alors renvoie le gisement/orientement (suivant AngleSens) de num2 vers num1/num3

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Point inconnu | ERR | si un numéro de point donné n'existe pas |
| <Angle Num1−Num2−Num3> | ANG | la valeur renvoyée est un angle (type STANDARD) sous forme par ex "235.2147000000gv" de sens et d'unité courants (la précision est toujours de 10 décimales) |

## Exemples:

| | |
|---|---|
| @angle(25,12,47) | renvoie l'angle correspondant au gisement de la direction 12 vers 47 moins le gisement de la direction 12 vers 25 |

# @appartienta

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eltobj1 | STANDARD | Nom d'une variable contenant un objet ou un élément |
| | OBJ | Objet |
| | PT | Element Point |
| | LI | Element Liaison |
| | FC | Element Face |
| | EC | Element Ecriture |
| eltobj2 | STANDARD | Nom d'une variable contenant un objet ou un élément |
| | OBJ | Objet |
| | PT | Element Point |
| | LI | Element Liaison |
| | FC | Element Face |
| | EC | Element Ecriture |

## Action:

détermine si *eltobj1* appartient à *eltobj2*.

Un *point* appartient à un *point* si les coordonnées sont identiques à EpsilonAppFace près.
Un *point* appartient à une *liaison* si est sur la liaison (tolérance à l'écartement de la liaison nulle)
Un *point* appartient à une *face* si il se trouve dans la face (si il constitue un des points du polygone de la face)
Une *liaison* appartient à une *liaison* si elle est contenue dans cette liaison (tolérance à l'écartement de chacun des points de la liaison nulle)
Une liaison appartient à une face si un point quelconque de la liaison est dans la face.
Une face appartient à une face si l'intersection des deux faces est non nulle
Une écriture appartient à un point si les coordonnées du point d'insertion de l'écriture sont identiques à celles du point à EpsilonAppFace près.
Une écriture appartient à une face si le point d'insertion de l'écriture est dans la face.
Le test d'appartenance d'un élément ou d'un objet à un autre objet est fait de manière identique suivant la nature des objets
Tous les autres types de test d'appartenance échouent (par ex: eltobj1=FC et eltobj2=PT ) en renvoyant "ERREUR: Test d'appartenance impossible"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| ERREUR: Test d'appartenance impossible | ERR | test incohérent (ex une face appartient à une liaison...) |
| 0 ou 1 | BOOL | résultat du test |

## Exemples:

| | |
|---|---|
| @appartienta(@var1,var2) | teste si le contenu de la variable var1 (par ex: var1 est le nom d'une variable de type OBJ) appartient à l'objet de nom var2 |
| @appartienta(var1,var2) | idem (var1 étant de type standard, c'est la variable de ce nom qui est recherchée) |

# @appliquemaskcla

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche à traiter (ou −1 si toutes) |
| classe | STANDARD | classe concernée |
| sel | STANDARD | sélection à traiter |
| telt | STANDARD | Type d'élément sur lequel appliquer le masque de modification<br>1=Points<br>2=Liaisons<br>4=Ecritures<br>16=Faces |

## Action:

modifie tous les éléments de la couche "couche" sélectionnés au niveau "sel" avec le masque de modification de la classe "classe". La modification ne porte que sur les types d'éléments choisis par "telt"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | nombre d'éléments modifiés |

## Exemples:

| | |
|---|---|
| @appliquemaskcla(@couchetravail,@classetravail,@selecttravail,18) | applique les masques de modification de classe aux liaisons et faces sélectionnées |

# @backwards

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | PT | élement (ou éléments de l'objet) à déplacer dans la liste de priorité d'affichage |
| | LI | |
| elt | FC | |
| | EC | |
| | OBJ | |

## Action:

cette fonction déplace l'élément en tête de liste des éléments à afficher pour la couche, ce qui a pour effet de l'afficher en premier lors de l'affichage de la couche et donc de le mettre à l'arrière plan par rapport à tous les autres éléments de la couche.
Pour un objet ce sont les éléments de l'objet qui sont déplacer en tête de leur couche respective

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | l'élément a été déplacé |

## Exemples:

| | |
|---|---|
| @backwards(@elt) | met l'élément à l'arrière plan |

# @beep

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| freq | STANDARD | >0 : frequence en Hertz<br><0 : son windows<br>=0 : indicateur MSG dans la barre de statut |
| duration | STANDARD | suivant la valeur de freq :<br>−durée en millisecondes<br>−type de son windows<br>−modification (0 ou 1) ou interrogation (tte autre valeur) de l'indicateur MSG |

## Action:

émet un avertissement sonore :
si freq>0, alors il s'agit d'un bip du haut parleur de durée donné par la valeur absolue de 'duration'
sinon émet un son windows équivalent aux sons standards suivant la valeur de 'duration' :
1 = MB_ICONHAND
2 = MB_ICONQUESTION
3 = MB_ICONASTERISK
tout autre valeur = MB_ICONEXCLAMATION
si freq=0, alors 'duration' indique si on doit mettre ON (1) ou OFF (0) l'indicateur MSG de la barre de statut, ou interroge cet indicateur (pour toute autre valeur)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | son émis |
| 1 | STR | valeur de l'indicateur MSG de la barre de statut |
| 0 | STR | valeur de l'indicateur MSG de la barre de statut |

## Exemples:

| | |
|---|---|
| @beep(900,200) | émet un bip haut parleur |
| @beep(−1,0) | émet le son MB_ICONEXCLAMATION (carte son) |
| @beep(0,0) | met OFF l'indicateur MSG de la barre de statut |
| @beep(0,−1) | renvoie "0" ou "1" suivant si l'indicateur de la barre de statut MSG est OFF ou ON |

# @bellman

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | PT | point source |
| dest | PT | point destination |
| selsrce | STANDARD | niveau de sélection représentant le graphe ou −1 si toute la couche est à considérer comme étant le graphe |
| seldest | STANDARD | niveau de sélection qui servira à la sélection du chemin le plus court trouvé ou −1 si le chemin le plus court ne doit pas être sélectionné |
| dirname | STANDARD | nom de la donnée extra donnant le poids dans le sens direct de la liaison |
| revname | STANDARD | nom de la donnée extra donnant le poids dans le sens inverse de la liaison |

## Action:

Effectue un calcul par l'algorithme de bellman. cet algorithme calcule dans un graphe le plus court chemin d'un point source à un point destination.
Le graphe est représenté par l'ensemble des liaisons liées entre elles composant la couche des points source et destination. Il est possible de limiter le graphe aux liaisons sélectionnées de la couche en indiquant un niveau de sélection "selsrce". A l'issue du calcul, les liaisons composant le plus court chemin sont sélectionnées au niveau de sélection "seldest" s'il est fourni (différent de −1).
La fonction renvoie la plus courte distance parcourue du point "srce" au point "dest"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <dist> | NUM | fournit la distance du plus court chemin. |
| Calcul impossible! | ERR | impossible d'atteindre le point "dest" à partir du point "srce" |

## Exemples:

| | |
|---|---|
| @bellman(@getpoint(1), @getpoint(9),−1,0,pd,pi) | sélectionne au niveau 0 le plus court chemin du point 1 au point 9 pour l'ensemble des liaisons de la couche du point 1. |

# @binand

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | première valeur |
| val2 | STANDARD | deuxieme valeur |

## Action:

val1 ET val2 (binaire)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| val1 ET val2 (NB: exemple pour version 32 bits) | NUM | renvoie le résultat de (val1 ET val2) |

## Exemples:

| | |
|---|---|
| @setvar(var1,1,NUM),<br>@setvar(var2,0,NUM),<br>@binand(@var1,@var2) | renvoie [var1] AND [var2] = 0, var1 et var2 étant des noms de variables de type STANDARD |
| @setvar(var1,3,NUM),<br>@binand(2 liaisons sélectionnées,@var1) | la première valeur (STANDARD) est traduite en 2 soit 00000010 et la seconde est 00000011<br>Le résultat est donc 00000010 soit 2 |

# @binnot

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | première valeur |

## Action:

NOT val1 (binaire) soit le complément à 2

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| NOT val1 | NUM | renvoie le résultat de NOT val1 |

## Exemples:

@setvar(var1,5,NUM),
@binnot(@var1)

(NB: exemple pour version 32 bits)

renvoie NOT [var1] = ~00000101 = 11111111 11111111 11111111 11111010 = 4294967290 soit comme les entiers sont signés −6

# @binor

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | première valeur |
| val2 | STANDARD | deuxieme valeur |

## Action:

val1 OU val2 (binaire)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| val1 OU val2 (NB: exemple pour version 32 bits) | NUM | renvoie le résultat de (val1 OU val2) |

## Exemples:

| | |
|---|---|
| @setvar(var1,1,NUM),<br>@setvar(var2,0,NUM),<br>@binor(@var1,@var2) | renvoie [var1] OR [var2] = 1, var1 et var2 étant des noms de variables de type STANDARD |
| @setvar(var1,3,NUM),<br>@binor(2 liaisons sélectionnées,@var1) | la première valeur (STANDARD) est traduite en 2 soit 00000010 et la seconde est 00000011<br>Le résultat est donc 00000011 soit 3 |

# @binshift

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val | STANDARD | valeur entière (signée) à décaler ou ordre du bit (0 à 31) |
| decal | STANDARD | décalage sous forme d'entier signé (−31 à +31 en principe) |
| reverse | STANDARD | 0 ou 1 : indique si un decalge est opéré (0) ou bien une position d'un bit dans une valeur |

## Action:

si reverse = 0:
effectue un decalage binaire (shift) de la droite vers la gauche de "decal" bits si "decal" est positif, de la gauche vers la droite de "−decal" bits si "decal" est négatif
si reverse = 1:
effectue l'opération inverse c'est à dire retrouve la position de l'unique bit positionné (de 0 à 31) de la valeur fournie en "val"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| newval (NB: exemple pour version 32 bits) | NUM | renvoie le résultat |

## Exemples:

| | |
|---|---|
| @setvar(var1,3,NUM),<br>@setvar(var2,2,NUM),<br>@binshift(@var1,@var2,0) | renvoie 0x00000011 décalé de 2 soit 0x00001100 = 12 |
| @setvar(var1,12,NUM),<br>@setvar(var2,−2,NUM),<br>@binshift(@var1,@var2,0) | renvoie 0x00001100 décalé de 2 soit 0x00000011 |
| @setvar(var1,3,NUM),<br>@binshift(@var1,0,1) | renvoie erreur : 3 = 0x00000011 a plusieurs bits de positionnés |
| @setvar(var1,8,NUM),<br>@binshift(@var1,0,1) | renvoie 3 car 8 = 0x00001000 est le bit d'indice 3 positionné |

# @bissectrice

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ang1 | ANG | premier angle |
| ang2 | ANG | second angle |

## Action:

fournit l'angle de la bissectrice entre ang1 et ang2.
si l'écart entre ang1 et ang2 est supérieur à PI/2, renvoie une erreur

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de calculer une moyenne d'angles écartés de plus de PI/2 | ERR | valeurs hors norme |
| <ang_bissectrice> | ANG | renvoie l'angle bissecteur (ang1+ang2)/2 toujours en valeur [0,2*PI[ dans l'unité et sens courant. |

## Exemples:

| | |
|---|---|
| @setvar(ang1,30gv,ANG),<br>@setvar(ang2,50gv,ANG),<br>@bissectrice(@ang1,@ang2) | renvoie 40.0000gv de type ANG |
| @setvar(ang1,30gv,ANG),<br>@setvar(ang2,390gv,ANG),<br>@bissectrice(@ang1,@ang2) | renvoie 10.0000gv de type ANG |
| @setvar(ang1,300gv,ANG),<br>@setvar(ang2,90gv,ANG),<br>@bissectrice(@ang1,@ang2) | renvoie une erreur |
| @setvar(ang1,30gv,NUM),<br>@setvar(ang2,50gv,ANG),<br>@bissectrice(@ang1,@ang2) | renvoie une erreur (il ne s'agit pas de deux angles) |

# @break

*Nb breakètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | STANDARD | renvoie le paramètre fourni et provoque une rupture dans une liste, un "while". |

## Action:

renvoie simplement le paramètre et interrompt le déroulement d'une boucle ou d'une liste

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| parm | ??? | |

## Exemples:

```
@while(@inf(vcou,@nbcou),
   @list(
      @if(@equ(@layername(@vcou,""),"COUCHESTOP"),
         @break(@vcou),
         @next(vcou,-1)
         )
       )
   )
```

parcoure les couches et quitte la boucle à la rencontre d'une couche nommée "COUCHESTOP". Renvoie cette couche.

# @c_str

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | STANDARD | chaine de caractère |
| | ou | ou |
| | PT, LI, FC, OBJ, MASK... | valeur d'une variable pointeur à traduire en chaine de caractère compréhensible ("Point ...", "NULL" si pointeur nul...etc) |

## Action:

convertit tout type de résultat en chaine de caractère.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| parm | STR | le paramètre fourni est renvoyé sous forme de chaine de caractère dans tous les cas. |

## Exemples:

| | |
|---|---|
| @c_str(@var) | sortie de la valeur de var |

# @cadreformatpage2bk

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice du cadre dans le format de page. Cet indice doit être un indice valide pour le format de page courant. |

## Action:

déplace le cadre d'indice donné en fin de liste afin de le mettre en arrière plan.
renvoie le nombre de cadres du format de page courant

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <NbCadres> | NUM | nombre de cadres du format de page courant |

## Exemples:

| | |
|---|---|
| @cadreformatpage2bk(4) | met le cadre d'indice 4 dans le format de page courant (soit le cadre plan) en arrière plan (se dessine avant tous les autres) |

# @calculeinterp

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type d'interpolation à faire sous forme d'entier (champ de bits)<br>bit 0 :<br>• 0 = vecteur<br>• 1 = scalaire<br>bit 1 :<br>• 0 = pas de modif des points<br>• 1 = modif des points en cas d'interpolation vectorielle<br>bit 2 :<br>• 0 = interpolation gravitaire<br>• 1 = interpolation bilinéaire |
| sel | STANDARD | niveau de sélection des points à interpoler (0 à 31, −1 si on ne tient pas compte du niveau de sélection) |
| couche | STANDARD | couche à interpoler (−1 si ttes les couches) |
| nomxdata | STANDARD | nom de la donnée extra à initialiser (en cas d'interpolation scalaire) |

## Action:

Calcule une interpolation gravitaire ou bilinéaire à partir des correspondances trouvées dans le document et charge les résultats dans les données extra des éléments.

Si il s'agit d'une interpolation vectorielle gravitaire:

Les coordonnées des points peuvent ne pas être modifiées mais l'offset à réaliser est chargé dans les 2 cas dans les données extra des points sources, données extra de noms "dx" et "dy". dx et dy sont donc le résultat de la transformation élastique (se composant obligatoirement au début d'une transformation d'helmert, puis d'une interpolation vectorielle d'ordre 2 par défaut)

Si il s'agit d'une interpolation vectorielle bilinéaire:

Les coordonnées des points peuvent ne pas être modifiées mais les nouvelles coordonnées sont alors chargées dans les données extra des points sources, données extra de noms "cx" et "cy". cx et cy sont donc les coordonnées résultat de l'interpolation.

Si il s'agit d'une interpolation scalaire :

La valeur scalaire à interpolée doit se trouver dans la donnée extra de nom "nomxdata" des poles. Si une donnée extra est innexistante dans un pole elle est considérée alors nulle pour l'interpolation gravitaire, cela provoque une erreur pour l'interpolation bilinéaire. Les points sources sont alors annotés avec la valeur scalaire interpolée stockée dans la donnée extra de nom fourni par "nomxdata". (Il n'y a pas de transformation d'helmert préalable dans ce cas pour l'interpolation gravitaire).

Pour l'interpolation gravitaire, dans les deux cas les valeurs des vecteurs (après transformation pour l'interpolation vectorielle et brute pour l'interpolation scalaire) aux poles sont stockées dans les données extra des points sources (poles) de noms "vx" et "vy"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <N> | NUM | Renvoie le nombre de points faisant objet de l'annotation des données extra ou modifiés |

## Exemples:

| | |
|---|---|
| @calculeinterp(0,−1,5,"") | calcule et effectue une transformation élastique de la couche 5 du plan avec les correspondances du document |

# @calculetransf

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de transformation sous forme d'entier<br>0 = isométrique<br>1 = helmert<br>2 = harmonique<br>3 = translation optimale |
| nomfich | STANDARD | nom du fichier de rapport de la transformation à générer (chemin complet)<br>si "" est fournit, alors un fichier par défaut est fourni dans le répertoire temporaire de l'application sous format TRANSFxx.TXT<br>si "0" est fourni, aucun fichier n'est généré |

## Action:

Calcule une transformation à partir des correspondances trouvées dans le document et la charge dans la fenêtre courante.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <EMQ> | NUM | Renvoie l'Erreur Moyenne Quadratique de la transformation |

## Exemples:

| | |
|---|---|
| @calculetransf(1,"") | calcule une transformation d'helmert qui est chargée dans la transformation courante de la fenêtre |

# @cardinality

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

controle la cardinalité des objets de classes actives pour la couche courante et sélectionne ceux dont la cardinalité n'est pas conforme au SCD.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Impossible de controler la cardinalité des objets : des objets ne sont pas intègres (éléments sur plusieurs couches) | ERR | si les paramètres ont des types non appropriés |
| Controle de la cardinalité des objets de classe active de la couche X : NN anomalies détectées. | STR | renvoie un message indiquant le résultat de l'opération |

## Exemples:

@cardinality

# @cardinalityrel

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| dir | STANDARD | direction de relation:<br>1=directe<br>2=inverse<br>3=les deux |
| type | STANDARD | type de relation (de 0 à NbTRel−1) |
| couchesrce | STANDARD | couche source (de 0 à NbCouches−1) ou −1 si ttes |
| couchedest | STANDARD | couche destination (de 0 à NbCouches−1) ou −1 si ttes |
| nivsel | STANDARD | niveau de sélection à positionner (de 0 à 31) |

## Action:

controle la cardinalité des relations de type donné entre les objets ou élements des couches couchesource et couchedest fournies et sélectionne les éléments ou objets sources ou destination dont la cardinalité des relations est incorrecte.
ATTENTION: Les objets doivent avoir leurs couches et classes mise à jour préalablement par *ObjIntegrity* avant de lancer cette procédure

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N elt/obj ont des relations anormales | NUM | renvoie un message indiquant le résultat de l'opération |

## Exemples:

| | |
|---|---|
| @cardinalityrel(1,9,0,1,0) | controle des relations directes Parcelle −> Subdiv de section et sélection des parcelles ayant des relations anormales. |

# @casestr

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| texte | STANDARD | texte dont il faut extraire la chaine |
| options | STANDARD | champ de bits :<br>bit 0 = majuscule (1) ou minuscules (0)<br>bit 1 = conversion maj/min (1) ou pas de conversion (0)<br>bit 2 = conversion en lettres non accentuées (1) ou pas de conversion (0) |

## Action:

convertit une chaine en majuscules ou minuscules

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <texte> | STR | renvoie la chaine convertie |

## Exemples:

| | |
|---|---|
| @casestr(commenter,3) | renvoie "COMMENTER" |
| @casestr(Test,2) | renvoie "test" |
| @casestr(Test,1) | renvoie "Test" |
| @casestr("reçu",7) | renvoie "RECU" |
| @casestr("où est l'été?",4) | renvoie "ou est l'ete" |

# @cassertopologie

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| objsrce | STANDARD<br>OBJ<br>FC | objet/élément ou nom de variable contenant l'objet/l'élément source |
| objdest | STANDARD<br>ou<br>OBJ<br>FC<br>LI<br>PT<br>EC | objet/élément ou nom de variable contenant l'objet/l'élément destination |

## Action:

cette fonction casse la topologie entre objsrce et objdest. TopoCad détache "objdest" de "objsrce" et de tout autre objet s'ils sont attachés. La fonction peut agir sur les objets et éléments suivants

| ***source*** | ***destination*** |
|---|---|
| Objet | Objet |
| Objet | Face |
| Objet | Liaison |
| Objet | Point |
| Face | Face |

La fonction renvoie une erreur dans tout autre cas.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STANDARD | traitement réalisé |

## Exemples:

| | |
|---|---|
| @cassertopologie(obj1,obj2) | sépare les deux objets |

# @checkmodele

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche concernée (0 à NbCouches−1) ou −1 si toutes |
| nivselsrce | STANDARD | niveau de sélection des éléments à scruter (−1 si tous) |
| telt | STANDARD | type d'élément à scruter : combinaison binaire de :<br>1 = Points<br>2 = Liaisons<br>4 = écritures<br>16 = faces |
| nivseldest | STANDARD | niveau de sélection à positionner |
| priorityobj | STANDARD | considère ou non l'appartenance à l'objet dans le controle de conformité |

## Action:

scrute tous les éléments concernés, et fournit leur classe théorique d'après la hierarchie du SCD et en regardant la classe des éléments en relation directe avec celui−ci (les faces pour les liaisons et les liaisons pour les points). Sélectionne les éléments ayant une classe de priorité inférieure à celle théorique (un trait qui devrait être parcelle au lieu de bati dur) et les élements dont le type (pt, li, fc,ec) ne devrait pas exister pour la classe.
On ne peut donner à cette fonction un niveau de sélection source et destination identique (renvoie alors "Paramètre invalide").
Si priorityobj est 0, l'appartenance d'un élément à l'objet ne rentre pas en ligne de compte dans le calcul de la classe théorique

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N éléments sélectionnés | NUM | nb d'éléments non conformes au SCD |

## Exemples:

| | |
|---|---|
| @checkmodele(−1,−1,23,0,1) | sélectionne au niveau 0 tous les éléments du document non conforme à la hiérarchie du SCD |

# @chgtsysteme

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| srce | STANDARD | système source sous forme de code Edigéo du systeme |
| dest | STANDARD | système destination sous forme de code Edigéo du systeme |
| vect_rast | STANDARD | indique ce qui est transféré de zones de la couche<br>0x01 = raster<br>0x10 = vectoriel<br>0x11 = raster + vectoriel |
| repercute | STANDARD | indique si la zone destination est répercutée sur le document (1) ou non (0) |

## Action:

Effectue un changement de système de coordonnées pour la couche "couche".
Le changement peut intervenir sur la partie vectorielle de la couche ainsi que sur la partie raster. Dans ce dernier cas, le nouveau raster est calculé par une approximation faite par une transformation d'helmert effectuée sur les 4 points extrèmes du raster complet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Changement de système impossible | ERR | le système géodésique/ la projection ne sont pas correctement définies dans TOPOCAD.INI |
| Ok | STD | Transfert réalisé |

## Exemples:

| | |
|---|---|
| @chgtsysteme(2,LAMB3,LAMB2,0x11,0) | effectue un changement de zone lambert de la couche 2 de Lambert III à Lambert II pour l'ensemble de la couche (vectoriel et raster) sans modifier les propriétés du document. |

# @chgtsystemept

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| pt | PT | point à modifier |
| srce | STANDARD | systeme de coordonnée source sous forme de code EDIGEO du système |
| dest | STANDARD | systeme de coordonnée destination sous forme de code EDIGEO du système |

## Action:

Effectue un changement de système de coordonnées pour le point "pt"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STD | Conversion réalisée |

## Exemples:

| | |
|---|---|
| @chgtsystemept(@monpt,LAMB2,LAMB3) | effectue un changement de zone lambert de Lambert II à Lambert III pour le point "monpt" |

# @classesansobj

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe concernée à scruter (de 1 à NbTObj−1) |
| nivsel | STANDARD | niveau de sélection à positionner (de 0 à 31) |
| direct | STANDARD | indique si la relation est recherchée directement ou indirectement<br>1 (bit 0) = direct<br>2 (bit 1) = indirectement |

## Action:

Sélectionne tous les éléments d'une classe n'appartenant pas directement ou indirectement (la liaison d'une face constituant l'objet) à un objet de la même classe.
Si on veut effectuer la recherche de la relation directement et indirectement en même temps, alors combiner les 2 bits (soit 3).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N éléments ont été sélectionnés | NUM | N = Nombre d'éléments en anomalie |

## Exemples:

| | |
|---|---|
| @classesansobj(0,5,0,2) | recherche toutes les liaisons entourant les objets parcelles et n'étant pas de classe parcelle sur la couche 0. |

# @classetravail

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie la classe de travail (de 0 à ...).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [ClasseTravail] | NUM | valeur de la classe de travail (0 = <sans classe>) |

## Exemples:

@classetravail

# @clearallsm

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | niveau de sélection à considérer (de 0 à 31) ou −1 si toute la couche est concernée |

## Action:

supprime tous les symboles signes de mitoyenneté de la couche de travail dont les liaisons sont sélectionnées au niveau "nivsel".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | signes supprimés |

## Exemples:

@clearallsm(0)

# @clearatt

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément, un objet ou un masque |
| | ou | ou |
| eltobjmask | PT, LI, FC, EC, OBJ | élément dont les attributs sont à effacer |
| | ou | ou |
| | MASK... | masque dont les attributs sont à effacer |
| att | STANDARD | attributs à effacer à l'élément, l'objet ou le masque sous forme de champs binaires. Les valeurs sont une combinaisons de :<br><br>1= Numbered (numéroté, présentation)<br>2= HiddenMono<br>4= HiddenCoul<br>8= Simple<br>0x10= Locked<br>0x100= Italique<br>0x200= Gras<br>0x400= Barre<br>0x800= Souligne<br>0x1000= Opaque<br>0x1000000= AAjouter<br>0x2000000= ASupprimer |
| withsymb | STANDARD | agit sur les symboles de l'élément ou non (en cas de liaison ou écriture) ou sur les éléments en cas d'objet |

## Action:

efface les attributs d'un élément, un objet ou un masque. Pour un masque de modification, fait en sorte que ce dernier efface l'attribut, pour un masque de recherche fait en sorte qu'une recherche de l'absence d'attribut se fasse.
efface éventuellement les attributs des symboles associés de l'élément (écriture ou liaison)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @clearatt(@pt,1,0) | efface l'attribut "numéroté" au point "pt" |

# @clearselect

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | niveau de sélection à effacer |
| | STANDARD | nom de la variable contenant l'élément ou objet à désélectionner |
| eltobj | PT, LI, FC, EC,RELSEM ou OBJ | élément, relation ou objet à désélectionner |

## Action:

ote la sélection de l'élément ou l'objet fourni au niveau demandé

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <elt/obj> | ??? | renvoie l'élément ou l'objet |

## Exemples:

@clearselect(0,mavar)

# @clearselectcou

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| select | STANDARD | niveau de sélection (de 0 à 31) ou −1 si tous niveaux de sélection |
| couche | STANDARD | numéro de la couche à désélectionner (de 0 à NbCouches−1) ou −1 si toutes les couches sont concernées |

## Action:

désélectionne la couche (au niveau de sélection "select") dont le numero d'ordre est donné mais ne sélectionne pas les éléments composant cette couche

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @clearselectcou(0,0) | déselectionne la couche 0 au niveau de sélection 0 |
| @clearselectcou(−1,0) | efface tous les niveau de sélection de la couche 0 |
| @clearselectcou(−1,−1) | efface tous les niveaux de sélection de toutes les couches |

# @closeview

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| force | STANDARD | 1 => ferme la vue même si le document a été modifié sans avertissement<br>0 => ferme la vue et demande confirmation si le document a été modifié |

## Action:

Ferme la vue plan courante (il est également possible de fermer la vue texte courante).
Attention : Aucun document et vue n'est attribué à l'issue de cette commande au programme TED qui s'execute. Il est donc impératif soit de gérer soit même ses documents−vues, soit de faire suivre cette commande par une commande @loaddoc, @setdocvue ou @newdoc, soit de n'utiliser par la suite que des commandes ne demandant pas qu'un environnement document−vue soit présent.
Un environnement document−vue n'est valide que si une fenêtre plan est attribué au programme TED.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| 1 | STR | fenetre fermée |
| 0 | STR | fenêtre non fermée |

## Exemples:

| | |
|---|---|
| @closeview(0) | fermeture de la vue courante avec interrogation |
| @setvar(vue,@getvue(−1),PLANVIEW),<br>@setvar(doc,@getdoc(""),MAPDOC),<br>@loadtxt(d:\bal\test.txt,1),<br>@setvar(rep,@msgdlg("","Appuyez sur une touche pour continuer"),NUM),<br>@closeview(0),<br>@setdocvue(@doc,@vue) | récupère vue plan courante<br>récupère doc MAP courant<br>charge un document texte et le rend actif et visible<br>pause<br>ferme la vue texte (et donc le document)<br>repasse surdocument précédent |

# @codeinsee

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le code INSEE du document.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [CodeInsee] | STR | code INSEE du document |

## Exemples:

@codeinsee

# @commune

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le nom de la commune du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [NomCommune] | STR | propriété du document |

## Exemples:

@commune

# @compareelt

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe des éléments à considérer (de 0 à ...) ou (−1) si les éléments de toute classe doivent être considérés |
| couchesrce | STANDARD | numéro de la couche source à comparer |
| couchedest | STANDARD | numéro de la couche destination à comparer |
| epsilon | STANDARD | écart maximum en position au dela duquel il n'y a plus égalité pour les points. |
| epsilone | STANDARD | écart maximum en position au dela duquel il n'y a plus égalité pour les écritures. |
| deport | STANDARD | indque si la comparaison des écritures (en position) doit se faire sur le déport ou l'écriture<br>0 = sur l'écriture<br>1 = sur le déport (extrémité de la flèche) |

## Action:

compare les éléments un à un des deux couches source et destination et sélectionne au niveau 0 les éléments qui sont dans une couche et pas dans l'autre

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Intégrité des objets ou éléments non respectée! | ERR | les couches doivent être intègres pour que la fonction puisse s'effectuer<br>(NB: une liaison n'est pas intègre si un de ses points est dans une autre couche que celle de la liaison) |
| OK | STR | les éléments sont sélectionnés au niveau 0 |

## Exemples:

| | |
|---|---|
| @compareelt(5,0,1,0.1,0.1,0) | sélectionne (au niveau 0) tous les éléments de classe "Parcelle" (5) de la couche (0) et (1) qui sont dans une couche et pas dans l'autre en considérant que les points sont égaux si leurs coordonnées sont identiques à 10 cm près et que les écritures sont identiques si les coordonnées de leur point d'insertion sont identiques à 10 cm près. |

# @compareobj

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe des objets à considérer de 1 à ..., 0 n'ayant pas de sens (n'effectue rien dans ce cas) |
| couchesrce | STANDARD | numéro de la couche source concernée (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche destination concernée (de 0 à Nb_Couches_document−1) |
| epsilon | STANDARD | écart maxi en position accepté |
| code | STANDARD | type de comparaison effectuée (ajouter les valeurs pour autant d'options désirées)<br><br>1 = OBJ_COMPARE_CARD = teste la cadinalité des objets<br><br>test de la cardinalité des types d'éléments suivants selon la nature de l'objet<br>PT : pt+ec<br>LI : ec<br>FC : fc+ec<br>EC : ec<br><br>2 = OBJ_COMPARE_POS= teste la position des objets<br><br>teste suivant la nature de l'objet:<br>PT : position<br>LI : emprise<br>FC : emprise<br>EC : position<br><br>4 = OBJ_COMPARE_DIM= teste la dimension des objets<br><br>teste suivant la nature des objets:<br>PT : néant (pas de test)<br>LI : longueur<br>FC : surface<br>EC : néant<br><br>8 = OBJ_COMPARE_TEXT= teste si le texte total (ensemble ordonné des écritures) de l'objet est identique à celui de l'autre objet<br><br>16= OBJ_COMPARE_EXTCARD=teste de cardinalité totale de l'objet (tous les types d'éléments pt+li+fc+ec) |
| maj | STANDARD | indique si les éléments qui disparaissent de la destination doivent être traités comme des éléments qui vont être supprimés et donc ne sélectionne parmi les éléments des objets de la couche destination qui vont être supprimés que les éléments qui n'entrainent pas une suppression d'un autre élément ou objet<br>*ex: sur la couche destination exite la parcelle 125 entourée de parcelles sauf sur un coté, sur la couche source elle n'existe pas, alors sur la couche destination :*<br>*si l'option maj=1, seule la face sera sélectionnée avec la liaison qui ne jouxte pas une autre parcelle.*<br>*si l'option maj=0, la face et les liaisons de la parcelle sont sélectionnées.*<br>*L'écriture du numéro de la parcelle est sélectionné dans les 2 cas.* |

## Action:

Effectue la différence entre les objets de deux couches. Les différences recherchées sont données par "code".
La comparaison se comporte comme une mise à jour de la destination par la source (si maj=1).
Les niveaux de sélections de 0 à 3 sont modifiés comme suit:
Niv 0 : couche source : éléments devant être ajoutés.
Niv 0 : couche destination : éléments devant être supprimés.
Niv 1 : pour les 2 couches : éléments devant être modifiés car cardinalité différente
Niv 2 : pour les 2 couches : éléments devant être modifiés car position ou dimension différente
Niv 3 : pour les 2 couches : éléments devant être modifiés car textes différents
L'application doit être en capacité de calculer l'identifiant de chacun des objets de type (classe) choisi : une méthode de calcul de l'identifiant pour le type d'objet doit avoir été défini.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Intégrité des objets ou éléments non respectée! | ERR | les éléments et objets doivent être intègres pour que la procédure aboutisse |
| OK | STR | comparaison a abouti |

## Exemples:

| | |
|---|---|
| @compareobj(5,0,1,0.1,0x15,1) | compare les objets parcelles des couches 0 et 1 en regardant si la cardinalité des objets de même identifiant est strictement identique et si les surfaces des objets sont identiques. Etablit les sélections en vue d'une mise à jour de la couche 1 par la couche 0 de manière à ce qu'aucun élément de la destination ne soit supprimé autre que ceux appartenant exclusivement aux objets devant être supprimés. |

# @compidu

*Nb paramètres*
8

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvardest | STANDARD | nom de la variable devant accueillir le résultat (cette variable doit exister) |
| codeinsee | STANDARD | nom de la variable donnant le code INSEE |
| prefsect | STANDARD | nom de la variable donnant le préfixe de section |
| sect | STANDARD | nom de la variable donnant la section |
| subdsect | STANDARD | nom de la variable donnant la subdivision de section |
| parcelle | STANDARD | nom de la variable donnant le numéro de parcelle |
| extra | STANDARD | nom de la variable donnant la donnée extra (soit l'étiquette, soit le texte, soit le numéro d'objet) |
| idalgo | STANDARD | algorithme de composition à utiliser pour fournir l'IDU<br>doit être une combinaison des valeurs suivantes:<br><br>1 = IDALGO_COMMUNE<br>2 = IDALGO_PREFSECT<br>4 = IDALGO_SECTION<br>8 = IDALGO_SUBDSECT<br>0x10 = IDALGO_PARCELLE<br>0x20 = IDALGO_TEXT<br>0x40 = IDALGO_LABEL<br>0x80 = IDALGO_ID<br>0x100 = IDALGO_XDATA<br><br>ces trois dernieres valeurs s'excluent mutuellement |

## Action:

compose un identifiant suivant les différentes valeurs et d'après l'algorithme donnés.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Composition de l'identifiant impossible! | ERR | une valeur n'est pas exploitable pour composer l'identifiant |
| IDU | STR | identifiant constitué (également enregistré dans [nomvardest]) |

## Exemples:

| | |
|---|---|
| @compidu(idu,insee,0,0,0,0,lbl,0x41) | fourni un identifiant composé du code INSEE donné par la variable "insee" et d'une étiquette donnée par la variable "lbl" et le stocke dans idu |

# @concat

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| texte1 | STANDARD | texte 1 |
| texte2 | STANDARD | texte 2 à concaténer avec texte 1 |
| avec_esp | STANDARD | indique si un espace est inséré entre les deux textes concaténés |

## Action:

<texte1> + " " + <texte2>
ou
<texte1> + <texte2>

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <resultat> | STR | valeur après concaténation |

## Exemples:

| | |
|---|---|
| @concat(@mavar,du boeuf,1) | si la variable "mavar" contient "rue", alors retourne "rue du boeuf" |
| @concat(@mavar,euil,0) | si la variable "mavar" contient "rue d'argent", alors retourne "rue d'argenteuil" |
| @concat(@mavar,@mavar2,1) | si la variable "mavar" contient "rue" et mavar2 contient "courbe" , alors retourne "rue courbe" |

# @concateneli

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des liaisons à concaténer (de 0 à 31) |

## Action:

fusionne toutes les liaisons sélectionnées au niveau "nivsel" de la couche "couche"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Fusion des liaisons multiples sélectionnées : NN fusions opérées !!! | STR | NN = Nombre de fusions opérées. |

## Exemples:

| | |
|---|---|
| @concateneli(0,0) | fusionne les liaisons sélectionnées au niveau 0 de la couche 0 |

# @concatenept

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des points à fusionner (de 0 à 31) |
| tolérance | STANDARD | écart maximum au dela duquel deux points sont considérés différents |

## Action:

fusionne tous les points sélectionnés au niveau "nivsel" de la couche "couche" présentant un écart en position entre eux inférieur à "tolérance".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Fusion des points doublons : NN points ont été fusionnés | STR | NN = Nombre de points fusionés |

## Exemples:

| | |
|---|---|
| @concatenept(0,0,0.1) | fusionne tous les points sélectionnés de la couche 0 qui ont un écart entre eux inférieur à 10 cm |

# @concatptli

*Nb paramètres*
9

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des points à fusionner sur les liaisons sélectionnées au même niveau (de 0 à 31) |
| tolerance | STANDARD | ecart maxi entre le point et la liaison au dela duquel le point ne peut être fusionné sur la liaison. |

## Action:

fusionne tous les points sélectionnés au niveau "nivsel" de la couche "couche" sur les liaisons de la couche sélectionnées au niveau "nivsel" si l'écart en position entre le point et la liaison est inférieur à "tolerance".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Fusion de points sur liaisons : NN fusions ont été réalisées | STR | NN = Nombre de fusions réalisées |

## Exemples:

| | |
|---|---|
| @concatptli(0,0,0.2) | fusionne les points sélectionnés (au niveau 0) de la couche 0 sur les liaisons sélectionnées (au niveau 0) proches à moins de 20cm |

# @concatvar

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable destination |
| texte | STANDARD | texte à concaténer avec la variable |
| avec_esp | STANDARD | indique si un espace est inséré entre les deux textes concaténé |

## Action:

[nomvar] <== [nomvar] + " "+texte
ou
[nomvar] <== [nomvar] + texte

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Variable inconnue | ERR | si le nom ne correspond pas à une variable |
| Variable incompatible | ERR | si la variable fournie n'est pas de type standard |
| [nomvar] | STR | valeur de la variable après concaténation |

## Exemples:

| | |
|---|---|
| @concatvar(mavar,du boeuf,1) | si la variable "mavar" contient "rue", alors retourne "rue du boeuf" qui est alors la valeur de "mavar" |
| @concatvar(mavar,euil,0) | si la variable "mavar" contient "rue d'argent", alors retourne "rue d'argenteuil" qui est alors la valeur de "mavar" |
| @concatvar(mavar,@mavar2,1) | si la variable "mavar" contient "rue" et mavar2 contient "courbe" , alors retourne "rue courbe" qui est alors la valeur de "mavar" |

# @conformemodele

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche concernée (0 à NbCouches−1) ou −1 si toutes |
| nivsel | STANDARD | niveau de sélection des éléments à scruter (−1 si tous) |
| telt | STANDARD | type d'élément à scruter : combinaison binaire de :<br>1 = Points<br>2 = Liaisons<br>4 = écritures<br>16 = faces |
| opt | STANDARD | indique l'action à entreprendre<br><br>bit 0 et 1:<br>0 = modifie les éléments dont la classe ne correspond pas<br>1 = modifie les élements de classe calculée 0<br>2 = modifie les élements de classe calculée différente de 0<br>3 = modifie tous les éléments<br><br>bit 4:<br>1 = l'appartenance à l'objet (de plus haute priorité pour lélément détermine la classe)<br>0 = pas de recollement avec la classe de l'objet |

## Action:

scrute tous les éléments concernés, et fournit leur classe théorique d'après la hierarchie du SCD et en regardant la classe des éléments en relation directe avec celui−ci (les faces pour les liaisons et les liaisons pour les points). Puis si la classe théorique est de priorité supérieure, modifie les éléments avec le masque de modification de cette nouvelle classe.
La modification est faite pour les faces, puis les liaisons, puis les points, puis les écritures afin de respecter une cohérence dans la hiérarchie si plusieurs types d'éléments sont demandés.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N éléments modifiés | NUM | nb d'éléments non conformes au SCD modifiés |

## Exemples:

| | |
|---|---|
| @conformemodele(−1,−1,23,3) | modifie tous les éléments du document non conforme à la hiérarchie du SCD |

# @contains

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm1 | STANDARD | chaine de caractères |
| parm2 | STANDARD | chaine de caractères |

## Action:

teste si parm2 est incluse dans parm1 (la casse des caractères n'est pas considérée)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| 0 ou 1 | BOOL | renvoie true si parm2 est dans parm1 |

## Exemples:

| | |
|---|---|
| @contains(@var,rue) | si var est le nom d'une variable contenant "rue des boeufs", la fonction renvoie alors 1 (vrai) |

# @controleid

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe des objets dont l'identifiant est à controler |
| couche | STANDARD | couche des objets à controler (ou −1 si ts les objets sont à controler) |
| nivselsrce | STANDARD | niveau de sélection des objets à controler (−1 si tous les objets) |
| nivseldest | STANDARD | niveau de sélection à positionner pour les objets dont l'identifiant est invalide ou non unique |

## Action:

controle les identifiants des objets de classe "classe" en vérifiant leur validité et leur unicité
nivselsrce doit être différent de nivseldest (sinon une erreur est générée)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N objets avec identifiants invalides et M objets avec identifiants non uniques ont été trouvés et sélectionnés ! | NUM | controleid réalisé avec succès |

## Exemples:

| | |
|---|---|
| @controleid(5,−1,−1,0) | controle les identifiants des parcelles et sélectionne les invalides au niveau 0 |

# @copl

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie une chaine indiquant le mode de confection du plan

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <mode de confection du plan> | STR | |

## Exemples:

@copl

# @copy

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| select | STANDARD | niveau de sélection pour une copie de couche(s) ou '−1' pour une copie de texte |
| couche/texte | STANDARD | copie de couche : valeur numérique de la couche dont les éléments sélectionnés sont à copier dans le presse papier (−1 si toutes)<br>copie de texte : dans ce cas c'est le texte de ce paramètre qui est transmis au presse papier |
| option | STANDARD | option sous forme de champ de 2 bits (format hexa, decimal ou octal):<br>0x100 = 256 = EltSigne = force les signes des liaisons à se copier même s'ils ne sont pas sélectionnés du moment que la liaison est sélectionnée<br>0x200 = 512 = EltDeport = force les déports des écritures à se copier même s'ils ne sont pas sélectionnés du moment que l'écriture est sélectionnée<br>0x1000 = 4096 = Sans dépendance : la bordure des élements sélectionnés est copiée sans les caractéristiques des éléments de cette bordure ni objets éventuels composant la bordure (par ex: un point de canevas formant le pourtour d'une parcelle).<br>Les autres bits de la valeurs sont ignorés |

## Action:

Copie de la sélection courante de la couche donnée (ou de toutes les couches si "couche"=−1) dans le presse papier
NB: on ne peut copier un symbole qu'avec son élément porteur.
ou
Copie du texte fourni dans le presse papier

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Erreur dans la copie vers le presse papier | ERR | erreur lors de la copie dans le presse papier |
| Sélection de la couche transférée dans le presse papier | STR | |
| Texte transféré dans le presse papier | STR | |

## Exemples:

| | |
|---|---|
| @copy(@selecttravail,@couchetravail,0) | copie la sélection de la couche courante |
| @copy(@selecttravail,@couchetravail,0x300) | copie la sélection de la couche courante avec tous les symboles associés aux éléments sélectionnés même si ces derniers ne sont pas sélectionnés. |
| @copy(@selecttravail,@couchetravail,0x1000) | Un batiment en bordure d'une parcelle est sélectionné puis cette commande @copy est lancée : la bordure du batiment copié sera de classe batiment au lieu de parcelle. |
| @copy(−1,"texte copié",0) | copie "texte copié" dans le presse papier |

# @cos

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | ANG | fournit le cosinus du paramètre exprimé sous forme normalisé des angles (ex: 125.3342gv) |

## Action:

renvoie le cosinus de l'angle

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <cos(parm)> | NUM | renvoie le cosinus |

## Exemples:

| | |
|---|---|
| @cos(@var) | renvoie le cosinus de la variable |

# @couchetravail

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie l'indice de la couche de travail en cours (de 0 à...)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [CoucheTravail] | NUM | indice de la couche de travail |

## Exemples:

@couchetravail

# @createautoobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si toutes les couches sont concernées |
| classe | STANDARD | classe des objets à créer ou −1 si toutes les classes sont concernées |

## Action:

crée les objets automatiquement de la classe "classe" pour la couche "couche".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| NN objets (de classe C) ont été créés sur KK occurences possibles. | NUM | NN =nombre d'objets créés, KK = nombre d'objets possibles |
| Classe NN ne peut construire ses objets automatiquement! | NUM | si le type d'objet ne peut construire ses objets automatiquement |

## Exemples:

| | |
|---|---|
| @createautoobj(0,5) | crée automatiquement tous les objets parcelles de la couche 0 |

# @createdeport

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des liaisons constituant les flèches de rattachement (déport d'écritures) à transformer |
| classeec | STANDARD | classe des écritures devant être rattachées |
| epsilon | STANDARD | écart maximum pour la reconnaissance de l'écriture qui est à proximité (en mm papier) |
| asuppr | STANDARD | indique si la polyligne trouvée représentant le rattachement d'une écriture doit être supprimée une fois le déport d'écriture créé. |

## Action:

crée de nouveaux déport d'écriture en scrutant toutes les liaisons sélectionnées et les écritures de classe "classe" qui sont à proximité afin d'en déduire le déport. Les liaisons reconnues peuvent constituer une polyligne à une ou plusieurs liaisons. Dans le cas d'une seule liaison, TopoCad recherche l'écriture à partir des 2 extrémités de la liaison, dans le cas de plusieurs liaisons, TopoCad recherche l'écriture à partir du milieu de la première et dernière liaison de la polyligne.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N échecs dans la conversion des polylignes en déports d'écriture | STR | N = Nombre d'échecs dans la détermination des déports, dans ce cas la polyligne est laissée sélectionnée |

## Exemples:

| | |
|---|---|
| @createdeport(0,0,5,10,1) | crée les déports d'écritures en recherchant les liaisons sélectionnée et les écritures de classe parcelles à proximité (à moins de 10 mm papier) et supprime les polylignes en cas de succès. |

# @createec

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche de l'écriture à créer (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe de l'écriture à créer (de 1 à NbTObj−1) |
| X | STANDARD | coordonnee X de l'écriture |
| Y | STANDARD | coordonnee Y de l'écriture |
| texte | STANDARD | texte de l'écriture |

## Action:

crée une écriture de classe "classe" et de coordonnées données

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Ecriture> | EC | l'écriture créée ou NULL |

## Exemples:

| | |
|---|---|
| @createec(0,0,856543.76,121000.60,"puits") | crée une écriture |

# @createfcwithaddfc

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classefc | STANDARD | classe des faces à créer (de 1 à NbTObj−1) |
| nivsel | STANDARD | niveau de sélection des faces à concaténer (de 0 à 31) |

## Action:

Opère une union de faces ***dans un milieu topologique***. Cette fonction unit toutes les faces sélectionnées et en tire une ou des faces créées de classe "classefc". La fonction opérant par contour, les trous sont traités comme des faces, il convient donc de les traiter à postériori par la fonction @makefacesatrou

Si la topologie des faces sélectionnées n'existe pas et donc que deux faces sélectionnées se recouvrent (ont même liaison) la fonction échoue, envoie un message sur ERRORxxx.LOG et renvoie NULL.

Si la fonction abouttit, un message indiquant le nombre de faces créées est envoyé à ERRORxxx.LOG et la première face créée est renvoyée, ce qui permet, toutes les autres faces étant créées en aval, de parcourir ces faces créées par la suite grace à la fonction @next

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Face> | FC | la première face créée ou NULL |

## Exemples:

| | |
|---|---|
| @createfcwithaddfc(0,5,0) | crée une face (ou plusieurs) de classe parcelle avec les faces sélectionnées au niveau 0 de la couche 0 |

# @createfcwithfc

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classefc | STANDARD | classe des faces à créer (de 1 à NbTObj−1) |
| nivsel1 | STANDARD | niveau de sélection des faces 1° terme (de 0 à 31) |
| nivsel2 | STANDARD | niveau de sélection des faces 2° terme (de 0 à 31) |
| op | STANDARD | opération à réaliser sur les deux groupes de faces<br><br>0 = XOR<br>1 = AND (intersection)<br>2 = OR (union)<br>3 = faces 1 − faces 2<br>4 = faces 2 − faces 1 |

## Action:

Effectue une opération arithmétique de faces ***dans un milieu topologique***. Cette fonction opère sur deux ensembles de faces sélectionnées au niveau nivsel2 et sélectionnées au niveau nivsel1 et en tire une ou des faces créées de classe "classefc".La fonction opérant par contour, les trous sont traités comme des faces, il convient donc de les traiter à postériori par la fonction @makefacesatrou

Le premier et le second groupe sont vus comme une seul surface de surfaces jointes ou disjointes éventuellement.

Si la topologie ou l'intégrité des faces sélectionnées n'existe pas la fonction échoue, envoie un message sur ERRORxxx.LOG et renvoie NULL.

Si la fonction abouttit, un message indiquant le nombre de faces créées est envoyé à ERRORxxx.LOG et la première face créée est renvoyée, ce qui permet, toutes les autres faces étant créées en aval, de parcourir ces faces créées par la suite grace à la fonction @next

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Face> | FC | la première face créée ou NULL |

## Exemples:

| | |
|---|---|
| @createfcwithfc(0,5,0,1,3) | crée une face (ou plusieurs) de classe parcelle en soustrayant les faces sélectionnées au niveau 0 d'avec les faces sélectionnées au niveau 1 de la couche 0 |

# @createfcwithli

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classefc | STANDARD | classe des faces à créer (de 1 à NbTObj−1) |
| nivsel | STANDARD | niveau de sélection des liaisons représentant l'enveloppe de la face (de 0 à 31) |

## Action:

Scrute toutes les liaisons de la couche "couche" qui sont sélectionnées au niveau "nivsel" et écarte dans un premier temps toutes les polylignes orphelines (non fermées). Puis essaye de constituer une face externe avec les polylignes restantes. Seule la première face englobante de plus faibles coordonnées est constituée si plusieurs blocs sont possibles et ses liaisons sont déselectionnées ce qui permet éventuellement de réitérer l'opération si les liaisons constituent plusieurs faces externes.

La fonction renvoie la face créée (qui est ajouttée au document) ou NULL si aucune face n'a pu être créée.

Cette fonction échoue (NULL) si les liaisons sélectionnées se recoupent entre elles

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Face> | FC | la face créée ou NULL |

## Exemples:

| | |
|---|---|
| @createfcwithli(0,5,0) | crée une face parcelle englobant les liaisons sélectionnées au niveau 0 |

# @createliwithpt

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classeli | STANDARD | classe de la liaison à créer (de 1 à NbTObj−1) |
| nivsel ou | STANDARD | niveau de sélection des 2 points représentant la liaison (de 0 à 31) |
| ptsrce | PT | point source de la liaison à créer |
| ptdest | PT | point destination de la liaison à créer |

## Action:

cette fonction crée une liaison :
−soit à partir de deux points fournis qui doivent être différents et valides (sinon une erreur est déclenchée)
−soit à partir de deux points sélectionnés au niveau "nivsel" dans la couche fournie : dans ce cas le nombre de points sélectionnés de la couche doit être de 2 sinon un message d'erreur est transmis à ERRORxxx.LOG et la fonction renvoie NULL

la liaison créée est de classe "classeli".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Liaison> | LI | la liaison créée ou NULL |

## Exemples:

| | |
|---|---|
| @createliwithpt(0,5,0,0) | crée une liaison parcelle avec les points sélectionnés au niveau 0 de la couche 0 |

# @createobjwext

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe de l'objet à créer |
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des éléments devant faire partie de l'objet (de 0 à 31) |
| extradataname | STANDARD | nom de la donnée extra unifiant les éléments en objet |

## Action:

crée les objets de classe "classe" composés des éléments sélectionnés et possédant une même donnée extra de nom "extradataname". Les éléments sélectionnés au niveau "nivsel" de la couche "couche" sont scrutés. Les éléments avec une donnée extra vide forment également un objet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N objets ont été créés | NUM | nombre d'objets créés |

## Exemples:

| | |
|---|---|
| @createobjwext(0,0,5,idu) | crée les objets parcelles avec les éléments sélectionnés au niveau 0 de la couche 0 possédant une une donnée extra "idu" (identifiant) identique. |

# @createobjwlbl

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe de l'objet à créer |
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des éléments devant faire partie de l'objet (de 0 à 31) |

## Action:

crée les objets de classe "classe" composés des éléments sélectionnés et possédant une même étiquette. Les éléments sélectionnés au niveau "nivsel" de la couche "couche" sont scrutés. Les éléments sans étiquette forment également un objet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N objets ont été créés | NUM | nombre d'objets créés |

## Exemples:

| | |
|---|---|
| @createobjwlbl(0,0,5) | crée les objets parcelles avec les éléments sélectionnés au niveau 0 de la couche 0 possédant une étiquette identique. |

# @createobjwselelt

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des éléments devant faire partie de l'objet (de 0 à 31) |
| classe | STANDARD | classe de l'objet à créer |

## Action:

crée un objet de classe "classe" avec les éléments sélectionnés au niveau "nivsel" de la couche "couche"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Objet créé> | OBJ | objet créé ou O si impossible (pas d'éléments) |

## Exemples:

| | |
|---|---|
| @createobjwselelt(0,0,12) | crée un objet voie publique avec les éléments sélectionné au niveau 0 de la couche 0 |

# @creatept

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche du point à créer (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe du point à créer (de 1 à NbTObj−1) |
| X | STANDARD | coordonnee X du point |
| Y | STANDARD | coordonnee Y du point |
| Z | STANDARD | altitude du point (peut être "" pour indéfini) |

## Action:

crée un point de classe "classe" et de coordonnées données

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Point> | PT | le point créé ou NULL |

## Exemples:

| | |
|---|---|
| @creatept(0,0,856543.76,121000.60,"") | crée un point simple sans altitude |

# @createsmwithli

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche dans laquelle rechercher les polylignes sélectionnées symbolisant un signe de mitoyenneté (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des polylignes à transformer représentant les signes |
| coucheli | STANDARD | numéro de la couche dans laquelle rechercher les liaisons sur lesquelles attacher les signes de mitoyenneté (de 0 à Nb_Couches_document−1) |
| classeli | STANDARD | classe des liaisons devant supporter les signes de mitoyenneté |
| epsilonmax | STANDARD | distance maximale entre le point moyen de la polyligne représentant le signe et la liaison supportant le signe au dela de laquelle le signe ne peut appartenir à la liaison (en mm papier) |
| epsilonmitoy | STANDARD | distance entre le point moyen de la polyligne représentant le signe et la liaison supportant le signe en dessous de laquelle le signe est considéré comme mitoyen.(en mm papier) |
| asuppr | STANDARD | indique si la polyligne trouvée représentant le signe de mitoyenneté doit être supprimée une fois le signe créé. |

## Action:

crée de nouveaux signes de mitoyenneté en scrutant toutes les liaisons sélectionnées (formant une polyligne représentant le signe) et en recherchant la proximité du centre de cette polyligne avec les liaisons de classe "classe" de la couche "coucheli" .

Le centre est calculé en faisant la moyenne des coordonnées extrèmes de la polyligne.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N échecs dans la conversion des polylignes en signes de mitoyenneté | NUM | N = Nombre d'échecs dans la détermination des déports, dans ce cas la polyligne est laissée sélectionnée |

## Exemples:

| | |
|---|---|
| @createsmwithli(0,0,0,5,10,0,1) | crée les signe de mitoyenneté sur la couche 0 à partir des liaisons sélectionnées sur les liaisons "Parcelle". Le signe est un signe non mitoyen quelle que soit la position du point (la distance ne peut être inférieure à 0). Le rayon maxi de détection est de 10 mm papier (par rapport à l'échelle d'origine du document) |

# @createsmwithpt

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche dans laquelle rechercher les points sélectionnées symbolisant un signe de mitoyenneté (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des points à transformer représentant les signes |
| coucheli | STANDARD | numéro de la couche dans laquelle rechercher les liaisons sur lesquelles attacher les signes de mitoyenneté (de 0 à Nb_Couches_document−1) |
| classeli | STANDARD | classe des liaisons devant supporter les signes de mitoyenneté |
| epsilonmax | STANDARD | distance maximale entre le point représentant le signe et la liaison supportant le signe au dela de laquelle le signe ne peut appartenir à la liaison (en mm papier) |
| epsilonmitoy | STANDARD | distance entre le point représentant le signe et la liaison supportant le signe en dessous de laquelle le signe est considéré comme mitoyen.(en mm papier) |
| asuppr | STANDARD | indique si le point trouvé représentant le signe de mitoyenneté doit être supprimé une fois le signe créé. |

## Action:

crée de nouveaux signes de mitoyenneté en scrutant tous les points sélectionnés (représentant le signe) et en recherchant la proximité de ce point avec les liaisons de classe "classe" de la couche "coucheli" .

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N échecs dans la conversion des points en signes de mitoyenneté | NUM | N = Nombre d'échecs dans la détermination des déports, dans ce cas le point est laissé sélectionné |

## Exemples:

| | |
|---|---|
| @createsmwithpt(0,0,0,5,10,0,1) | crée les signe de mitoyenneté sur la couche 0 à partir des points sélectionnés sur les liaisons "Parcelle". Le signe est un signe non mitoyen quelle que soit la position du point (la distance ne peut être inférieure à 0). Le rayon maxi de détection est de 10 mm papier (par rapport à l'échelle d'origine du document) |

# @createtext

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| objet | OBJ | objet au centroide duquel mettre le texte |
| texte | STANDARD | texte à créer |
| maskec | STANDARD ou MASKCREC | numéro de masque de création d'écriture (0 à NbMaskCrEc−1) ou masque de création d'une écriture |

## Action:

crée un texte "texte" au centroide de l'objet "objet" grace au masque de création d'écriture fourni

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Numéro de masque invalide | ERR | le numéro donné est invalide |
| <ecriture> | EC | écriture créée si OK |

## Exemples:

| | |
|---|---|
| @createtext(@monobj,@var,5) | crée un texte indiqué par la variable "var" au centroide de l'objet indiqué par la variable "monobj" en utilisant le masque de création des écritures n° 5 (parcelles) |

# @createtextobj

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| objet | OBJ | objet auquel fournir l'écriture |
| texte | STANDARD | texte à créer et à attacher à l'objet |
| flags | STANDARD | indique si les précédentes écritures de l'objet doivent être suprimées:<br>0= rien n'est supprimé<br>1 = les précedentes écritures sont détachées de l'objet mais non supprimées<br>2 = les précedentes écritures sont supprimées et balayées de l'objet |

## Action:

crée un texte au centroide de l'objet "objet" et l'attribue à l'objet "objet". Le masque de création d'écriture de la classe de "objet" est utilisé pour créé l'écriture. Le texte de l'écriture créée est fourni par "texte". "flags" indique si les précédentes écritures de l'objet doivent être supprimées ou non.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | objet rectifié |

## Exemples:

| | |
|---|---|
| @createtextobj(@varobj,@vartexte,2) | affecte à l'objet contenu dans la variable "varobj" le texte contenu dans la variable "vartexte" en supprimant toutes anciennes écritures de l'objet |

# @crois2faces

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classeli | STANDARD | classe des liaisons à scruter |
| classefc | STANDARD | classe des faces à créer |
| nivsel | STANDARD | niveau de sélection pour les couple de croisillons n'ayant pu créer une face |

## Action:

Constitue les faces de la couche "couche" à l'intersection des croisillons de classe "classeli" (pour chaque couple de liaisons). Si une face de classe "classefc" existe à l'intersection, alors ignore le couple. Si un couple ne peut créer de faces alors celui ci est sélectionné au niveau "nivsel".
La face est créée en tenant compte des règles courantes de création de faces, aussi il est peut être préférable de réaliser cette fonction en vision monochrome (avec classe de travail=bati léger) afin que topocad donne une priorité dans la reconnaissance de la face aux contours de batis légers (si il s'agit par ex de croisillons de batis légers)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @crois2faces(0,9,8,0) | crée toutes les faces possibles, si elles n'existent pas déjà, de classe "bati légér" (8) à partir des croisillons (9) et sélectionne au niveau 0 les croisillons dont TopoCad n'a pas trouvé ni pu créer de faces. |

# @crois2obj

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classeli | STANDARD | classe des liaisons à constituer en objets |
| nivsel | STANDARD | niveau de sélection à positionner pour les cas ambigus non résolus (de 0 à 31) |

## Action:

Constitue les objets croisillons de classe "classe" pour chaque couple de liaisons de la couche "couche" et de classe "classe". Ignore les liaisons faisant déjà partie d'un objet de classe "classe" et sélectionne au niveau "nivsel" les liaisons n'ayant pu être constituées en objet (pas d'intersection avec une autre liaison ou intersection multiple).
Cette fonction constitue des objets "bilinéaires". Cependant à la différence de la constitution automatique des objets bilinéaires (c'est à dire dont la cardinalité des liaisons est 2), elle ne crée pas d'objet si une liaison est orpheline.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @crois2obj(0,9,0) | constiitue les objets croisillons (9) de la couche 0 et sélectionne au niveau 0 les impossibilités. |

# @currenttrelsem

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le type de relation sémantique courant de la fenetre, sous forme d'indice de 0 à NbTRelSem−1 (0 étant le type des correspondances pt à pt)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [currenttrelsem] | NUM | indice du type de relation sémantique courante |

## Exemples:

@currenttrelsem

# @date

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la date courante au format date en vigueur (par défaut "JJ/MM/AAAA" accepté par la base de donnée de TopoCad)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <date> | STR | date courante au format date en vigueur |
| "????" | ERR | si la date courante est invalide (par ex: mois>12) |

## Exemples:

| | |
|---|---|
| @date | renvoie par ex: "08/02/1960" |

# @dateedi

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la date d'édition du document au format date en vigueur (par défaut "JJ/MM/AAAA")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <date> | | date d'édition au format date en vigueur |
| ou | STR | ou |
| "????" | | si la date d'édition est invalide (par ex: mois>12) |

## Exemples:

| | |
|---|---|
| @dateedi | renvoie par ex: "08/02/1960" |

# @dateincorp

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la date d'incorporation au SIG du document au format date en vigueur (par défaut "JJ/MM/AAAA")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <date> | | date d'incorporation au format date en vigueur |
| ou | STR | ou |
| "????" | | si la date d'incorporation est invalide (par ex: mois>12) |

## Exemples:

| | |
|---|---|
| @dateincorp | renvoie par ex: "08/02/1960" |

# @dateredi

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la date de réédition du document au format date en vigueur (par défaut "JJ/MM/AAAA")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <date> | | date de réédition au format date en vigueur |
| ou | STR | ou |
| "????" | | si la date de réédition est invalide (par ex: mois>12) |

## Exemples:

| | |
|---|---|
| @dateredi | renvoie par ex: "08/02/1960" |

# @dbbrowse

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

Edition d'une base de donnée par une boite de dialogue.
Chaque fiche correspond à l'ensemble des propriétés de l'objet identifiable (objet graphique ou non graphique)
Pour la base standard : c'est donc l'ensemble des enregistrements ayant un même identifiant
Pour les bases de classe ou auxiliaires : c'est un enregistrement dont les champs représentent les différentes propriétés.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <IDU de l'objet édité> | STR | L'identifiant de l'objet qui était en cours d'édition est renvoyé (débarrassé des caractères espaces avant et après la chaine) si on appuie sur OK dans la boite de dialogue |
| Abandon | STR | renvoyé si on sort de la boite de dialogue par "Abandon" ou "Esc/Echap" |

## Exemples:

| | |
|---|---|
| @dbbrowse(C5) | édite la base de donnée des parcelles courante |

# @dbclass2dbstand

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |

## Action:

copie toutes les propriétés concernant la classe "classe" d'une base de donnée de classe vers la base de donnée standard.
Les 2 bases (de classe "classe" et standard) doivent avoir été "ouvertes" au niveau de l'application.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| ERREUR : base non prête | ERR | si une des 2 bases n'a pas été ouverte |
| ERREUR : aucune propriété n'a été copiée | ERR | si la base de classe est vide ou impossible de copier les données |
| ERREUR : NN propriétés seulement ont été transférées | ERR | si TopoCad a détecté des erreurs et n'a pu copier toutes les propriétés |
| N propriétés ont été transférées | STR | N = Nombre de propriétés copiées |

## Exemples:

| | |
|---|---|
| @dbclass2dbstand(5) | copie toutes les propriétés de la base de classe de parcelles ouverte vers la base standard |

# @dbclose

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis).<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

ferme la base de donnée d'indice "indbase" dans la table des bases de données de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | base fermée avec succès |

## Exemples:

| | |
|---|---|
| @dbclose(0) | ferme la base de donnée standard |
| @dbclose(5) | ferme la base de donnée de classe des parcelles |
| @dbclose(25) | ferme la première base de donnée auxiliaire (si il y 24 classes+1 "non classe" dans le SCD) |

# @dbcreateindex

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| expr | STANDARD | expression servant de clé pour l'indexation (nom d'un champ par ex) |
| subst | STANDARD | substantif ajoutté au nom du fichier d'index (ex: "filename.dbf" étant le fichier dbase, "nom" étant le substantif, alors le fichier d'index se nomme "filename_nom.ndx" ) |
| unique | STANDARD | 0 pour une clé pouvant être répétée sur plusieurs enregistrements<br>1 pour une clé unique |
| overlay | STANDARD | 0 pour une création d'un index sans écrasement d'un précédent existant<br>1 pour une création d'un index avec écrasement d'un précedent index existant |

## Action:

Cette commande crée un index supplémentaire pour la base donnée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de créer l'index! | ERR | si fichier d'index existe déjà et overlay=0 ou pour toute autre erreur |
| OK | STR | index créé et rajouté à la table des indexs de la base |

## Exemples:

| | |
|---|---|
| @dbcreateindex(A2,LIFICDNOM,nom,0,1) | crée un index sur le champ LIFICDNOM qui a des valeurs pouvant se répéter d'un enregistrement à l'autre, en écrasant tout précédent index. |

# @dbdelete

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| idu | STANDARD | identifiant (complet avec prefixe) à supprimer<br>ou<br>"" (non fourni) pour la fiche courante de la base |
| opt | STANDARD | options:<br>0 = pas de confirmation de suppression<br>1 (ou tte autre valeur) = confirmation de suppression |

## Action:

supprime la fiche d'identifiant de la fiche courante ou la fiche d'identifiant donné.
pour une base de classe, il s'agit de la suppression de l'enregistrement
pour une base standard, il s'agit de la suppression de tous les enregistrements de même identifiant
NB: il n'y a pas de vérification que le préfixe donné dans l'identifiant corresponde au prefixe de ce type de base, ainsi il est possible de supprimer une fiche n'ayant rien a faire dans la base (parce que pas de bon préfixe)
NB: la base pointe sur la fiche précédent celle qui a été supprimée à l'issue de l'opération.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| base non prète | ERR | si base non ouverte ou autre problème |
| impossible d'acquérir le champ IDU de l'enregistrement | ERR | si erreur |
| Impossible de supprimer les propriétés de l'identifiant 'xxxxxxxx' : cet identifiant n'existe pas dans la base ! | STR | pas d'identifiant identique trouvé dans la base (peut être erreur dans le préfixe saisi) |
| Suppression Abandonnée !!! | STR | abandon après confirmation |
| Impossible de supprimer la fiche d'identifiant : xxxxxxx | ERR | erreur à la suppression |
| 1 fiche supprimée : xxxxxx | STR | opération réussie : 'xxxxx' représente l'identifiant supprimé |

## Exemples:

| | |
|---|---|
| @dbdelete(5,"PARC536000AB0155",0) | recherche le numéro de parcelle (champ de la base de donné parcelle ayant comme identifiant PARC536000AB0155) et supprime la fiche. |

# @dbgetfield

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| fieldname | STANDARD | nom du champ à obtenir |

## Action:

fournit le champ de nom "fieldname" de l'enregistrement courant de la base de donnée d'indice "indbase" de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des type ou valeur non appropriés |
| ERREUR : base non prète | ERR | si base non ouverte ou autre problème |
| ERREUR : impossible d'acquérir le champs demandé | ERR | si nom de champ erroné ou autre problème |
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <valeur champ> | STR | renvoie la valeur du champ |

## Exemples:

| | |
|---|---|
| @dbgetfield(5,SUPF) | renvoie le champ SUPF de l'enregistrement courant de la base de parcelles |

# @dbnext

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

positionne le curseur sur le prochain enregistrement.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prête | ERR | si base non ouverte ou autre problème |
| ERREUR : impossible de positionner la table | ERR | si erreur |
| OK | STR | enregistrement suivant devient l'enregistrement courant |
| EOF | STR | la fin de fichier a été atteinte |

## Exemples:

```
@dbnext(C5)
```

enregistrement suivant de la base de parcelles de l'application

```
@setlvar(isok,@dbsettobegin(@basepermis),STR),
@while(@equ(@isok,"OK"),
   @list(

@setlvar(vnature,@dbgetfield(@basepermis,NATURE),STR),
      @if(@contains(@vnature,"piscine"),
         @out(@dbgetfield(@basepermis,IDU)),
         ""
         ),
      @setlvar(isok,@dbnext(@basepermis),STR)
      )
   )
```

exemple de procedure listant tous les permis de la base concernant une piscine

# @dbprevious

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

deplace le curseur vers le précédent enregistrement

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prête | ERR | si base non ouverte ou autre problème |
| ERREUR : impossible de positionner la table | ERR | si erreur |
| OK | STR | enregistrement précédent devient l'enregistrement courant |
| EOF | STR | la fin de fichier a été atteinte |

## Exemples:

```
@dbprevious(C5)
```

enregistrement précédent de la base de parcelles de l'application

```
@setlvar(isok,@dbsettoend(@basepermis),STR),
@while(@equ(@isok,"OK"),
   @list(

@setlvar(vnature,@dbgetfield(@basepermis,NATURE),STR),
      @if(@contains(@vnature,"piscine"),
         @out(@dbgetfield(@basepermis,IDU)),
         ""
         ),
      @setlvar(isok,@dbprevious(@basepermis),STR)
      )
   )
```

exemple de procedure listant tous les permis de la base concernant une piscine

# @dbopen

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| filename | STANDARD | nom de la base d'extension DBF |

## Action:

ouvre la base de donnée dont le chemin est donné par "filename" comme une base de donnée d'indice "indbase" de l'application. Le nom de fichier doit exister pour une base non standard (de classe ou auxiliaire). En cas de base standard, si le nom de fichier n'existe pas, la base est créée.
TopoCad tente de positionner le curseur au début de la base de donnée à l'issue de l'ouverture, mais ne renvoie pas d'erreur en cas d'échec, la fonction @dbsettobegin permet d'envoyer une erreur en cas de problème (base vide, probleme de lecture ...)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prète | ERR | si base non ouverte ou autre problème |
| ERREUR : création de base non standard impossible | ERR | si erreur |
| ERREUR : Impossible d'ouvrir la table... | ERR | base verouillée par une autre application ou en lecture seule ... |
| Connexion avec la table...réalisée avec succès. | STR | |

## Exemples:

| | |
|---|---|
| @dbopen(5,c:\PERMIS.DBF) | ouverture de la base spécifiée comme base de parcelle |

# @dbpack

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis).<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

compacte la base de donnée et reconstruit les index

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| impossible de compacter la base! | STR | erreur dans le compactage |
| OK | STR | base compactée avec succès |

## Exemples:

| | |
|---|---|
| @dbpack(5) | compacte la base de donnée de classe des parcelles |

# @dbseek

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| fieldname | STANDARD | nom du champ |
| value | STANDARD | valeur à rechercher pour le champ |
| opt | STANDARD | options:<br>0 = égalité stricte (nombres ou chaines de car),<br>1 = champ>= value ('commence par' pour les chaines de car)<br>2 = champ>value ('contient' pour les chaines de caractères) |

## Action:

recherche une valeur dans le champ d'une base de donnée et se positionne sur l'enregistrement dont le champ satisfait aux conditions de comparaison fournies par "opt".

NB: une recherche dans un index est sensible à la casse (rechercher "pc003M1054" ne donnera pas la même chose que rechercher "PC003M1054"), la recherche sans index (en parcourant toute la base) n'est pas sensible à la casse (dans ce cas "rues" > "rue", "RuE"=="rUe")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prète | ERR | si base non ouverte ou autre problème |
| ERREUR : impossible de rechercher la donnée | ERR | si erreur |
| 1 Trouvé ! | STR | base positionnée sur l'enregistrement trouvé |
| Non trouvé ! | STR | |

## Exemples:

| | |
|---|---|
| @dbseek(5,numero_parcelle,@var,0) | recherche le numéro de parcelle (champ de la base de donné parcelle ayant comme nom "numero_parcelle") contenu dans la variable de nom "var" et se positionne sur l'enregistrement. |

# @dbsetdefault

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| idu | STANDARD | valeur du champ IDU |

## Action:

Initialise un enregistrement avec les valeurs par défaut, idu représente l'identifiant c'est à dire la valeur du champ IDU à initialiser. Cette procédure échoue sur des bases n'ayant pas de champs de nom IDU (bases auxiliaire non référencée par le système). La procédure recherche l'enregistrement et s'il n'existe pas en crée un nouveau. Pour la base standard, crée autant de propriétés que répertoriées par le système.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'initialiser un enregistrement! | ERR | si base ne possédant pas de champs IDU ou non ouverte ou autre problème |
| OK | STR | enregistrement initialisé et devient l'enregistrement courant |

## Exemples:

| | |
|---|---|
| @dbsetdefault(27,PERM003PC04M1021) | création et initialisation d'un enregistrement dans la base de permis d'IDU "PERM003PC04M1021" |

# @dbsetfield

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |
| fieldname | STANDARD | nom du champ |
| value | STANDARD | valeur à fournir au champ |

## Action:

affecte une valeur au champ "fieldname" de la base de donnée d'indice "indbase" de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prête | ERR | si base non ouverte ou autre problème |
| ERREUR : enregistrement non disponible | ERR | si base positionnée sur un "crack" (EOF, BOF...) |
| OK | STR | champ modifié |

## Exemples:

| | |
|---|---|
| @dbsetfield(5,SUPF,@surf) | affecte la valeur de la variable de nom "surf" au champ "SUPF" de l'enregistrement courant de la base de parcelles (5) de l'application |

# @dbsettobegin

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

positionne la base d'indice "indbase" de l'application sur son premier enregistrement

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prête | ERR | si base non ouverte ou autre problème |
| ERREUR : impossible de positionner la table | ERR | si erreur |
| OK | STR | le premier enregistrement devient l'enregistrement courant |
| EOF | STR | la base de données est vide |

## Exemples:

| | |
|---|---|
| @dbsettobegin(C5) | positionne la base de parcelles sur le premier enregistrement |

# @dbsettoend

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | indice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis)<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

positionne la base d'indice "indbase" de l'application sur son dernier enregistrement

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prête | ERR | si base non ouverte ou autre problème |
| ERREUR : impossible de positionner la table | ERR | si erreur |
| OK | STR | le dernier enregistrement devient l'enregistrement courant |
| EOF | STR | la base de données est vide |

## Exemples:

| | |
|---|---|
| @dbsettoend(C5) | positionne la base de parcelles sur le dernier enregistrement |

# @dbstand2dbclass

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe réceptrice (1 à ...) |

## Action:

copie toutes les propriétés concernant les objets de classe "classe" de la classe de base standard dans la base de classe correspondante de l'application
Les deux bases doivent avoir été ouvertes
Seules les propriétés définies dans la base de classe peuvent être copiées (pas de création de champ supplémentaire)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ERREUR : base non prête | ERR | si une des 2 bases n'a pas été ouverte |
| ERREUR : aucune propriété n'a été copiée | ERR | |
| ERREUR : NN propriétés seulement ont été transférées | ERR | si TopoCad a détecté des propriétés qu'il n'a pu copier |
| N propriétés ont été transférées | STR | N = Nombre de propriétés copiées (toutes ont été copiées) |

## Exemples:

| | |
|---|---|
| @dbstand2dbclass(5) | copie toutes les propriétés de la base standard vers la base de classe de parcelles |

# @decompidu

*Nb paramètres*
8

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable source contenant l'identifiant à décoder |
| commune | STANDARD | nom de la variable acueillant le code INSEE de la commune |
| prefsect | STANDARD | nom de la variable accueillant le préfixe de section |
| sect | STANDARD | nom de la variable accueillant la section |
| subdsect | STANDARD | nom de la variable accueillant la subdivision de section |
| parcelle | STANDARD | nom de la variable accueillant le numéro de parcelle |
| extra | STANDARD | nom de la variable accueillant la donnée extra (soit l'étiquette, soit le texte, soit le numéro d'objet) |
| idalgo | STANDARD | algorithme de composition utilisé pour l'IDU fourni<br>doit être une combinaison des valeurs suivantes:<br><br>1 = IDALGO_COMMUNE<br>2 = IDALGO_PREFSECT<br>4 = IDALGO_SECTION<br>8 = IDALGO_SUBDSECT<br>0x10 = IDALGO_PARCELLE<br>0x20 = IDALGO_TEXT<br>0x40 = IDALGO_LABEL<br>0x80 = IDALGO_ID<br>0x100 = IDALGO_XDATA<br><br>ces trois dernieres valeurs s'excluent mutuellement |

## Action:

décompose l'identifiant contenu dans la variable "nomvar" sur les différentes variables suivant l'algorithme qui a été utilisé pour composer l'identifiant

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Décomposition de l'identifiant impossible | ERR | si l'algorithme indiqué ne correspond pas à quelque chose de valide |
| OK | STR | décomposition réalisée avec succès |

## Exemples:

| | |
|---|---|
| @decompidu(idu,commune,0,0,0,parcelle,0,0x11) | décompose un identifiant de type "2980015" (commune + numero de parcelle) |

# @decompose

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| face | FC | face à décomposer |
| nivsel1 | STANDARD | niveau de sélection du contour englobant |
| nivsel2 | STANDARD | niveau de sélection des trous |

## Action:

décompose une face en sa face principale et ses trous. Le contour de la face sera sélectionné au niveau "nivsel1" et les trous seront sélectionnés au niveau "nivsel2"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de décomposer la face | ERR | face invalide impossible à décomposer |
| <nbfc> | NUM | nombre de contours extraits |

## Exemples:

@decompose(@varfc,0,1)

# @decoupecouche

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche à découper (de 0 à Nb_Couches_document−1) |
| obj/fc | OBJ ou FC | objet surfacique ou face effectuant la découpe |
| sel | STANDARD | niveau de sélection des éléments vectoriels découpés |
| type | STANDARD | indique comment est calculé le contour de découpe<br>0 = brut = les coordonnées brutes (ramenées au raster pour le raster) sont considérées<br>1 = par transf en cours<br>2 = par transf inverse en cours<br>3 = par calquage |
| vect_rast | STANDARD | indique en cas d'ajustement non virtuel ce qui est ajusté<br>0x01 = raster<br>0x10 = vectoriel<br>0x11 = raster + vectoriel |

## Action:

La découpe peut être vectorielle ou/et raster
La découpe raster utilise la face donnée ou l'ensemble des faces de l'objet pour découper le raster (les coordonnées des points de la face sont considérées suivant "type")
La découpe vectorielle se contente de sélectionner les éléments points et écritures (la pointe du déport est considérée s'il existe un déport) contenus dans la face ou les faces de l'objet fourni.

La découpe vectorielle considère les points de la face/objet comme étant des coordonnées pour la couche de laquelle appartient la face/l'objet (une objet non intègre dont la valeur de la couche est −1 provoque donc une erreur)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| OK | STR | le découpage a eu lieu |

## Exemples:

| | |
|---|---|
| @setselecttravail(0),<br>@decoupecouche(3,@vfc,0,0,0x11),<br>@delete(−1,3,−1) | découpe vectorielle et raster de la couche 3 suivant le contour de la face "vfc" |

# @defined

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à tester |

## Action:

détermine si une variable est définie ou non

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si le paramètre n'est pas standard |
| 1 | BOOL | le nom est le nom d'une variable définie |
| 0 | BOOL | le nom est un mot réservé ou n'est pas le nom d'une variable |

## Exemples:

| | |
|---|---|
| @defined(defined) | renvoie 0 |
| @defined(strangevar) | renvoie 0 |
| @setvar(mavar,"test",STR),<br>@defined(mavar) | renvoie 1 |

# @delallrelsem

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD<br>ou<br>PT | 0 |
| eltobj | LI<br>FC<br>EC<br>OBJ | ou<br>élement ou objet dont il faut trouver la prochaine relation (de type donné) |
| type | STANDARD | type de relation recherchée (si type==−1 alors toutes les relations sont considérées) |

## Action:

si eltobj= "0" alors supprime toutes les relations de type donné du document
sinon supprime toutes les relations de type donné de l'élément ou objet donné.
Si type est différent de −1, supprime quel que soit le type

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <nb> | NUM | renvoie le nbre de relations supprimées |

## Exemples:

| | |
|---|---|
| @delallrelsem(@obj,7) | supprime toutes les relations de type parcelle−>subdiv de sect de l'objet désigné par la variable "obj" . Renvoie le nombre de realations supprimées. |

# @delcadreformatpage

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice du cadre dans le format de page. Cet indice est formatté pour (0 si négatif, dernier cadre si indice élevé), mais doit être un indice différent du cadre plan pour le format de page courant sinon une erreur est renvoyée. |

## Action:

supprime le cadre de format de page du format de page courant. Il est impossible de supprimer le cadre plan (mettre son cadre à 0,0,0,0 s'il n'est pas désiré)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | indice fourni est celui du cadre plan |
| <Nb cadres> | NUM | nombre de cadres du format après la suppression |

## Exemples:

| | |
|---|---|
| @delcadreformatpage(2) | supprime le cadre indiquant l'échelle |

# @delcouche

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice de la couche à supprimer (de 0 à NbCouches) |

## Action:

supprime la couche et tous les éléments ou objets attachés à cette dernière

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Suppression de la couche impossible | ERR | des éléments ou la couche est verrouillée |
| OK | STR | couche supprimée |

## Exemples:

| | |
|---|---|
| @delcouche(1) | supprime la seconde couche |

# @deldatabase

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indbase | STANDARD | iindice de la table dans la base de donnée de TopoCad<br>0 = indice de la table standard<br>N = indice de la table correspondante<br>S = table standard<br>Cn = table de classe correspondante (ex: C5 est l'indice de la base de parcelles)<br>An = table auxiliaire correspondante (ex: A2 est l'indice de la base de permis).<br>En = table externe correspondante (ex: E0 est l'indice de la base servant de statistiques) |

## Action:

Ferme la base de donnée et la supprime de la table interne des bases de données de l'application et renvoie 1 en cas de succès.

La base standard ne peut pas être supprimée par l'utilisateur (renvoie 0).

Une base de classe ne peut être supprimée si une méthode de calcul de l'identifiant existe dans le modèle courant : il est donc néccessaire de modifier le modèle éventuellement avant sinon la procédure renvoie 0.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 1 ou 0 | STR | base supprimée (1) ou non supprimée (0) |

## Exemples:

| | |
|---|---|
| @deldatabase(C6) | ferme le DBF des subdivisions fiscales et supprime la base de donnée de l'application |

# @deleltdata

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | couche concernée ou −1 pour toutes les couches |
| elt | ou | ou |
| | PT, LI, FC, EC | élément dont une donnée est à fixer (un élement nul indique tout élement de ce type) |
| name | STANDARD | nom de la donnée à supprimer (composé de caractères alphanumériques et du signe '_' −underscore−) ou "<chaine vide>" pour supprimer toutes les données |

## Action:

supprime la donnée propre à un élément.
chaque élément possède une collection de données qui lui sont propre et qui meurent et vivent avec lui. Une fusion de deux éléments entraine irrémédiablement la disparition des données d'un des deux éléments fusionné. Ces données sont donc utilisées soit de manière temporaire soit dans le cadre d'une gestion statique du plan (ex: navigateur pour lequel les données ne changent pas).
Les données ont des types divers mais TED ne pourra enregistrer que des données de type numérique (réel) ou des chaines de caractères de faible taille (max 9 caractères).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <data> | STR | donnée de l'élémént |

## Exemples:

| | |
|---|---|
| @deleltdata(elt,debit) | supprime la donnée "débit" d'un élément |
| @deleltdata(elt,"") | supprime toutes les données d'un element |
| @setvar(varpt,0,PT),<br>@deleltdata(@varpt,debit) | supprime la donnée "débit" de tous les points du document |
| @deleltdata(2,"") | supprime toutes les données des éléments de la couche 2 |
| @deleltdata(−1,"") | supprime toutes les données de tous les éléments du document |

# @deleltobj

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément,objet qu'il faut supprimer |
| | ou | ou |
| elt | PT<br>LI<br>FC<br>EC<br>OBJ | élement, objet à supprimer |

## Action:

Supprime l'élément ou l'objet donné en argument.
La suppression de l'objet ne supprime pas les éléments contenus dans l'objet.
La suppression de l'élément supprime les objets éventuels auxquels était attaché l'élément.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Des éléments sont verrouillés | ERR | si un des éléments amenés à être supprimés est vérouillé |
| Ok | OK | suppression effectuée |

## Exemples:

| | |
|---|---|
| @deleltobj(obj) | supprime l'objet (sans les éléments) |

# @delelttoobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obj | OBJ | objet dont il faut retirer l'élément |
| elt | PT<br>LI<br>FC<br>EC | élement à retirer de l' objet |

## Action:

Supprime l'élément de l'objet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 | NUM | l'élément a été retranché de l'objet |
| 1 | NUM | l'élément n'appartient pas à l'objet |
| 2 | NUM | l'élément ne peut être retiré car unique constituant de l'objet |

## Exemples:

| | |
|---|---|
| @delelttoobj(@obj,@elt) | supprime l'élément de l'objet |

# @delete

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe des éléments à supprimer (de 0 à NbTObj−1 ou −1 pour toutes les classes) |
| couche | STANDARD | couche des éléments à supprimer (de 0 à NbCouches−1 ou −1 pour toutes les couches) |
| typeelt | STANDARD | type des éléments à supprimer sous forme de champ de bit<br>1 = EltPoint<br>2 = EltLiaison<br>4 = EltEcriture<br>16=EltFaces<br>−1 pour tous les types d'éléments (pts+lis+ecs+fcs)<br>ou<br>256 = signes de mitoyenneté<br>ou<br>512 = deports d'écriture |

## Action:

supprime tous les éléments sélectionnés au niveau de sélection courant et satisfaisant aux conditions fournies de classe, couche et type.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Des éléments sont verrouillés | ERR | échec de la suppression (aucun élément n'est supprimé) |
| OK | STR | en cas de réussite |

## Exemples:

| | |
|---|---|
| @delete(−1,0,−1) | supprime tous les éléments sélectionnés de la couche 0 |

# @delformeface

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

supprime la dernière forme de face de la table des formes de face.
cette opération n'est possible que si le nombre de forme de faces est suffisant pour le système c'est à dire est supérieur au nombre minimum préétabli et si aucun face ne référence cette forme de face dans aucun document ouvert par l'application.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Impossible de supprimer une forme de face supplémentaire! | ERR | si le nombre minimum de formes nécessaire à l'application est atteint |
| Impossible de supprimer la Forme de face qui est référencée par une Face | ERR | une face utilise cette forme pour se dessiner |
| OK | STR | forme de face supprimée |

## Exemples:

| | |
|---|---|
| @delformeface | supprime la dernière forme des faces |

# @delformeliaison

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

supprime la dernière forme de liaison de la table des formes de liaison.
cette opération n'est possible que si le nombre de forme de liaisons est suffisant pour le système c'est à dire est supérieur au nombre minimum préétabli et si aucun liaison ne référence cette forme de liaison dans aucun document ouvert par l'application.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Impossible de supprimer une forme de liaison supplémentaire! | ERR | si le nombre minimum de formes nécessaire à l'application est atteint |
| Impossible de supprimer la Forme de liaison qui est référencée par une Liaison | ERR | une liaison utilise cette forme pour se dessiner |
| OK | STR | forme de liaison supprimée |

## Exemples:

| | |
|---|---|
| @delformeliaison | supprime la dernière forme des liaisons |

# @delformepoint

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

supprime la dernière forme de point de la table des formes de point.
cette opération n'est possible que si le nombre de forme de points est suffisant pour le système c'est à dire est supérieur au nombre minimum préétabli et si aucun point ne référence cette forme de point dans aucun document ouvert par l'application.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Impossible de supprimer une forme de point supplémentaire! | ERR | si le nombre minimum de formes nécessaire à l'application est atteint |
| Impossible de supprimer la Forme de point qui est référencée par un Point | ERR | un point utilise cette forme pour se dessiner |
| OK | STR | forme de point supprimée |

## Exemples:

| | |
|---|---|
| @delformepoint | supprime la dernière forme des points |

# @delinvalrelsem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de relation (de 0 à NbTRel−1) ou −1 pour tout type de relations |

## Action:

supprime les relations sémantiques invalides de type spécifié ou de tout type si type<0

IMPORTANT : Cette fonction suppose que les informations concernant les objets sont mises à jour (ces informations sont mises à jour par @objintegrity)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de relations supprimées |

## Exemples:

| | |
|---|---|
| @delinvalrelsem(7) | supprime les relations sémantiques Parcelle −> Subdiv de section invalides. |

# @delobservation

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation à supprimer |

## Action:

supprime une observation  au document. Ne supprime pas le point résultat du calcul mais uniquement le calcul (l'observation)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 | STD | l'observation n'a pu être supprimée |
| 1 | STD | l'observation a été supprimée |

## Exemples:

| | |
|---|---|
| @delobservation(@monobs) | suppression de l'observation |

# @delpattern

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

supprime le dernier motif de l'application.
cette opération n'est possible que si le nombre de forme de motifs est suffisant pour le système c'est à dire est supérieur au nombre minimum préétabli et si aucune forme de face ou forme de point ne référence ce motif dans aucun document ouvert par l'application.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Impossible de supprimer un motif supplémentaire! | ERR | si le nombre minimum de motifs nécessaire à l'application est atteint |
| Impossible de supprimer le Motif qui est référencé par une Forme de Face! | ERR | une face utilise ce motif pour se dessiner |
| Impossible de supprimer le Motif qui est référencé par une Forme de Point | ERR | un point utilise ce motif pour se dessiner |
| OK | STR | forme de point supprimée |

## Exemples:

| | |
|---|---|
| @delpattern | supprime le dernier motif |

# @delrelsem

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | PT<br>LI<br>FC<br>EC<br>OBJ<br>RELSEM | variable indiquant la source ou directement la source de la relation ou pointeur nul si la source est indifférente |
| dest | PT<br>LI<br>FC<br>EC<br>OBJ<br>STANDARD | variable indiquant la destination ou directement la destination de la relation ou pointeur nul si la destination est indifférente |
| type | STANDARD | type de relation (de 0 à NbTRel−1) ou −1 si type indifférent |

## Action:

supprime les relations sémantiques d'une source, d'une destination, de type donné ou non ou une relation particulière entre une source et une destination

il est possible également de fournir comme premier parametre la relation elle même.  Dans ce ca le deuxième paramètrre doit être "0"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de relations supprimées |

## Exemples:

| | |
|---|---|
| @setvar(vobj2,0,OBJ),<br>@delrelsem(@vobj1,@vobj2,−1) | supprime toutes les relations dont la source est l'objet indiqué par la variable "vobj" |
| @setvar(rel,@marel,RELSEM),<br>@delrelsem(@rel,0,0) | supprime la relation "rel" |

# @delselrelsem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | niveau de sélection des relations à supprimer (de 0 à 31) |

## Action:

supprime toutes les relations sémantiques sélectionnées au niveau nivsel du document entier quel que soit leur type

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N relations ont été supprimées | NUM | renvoie le nombre de relations supprimées |

## Exemples:

| | |
|---|---|
| @delselrelsem(0) | supprime toutes les relations sélectionnées au niveau 0 |

# @deltextobj

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obj | OBJ | objet dont il faut supprimer les écritures |

## Action:

supprime toutes les écritures de l'objet "obj" mais ne détruit pas les écritures

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ne peut vider complètement l'objet | ERR | si l'objet est uniquement constitué d'écritures. |
| N | NUM | N = Nombre d'écritures supprimées |

## Exemples:

| | |
|---|---|
| @deltextobj(@obj) | supprime les écritures de l'objet contenu dans la variable "obj" |

# @dependde

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eltobj1 | STANDARD | Nom d'une variable contenant un objet ou un élément |
| | OBJ | Objet |
| | PT | Element Point |
| | LI | Element Liaison |
| | FC | Element Face |
| | EC | Element Ecriture |
| eltobj2 | STANDARD | Nom d'une variable contenant un objet ou un élément |
| | OBJ | Objet |
| | PT | Element Point |
| | LI | Element Liaison |
| | FC | Element Face |
| | EC | Element Ecriture |

## Action:

détermine si *eltobj1* dépend de *eltobj2*.

Un élément A dépend d'un élément B si la suppression de B entraine la suppression de A

Il en est de même pour les objets.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| ERREUR: Test de dépendance impossible | ERR | test incohérent (ex: test de dépendance d'un objet à un élément...) |
| 0 ou 1 | BOOL | résultat du test |

## Exemples:

| | |
|---|---|
| @dependde(@var1,var2) | teste si le contenu de la variable var1 (par ex: var1 est le nom d'une variable de type OBJ) dépend de l'objet de nom var2 |
| @dependde(var1,var2) | idem (var1 étant de type standard, c'est la variable de ce nom qui est recherchée) |

# @diffelt

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche1 | STANDARD | numéro de la première couche concernée (de 0 à Nb_Couches_document−1) |
| couche2 | STANDARD | numéro de la seconde couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection à modifier pour marquer la différence (de 0 à 31) |
| masque | STANDARD | masque de 32 bits indiquant les différences à considérer:<br>octet 0 = critères des points<br>octet 1 = critères des liaisons<br>octet 2 = critères des faces<br>octet 3 = critères des écritures<br><br>la partie basse de chacun des octets concerne les propriétés propres aux éléments :<br>bit 0 = Epaisseur<br>bit 1 = Couleur<br>bit 2 = Classe<br>bit 3 = Etiquette<br><br>les autres propriétés sont :<br>0x1000 = Forme (liaison)<br>0x200000 = Forme (face)<br>0x100000 = Couleur de fond (face)<br>0x10000000 = orientation (écriture)<br>0x20000000 = police (ecriture)<br>0x40000000 = hauteur (ecriture)<br>0x80000000 = texte (ecriture)<br><br>Une valeur de −1 (tous bits à 1) indique donc qu'un maximum de critères d'égalité doivent être testés.<br>Une valeur de 0 indique donc uniquement de tester en position les éléments. |
| telt | STANDARD | Type d'élément sur lequel faire l'ajustement. peut être :<br>1=Points<br>2=Liaisons<br>4=Ecritures<br>16=Faces |

## Action:

marque les différences entre deux couches en comparant les éléments en position à EpsilonAppFace près et suivant les différences à considérer données par le masque

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Nb différences = N | STR | |

## Exemples:

| | |
|---|---|
| @diffelt(0,1,0,8,1) | sélectionne tous les points (1) différents en position entre les couches 0 et 1 et les points de même position mais ayant une étiquette (bit 3=8) différente. |

# @diffpt

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche1 | STANDARD | numéro de la premiere couche concernée (de 0 à Nb_Couches_document−1) |
| couche2 | STANDARD | numéro de la seconde couche concernée (de 0 à Nb_Couches_document−1) |
| code | STANDARD | points à sélectionner (combinaison binaire)<br>0 = pas de sélection (juste un rapport d'effectué)<br>1 = les points communs aux 2 couches sont sélectionnés<br>2 = les points ambigüs sont sélectionnés<br>4 = les points isolés (uniques) sont sélectionnés |
| epsilon | STANDARD | écart maxi entre les points au dela duquel les points ne sont plus considérés comme commun. |

## Action:

fournit un rapport dans ERRORxxx.LOG indiquant les écarts constatés sur les points communs entre les deux couches ainsi que les statistiques sur les écarts, le nombre de points ambigus, isolé ... et sélectionne suivant "code" les points désirés

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @diffpt(0,1,2,0.1) | effectue un rapport sur les points communs aux 2 couches et sélectionne les points ambigus |

# @dijkstra

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | PT | point source |
| dest | PT | point destination |
| selsrce | STANDARD | niveau de sélection représentant le graphe ou −1 si toute la couche est à considérer comme étant le graphe |
| seldest | STANDARD | niveau de sélection qui servira à la sélection du chemin le plus court trouvé ou −1 si le chemin le plus court ne doit pas être sélectionné |

## Action:

Effectue un calcul par l'algorithme de Dijkstra. cet algorithme calcule dans un graphe le plus court chemin d'un point source à un point destination.
Le graphe est représenté par l'ensemble des liaisons liées entre elles composant la couche des points source et destination. Il est possible de limiter le graphe aux liaisons sélectionnées de la couche en indiquant un niveau de sélection "selsrce". A l'issue du calcul, les liaisons composant le plus court chemin sont sélectionnées au niveau de sélection "seldest" s'il est fourni (différent de −1). Les données utilisées pour le calcul peuvent être conservées si "savexdata" est à 1 (les données extra propres aux éléments sont volatiles et non sauvegardées avec le document et donc ont une portée limittée à la session de travail).
La fonction renvoie la plus courte distance parcourue du point "srce" au point "dest" (qui est la donnée de nom "D" du point "dest")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <dist> | NUM | fournit la distance du plus court chemin. |
| Données n'ont pu être conservées! | ERR | problème à l'écriture ou lecture d'un Xdata d'un élément |
| Calcul impossible! | ERR | impossible d'atteindre le point "dest" à partir du point "srce" |

## Exemples:

| | |
|---|---|
| @dijkstra(@getpoint(1), @getpoint(9),−1,0) | sélectionne au niveau 0 le plus court chemin du point 1 au point 9 pour l'ensemble des liaisons de la couche du point 1. |

# @display

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un objet ou élément |
| | PT | ou |
| eltobj | LI | |
| | FC | élément ou objet à afficher |
| | EC | |
| | OBJ | |

## Action:

raffraichit l'écran en déplacant la vue de manière à ce que l'élément ou l'objet soit au centre de la fenêtre

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Objet incohérent (nature différente de sa composition) | ERR | par ex: un objet surfacique qui n'aurait pas de face mais seulement une écriture |
| Display : ERREUR : Impossible de déplacer une couche gelée | ERR | si la couche contenant l'élément ou l'objet à visualiser est gelée |
| OK | STR | affichage effectué |

## Exemples:

| | |
|---|---|
| @display(@obj) | affiche l'objet contenu dans la variable "obj" |
| @display(obj) | affiche l'objet contenu dans la variable "obj" |
| @display(0) | raffrichit la vue |

# @displayat

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| X | STANDARD | Coordonnee X réelle d'un point sur la couche de travail à afficher |
| Y | STANDARD | Coordonnee X réelle d'un point sur la couche de travail à afficher |

## Action:

raffraichit l'écran en déplacant la vue de manière à ce que le point spécifié de la couche de travail soit au centre de la fenêtre

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| displayat : ERREUR : Impossible de déplacer une couche gelée | ERR | si la couche de travail est gelée |
| OK | STR | affichage effectué |

## Exemples:

| | |
|---|---|
| @displayat(800000,123000) | centre la fenetre sur le point |

# @dist

*Nb paramètres*
indéfini

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| num1 | STANDARD<br>PT | numéro du point 1<br>ou point 1 |
| num2 | STANDARD<br>PT | numéro du point 2<br>ou point 2 |
| num3 | STANDARD<br>PT | numéro du point 3<br>ou point 3 |
| num4 | STANDARD<br>PT | numéro du point 4<br>ou point 4 |

...etc

## Action:

fournit la distance de la polyligne définie par num1−num2−num3 ... etc.(distance cumulée des segments de droites)
2 paramètres au moins doivent être fournis

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Point inconnu | ERR | un numéro de point donné est inconnu dans le document ou le numéro donné n'est pas un numéro (par ex chaine de caractères) |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Nombre de paramètres insuffisant | ERR | si le nombre de points est insuffisant. |
| <dist cumulée> | NUM | distance cumulée de tous les segments entre points |

## Exemples:

| | |
|---|---|
| @dist(2,@varpt,9,2) | varpt étant une variable contenant le point 4, fournit la distance entre les points 2 et 4 + distance entre 4 et 9 + distance entre 9 et 2 (même si aucune liaison n'existe entre ces points) |
| @dist(5,7) | distance entre le point 5 et le point 7 |

# @disteltobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | PT ou LI ou FC ou EC | élément à considérer |
| obj | OBJ | objet à considérer |

## Action:

fournit la distance entre l'élément et l'objet (plus courte distance).
NB: la distance d'une face à une autre est la plus courte distance d'un point de la face à un autre point de la face

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <distance> | NUM | |

## Exemples:

| | |
|---|---|
| @disteltobj(@varelt,@varobj) | donne la distance entre l'élément contenu dans la variable de nom "varelt" et l'objet contenu dans la variable de nom "varobj". |

# @distentreelt

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt1 | PT ou LI ou FC ou EC | premier élément à considérer |
| elt2 | PT ou LI ou FC ou EC | second élément à considérer |

## Action:

fournit la distance entre deux éléments
NB: la distance d'une face à une autre est la plus courte distance d'un point de la face à un autre point de la face

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <distance> | NUM | |

## Exemples:

| | |
|---|---|
| @distentreelt(@varelt1,@varelt2) | donne la distance entre l'élément contenu dans la variable de nom "varelt1" et l'élément contenu dans la variable de nom "varelt2" |

# @distentreobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obj1 | OBJ | premier objet à considérer |
| obj2 | OBJ | second objet à considérer |

## Action:

fournit la distance entre deux objets(plus courte distance).
NB: la distance d'une face à une autre est la plus courte distance d'un point de la face à un autre point de la face

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <distance> | NUM | |

## Exemples:

| | |
|---|---|
| @distentreobj(@varobj1,@varobj2) | donne la distance entre l'objet contenu dans la variable de nom "varobj1" et l'objet contenu dans la variable de nom "varobj2". |

# @distobj

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obj | OBJ | objet concerné |

## Action:

fournit la longueur d'un objet linéaire ou le périmètre d'un objet surfacique. pour les objets d'une autre nature renvoie 0

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <périmètre/longueur> | NUM | |

## Exemples:

| | |
|---|---|
| @distobj(@varobj) | fournit la surface de l'objet surfacique contenu dans la variable de nom "varobj" |

# @distrib

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche source dans laquelle rechercher les objets/éléments de classe "classe" (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche destination dans laquelle copier les objets/éléments de classe "classe" (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des objets ou éléments à distribuer de "couchesrce" vers "couchedest" |
| parobj | STANDARD | indique si les objets doivent être copiés (1) ou si seulement les éléments de la classe doivent être copiés(0) |
| parmaskcr | STANDARD | somme d'indicateurs (EltPoint=1,EltLiaison=2,EltEcriture=4,EltFace=16) indiquant si les éléments distribués sont créés par le masque de création de la classe ou par une copie des propriétés de la source |
| parmaskmd | STANDARD | somme d'indicateurs (EltPoint=1,EltLiaison=2,EltEcriture=4,EltFace=16) indiquant si les éléments distribués sont systématiquement modifiés par le masque de modification de la classe. |

## Action:

copie tous les objets ou éléments de classe "classe" de la couche "couchesrce" vers la couche "couchedest". Les éléments sont copiés avec les propriétés intrinsèques à la classe ou/et au dessin.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| OK | STR | si tout s'est bien déroulé |

## Exemples:

| | |
|---|---|
| @distrib(0,2,7,1,3,0) | distribue tous les objets batis dur (7) de la couche 0 dans la couche 2. Les faces sont recopiées avec leurs couleurs et leurs propriétés d'origine, les liaisons par contre seront forcément des liaisons ayant les caractéristiques des traits de bati et les points des points simples (caractéristique des points de bati dur) |
| @distrib(0,2,7,1,0,0) | distribue tous les objets batis dur (7) de la couche 0 dans la couche 2. Les faces, liaisons, points sont recopiées avec leurs couleurs et leurs propriétés d'origine. Ainsi, si un point de polygo borde un bati dur, il sera recopié comme point de polygo (pas d'objet). |
| @distrib(0,2,7,1,23,0) | distribue tous les objets batis dur (7) de la couche 0 dans la couche 2. Les faces, liaisons, points sont recopiées comme s'il s'agissait d'éléments créés de toute pièces avec les propriétés d'origine de la classe bati dur (couleur, forme...) |
| @distrib(0,2,7,1,0,16) | distribue tous les objets batis dur (7) de la couche 0 dans la couche 2. Les éléments sont copiés avec leurs propriétés apparaissant dans la source puis les faces sont modifiées par le masque de modification de la classe. |

# @div

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | valeur1 |
| val2 | STANDARD | valeur2 |

## Action:

val1/val2

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <val1/val2> | NUM | founit la valeur val1 après division par la valeur val2 |

## Exemples:

| | |
|---|---|
| @div(@mavar,5) | divise le contenu de la variable "mavar" par 5 et renvoie le résultat |

# @divvar

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |
| value | STANDARD | diviseur |

## Action:

[nomvar] <== [nomvar]/value

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| impossible de diviser une variable pointeur | ERR | si la variable n'est pas standard |
| [nomvar] | NUM | founit la valeur de la variable après division par la valeur |

## Exemples:

| | |
|---|---|
| @divvar(mavar,5) | divise le contenu de la variable "mavar" par 5 et l'enregistre dans "mavar" |

# @dms2r

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| dms | STANDARD | chaine degrés,minutes,secondes de la forme 12d45'50.878" |

## Action:

transforme une chaine de caractère exprimant une valeur degrés, minutes, secondes en une valeur angulaire en radian

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si le paramètre n'est pas une chaine DMS |
| <r> | NUM | valeur de l'angle exprimée en radian |

## Exemples:

| | |
|---|---|
| @dms2r(45d12'35") | renvoie 0.7890585066898254674 |
| @dms2r(−45d12'35") | renvoie −0.7890585066898254674 |
| @dms2r(200d30') | renvoie 3.499385150248631504 |

# @dumpvar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| option | STANDARD | bit 0 = sortie des variables globales<br>bit 1 = sortie des variables de tous les documents (sinon seul le document courant) |

## Action:

Fonction de débogage : effectue une sortie dans le fichier ERRORxxx.LOG de toutes les variables TED présentes (nom et valeur)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| OK | STR | |

## Exemples:

| | |
|---|---|
| @dumpvar(0) | sortie de toutes les variables propres au document en cours |
| @dumpvar(1) | sortie de toutes les variables propres au document en cours et des variables globales |
| @dumpvar(3) | sortie de toutes les variables (globales et propres aux différents documents) |

# @echangecouches

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche source (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche destination (de 0 à Nb_Couches_document−1) |

## Action:

Echange deux couches entre elles. Les parties vectorielles et raster sont échangées et le transfert d'une couche à l'autre se fait de manière brute sans changement de coordonnées.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| OK | STR | le transfert a eu lieu |

## Exemples:

| | |
|---|---|
| @echangecouches(0,1) | échange les couche 0 et 1 |

# @echelle

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit l'échelle du document (avec 21 décimales)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <echelle> | NUM | |

## Exemples:

@echelle

# @echorigin

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit l'échelle d'origine du document (avec 21 décimales)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <echorigin> | NUM | |

## Exemples:

@echorigin

# @equ

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| term1 | STANDARD<br>PT<br>LI<br>FC<br>EC<br>MASK... | premier terme |
| term2 | STANDARD<br>PT<br>LI<br>FC<br>EC<br>MASK... | second terme |

## Action:

opérateur d'égalité. Renvoie 1 (true) si term1==term2. Pour les types pointeurs, l'objet pointé doit être identique, pour les types standards, c'est une comparaison de chaine de caractères qui est faite, sauf si les deux types sont numériques auquel cas c'est la conversion en flottant qui doit être identique : elles seront comparées comme étant des valeurs numériques décimales.Elles seront comparées comme étant des angles au format TopoCad (ex: 125.12dv) seulement si ce sont des valeurs de type angle
Si une variable numérique contient une valeur octale ou hexadécimale, la comparaison peut ne pas être celle désirée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| term1==term2 | BOOL | 1 pour vrai et 0 pour faux |

## Exemples:

| | |
|---|---|
| @equ(@var1,@var2) | teste pour savoir si le contenu de la variable "var1" est égal au contenu de la variable "var2" |
| @equ(3,3.25) | renvoie 0 (faux) |
| @equ(rue,rue) | renvoie 1 (vrai) |
| @equ("fée","FEE") | renvoie 0 (faux) |
| @equ("fee","FEE") | renvoie 1 (vrai) |
| @equ(charmes,rues) | renvoie 0 (faux) |
| @equ(0x30,0x40) | renvoie 0 (faux) |
| @equ(@num(0x30),@num(0x40)) | renvoie 1 (vrai) car la conversion en décimale donne 0 des deux cotés |
| @equ(@numi(0x30),@numi(0x40)) | renvoie 0 (faux) car ici on compare en décimale 48 et 64 |

# @error

*Nb errorètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | STR | provoque une erreur (sortie de TED) avec dans "parm" représentant la raison de l'erreur |

## Action:

provoque une erreur : ouvre une boite de dialogue d'erreur contenant la raison de l'erreur donnée par parm et positionne le fichier TED d'ou provient l'erreur.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| parm | ERR | raison de l'erreur |

## Exemples:

@error(Le nom de couche ne respecte pas le modèle) renvoie une erreur, stoppe le programme TED

# @estinclusdans

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eltobj1 | STANDARD | Nom d'une variable contenant un objet ou un élément |
| | OBJ | Objet |
| | PT | Element Point |
| | LI | Element Liaison |
| | FC | Element Face |
| | EC | Element Ecriture |
| eltobj2 | STANDARD | Nom d'une variable contenant un objet ou un élément |
| | OBJ | Objet |
| | PT | Element Point |
| | LI | Element Liaison |
| | FC | Element Face |
| | EC | Element Ecriture |

## Action:

détermine si *eltobj1* est inclus dans *eltobj2*.

Un *point* est inclus dans un *point* si les coordonnées sont identiques.
Un *point* est inclus dans une *liaison* si est sur la liaison (tolérance à l'écartement de la liaison EpsilonAppFace)
Un *point* est inclus dans une *face* si il se trouve dans la face (si il constitue un des points du polygone de la face ou est à l'intérieur)
Un *point* est inclus dans une *écriture* si ses coordonnées sont celles du point d'insertion de l'écriture (ou de son déport si cette dernière possède un déport d'écriture)
Une *liaison* est inclus dans une *liaison* si ses deux points sont inclus dans la liaison (tolérance à l'écartement de chacun des points de la liaison EpsilonAppFace)
Une *liaison* est inclus dans une *face* si tous les points de la liaison sont dans la face.
Une *face* 1 est inclus dans une *face* 2 si tous les points de la face 1 sont dans la face 2.
Une *écriture* est inclus dans un *élément* si le point d'insertion d'écriture ou son déport est inclus dans l'élément
Le test d'inclusion de l'objet dans un autre objet est fait de manière identique suivant la nature des deux objets
Tous les autres types de test d'inclusion échouent (par ex: eltobj1=FC et eltobj2=PT ) en renvoyant "ERREUR: Test d'inclusion impossible"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| ERREUR: Test d'inclusion impossible | ERR | test incohérent (ex une face est inclus dans une liaison...) |
| 0 ou 1 | BOOL | résultat du test |

## Exemples:

| | |
|---|---|
| @estinclusdans(@var1,var2) | teste si le contenu de la variable var1 (par ex: var1 est le nom d'une variable de type OBJ) est inclus dans l'objet de nom var2 |
| @estinclusdans(var1,var2) | idem (var1 étant de type standard, c'est la variable de ce nom qui est recherchée) |

# @etiqueter

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe des objets à étiqueter |
| lastnum | STANDARD | numero au dela duquel faire le numérotage |
| incr | STANDARD | incrément à réaliser |
| razlbl | STANDARD | 1 = force à étiqueter même si étiquette déjà existante<br>0 = n'étiquette que les étiquettes qui n'existent pas encore et avec un numéro non déjà existant |

## Action:

repositionne l'écriture ou la flèche de rattachement d'écriture (déport d'écriture) et la réoriente par rapport à un élément (point,liaison,face,écriture) suivant les critères donnés.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @etiqueter(13,0,1,0) | étiquette les points de canevas en laissant les numéros existant |

# @exec

*Nb paramètres*
indéfini

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom de fichier TED à executer |
| param1 | ??? | paramètre 1 |
| param2 | ??? | paramètre 2 |
| ....etc | | |

## Action:

exécute un fichier TED en fournissant éventuellement des paramètres : ceux ci seront récupérés par la procédure appelée comme si ils étaient des variables locales de nom "param1", "param2" ....etc.
Les paramètres sont fournis par valeur. Pour transmettre une référence à une variable, utiliser @refvar.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <résultat de la commande> | ??? | |

## Exemples:

```
@exec(c:\prog.ted)
```
execute le programme TED

```
@setlvar(nomfichier,"d:\BAL\maliste.txt",STR),
@setlvar(motrecherche,"PC07M1002",STR),
@exec("d:\bal\comptemot.ted",@nomfichier,@motrecherche)
;procedure comptemot.ted
@param(fichier,mot),
@setlvar(msg,"Recherche des mots'",STR),
@concatvar(msg,@mot,0),
@concatvar(msg,"' dans le fichier ",0),
@concatvar(msg,@fichier,0),
@out(@msg),
....
```
execution d'un programme avec des paramètres (fournis par valeur)

```
@setlvar(valeur,5,NUM),
@if(@exec("d:\bal\cube.ted",@refvar(valeur)),
  ......
; procedure cube.ted
@param(val),
@setlvar(result,@mul(@val,@val),NUM),
@mulvar(result,@val),
@subvar(result,10000),
@setvar(val,@result,NUM),
@if(@sup(@result,@num(0)),
  @return(1),
  @return(0)
  )
```
execution d'un programme avec un paramètre passé par référence (effectue le cube de valeur moins 10000 et renvoie 1 si positif)
Technique à favoriser

```
@setlvar(valeur,5,NUM),
@if(@exec("d:\bal\cube.ted",valeur),
  ......
; procedure cube.ted
@param(tmp),
@setvar(val,@tmp,NUM)
@setlvar(result,@mul(@val,@val),NUM),
@mulvar(result,@val),
@subvar(result,10000),
@setvar(@ivar(@tmp),@result,NUM),
@if(@sup(@result,@num(0)),
  @return(1),
  @return(0)
  )
```
execution du même programme avec un paramètre passé par référence avec une technique compatible avec la version 5 (aucune variable locale de nom "valeur" doit être définie dans cube.ted)

```
@setdebug(0x400),
@setvar(valeur,5,NUM),
@if(@exec("d:\bal\cube.ted",valeur),
  ......
; procedure cube.ted
@setlvar(result,@mul(@val,@val),NUM),
@mulvar(result,@val),
@subvar(result,10000),
@setvar(valeur,@result,NUM),
@if(@sup(@result,@num(0)),
  @return(1),
  @return(0)
  )
```

execution du même programme avec un paramètre passé par référence avec une technique compatible avec la version 5 (aucune variable locale de nom "valeur" doit être définie dans cube.ted, les variables initialisées par @setvar peuvent ici ne pas exister et être créées alors pour l'ensemble du programme)

# @exist

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom de fichier à rechercher |
| withrep | STANDARD | indique le type de recherche :<br>0 = un fichier (en excluant les répertoires)<br>1 = un fichier ou un repertoire<br>2 = un fichier ou un repertoire en excluant "." et ".."<br>ex: *.tif, 244_??.map ...etc |

## Action:

indique si un fichier ou un répertoire existe

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 ou 1 | BOOL | |

## Exemples:

| | |
|---|---|
| @exist(c:\edigeo01.thf,1) | renvoie 1 si le fichier existe, 0 sinon |
| @exist("c:\edigeo\com−003\.",1) | renvoie 1 si le répertoire c:\edigeo\com−003 existe |
| @exist("c::\edigeo*.*",2) | renvoie 1 si le répertoire c:\edigeo n'est pas vide |

# @exit

*Nb exitètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | ??? | renvoie le paramètre fourni en provoquant une sortie de TED |

## Action:

renvoie simplement le paramètre et termine TED

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| parm | ??? | type du paramètre (standard) |

## Exemples:

| | |
|---|---|
| @exit(OK) | renvoie OK et quitte TED |

# @export

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom du fichier pour exportation |
| type | STANDARD | type d'exportation<br>1 = NXY<br>2 = DXF<br>3 = EDIGEO/PCI<br>4 = BMP<br>5 = WMF<br>6 = MIF<br>7 = SHP<br>8 = EDIGEO UTILISATEUR<br>9 = LOC<br>10 = ASC (APIC)<br>11 = TXT (DA numérique)<br>12 = OSM<br>13 = KML |
| strict | STANDARD | indique si l'échec de l'export renvoie une erreur (strict=1) ou la valeur 0 (strict=0) |

## Action:

export du document ou partie du document (en fonction des paramètres présents au moment de l'exportation). Pour l'export BMP, une boite de dialogue demande des précisions

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Echec à l'exportation de ... | ERR | si l'export n'a pu se dérouler correctement (et strict=1) |
| 0 ou 1 | NUM | export réalisé avec succès (1) ou non (0) |

## Exemples:

| | |
|---|---|
| @export(c:\temp\edigeo01.thf,3,1) | export Edigeo du document |

# @extract

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | chaine dont on veut extraire une partie |
| pos | STANDARD | indique la position (0 à N) de la partie que l'on veut extraire |
| delim | STANDARD | chaine faisant office de délimiteur |
| sep | STANDARD | caractère (optionnel) indiquant le séparateur de texte |

## Action:

La chaine "val1" est sensée être une chaine composée de plusieurs parties séparées par une chaine délimiteur. La fonction renvoie la partie que l'on veut extraire entre ces délimiteurs.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <extrait de chaine> | STR | renvoie une partie de la chaine |

## Exemples:

| | |
|---|---|
| @extract("coucouetmeetvoilou",0,"et","") | renvoie la chaine "coucou" |
| @extract("coucouetmeetvoilou",1,"et","") | renvoie la chaine "me" |
| @extract("coucou",0,"cou","") | renvoie la chaine vide "" |
| @extract("coucou,me,voilou",5,",","") | renvoie la chaine vide "" |
| @extract("cou;−cou;me−;voilou",2,";","−") | renvoie la chaine "voilou" |
| @setvar(chaine,"123,\x2212,rue des myosotis\x22,PARIS",STR)<br>@extract(@chaine,1,",","\x22") | renvoie la chaine "12,rue des myosotis" |
| @setvar(chaine,"123,\x2212,rue des myosotis\x22,PARIS",STR)<br>@extract(@chaine,1,",","22") | erreur: le séparateur de texte doit être un seul caractère ou la chaine vide |

# @extractword

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | chaine dont on veut extraire un mot |
| pos | STANDARD | indique la position (0 à N) du mot que l'on veut extraire |
| chaine_esp | STANDARD | chaine des caractères à ignorer et à considérer comme des espaces |

## Action:

La chaine "val1" est sensée être une chaine composée de plusieurs parties séparées par des caractères assimilés à des espace (l'espace, la tabulation, le retour chariot...et les caractères de la chaine "chaine_esp"). La fonction renvoie la partie que l'on veut extraire entre ces espaces, c'est à dire le mot dans une phrase ordinaire.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <extrait de chaine> | STR | renvoie le mot |

## Exemples:

| | |
|---|---|
| @extractword("coucou  me voilou",0,"") | renvoie la chaine "coucou" |
| @extractword("coucou  me    voilou",1,"") | renvoie la chaine "me" |
| @extractword("   coucou",0,"") | renvoie la chaine "coucou" |
| @extractword("coucou me voilou",3,"") | renvoie la chaine vide "" |
| @extractword("coucou me voilou",50,"") | renvoie la chaine vide "" |
| @extractword("coucou me voilou",−1,"") | renvoie erreur |
| @extractword("coucou me voilou",4,"ou") | renvoie "il" (5° parametre dans la chaine "c  c    me v il  ") |

# @faces2crois

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classeli | STANDARD | classe des liaisons à créer dans les faces |
| nivsel | STANDARD | niveau de sélection (de 0 à 31) des faces à considérer : la sélection est effacée si l'opération a réussi pour la face |
| createobj | STANDARD | indique si l'objet (constitué des 2 croisillons) de classe "classeli" doit être créé (1) ou non (0) |

## Action:

crée des croisillons à l'intérieur des faces de classe "classefc" et laisse sélectionné les cas non résolus

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @faces2crois(0,9,0,1) | crée les croisillons à partir des faces "bati léger" sélectionnées et crée les objets "croisillons". |

# @feuille

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le nom de la feuille (ou subdivision de section) du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <nom de feuille> | STR | nom de la feuille |

## Exemples:

| | |
|---|---|
| @feuille | valeur renvoyée = "0" |

# @fileopensavedlg

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable qui va recevoir le nom du fichier (cette variable doit etre de type standard)<br>la valeur de cette variable renseigne le nom du fichier par défaut en entrée |
| opt | STANDARD | option sous forme de champ de bit<br><br>bit 0 : à 0 => une boite de dialogue d'ouverture de fichier est demandée<br>bit 0 : à 1 => une boite de dialogue de sauvegarde de fichier est demandée<br>bit 1 : à 1 => indique que l'on doit choisir un fichier existant<br>bit 1 : à 0 => indique que l'on peut choisir un nouveau nom de fichier<br>bit 2 : à 0 => indique un choix de fichier<br>bit 2 : à 1 => indique un choix de répertoire |
| initdir | STANDARD | indique le chemin initial de recherche de la boite de dialogue (sous forme "C:\TOPOCAD\DOC" par exemple) |
| ext | STANDARD | extension des fichiers devant être affichés (sous forme "MAP" par exemple) |

## Action:

amène une boite de dialogue de saisie d'un fichier afin de choisir un fichier en parcourant l'arborescence. Nomvar est initialisé avec le nom complet du fichier qui est également retourné par la fonction. Si il s'agit d'un choix de répertoire le répertoire est fourni avec le '\' final

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <divers message> | ERR | en cas d'interruption par abandon ou variable non appropriée ou variable inconnue |
| <longueur> | STR | en cas de réussite, longueur du nom du fichier sans chemin et avec l'extension, en cas de demande de répertoire renvoie la longueur de la chaine de répertoire dans le nom du fichier fourni (donc avec \ final). |

## Exemples:

@fileopensavedlg(destname,0,"c:\topocad\doc","ted") demande d'ouverture d'un fichier d'extension *.TED

# @first

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à initialiserqui peut être de type suivant<br>PT,LI, FC, EC, MAPDOC, PLANVIEW, NUM, STR |
| couche | STANDARD | couche concernée (de 0 à NbCouches) ou −1 si toutes les couches sont concernées (soit donc le premier du document) |

## Action:

initialise une variable avec sa première valeur
dans le cas d'une variable pointeur (PT, LI...) c'est le premier point, liaison... qui est pris, dans les autres cas c'est "0"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre non valide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Erreur de syntaxe : variable attendue | ERR | le nom de la variable fournie n'existe pas |
| <valeur> | ??? | premiere valeur renvoyée |

## Exemples:

| | |
|---|---|
| @first(varpt) | initialise la variable avec sa première valeur |

# @firstobjet

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe de l'objet recherché. Si 0 est fourni, alors les objets de toute classe sont considérés. Si −1 est fourni, les objets composés d'éléments de classes différentes sont considérés |
| couche | STANDARD | couche de l'objet recherché. Si −1 est fourni, alors les objets de toute couches sont considérés. |

## Action:

fourni le premier objet de la classe et de la couche fournies.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <objet> | OBJ | renvoie l'objet ou NULL si non trouvé |

## Exemples:

| | |
|---|---|
| @firstobjet(5,−1) | renvoie le premier objet de classe parcelle du document |

# @firstobservation

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| num | STD | numéro du point résultat de l'observation à rechercher ou 0 si on desire avoir la première observation |

## Action:

recherche la premiere observation dont le résultat a pour numero "num"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <obs> | OBS | l'observation ou NULL si aucune trouvée |

## Exemples:

| | |
|---|---|
| @firstobservation(127) | fournit la premiere observation de numero 127 |
| @firstobservation(0) | fournit la première observation du document |

# @firstrelsem

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD ou PT | 0 |
| eltobj | LI FC EC OBJ | ou élement ou objet dont il faut trouver la première relation (de type donné) |
| type | STANDARD | type de relation recherchée (si type==−1 alors toutes les relations sont considérées) |

## Action:

si eltobj == "0" alors fournit la première relation du document de type donné si type est différent de −1
sinon fournit la première relation d'un élément ou objet (et de type donné si type est différent de −1).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <relation> | RELSEM | renvoie la relation ou NULL si non trouvé |

## Exemples:

| | |
|---|---|
| @firstrelsem(@obj,7) | renvoie la premiere relation de type parcelle−>subdiv de sect de l'objet désigné par la variable "obj". Renvoie une erreur si la variable "obj" ne contient pas un objet. |

# @firstwithelt

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à initialiser qui peut être de type PT, LI, FC,OBJ |
| nomelt | PT,LI,FC,EC | élément dont il faut scruter les éléments en relation. |

## Action:

initialise une variable de type élément avec le premier élément des éléments en relation avec l'élément donné (PT, LI, ou FC) ou  le premier objet qui contient l'élément..

si nomvar est PT, nomelt ne peut être que LI
si nomvar est LI, nomelt ne peut être que PT ou FC
si nomvar est FC, nomelt ne peut être que LI
si nomvar est OBJ, nomelt ne peut etre que PT, LI, FC, ou EC

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | PT,LI,FC,OBJ | renvoie la variable après initialisation |

## Exemples:

| | |
|---|---|
| @firstwithelt(mavar,@elt) | initialise "mavar" avec le premier point appartenant à la liaison @elt (le point source), mavar est de type PT |

# @firstwithmask

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à initialiser qui peut être de type PT, LI, FC, ou EC |
| nommasque | STANDARD | nom de la variable contenant le masque de recherche permettant de choisir le premier élément. La variable peut être de type MASKCHPT,MASKCHLI, MASKCHFC, ou MASKCHEC |
| couche | STANDARD | couche à scruter ou toutes si −1 |

## Action:

initialise une variable de type élément avec le premier élément répondant au masque de recherche fourni dans la couche donnée ou parmi toutes les couches si "couche"==−1.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | ??? | renvoie la variable après initialisation |

## Exemples:

| | |
|---|---|
| @firstwithmask(mavar,mask,−1) | initialise "mavar" avec le premier point satisfaisnt au masque mask (mavar est de type PT et mask de type MASKCHPT) |

# @firstwithobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à initialiser qui peut être de type PT, LI, FC, ou EC |
| nomobj | OBJ | objet dont il faut scruter les éléments. |

## Action:

initialise une variable de type élément avec le premier élément de l'objet du même type que la variable (PT, LI, ou FC ou EC).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | ??? | renvoie la variable après initialisation |

## Exemples:

| | |
|---|---|
| @firstwithobj(mavar,@obj) | initialise "mavar" avec le premier point appartenant à l'objet "obj" (mavar est de type PT et obj de type OBJ) |

# @fixdec

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de couche (de 0 à NbCouches−1) |
| select | STANDARD | niveau de sélection des déports d'écritures à considérer (de 0 à 31 ou −1 si tous sont à considerer) |
| angle | STANDARD | valeur de l'angle représentant la verticale de la feuille par rapport au nord géographique |

## Action:

fixe les déports d'écritures étant sélectionnés au niveau 'nivsel' de la couche "couche" (par l'écriture) à droite ou à gauche comme lors de la présentation en considérant que la feuille fait une orientation de "angle" par rapport au Nord géographique.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <nb dec> | NUM | nombre de déports positionnés |

## Exemples:

| | |
|---|---|
| @fixdec(0,−1,0) | fixe les déport d'écriture de la couche 0 |

# @for

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| init | ??? | commande d'initialisation quelconque |
| cond | ??? | condition d'exécution. (un opérateur booléen existe sur tout type) |
| exec | ??? | commande (pour une liste de commandes à executer voir @list ou @exec) à éxécuter si la condition est remplie. |
| incr | ??? | commande quelconque d'incrémentation |

## Action:

boucle conditionnelle
Il est possible d'interrompre la boucle par ESC

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Interruption utilisateur | ERR | boucle interrompue par l'appui de ESC de l'utilisateur |
| | ??? | renvoie le résultat de la dernière commande |

## Exemples:

```
@for(@setvar(vcou,0,NUM),
   @inf(@vcou,@nbcouches),
   @list(
      @if(@equ(@layername(@vcou,""),"PERMIS"),
         @layerinit(@vcou,0,0x3E6),
         @layerinit(@vcou,0x22,0x3C4)
         ),
      @if(@equ(@layername(@vcou,""),"TA"),
         @layerinit(@vcou,0,0x3E6),
         @layerinit(@vcou,0x22,0x3C4)
         )
      ),
   @next(vcou,−1)
   )
```

effectue une boucle : tant que la variable "vcou" créée dans l'initialisation est inférieure au nombre de couche, effectue alors une série d'opération diverses suivant le nom de la couche.
ce qui pourait ce traduire en C :

```
for (int i=0;i<nbcouches;i++)
   {
   liste de commandes
   }
```

```
@for(@setvar(i,0,NUM),
   @inf(@i,@num(30000)),
   @hinttext("valeur=",@i),
   @next(i,−1)
   )
```

effectue un affichage en incrémentant de 0 à 30000

# @foreach

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de variable contenant le nom du fichier parmi les noms de fichiers répondant au masque |
| filemask | STANDARD | masque de recherche de fichier (wildchars format DOS)<br>ex : c:*.map, c:\edigeo??.thf ... |
| dir | STANDARD | indique si on recherche les fichiers(0) ou les sous répertoires(1) |
| action | STANDARD | liste ou commande à executer pour chaque occurence de fichier trouvée |

## Action:

effectue une action pour tous les fichiers ou sous répertoires répondant au masque donné. Pour chaque occurence, "nomvar" reflète alors le nom de fichier ou répertoire complet trouvé.
Il est possible d'interrompre la boucle par ESC

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [filemask] | STR | |

## Exemples:

| | |
|---|---|
| @foreach(mavar,c:*.map,0,@loaddoc(mavar)) | ouvre tous les documents MAP se trouvant dans C:\. "mavar" sert de tampon d'acceil pour les noms de fichiers MAP |

# @format

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| text | STANDARD | texte à formater |
| type | STANDARD | indicateur de ce que représente le texte<br>0 = une chaine de caractère<br>1 = un nombre décimal |
| largeur | STANDARD | largeur de la chaine de sortie |
| precision | STANDARD | *type 0:*<br>0 = supprime les espaces derrière la chaine<br>1 = supprime les espaces devant la chaine<br>2 = supprime les espaces devant et derrière la chaine<br>*type 1:*<br>indique le nombre de décimales à fournir |
| fill | STANDARD | *type 0:*<br><0 = indique le caractère de remplissage (de code = −fill) sur la gauche, lachaine étant cadrée à droite<br>>0 = indique le caractère de remplissage (de code = +fill) sur la droite, lachaine étant cadrée à gauche<br>0 = il n'y a pas de remplissage (largeur n'est pas utilisé)<br>*type 1:*<br>indique le caractère de remplissage (un nombre est cadré à droite)<br>si 0, alors le caractère par défaut est utilisé (espace) |

## Action:

Formatte un texte avec certains critères de cadrage. La largeur n'est qu'indicative et peut être plus petite en sortie si il s'agit d'un texte et que fill=0 ou peut être plus longue si le texte est de plus grande largeur ou si le nombre est plus grand en largeur que le format proposé.
Il n'y a pas de troncature (voir @substr pour cela), mais seulement un arrondi pour les nombres.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <chaine> | STR | chaine de sortie formattée |

## Exemples:

| | |
|---|---|
| @format("121.25",1,5,1,48) | renvoie "0121.3" |
| @format("125873.554",1,5,2,48) | renvoie "125873.55" |
| @format("121.25",1,9,5,48) | renvoie "121.25000" |
| @format("121.25",1,10,5,0) | renvoie " 121.25000" |
| @format(" cou ",0,5,0,−48) | renvoie "00 cou" |
| @format(" cou ",0,5,0,48) | renvoie " cou00" |

# @forwards

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | PT | élement (ou éléments de l'objet) à déplacer dans la liste de priorité d'affichage |
| | LI | |
| elt | FC | |
| | EC | |
| | OBJ | |

## Action:

cette fonction déplace l'élément en fin de liste des éléments à afficher pour la couche, ce qui a pour effet de l'afficher en dernier lors de l'affichage de la couche et donc de le mettre à l'avant plan par rapport à tous les autres éléments de la couche. Pour un objet ce sont les éléments de l'objet qui sont déplacer en fin de leur couche respective

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | l'élément a été déplacé |

## Exemples:

| | |
|---|---|
| @forwards(@elt) | met l'élément à l'avant plan |

# @fusiondblfaces

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection (de 0 à 31) utilisé pour noter les faces étant à fusionner |
| epsilon | STANDARD | écart maxi en position entre les points de la face au dela duquel on doit considérer que les faces ne sont plus identiques |
| objtoo | STANDARD | indique si toutes les faces sont à traiter ou si seulement ne pourront être fusionnées à d'autres faces les faces ne faisant partie d'aucun objet et de même classe si celle ci est fusionnée à une face d'objet. |

## Action:

fusionne toutes les faces semblables (à epsilon près) de la couche "couche" qui sont sélectionnées au niveau "nivsel".
Voici les différents cas de figures dans le cas ou l'option "objtoo" est à 0:

- face objet + face objet => pas de fusion
- face objet + face de même classe => fusion en face objet
- face objet + face de classe différente => pas de fusion
- face + face => fusion

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Traitement des doublons : N doublons ont été traités | STR | N = Nombre de faces doublons traitées |

## Exemples:

| | |
|---|---|
| @fusiondblfaces(0,0,0.1,1) | fusionne les faces sélectionées de la couche 0 à 10cm près |

# @fusiondblobj

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fusionne tous les objets doublons c'est à dire les objets ayant les mêmes éléments.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Traitement des doublons : N objets doublons ont été traités | STR | N = Nombre d'objets doublons traités |

## Exemples:

@fusiondblobj

# @fusionlili

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des liaisons à intersecter (de 0 à 31) |

## Action:

scrute toutes les liaisons de la couche "couche" qui sont sélectionnées au niveau "nivsel" et fait en sorte que chacune des intersections de ces liaisons soient représentées par un point auquelles sont attachées de nouvelles liaisons remplacant les anciennes, autrement dit crée l'intersection de toutes ces liaisons qui se coupent en partageant chaque liaison coupée en deux liaisons (permet de réaliser la topologie de niveau 2).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N points ont été créés et sélectionnés à l'intersection des liaisons | STR | N = Nombre de points créés |

## Exemples:

| | |
|---|---|
| @fusionlili(0,0) | crée les intersections des liaisons sélectionnées de la couche 0 |

# @getapicce

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom/valeur à rechercher | STANDARD | nom ou valeur à rechercher et à convertir |
| conv | STANDARD | type de conversion à faire sous forme de champ binaire:<br>*source:* (indique le type de donnée fournie dans le premier paramètre)<br>0x10 : nom du code etat (ex: "E_CADAS")<br>0x20: valeur binaire du code état (2 puissance 0 à 31, soit 1,2,4,8,16...)<br>0x30: valeur Apic du code etat (dans la table des états : soit 1 à 32)<br>*destination:*<br>0x1 : nom du code etat (ex: "E_CADAS")<br>0x2: valeur binaire du code état (2 puissance 0 à 31, soit 1,2,4,8,16...)<br>0x3: valeur Apic du code etat (dans la table des états : soit 1 à 32) |

## Action:

recherche et fournit un code état sous ses différentes formes.
le paramètre "conv" indique ce que l'on fournit dans le premier paramètre et ce que l'on veut obtenir.
la valeur Apic du code état est en fait un indice basé à 1. La valeur binaire s'obtient par 2 puissance (valeur_apic−1).

Si le code etat n'est pas trouvé, renvoie alors une chaine vide "" de type STR

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Aucun codes états de définis dans la configuration! | ERR | pas de codes états fournis par TopoCad.ini |
| <nom> | STR | nom du code etat renvoyé |
| <valeur> | NUM | valeur du code etat renvoyé |

## Exemples:

| | |
|---|---|
| @getapicce(E_CADAS,0x13) | renvoie la valeur du code etat dans la table Apic soit ici "1" |
| @getapicce(E_ARPEN,0x12) | renvoie la valeur entiere du code etat soit ici "8" |

# @getatt

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément,objet ou masque dont il faut connaitre les attributs |
| | ou | ou |
| elt | PT<br>LI<br>FC<br>EC<br>OBJ<br>MASKCRPT<br>MASKCRLI<br>MASKCRFC<br>MASKCREC<br>MASKMDPT<br>MASKMDLI<br>MASKMDFC<br>MASKMDEC<br>MASKCHPT<br>MASKCHLI<br>MASKCHFC<br>MASKCHEC | élement, objet ou masque |
| op | STANDARD | opérateur & logique à appliquer au résultat<br>les attributs sont:<br>1= Numbered (numéroté, présentation)<br>2= HiddenMono<br>4= HiddenCoul<br>8= Simple<br>0x10= Locked<br>0x100= Italique<br>0x200= Gras<br>0x400= Barre<br>0x800= Souligne<br>0x1000= Opaque<br>0x1000000= AAjouter<br>0x2000000= ASupprimer |

## Action:

Fournit les attributs d'un élément, objet ou masque à travers ou non une variable. S'il s'agit de masque de modification ou recherche fournit les attributs positionnés par le masque de modification ou recherchés par le masque de recherche.

l'opération effectuée est la suivante : (att & op == op ) ? 1 : 0

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <attributs> | NUM | attributs recherchés |

## Exemples:

| | |
|---|---|
| @getatt(obj,0x1000000) | donne l' attribut AAjoutter de l'objet contenu dans la variable "obj" |

# @getbmp2tp

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de couche (de 0 à NbCouches−1) |
| nom | STANDARD | nom de la valeur à extraire sous forme de chaine de caractère:<br><br>a = coef A de la matrice de transfert des coordonnées Bitmap vers coordonnées Terrain/Papier<br>b = coef B de la matrice de transfert des coordonnées Bitmap vers coordonnées Terrain/Papier<br>c = coef C de la matrice de transfert des coordonnées Bitmap vers coordonnées Terrain/Papier<br>d = coef D de la matrice de transfert des coordonnées Bitmap vers coordonnées Terrain/Papier<br>p = coef P de la matrice de transfert des coordonnées Bitmap vers coordonnées Terrain/Papier<br>q = coef Q de la matrice de transfert des coordonnées Bitmap vers coordonnées Terrain/Papier<br>ai = coef AI de la matrice de transfert des coordonnées Terrain/Papier vers coordonnées Bitmap<br>bi = coef BI de la matrice de transfert des coordonnées Terrain/Papier vers coordonnées Bitmap<br>ci = coef CI de la matrice de transfert des coordonnées Terrain/Papier vers coordonnées Bitmap<br>di = coef DI de la matrice de transfert des coordonnées Terrain/Papier vers coordonnées Bitmap<br>pi = coef PI de la matrice de transfert des coordonnées Terrain/Papier vers coordonnées Bitmap<br>qi = coef QI de la matrice de transfert des coordonnées Terrain/Papier vers coordonnées Bitmap<br>h = Hauteur du Bitmap (toujours positif)<br>w = Largeur du Bitmap.<br>dpi = nombre de pixels par pouces (en X et sensé etre identique en Y) |

## Action:

Obtient un paramètre concernant les transfert du Bitmap de la couche "couche"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <valeur> | NUM | valeur recherchée |

## Exemples:

| | |
|---|---|
| @getbmp2tp(2,w) | donne la largeur du bitmap d'origine de la couche 2 |

# @getbmpname

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de couche (de 0 à NbCouches−1) |

## Action:

Fournit le nom du fichier BMP associé à la couche "couche" avec son chemin complet

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <nom fichier> | STR | chemin+nom du fichier |

## Exemples:

| | |
|---|---|
| @getbmpname(2) | donne le nom du BMP (ex: "C:\topocad\doc\essai.bmp") |

# @getcentroide

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | PT<br>LI | élement ou objet |
| eltobj | FC<br>EC<br>OBJ | |
| withdec | BOOL | indique si on doit tenir compte des déports d'écriture ou de l'écriture dans le calcul du centroide |
| type | NUM | indique le type de centroide à calculer :<br>0 = centroide simple (centre du rectangle englobant minimum)<br>1 = centroide complexe (toujours dans la face)<br>2 = centre de gravité (pas toujours dans la face). |

## Action:

fournit le centroide de l'élément ou de l'objet. La fonction échoue si elle ne trouve pas deux variables de noms "Xc" et "Yc"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | les variables "Xc" et "Yc" ont été initialisées avec le centroide |

## Exemples:

```
@getcentroide(@obj,0,1),
@out("X="),
@out(@Xc),
@out("Y="),
@out(@Yc)
```

donne le centroide de l'objet et ecrit ses coordonnées dans le fichiers des messages

# @getchoice

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| prompt | STANDARD | texte devant être affiché sur la boite de dialogue de choix |
| liste | STANDARD | chaine de caractère représentant une liste de choix à opérer séparés par le caractère '\|'<br>ex: "choix 1\|choix 2\|choix 3" |
| ind | STANDARD | indice du choix par défaut (de 0 à ...) |

## Action:

saisie d'un choix donné par l'utilisateur par l'intermédiaire d'une boite de dialogue comportant une liste de choix définis et renvoie le choix effectué de 0 à N.
si une chaine commence par un crochet ouvrant et contient un crochet fermant, alors la chaine contenue dans ces crochets est la chaine de retour du choix de l'option, sinon le retour est l'indice basé à 0 de l'option choisie.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Interruption Utilisateur | ERR | Si on termine la sélection par ESC |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [choix] | STR | renvoie l'option choisie |

## Exemples:

| | |
|---|---|
| @getchoice("Type de Permis SVP ?","Permis de construire\|Permis de démolir\|Déclaration Préalable",0) | selectionne un choix pour le reste du programme. renvoie 0 si on choisi "Permis de construire" |
| @getchoice("Type de Permis SVP ?","[C]Permis de construire\|[D]Permis de démolir\|[DP]Déclaration Préalable",0) | selectionne un, choix parmi les 3 suivants :<br>−Permis de construire<br>−Permis de démolir<br>−Déclaration préalable<br>renvoie "C" si on choisit la première option et "DP" si on choisit la 3° option.<br>NB: "[C]" ne s'affiche pas dans les choix proposés. |

# @getclasse

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément,objet ou masque dont il faut connaitre la classe |
| | ou | ou |
| | PT | élement, objet ou masque |
| | LI | |
| | FC | |
| | EC | |
| | OBJ | |
| | MASKCRPT | |
| elt | MASKCRLI | |
| | MASKCRFC | |
| | MASKCREC | |
| | MASKMDPT | |
| | MASKMDLI | |
| | MASKMDFC | |
| | MASKMDEC | |
| | MASKCHPT | |
| | MASKCHLI | |
| | MASKCHFC | |
| | MASKCHEC | |

## Action:

fourni la classe d'un élément, objet ou masque à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <classe> | NUM | classe recherchée |

## Exemples:

| | |
|---|---|
| @getclasse(obj) | donne la classe de l'objet contenu dans la variable "obj" |

# @getcolor

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD | nom d'une variable contenant l'élément,objet ou masque dont il faut connaitre la couleur |
| | ou | ou |
| | PT<br>LI<br>FC<br>EC<br>MASKCRPT<br>MASKCRLI<br>MASKCRFC<br>MASKCREC<br>MASKMDPT<br>MASKMDLI<br>MASKMDFC<br>MASKMDEC<br>MASKCHPT<br>MASKCHLI<br>MASKCHFC<br>MASKCHEC | élement ou masque |

## Action:

fourni la couleur d'un élément ou masque à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <couleur> | NUM | couleur recherchée |

## Exemples:

| | |
|---|---|
| @getcolor(mafc) | donne la couleur de la face contenue dans la variable "mafc" |

# @getcouche

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD | nom d'une variable contenant l'élément,objet, masque ou relation dont il faut connaitre la couche |
| | ou | ou |
| | PT<br>LI<br>FC<br>EC<br>OBJ<br>MASKCRPT<br>MASKCRLI<br>MASKCRFC<br>MASKCREC<br>MASKMDPT<br>MASKMDLI<br>MASKMDFC<br>MASKMDEC<br>MASKCHPT<br>MASKCHLI<br>MASKCHFC<br>MASKCHEC<br>RELSEM | élement, objet, masque ou relation<br><br>(Les relations n'ayant pas de couche, il est fourni pour ces dernières la couche de l'élément ou objet source de la relation) |

## Action:

fourni la couche d'un élément, objet ou masque à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <couche> | NUM | couche recherchée |

## Exemples:

| | |
|---|---|
| @getcouche(obj) | donne la couche de l'objet contenu dans la variable "obj" |

# @getdata

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom de l'option | STANDARD | nom de l'option à lire |

## Action:

recherche et fournit une option globale  de l'application.

Il peut s'agir du nom des variables globales présentes dans la section DATA du fichier TOPOCAD.INI. Les variables disponibles sont alors:

- PrecisionCoord
- PrecisionAltitude
- PrecisionDistance
- PrecisionAngle
- AngleUnit
- AngleSens
- BmpPath
- DocPath
- AuxPath
- DbaPath
- HlpPath
- CarPath
- TmpPath
- DxfPath
- TopoCadPath
- PreExec
- PostExec
- HauteurNumero
- HauteurDistance
- HauteurSurface
- UnitSurface
- TailleFlecheNord
- CouleurFlecheNord
- ManipCoucheTransf
- ManipCoucheCopie
- ModeSelect
- PtEcVisible
- PtSimpleVisible
- HideClassMode
- SigneMitoyOn
- DeportEcOn
- DrawPolyline
- FormatDates
- ImprHasFNord
- CouleurQuadrillage
- Quadrillage
- HauteurCoordQuadr
- SelVisibleOnPrint
- DefFormatPage
- PointCache
- ModeReelVision
- EspactHachMonochrome
- AngleHachMonochrome
- CAPointProlLi
- CopieSymForCut
- CAInserePtDsLi
- PrecisionSurface
- NXYUser
- NXYWithLabel

- LOCUser
- VisionRelSem
- OrdreInterpolGravitaire
- FormatCarnetListe (liste des formats de carnet disponibles séparés par "|")
- NoAutoAddField
- Tmw (TopoCad Mouse Wheel : cf fonctions de zoom)
- MethodeCodif

Il peut s'agir des noms suivants (qui parlent d'eux−mêmes):

- LargeurIdent
- LargeurCoord
- LargeurAltitude
- LargeurDistance
- LargeurAngle
- PrecisionAngle
- PCIConfigLot
- EcrNumero
- IncrEcrNumero
- LastNum (propre au document)
- NbMaskCrPt
- NbMaskMdPt
- NbMaskChPt
- NbMaskCrLi
- NbMaskMdLi
- NbMaskChLi
- NbMaskCrFc
- NbMaskMdFc
- NbMaskChFc
- NbMaskCrEc
- NbMaskMdEc
- NbMaskChEc

Il peut s'agir de "InfoLabel" : cette variable interne indique si l'affichage des numeros de points, des distances de laisons, de surfaces de faces est substitué par l'affichage des étiquettes de ceux ci. Cette variable interne est toujours à 0 au démarrage de l'application (pas d'affichage d'étiquettes). L'accélérateur CTL+ALT+I permet de switcher d'un mode à l'autre pour un type d'élément en fonction du mode courant (Pt,Li,Fc).

Il peut s'agir de "FormatPoints" qui fournit alors le format utilisateur de sortie des points (6° format dans la section FORMATS_POINTS du fichier de configuration TOPOCAD.INI).

Il peut s'agir de "FormatEcritures" qui fournit alors le format utilisateur de sortie des écritures (6° format dans la section FORMATS_ECRITURES du fichier de configuration TOPOCAD.INI). Dans ce cas la chaine de format est fournie.

Il peut s'agir de "UserHelp" indiquant le nom du module en cours (nom du fichier TED de configuration sans chemin ni extension).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <parametre> | ??? | fournit la propriété. |

## Exemples:

| | |
|---|---|
| @getdata(PrecisionCoordonnee) | donne la precision des coord sous forme d'entier |

# @getdbprop

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable de type OBJ |
| obj | ou | ou |
| | OBJ | objet dont il faut extraire la propriété |
| prop | STANDARD | nom de la propriété à extraire |
| strict | STANDARD | 1 = renvoie une erreur si la propriété n'est pas trouvée<br>0 = renvoie "<AUCUNE_PROPRIETE>" si la propriété n'est pas trouvée |

## Action:

fournit la propriété désignée d'un objet. Même si strict=1, la fonction peut renvoyer des erreurs sous forme "<erreur...>" et donc sans blocage, il est donc nécessaire de tester toujours la donnée recue.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de calculer l'identifiant de l'objet | ERR | une propriété d'un objet dont le type ne peut s'identifier est demandé ou l'objet n'a pas suffisamment d'infos pour calculer son identifiant (n° insee manquant ...) |
| Impossible d'obtenir la propriété de l'objet | ERR | aucune base n'est ouverte ou l'objet n'a pas de propriété ou pas la propriété demandée |
| <propriété> | STR | propriété renvoyée |

## Exemples:

| | |
|---|---|
| @getdbprop(monobj,surface) | fournit la propriété de nom "surface" de l'objet contenu dans la variable "monobj" |

# @getdbpropwithid

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| idu | STANDARD | identifiant de l'objet dont il faut acquérir la propriété |
| prop | STANDARD | nom de la propriété à extraire |
| strict | STANDARD | 1 = renvoie une erreur si impossible d'acquerir la propriété<br>0 = renvoie "<AUCUNE_PROPRIETE>" si la propriété n'est pas trouvée |

## Action:

fournit la propriété désignée d'un objet d'identifiant "idu"
(même si strict=1, la fonction peut renvoyer des erreurs sous forme "<erreur...>" et donc sans blocage, il est donc nécessaire de tester toujours la donnée recue)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'obtenir la propriété de l'objet d'IDU xxx | ERR | aucune base n'est ouverte ou l'objet n'a pas de propriété ou pas la propriété demandée |
| <propriété> | STR | propriété renvoyée |

## Exemples:

| | |
|---|---|
| @getdbpropwithid(idu,surface) | fournit la propriété de nom "surface" de l'objet d'identifiant contenu dans la variable "idu" |

# @getdestli

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| li | STANDARD ou LI | liaison dont il faut extraire le point destination |

## Action:

fournit le point destination de la liaison

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <srce> | PT | renvoie le point destination de la liaison |

## Exemples:

| | |
|---|---|
| @getdestli(vli) | fournit le point destination de la liaison |

# @getdestrelsem

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable (de type quelconque) accueillant la donnée |
| rel | RELSEM | relation sémantique dont il faut extraire une donnée |

## Action:

fournit la destination de la relation (la variable est alors du type OBJ, PT, LI, FC ou EC)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | renvoie OK si la variable est affectée |

## Exemples:

```
@setvar(vrel,@firstrelsem(@vpt,10),RELSEM),
@while(@vrel,
  @list(
    @getdestrelsem(var,@vrel),
    @out(@var),
    @setvar(vrel,@nextrelsem(@vpt,10,@vrel),RELSEM)
    )
  ),
```

liste les différents éléments ayant une relation avec le point dans la variable vpt

# @getdoc

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| file | STANDARD | nom du fichier MAP à chercher |

## Action:

Recherche un document MAP ouvert par TopoCad et le fournit si il est chargé par l'application, sinon renvoie un pointeur NUL.
si une chaine vide est fournie en entrée, TopoCad renvoie le document MAP courant.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <document> | MAPDOC | document trouvé ou NUL |

## Exemples:

| | |
|---|---|
| @getdoc(c:\topocad\doc\test.map) | renvoie une variable de type pointeur MAPDOC sur le document présent dans l'application |
| @getdoc("") | renvoie une variable de type pointeur MAPDOC ayant la valeur du document courant. |
| @getdoc(" ") | idem |

# @getdocname

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| mapdoc | MAPDOC | document MAPDOC dont le nom est à rechercher |
| withpath | STANDARD | indique si l'on recherche le nom complet avec chemin (1) ou seulement le nom du fichier (0) |

## Action:

Recherche le nom du fichier associé à un document MAP ouvert par TopoCad et le fournit avec le chemin si withpath=1 ou sans le chemin si withpath=0. Si aucun fichier n'est associé (dans le cas de nouveau document venant d'être créé et non sauvegardé), une chaine vide est renvoyée.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <file_name> | STR | document trouvé ou chaine nulle en cas de nouveau document |

## Exemples:

| | |
|---|---|
| @getdocname(@getdoc(""),1) | renvoie "W:\COMMUNES\101.MAP" c'est à dire le chemin du document courant qui est en cours d'édition |

# @getelementat

*Nb paramètres*
8

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| X | STANDARD | coordonnée X du point de recherche |
| Y | STANDARD | coordonnée Y du point de recherche |
| couche | STANDARD | couche de l'élément à rechercher ou −1 si ttes les couches |
| classe | STANDARD | classe de l'élément à rechercher ou −1 si ttes les classes |
| sel | STANDARD | niveau de sélection des éléments à considérer (ou −1 si tous les éléments) |
| telt | STANDARD | type d'élément à rechercher (1=point,2=liaisons,16=faces,4=écritures) |
| eps | STANDARD | rayon de recherche |
| ordre | STANDARD | si une liste d'éléments satisfait aux conditions, alors ordre indique l'indice de l'élément dans la liste (de 0 à N−1 éléments). La liste est triée par ordre croissant des distances au point de référence. |

## Action:

fournit l'élément satisfaisant aux conditions de proximités définies. Si plusieurs éléments satisfont aux propriétés, alors fournit l'élément d'ordre 'ordre' dans la liste des éléments trouvés. Pour rechercher UN élément, il est donc nécessaire de mettre ordre=0.
Si on désire rechercher le plus proche élément alors il faut fournir eps < 0 et ordre=0.
si aucun élément n'est trouvé alors renvoie un élément NUL.
la fonction renvoie également un pointeur nul si l'élément d'indice 1 est demandé (2° élément) et qu'un seul élément est trouvé.
En cas de recherche d'écriture c'est la position du déport d'écriture qui est considéré s'il existe plutot que l'écriture.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <élément> | PT, LI, FC ou EC | l'élément renvoyé ou 0 (pointeur nul) si aucun élément de cet ordre trouvé |

## Exemples:

| | |
|---|---|
| @getelementat(786000,126000,0,5,−1,4,−1,0) | fournit l'écriture de classe parcelle dont le point d'insertion ou le déport se trouve le plus près de la position x=786000 et y=126000 et qui est sur la couche 0. |

# @geteltdata

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément |
| elt | ou | ou |
| | PT, LI, FC, EC | élément dont une donnée est à acquérir |
| name | STANDARD | nom de la donnée à acquèrir (composé de caractères alphanumériques et du signe '_' (underscore) |

## Action:

fournit la donnée propre à un élément.
chaque élément possède une collection de données qui lui sont propre et qui meurent et vivent avec lui. Une fusion de deux éléments entraine irrémédiablement la disparition des données d'un des deux éléments fusionné. Ces données sont donc utilisées soit de manière temporaire soit dans le cadre d'une gestion statique du plan (ex: navigateur pour lequel les données ne changent pas).
Les données ont des types divers mais seront toujours récupérées sous forme de chaine de caractère.

Si name == #nb alors la fonction renvoie le nombre de données extra présentes dans l'élément
Si name == #X avec X étant un nombre de 0 à (nb−1), alors la fonction renvoie le nom de la donnée extra suivie de son type (séparé par la barre verticale) :
les types S, I, R correspondant aux type chaines de caractères, entiers et réels sont les seuls accessibles.
les types P,L,F,E correspondant aux types pointeurs sur point, liaison, face, écritures ne sont pas accessibles (usage interne volatile).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <data> | STR | donnée de l'élémént |
| "<Absent>" | STR | donnée inexistante dans cet élément |
| <nb> | NUM | nombre de données extra de l'élément |
| "<nom>\|<type>" | STR | nom de la donnée extra suivi de son type. |

## Exemples:

| | |
|---|---|
| @geteltdata(elt,debit) | fournit une donnée d'un élément dans un graphe pour calcul de flux |
| @geteltdata(elt,#nb) | renvoie le nombre de données extra (y compris données internes) |
| @geteltdata(elt,#0) | renvoie le nom de la première donnée extra de l'élément |
| @geteltdata(elt,#1) | renvoie le nom de la seconde donnée extra de l'élément |

# @getenv

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom | STANDARD | nom de la variable d'environnement à rechercher |

## Action:

recherche la valeur d' une variable d'environnement.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Var...trop longue | ERR | si la variable d'environnement est trop longue |
| <valeur> | STR | renvoie la chaine valeur de la variable d'environnement (ou une chaine vide si la variable d'environnement n'existe pas) |

## Exemples:

| | |
|---|---|
| @getenv(communespath) | renvoie "d:\communes" |
| @getenv(ignpath) | renvoie "d:\ign" |

# @getepaisseur

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD | nom d'une variable contenant l'élément,objet ou masque dont il faut connaitre l'épaisseur |
| | ou | ou |
| | PT<br>LI<br>FC<br>EC<br>MASKCRPT<br>MASKCRLI<br>MASKCRFC<br>MASKCREC<br>MASKMDPT<br>MASKMDLI<br>MASKMDFC<br>MASKMDEC<br>MASKCHPT<br>MASKCHLI<br>MASKCHFC<br>MASKCHEC | élement ou masque |

## Action:

fourni l'épaisseur d'un élément ou masque à travers ou non une variable (en 1/100° de mm)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <épaisseur> | NUM | épaissseur recherchée |

## Exemples:

| | |
|---|---|
| @getepaisseur(mafc) | donne l'épaisseur de la face contenue dans la variable "mafc" |

# @getextents

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche concernée ou (−1 si toutes les couches actives) |
| sel | STANDARD | niveau de sélection des éléments vectoriels dont on veut connaitre les limites (−1 si pas de niveau de sélection) |
| vect_rast | STANDARD | champ de bit déterminant le type d'étendue souhaitée:<br>0x01 = étendue vectorielle<br>0x10 = étendue raster |

## Action:

fournit dans 4 variables "Xmin", "Xmax", "Ymin" et "Ymax" l'étendue des éléments vectoriels (sélectionnés) et/ou des rasters de la couche/des couches actives.
si vect_rast valide l'étendue vectorielle, sont pris en compte alors tous les éléments (sélectionnés si sel différent de −1) de la ou des couches actives
si vect_rast valide l'étendue raster, sont pris en compte les rasters sur les couches actives dont le bitmap est validé (drapeau Bmp est ON)
La fonction échoue si elle ne trouve pas les variables.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Affectation impossible : types incompatibles | ERR | problème à l'écriture d'une variable : une variable de même nom existe et de type différent |
| 0 ou 1 | BOOL | 0 si aucun élément trouvé (les variables sont toutes à 0)<br>1 si les variables contiennent les mini et maxi de la sélection |

## Exemples:

| | |
|---|---|
| @getextents(0,0,1) | fournit l'étendue de la sélection au niveau 0 de la couche 0 |

# @getforme

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD | nom d'une variable contenant l'élément ou masque dont il faut connaitre la forme |
| | ou | ou |
| | PT<br>LI<br>FC<br>MASKCRPT<br>MASKCRLI<br>MASKCRFC<br>MASKMDPT<br>MASKMDLI<br>MASKMDFC<br>MASKCHPT<br>MASKCHLI<br>MASKCHFC | élement ou masque |

## Action:

fournit la forme d'un élément ou masque à travers ou non le nom d'une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <forme> | NUM | forme cherchée (entier) |

## Exemples:

| | |
|---|---|
| @getforme(monpt) | founit la forme du point contenu dans la variable "monpt" |

# @getidalgo

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |

## Action:

fournit l'IdAlgo (algorithme de calcul de l'identifiant) d'un type d'objet c'est à dire un entier dont les bits ont les significations suivantes:
bit 0 = IDALGO_COMMUNE = code insee
bit 1 = IDALGO_PREFSECT = préfixe section
bit 2 = IDALGO_SECTION = section
bit 3 = IDALGO_SUBDSECT = subdiv section
bit 4 = IDALGO_PARCELLE = parcelle
bit 5 = IDALGO_TEXT = unique texte de l'objet
bit 6 = IDALGO_LABEL = étiquette de l'objet
bit 7 = IDALGO_ID = numéro de l'objet
bit 8 = IDALGO_XDATA = valeur de la donnée extra de nom "IDENT" de l'élément directeur de l'objet
on a également
0 = pas d'identifiant

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <idalgo> | NUM | |

## Exemples:

| | |
|---|---|
| @getidalgo(5) | renvoie 0x27 = 39 = 0010 0111 = codeinsee+prefsect+section+texte de l'objet |

# @getidparm

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |

## Action:

renvoie l'IdParm du type d'objet désigné c'est à dire une valeur entière ayant une signification diverse suivant la classe auquel il s'aplique, à savoir :

Points de canevas :
N = largeur de numéro−nbre de digits
Commune, Section, subd sect :(la parcelle est tjrs recherchée par appartenance)
donne la manière dont la composante concernant l'objet des autres types d'objets est calculée
0=IDPARM_FROMDOC = les infos du documents sont prises pour réf l'objet
1=IDPARM_FROMCOUCHE = le nom de la couche est pris pour réf l'objet
2=IDPARM_FROMOBJ = recherche de l'appartenance faite pour réf l'objet
*combinaisons pour la méthode de recherche de l'objet d'appartenance*
IdParm des classes :

| COMMUNE | SECTION | SUBDSECT | NOM DE LA COUCHE |
|---|---|---|---|
| 0 | 0 | 0 | <non utilisé> |
| 0 | 0 | 1 | F ou FF |
| 0 | 1 | 0 | S ou SS ou SSSSS |
| 0 | 1 | 1 | SSSSSF ou SSSSSFF |
| 1 | 0 | 0 | CCC |
| 1 | 0 | 1 | CCCF ou CCCFF |
| 1 | 1 | 0 | CCCS ou CCCSS ou CCCSSSSS |
| 1 | 1 | 1 | CCCSSSSSF ou CCCSSSSSFF |

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <IdParm> | NUM | |

## Exemples:

| | |
|---|---|
| @getidparm(2) | renvoie IdParm des sections |

# @getidu

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom de la variable contenant l'objet dont déterminer l'identifiant |
| obj | ou | ou |
| | OBJ | objet dont déterminer l'identifiant |
| strict | STANDARD | indique si renvoie une erreur (1) ou une chaine (0) en cas de calcul impossible de l'identifiant |

## Action:

obtient l'identifiant de l'objet

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de calculer l'identifiant de l'objet n°NNN | STR<br>ERR | impossibilité de calculer l'identifiant (renvoie ERR si strict sinon renvoie STR) |
| <Identifiant> | STR | identifiant avec préfixe (PARC pour parcelles ...etc) |

## Exemples:

| | |
|---|---|
| @getidu(@varobj) | fournit l'identifiant de l'objet contenu dans la variable de nom "varobj" |

# @getinit

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe dont on cherche la valeur Init |
| masque | STANDARD | masque permettant d'isoler les valeurs à chercher :<br>valeur sous forme de champs de bits :<br>bit 5 = 0x20 = Actif<br>bit 4 = 0x10 = Element Face visible<br>bit 2 = 0x04 = Element Ecriture visible<br>bit 1 = 0x02 = Element Liaison visible<br>bit 0 = 0x01 = Element Point visible |

## Action:

fournit la valeur Init de la classe ou une partie suivant le masque. Cette valeur indique si la classe est active ou inactive, ou si des types d'élments de la classe sont invisibles

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Init> | NUM | Init de la classe |

## Exemples:

| | |
|---|---|
| @getinit(5,−1) | fournit la valeur Init de la classe |
| @getinit(5,0x20) | indique si la classe est active ou pas |
| @getinit(5,0x10) | inidque si les faces de la classe sont invisibles ou non pour les visions polychromes |

# @getlabel

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément ou objet |
| eltobj | ou | ou |
| | PT, LI, FC, EC, OBJ | élément ou objet dont l'étiquette est à acquérir |

## Action:

fournit l'étiquette d'un élément ou d'un objet

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <label> | STR | Etiquette de l'élémént ou l'objet |

## Exemples:

| | |
|---|---|
| @getlabel(elt) | fournit l'étiquette de l'élément contenu dans la variable "elt" |

# @getlayerinit

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| no | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| masque | STANDARD | valeur du masque permettant d'isoler l' option à rechercher sous forme de champ de bit avec chaque bit signifiant :<br>• bit 1 = 0x0002 = Visible<br>• bit 2 = 0x0004 = Monochrome<br>• bit 5 = 0x0020 = Actif<br>• bit 6 = 0x0040 = Bitmap<br>• bit 7 = 0x0080 = BmpTransparent<br>• bit 8 = 0x0100 = BmpAll<br>• bit 9 = 0x0200 = BmpBkTransparent<br>• bit 10 = 0x0400 = BmpMonoNoPreserv |

## Action:

Permet de connaitre la valeur des données d'une couche de la fenetre courante

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <valeur> | NUM | |

## Exemples:

| | |
|---|---|
| @getlayerinit(0,0x02) | indique si la couche 0 est visible |
| @getlayerinit(0,0x0100) | indique si la couche 0 n'a pas validé la découpe de la couche |

# @getline

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom d'un fichier texte |
| varname | STANDARD | nom de la variable à initialiser avec la ligne recherchée |
| no | STANDARD | numéro de la ligne (de 0 à ...) |

## Action:

extrait la ligne en position "no" d'un fichier de nom "filename". 0 est la première ligne, 1 la seconde ligne du fichier ...etc.
la ligne est copiée dans la variable de nom "varname" si la fonction n'échoue pas.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 ou 1 | BOOL | 1 => la ligne a été copiée<br>0 => index non valide (fichier comportant moins de 'no' lignes...) |

## Exemples:

| | |
|---|---|
| @getline(c:\batch.txt,mavar,@ind) | fournit la ligne du fichier c:\batch.txt qui est en position donnée par la variable de nom "ind" |

# @getmask

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de masque (de point, liaison, face ou écriture, de recherche, de modification, ou de création : ex MASKCHPT) |
| classe | STANDARD | classe |

## Action:

fournit le masque de type donné pour la classe "classe".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <masque> | MASK... | renvoie le masque de classe |

## Exemples:

| | |
|---|---|
| @getmask(MASKCHFC,5) | fournit le masque de recherche de face pour la classe parcelle |

# @getmask2str

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de masque (de point, liaison, face ou écriture, de recherche, de modification, ou de création : ex MASKCHPT) |
| classe | STANDARD | classe |

## Action:

fournit le masque de type donné pour la classe "classe" sous forme de chaine de caractère de description du masque.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <masque> | STR | renvoie le masque de classe |

## Exemples:

| | |
|---|---|
| @getmask2str(MASKCHFC,5) | fournit le masque de recherche de face pour la classe parcelle |

# @getmaskno

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de masque (de point, liaison, face ou écriture, de recherche, de modification, ou de création : ex MASKCHPT) |
| no | STANDARD | indice du masque (de 0 à NbMask...−1) |

## Action:

fournit le masque de type donné d'indice no

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <masque> | MASK... | renvoie le masque |

## Exemples:

| | |
|---|---|
| @getmaskno(MASKCHFC,0) | fournit le premier masque de recherche de face |

# @getmaskno2str

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de masque (de point, liaison, face ou écriture, de recherche, de modification, ou de création : ex MASKCHPT) |
| indice | STANDARD | indice du masque (de 0 à ...) |

## Action:

fournit le masque de type "type" d'indice "indice" dans la table des masques de type "type" et ceci sous forme de chaine de caractère décrivant le masque

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <masque> | STR | renvoie le masque voulu |

## Exemples:

| | |
|---|---|
| @getmaskno2str(MASKCHFC,0) | fournit le premier masque de recherche de face. |

# @getnum

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD<br>ou<br>PT<br>OBJ | nom d'une variable contenant l'élément ou l'objet dont il faut connaitre le numéro<br>ou<br>élement ou objet |

## Action:

fourni le numéro (unique) d'un élément ou objet à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <num> | NUM | numéro recherché |

## Exemples:

| | |
|---|---|
| @getnum(@monpt) | donne le numéro du point |
| @getnum(monpt) | idem |

# @getnumobservation

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation dont le numero de point est à extraire |
| opt | STD | options:<br>0 = pt est le point résultat de l'observation<br>1 = pt est la station de l'observation<br>2 = pt est la référence de l'observation<br>3 = la remarque de l'observation<br>11 à ... = valeur propre à l'observation (correspondant aux valeurs fournies dans la boite de dialogue) |

## Action:

fournit le numero de point relatif à une observation ou une valeur d'un paramètre de l'observation
ex: le rayonnement peut fournir ptA(11),ptB(12 ou 2),ptC(13 ou 1),Dcx(14),AngBAX(15), et evidemment le résultat (0) et la remarque(3)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <numpt> | STR | numero du point, distance, angle, valeur recherché |

## Exemples:

| | |
|---|---|
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getnumobservation(@curobs,0) | fournit le numero 37320 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getnumobservation(@curobs,1) | fournit le numero 37318 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getnumobservation(@curobs,2) | fournit le numero 37318 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37318,B=37319,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getnumobservation(@curobs,2) | fournit le numero 37319 |

# @getobjet

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| num | STANDARD | numéro de l'objet à rechercher |

## Action:

fournit l'objet de numéro "num"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <objet> | OBJ | l'objet renvoyé ou 0 (pointeur nul) si aucun objet de ce numéro trouvé |

## Exemples:

| | |
|---|---|
| @getobjet(125) | fournit l'objet de numéro 125 |

# @getobjetat

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| X | STANDARD | coordonnée X du point de recherche |
| Y | STANDARD | coordonnée Y du point de recherche |
| couche | STANDARD | couche de l'objet à rechercher ou −1 si ttes les couches |
| classe | STANDARD | classe de l'objet à rechercher |
| sel | STANDARD | niveau de sélection des éléments à considérer (ou −1 si tous les éléments) |
| eps | STANDARD | rayon de recherche |
| ordre | STANDARD | si une liste d'objets satisfait aux conditions, alors ordre indique l'indice de l'objet dans la liste (de 0 à N−1 objets). La liste est triée par ordre croissant des distances au point de référence. |

## Action:

fournit l'objet satisfaisant aux conditions de proximités définies. Si plusieurs objets satisfont aux propriétés, alors fournit l'objet d'ordre 'ordre' dans la liste des objets trouvés. Pour rechercher UN objet, il est donc nécessaire de mettre ordre=0.
Si on désire rechercher le plus proche objet alors il faut fournir eps < 0.
si aucun objet n'est trouvé alors renvoie un objet NUL.
la fonction renvoie également un pointeur nul si l'objet d'indice 1 est demandé (2° objet) et qu'un seul objet est trouvé.
En cas de recherche d'objets écriture c'est la position du déport d'écriture qui est considéré s'il existe plutot que l'écriture.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <objet> | OBJ | l'objet renvoyé ou 0 (pointeur nul) si aucun objet de cet ordre trouvé |

## Exemples:

| | |
|---|---|
| @getobjetat(786000,126000,0,5,−1,0,0) | fournit la parcelle se trouvant à la position x=786000 et y=126000 et sur la couche 0. |

# @getobjetwithid

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| idu | STANDARD | identifiant complet de l'objet à rechercher avec préfixe (pour une parcelle sous forme PARC... etc) |

## Action:

fournit un objet à partir d'un identifiant complet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <objet> | OBJ | l'objet recherché ou 0 (pointeur nul) si non trouvé |

## Exemples:

| | |
|---|---|
| @getobjetwithid(PARC298000AB0524) | fournit l'objet parcelle d'identifiant donné |

# @getpoint

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| num | STANDARD | numéro du point à rechercher |

## Action:

fournit le point de numéro "num" ou NULL si le point de numéro "num" n'existe pas dans le document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <point> | PT | le point renvoyé ou 0 (pointeur nul) si aucun point de ce numéro trouvé |

## Exemples:

| | |
|---|---|
| @getpoint(125) | fournit le point de numéro 125 |

# @getprid

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |

## Action:

renvoie le préfixe d'identifiant de 4 caractères de la classe fournie.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <IdParm> | STR | chaine de 4 caractères |

## Exemples:

| | |
|---|---|
| @getprid(5) | renvoie IdParm des parcelles soit "PARC" |

# @getproj

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| code | STANDARD | nom du système de coordonnée sous forme de code conforme à la norme Edigeo (zonegeo) |

## Action:

Fournit la définition du système de coordonnée dont le code edigeo est "code".
Ce code edigeo doit être répertorié par l'application (dans la liste des systèmes géographiques utilisables).
La définition du système est fournie sous forme de definition suivant le standard proj4
Si code est nul (chaine vide), la définition de la projection courante du document sera fournie.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si code edigeo inconnu ou le système de projection du document courant est indéfini. |
| <chaine_def> | STR | chaine de définition du système géographique utilisé. |

## Exemples:

| | |
|---|---|
| @getproj("") | fournit la définition du système courant soit "+proj=lcc +ellps=clrk80IGN +towgs84=0,0,0 +x_0=600000 +y_0=200000 +lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859 +lat_2=44d59'45.93773" le document ayant comme zonegeo "LAMB3" |
| @getproj(LAMB93) | fournit la définition du système Lambert 93 soit "Lambert 93" "+proj=lcc +ellps=GRS80 +grids=c:\topocad4\debug\exe\ign.dat +x_0=700000 +y_0=6600000 +lon_0=3d +lat_0=46d30' +lat_1=44d +lat_2=49d" |
| @getproj(UTM31) | erreur : le sytème n'étant pas définit dans la configuration standard de l'application. |

# @getpropinfo

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| propname | STANDARD | nom d'une propriété à rechercher |
| db | STANDARD | indice de la base de donnée concernée (sous forme symbolique ou non) pour une info sur un champ<br>prefixe du type d'objet concerné pour une info sur une règle de propriété |
| infoname | STANDARD | nom de l'info à rechercher. Peut être :<br>fieldlen = longueur du champ<br>fieldtype = type du champ (renvoie une lettre majuscule)<br>fielddecimal = nombre de décimale<br><br>ruleclasse = classe de la propriété (0 si pas de classe)<br>ruletype= type de la règle de propriété (sous forme de chaine de caractère décrivant le type)<br>ruledefvalue= valeur par défaut de la propriété |

## Action:

fournit une information sur la propriété désignée.
NB: on peut tester également l'existence du champ en demandant "fieldlen" : si le champ n'existe pas, alors renverra une chaine vide (soit 0)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <propriété> | STR | propriété renvoyée |

## Exemples:

| | |
|---|---|
| @getpropinfo(surface,C5,fielddecimal) | donne le nombre de décimales de la propriété surface pour les objets de classe parcelles (5) |

# @getptobservation

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation dont le point est à extraire |
| opt | STD | options:<br>0 = pt est le point résultat de l'observation<br>1 = pt est la station de l'observation (Rayonnt,Visee,Mesurage)<br>2 = pt est la référence de l'observation (Rayonnt,Visee,Mesurage) |

## Action:

fournit le point relatif à une observation ou NUL si le point n'existe pas (par ex observation non encore calculée ou une référence de mesurage)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <pt> | PT | point cherché fourni |
| 0 | PT | point recherché non trouvé |

## Exemples:

| | |
|---|---|
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getptobservation(@curobs,0) | fournit le point 37320 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getptobservation(@curobs,1) | fournit le point 37318 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getptobservation(@curobs,2) | fournit le point 37318 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37318,B=37319,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @getptobservation(@curobs,2) | fournit le point 37319 |

# @getrelsem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | numéro d'ordre de la relation dans l'ensemble du document (de 0 à NbRelSem−1) |

## Action:

fourni la relation d'indice "ind" du document.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés (hors limite) |
| <relation> | RELSEM | renvoie la relation |

## Exemples:

| | |
|---|---|
| @getrelsem(0) | fournit la premiere relation sémantique du document |

# @getselect

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | niveau de sélection à rechercher |
| | STANDARD | nom d'une variable contenant l'élément,objet, ou relation dont il faut connaitre la sélection |
| | ou | ou |
| elt | PT<br>LI<br>FC<br>EC<br>OBJ<br>RELSEM | élement, objet, relation |

## Action:

fournit la sélection au niveau demandé d'un élément ou objet à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 ou 1 | BOOL | indique la sélection |

## Exemples:

| | |
|---|---|
| @getselect(0,obj) | donne la sélection (au niveau 0) de l'objet contenu dans la variable "obj" |

# @getselectcou

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| select | STANDARD | niveau de selection (de 0 à 31) |
| couche | STANDARD | numero de la couche dont il faut connaitre la selection courante (de 0 à NbCouches−1) |

## Action:

fournit la sélection au niveau courant d'une couche dont le numero est fourni

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 ou 1 | BOOL | indique la sélection |

## Exemples:

| | |
|---|---|
| @getselectcou(0,0) | donne la sélection au niveau de sélection 0 de la couche 0 |

# @getsens

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| li | LI | liaison à tester |
| fc | FC | face à tester |

## Action:

indique la relation de la liaison à la face

0 = pas de relation
1 = la face est à droite de la liaison
−1 = la face est à gauche de la liaison

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <1 ou 0 ou −1> | NUM | valeur indiquant le sens |

## Exemples:

| | |
|---|---|
| @getsens(@vli,@vfc) | fourni le sens de la relation de vli à vfc |

# @getsrceli

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| li | STANDARD ou LI | liaison dont il faut extraire le point source |

## Action:

fournit le point source de la liaison

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <srce> | PT | renvoie le point source de la liaison |

## Exemples:

| | |
|---|---|
| @getsrceli(vli) | fournit le point source de la liaison |

# @getsrcerelsem

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable (de type quelconque) accueillant la donnée |
| rel | RELSEM | relation sémantique dont il faut extraire une donnée |

## Action:

fournit la source de la relation (la variable est alors du type OBJ, PT, LI, FC ou EC)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | renvoie OK si la variable est affectée |

## Exemples:

| | |
|---|---|
| @setvar(var,0,PLANVIEW),<br>@getsrcerelsem(var,@rel) | fournit la source de la relation contenue dans la variable "rel". var peut être de type quelconque car le type sera établi par le résultat de la recherche. |

# @gettext

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément dont il faut connaitre le texte |
| elt | ou | ou |
| | EC | élement écriture |

## Action:

fourni le texte d'un élément écriture à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <texte> | STR | texte recherché |

## Exemples:

| | |
|---|---|
| @gettext(@varec) | donne le texte de l'écriture contenue dans 'varec' |

# @gettobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe (indice) du type d'objet |
| nom parametre | STANDARD | nom du paramètre à rechercher |

## Action:

fournit une propriété du type d'objet de classe "classe". Cette fonction renvoie une erreur si le nom de la propriété est non valide. Les propriétés du type d'objet sont :

- Nom = nom en clair de la classe
- NbFcMin = nombre mini de face de ce type d'objet
- NbLiMin = nombre mini de liaisons de ce type d'objet
- NbPtMin = nombre mini de points de ce type d'objet
- NbEcMin = nombre mini d'écritures de ce type d'objet
- NbFcMax = nombre maxi de face de ce type d'objet
- NbLiMax = nombre maxi de liaisons de ce type d'objet
- NbPtMax = nombre maxi de points de ce type d'objet
- NbEcMax = nombre maxi d'écritures de ce type d'objet
- FcDesPriority = priorité de dessin des faces de cette classe
- LiDesPriority = priorité de dessin des liaisons de cette classe
- PtDesPriority = priorité de dessin des points de cette classe
- EcDesPriority = priorité de dessin des écritures de cette classe
- MaskCrFc = numéro d'ordre du masque de création de face pour cette classe
- MaskCrLi = numéro d'ordre du masque de création de liaison pour cette classe
- MaskCrPt = numéro d'ordre du masque de création de point pour cette classe
- MaskCrEc = numéro d'ordre du masque de création d'écriture pour cette classe
- MaskMdFc = numéro d'ordre du masque de modification de face pour cette classe
- MaskMdLi = numéro d'ordre du masque de modification de liaison pour cette classe
- MaskMdPt = numéro d'ordre du masque de modification de point pour cette classe
- MaskMdEc = numéro d'ordre du masque de modification d'écriture pour cette classe
- MaskChFc = numéro d'ordre du masque de recherche de face pour cette classe
- MaskChLi = numéro d'ordre du masque de recherche de liaison pour cette classe
- MaskChPt = numéro d'ordre du masque de recherche de point pour cette classe
- MaskChEc = numéro d'ordre du masque de recherche d'écriture pour cette classe
- Select = Elément de sélection de cette classe (1=point,2=liaison,4=écriture,16=face)
- Nature = Nature des objets de cette classe (1=point,2=liaison,4=écriture,16=face)
- AutoDetect = Indique si détection automatique de l'objet (1=oui, 0=non)
- DefCouche = couche par défaut pour ce type d'objet.
- Topology = réservé
- CouleurFc = couleur sous forme 0x00BBGGRR des faces de ce type d'objet (en visions polychromes)
- CouleurLi = couleur sous forme 0x00BBGGRR des liaisons de ce type d'objet (en visions polychromes)
- CouleurPt = couleur sous forme 0x00BBGGRR des points de ce type d'objet (en visions polychromes)
- CouleurEc = couleur sous forme 0x00BBGGRR des écritures de ce type d'objet (en visions polychromes)
  *NB: une couleur à −1 indique pas de tracé (la valeur stockée dans le fichier INI est alors 0xFF000000)*
- IdAlgo = algorithme de calcul de l'identifiant : la commande est alors identique à GetIdAlgo
- IdParm = paramètre pour le calcul de l'identifiant : la commande est alors identique à GetIdParm
- RayonDetect = rayon de détection en cas de construction automatique de l'objet
- PrId = Préfixe d'identifiant : la commande est alors identique à GetPrId.

La valeur Init du type d'objet peut être lue à partir de la fonction @getinit

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <valeur_parametre> | STR<br>NUM | fournit la valeur du paramètre |

## Exemples:

| | |
|---|---|
| @gettobj(5,CouleurFc) | renvoie la couleur des faces de parcelles |
| @gettobj(12,NbEcMin) | renvoie le nombre d'écriture minimal des objets de type "voie publique" |

# @gettrel

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| trelsem | STANDARD | classe (indice) du type de relation sémantique |
| nom parametre | STANDARD | nom du paramètre à rechercher |

## Action:

fournit une propriété du type de relation sémantique d'indice "trelsem" (de 0 à ...). Cette fonction renvoie une erreur si le nom de la propriété est non valide. Les propriétés du type de relation sémantique sont :

- Nom = nom en clair du type de relation sémantique
- NbRelMin = nombre mini de relations directes de ce type de relation sémantique
- NbIRelMin = nombre mini de relations inverses de ce type de relation sémantique
- NbRelMax = nombre maxi de relations directes de ce type de relation sémantique
- NbIRelMax = nombre maxi de relations inverses de ce type de relation sémantique
- ClasseSrce = classe de l'objet source ou 0 si la source est un élément
- ClasseDest = classe de l'objet destination ou 0 si la destination est un élément
- TEltSrce = 0 si la source est un objet, sinon combinaison des types d'éléments pouvant se trouver en source (1=EltPoint,2=EltLiaison,4=EltEcriture,16=EltFace)
- TEltDest = 0 si la destination est un objet, sinon combinaison des types d'éléments pouvant se trouver en destination (1=EltPoint,2=EltLiaison,4=EltEcriture,16=EltFace)
- MaskChSrce = numéro d'ordre du masque de recherche de face, liaison, écriture, point ou élément suivant la valeur dans TEltSrce
- MaskChDest = numéro d'ordre du masque de recherche de face, liaison, écriture, point ou élément suivant la valeur dans TEltDest
- Methode = Methode de détermination automatique de la relation (TEltSrce et TEltDest sont alors des valeurs simples et non des combinaisons de valeurs)
  0 = RELSEMMETHODE_MANUEL = calcul manuel
  1 = RELSEMMETHODE_APP = calcul par appartenance
  2 = RELSEMMETHODE_INCL = calcul par inclusion
  3 = RELSEMMETHODE_PROX = calcul par proximité (seuil maxi fixé par Param)
  4 = RELSEMMETHODE_DEPEND = calcul par dépendance
- Couleur = couleur sous forme 0x00BBGGRR de représentation de ce type de relation sémantique (en visions polychromes)
- Epaisseur = épaisseur de représentation de ce type de relation sémantique.
- Param = paramètre précisant la Methode (valeur maxi en mètre de la proximité)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <valeur_parametre> | STR NUM | fournit la valeur du paramètre |

## Exemples:

| | |
|---|---|
| @gettrel(0,Couleur) | renvoie la couleur de tracé des "correspondances" |
| @gettrel(3,NbRelMin) | renvoie le nombre mini de relations devant exister entre une borne et la parcelle |

# @gettrf

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| param | STANDARD | nom du paramètre de la transformation en cours dans la fenetre courante (a,b,c,d,p,q) ou 0 si tous les paramètres à enregistrer dans les variables (de noms "va","vb","vc","vd","vp","vq") |

## Action:

Fournit un paramètre de la transformation de la fenêtre courante ou enregistre tous les paramètres dans les variables de noms prédéterminés (va,vb...)
Les paramètres sont ceux de la transformation telle que
ax+by+p=X
cx+dy+q=Y.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <param> | NUM | la valeur du parametre recherché est renvoyé |
| OK | STD | tous les paramètres ont été enregistrés |

## Exemples:

| | |
|---|---|
| @gettrf(a) | renvoie le paramètre |
| @gettrf(0) | enregistre tous les paramètres dans les variables va,vb,vc,vd,vp,vq (toute ces variables doivent exister) |

# @gettyperelsem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| rel | STANDARD ou RELSEM | relation sémantique dont il faut extraire une donnée |

## Action:

fourni le type de la relation

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <type> | NUM | renvoie le type de la relation |

## Exemples:

| | |
|---|---|
| @gettyperelsem(@rel) | fournit le type de la relation "rel" |

# @getvar

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à initialiser avec la saisie effectuée |
| prompt | STANDARD | texte devant être affiché dans la boite de dialogue de saisiie |
| type | STANDARD | type de la variable à initialiser |

## Action:

saisie d'une donnée par l'utilisateur par l'intermédiaire d'une boite de dialogue et affectation à une variable (qui doit exister).
La valeur saisie représente suivant les types de variables :

| | |
|---|---|
| standard | la valeur |
| PT | le numéro du point |
| OBJ | le numéro de l'objet |
| MASKCRPT | indice dans la table des masques de création de points |
| MASKMDPT | indice dans la table des masques de modification de points |
| MASKCHPT | indice dans la table des masques de recherche de points |
| MASKCRLI | indice dans la table des masques de création de liaisons |
| MASKMDLI | indice dans la table des masques de modification de liaisons |
| MASKCHLI | indice dans la table des masques de recherche de liaisons |
| MASKCRFC | indice dans la table des masques de création de faces |
| MASKMDFC | indice dans la table des masques de modification de faces |
| MASKCHFC | indice dans la table des masques de recherche de faces |
| MASKCREC | indice dans la table des masques de création d'écritures |
| MASKMDEC | indice dans la table des masques de modification d'écritures |
| MASKCHEC | indice dans la table des masques de recherche d'écritures |

la variable doit être standard pour un type demandé standard (éventuelle conversion) ou du type fourni pour une variable non standard

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Interruption Utilisateur | ERR | Si la saisie s'est terminée par abandon (ou ESC) |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Affectation impossible : types incompatibles | ERR | si le type demandé ne correspond pas au type de la variable et n'est pas traductible (pas de conversion entre types standard) |
| [nomvar] | ??? | renvoie la variable modifiée ou créée |

## Exemples:

| | |
|---|---|
| @getvar(mavar,Entrer un numéro de point,PT) | saisie d'un numéro de point et affectation du point à la variable de nom "mavar" si le numéro existe sinon affectation de 0 (pointeur nul) |

# @getvue

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| indice | STANDARD | indice de la vue (de 0 à ...) ou (−1) si on désire récupérer la vue courante |

## Action:

recherche une vue (visualisation) et la fournit en retour.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <vue> | PLANVIEW | vue recherchée ou NUL si indice d'une vue inexistante |

## Exemples:

| | |
|---|---|
| @getvue(0) | fournit la premiere vue du document courant |
| @getvue(−1) | fournit la vue courante du document courant |
| @getvue(2) | fournit la 3° visualisation (fenêtre) ouverte du document courant |

# @getwindow

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche concernée ou<br>−1 si les coordonnées écran (de la fenêtre ou du moniteur principal) sont désirées |
| position | STANDARD | −1 = les dimensions de l'écran sont demandées (dans ce cas couche doit être −1) sinon :<br>position sur l'écran dont les coordonnées sont à rechercher<br>0 = centre de la fenêtre<br>1 = haut et gauche de la fenêtre<br>2 = haut et droite de la fenêtre<br>3 = bas et gauche de la fenêtre<br>4 = bas et droite de la fenêtre |
| varx | STANDARD | nom de la variable recevant la coordonnée X du point |
| vary | STANDARD | nom de la variable recevant la coordonnée Y du point |

## Action:

fournit les coordonnées réelles relatives à la couche 'couche' se trouvant à une position de la fenêtre plan.
Si couche==−1 alors les coordonnées écran sont alors renvoyées : on aura donc toujours @getwindow(−1,1,varx,vary) qui fournira les coordonnées 0,0 et @getwindow(−1,4,varx,vary) qui fournira la largeur et la hauteur de la fenêtre respectivement dans 'varx' et vary'.
Si couche==−1 ET position==−1, c'est alors les dimensions de l'écran principal qui sont renvoyées.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | les coordonnées ont été transmises aux variables |

## Exemples:

| | |
|---|---|
| @getwindow(2,0,varx,vary) | fournit les coordonnées correspondant au centre de la fenêtre plan pour la couche 2. |

# @getx

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément ou objet dont il faut connaitre la coordonnee |
| | ou | ou |
| elt | PT<br>LI<br>FC<br>EC<br>OBJ | élement ou objet |

## Action:

fourni la coordonnée X de l'élément ou du centroide de l'élément (suivant sa nature) ou de l'objet à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <coord> | NUM | coordonnée recherchée |

## Exemples:

| | |
|---|---|
| @getx(obj) | donne la coordonnée du centroide de l'objet contenu dans la variable "obj" |

# @gety

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément ou objet dont il faut connaitre la coordonnée |
| | ou | ou |
| elt | PT<br>LI<br>FC<br>EC<br>OBJ | élement ou objet |

## Action:

fourni la coordonnée Y de l'élément ou du centroide de l'élément (suivant sa nature) ou de l'objet à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <coord> | NUM | coordonnée recherchée |

## Exemples:

| | |
|---|---|
| @gety(obj) | donne la coordonnée du centroide de l'objet contenu dans la variable "obj" |

# @getz

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant le point dont il faut connaitre l'altitude |
| pt | ou | ou |
| | PT | point dont il faut connaitre l'altitude |

## Action:

fourni la coordonnée Z du point à travers ou non une variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <coord> | NUM | coordonnée Z recherchée |

## Exemples:

| | |
|---|---|
| @getz(pt) | donne l'altitudedu point contenu dans la variable "pt" |

# @global

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable globale à créer |
| value | STANDARD | valeur à affecter |

## Action:

initialisation et affectation d'une variable globale (visible durant toute l'instance de TopoCad).
La modification d'une variable globale et de sa valeur d'initialisation se fait par @setgvar
La modification d'une variable globale sans changer sa valeur d'initialisation se fait par @setvar
Cette instruction se trouve généralement en début de programme principal de configuration (déclaration des variables globales)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | STD | renvoie la variable |

## Exemples:

@global(monrep,"C:\topocad\doc")

# @graphinsert

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | PT | point source à insérer dans le graphe |
| selsrce | STANDARD | niveau de sélection des liaisons du graphe à considérer pour insertion (ou −1 si tous le graphe est à considérer) |
| seldest | STANDARD | niveau de sélection qui servira à la sélection des éléments insérés dans le graphe (ou −1 si rien n'est à sélectionner) |
| dirname | STANDARD | nom de l'Xdata donnant le poids dans le sens direct de la liaison |
| revname | STANDARD | nom de l'Xdata donnant le poids dans le sens inverse de la liaison |

## Action:

Effectue une einsertion du point "srce" dans le graphe représenté par les liaisons sélectionnées aun niveau "selsrce" de la couche du point à insérer.
Le point est relié au graphe par la plus proche liaison (sélectionnée si selsrce différent de −1) par une ou 3 liaison :
−si le pied de la perpendiculaire au point abaissée sur la liaison est dans la liaison alors une liaison du point source de la liaison à la perpendiculaire, une liaisson du point destination de la liaison à la perpendiculaire, et une liaison du pied de perpendiculaire au point "srce" sont créés et éventuellement sélectionné (si seldest est différent de −1)
−si le pied de la perpendicualire est extérieur à la liaison alors une seule liaison au point srce ou dest de la liaison est créé et sélectionnée (si seldest est différent de −1).

Effectue un calcul par l'algorithme de dijkstra.

Les données XData de ces liaisons créées sont mises à jour en fonctions des données du graphe

La fonction renvoie 0 (impossible d'insérer, 1 ou 3 représentant le nombre de liaisons créées.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0, 1 ou 3 | NUM | nombre de laison créées dans le graphe |

## Exemples:

| | |
|---|---|
| @graphinsert(@getpoint(1), −1,−1,D,RD) | insère le point dans le graphe de la couche du point 1 : les données D et RD sont mises à jour. |

# @graphnearest

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | PT | point source |
| selsrcept | STANDARD | niveau de sélection des points du graphe à considérer |
| selsrceli | STANDARD | niveau de sélection des liaisons du graphe à considérer (ou −1 si la totalité des liaisons est à considérer) |
| seldest | STANDARD | niveau de sélection qui servira à la sélection du chemin le plus court au point trouvé (ou −1 si le chemin le plus court au point ne doit pas être sélectionné) |
| dirname | STANDARD | nom de l'Xdata donnant le poids dans le sens direct de la liaison |
| revname | STANDARD | nom de l'Xdata donnant le poids dans le sens inverse de la liaison |

## Action:

Recherche dans le graphe parmi les points sélectionnés au niveau "selsrce" le point le plus proche du point "srce"
Effectue un calcul par l'algorithme de dijkstra.
Le graphe est représenté par l'ensemble des liaisons sélectionnées au niveau "selsrceli" de la couche du point source (ou toutes les liaisons si "selsrceli" est égal à −1). A l'issue du calcul, les liaisons composant le plus court chemin entre la source et le point destination trouvé sont sélectionnées au niveau de sélection "seldest" s'il est fourni (différent de −1). La "longueur" du chemin du point srce au point dest est donnée par l'XData de nom "dirname" du point dest et la "longueur" du point dest au point srce est donnée par l'XData de nom "revname" du point dest.
La fonction renvoie le point touvé ou NULL si aucun point trouvé
le point "srce" est désélectionné du niveau "selsrcept" à l'issue de la procédure.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <dest> | PT | point trouvé le plus proche dans le graphe |

## Exemples:

| | |
|---|---|
| @graphnearest(@getpoint(1), 0,−1,D,RD) | renvoie le point sélectionné au niveau 0 le plus proche dans le graphe représenté par les liaisons de la couche du point 1. |

# @graphselect

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| sel | STANDARD | niveau de sélection à positionner pour le graphe recherché |
| srceli | LI | liaison source à partir de laquelle déterminer le graphe |
| maskli | MASKCHLI | masque de recherche des liaisons indiquant l'appartenance au graphe (ou NULL) |

## Action:

Cette fonction recherche à partir de la liaison fournie toutes les branches de part et d'autre constituant donc un graphe. Seules les liaisons répondant au masque de recherche sont considérée lors de la recherche. Si la masque fourni est NULL, alors aucune restriction n'est apportée à la recherche

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <N> | NUM | nombre de laisons sélectionnées par l'opération et constituant le graphe (y compris la liaison originale répondant au masque) |

## Exemples:

| | |
|---|---|
| @setvar(mask,0,MASKCHLI),<br>@graphselect(0,@vli,@mask) | sélectionne au niveau 0 le graphe à partir de la liaison pointée par la variable "vli" |

# @graphshortestpath

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| srce | PT | point source |
| dest | PT | point destination |
| selsrce | STANDARD | niveau de sélection représentant le graphe ou −1 si toute la couche est à considérer comme étant le graphe |
| seldest | STANDARD | niveau de sélection qui servira à la sélection du chemin le plus court trouvé ou −1 si le chemin le plus court ne doit pas être sélectionné |
| dirname | STANDARD | nom de l'Xdata donnant le poids dans le sens direct de la liaison |
| revname | STANDARD | nom de l'Xdata donnant le poids dans le sens inverse de la liaison |

## Action:

Effectue un calcul par l'algorithme de dijkstra. cet algorithme calcule dans un graphe le plus court chemin d'un point source à un point destination.
Le graphe est représenté par l'ensemble des liaisons liées entre elles composant la couche des points source et destination. Il est possible de limiter le graphe aux liaisons sélectionnées de la couche en indiquant un niveau de sélection "selsrce". A l'issue du calcul, les liaisons composant le plus court chemin sont sélectionnées au niveau de sélection "seldest" s'il est fourni (différent de −1). La "longueur" du chemin du point srce au point dest est donnée par l'XData de nom "dirname" de la liaison et la "longueur" du point dest au point srce est donnée par l'XData de nom "revname" de la liaison.
La fonction renvoie la plus courte distance parcourue du point "srce" au point "dest" (qui est la donnée de nom "D" du point "dest")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <dist> | NUM | fournit la distance du plus court chemin. |
| Données n'ont pu être conservées! | ERR | problème à l'écriture ou lecture d'un Xdata d'un élément |
| Calcul impossible! | ERR | impossible d'atteindre le point "dest" à partir du point "srce" |

## Exemples:

| | |
|---|---|
| @graphshortestpath(@getpoint(1), @getpoint(9),−1,0,D,RD) | sélectionne au niveau 0 le plus court chemin du point 1 au point 9 pour l'ensemble des liaisons de la couche du point 1. |

# @hierarchise

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si ttes sont concernées |

## Action:

réordonne l'ordre d'affichage de l'ensemble des éléments sur la couche "couche" d'après leur priorités prédéfinies au niveau du SCD de TopoCad

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @hierarchise(0) | hiérarchise la couche 0 |
| @hierarchise(−1) | hiérarchise toutes les couches |

# @hinttext

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| texte1 | STANDARD | chaine de caractère |
| texte2 | STANDARD | chaine de caractère |

## Action:

affiche dans la barre d'état de l'application un message constitué de la concaténation de texte1 et texte2

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | affichage réalisé |

## Exemples:

| | |
|---|---|
| @hinttext("En cours de traitement...",@cmpt) | affiche un message indiquant la progression dans le déroulement d'une tache. |

# @if

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| condition | BOOL | il existe un opérateur booleen implicite pour tous les types de données |
| si_oui | commande | commande ou liste si la condition est réalisée |
| si_non | commande | commande ou liste si la condition n'est pas réalisée |

## Action:

execution conditionnelle de commande (ou liste).

le nombre de caractères total du premier terme ou du second terme ne peut excéder 1024

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| ECHEC | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ??? | ??? | renvoie le résultat de la commande exécutée |

## Exemples:

| | |
|---|---|
| @if(@test,@list(...),@list(....)) | execute la première ou seconde liste de commandes suivant le resultat donné par la variable de nom "test" |
| @if(@test,@list(...),) | |

# @import

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom du fichier d'importation |
| type | STANDARD | code donnant le type d'importation<br>1 = MAP<br>2 = NXY<br>3 = DXF<br>4 = EDIGEO/PCI<br>5 = MIF<br>6 = SHP<br>7 = LOC<br>8 = EDIGEO UTILISATEUR<br>9 = APIC<br>10 = DA Numérique<br>11 = OSM<br>12 = KML |
| strict | STANDARD | indique si l'échec de l'import renvoie une erreur (strict=1) ou la valeur 0 (strict=0) |

## Action:

importation dans le document (création de nouvelle(s) couche(s)) d'un fichier externe

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Echec à l'importation de <filename>. | ERR | impossible d'importer |
| 0 ou 1 | NUM | importation réalisée avec succès (1) ou non (0) |

## Exemples:

| | |
|---|---|
| @import(@file,2,1) | "file" contient un nom de fichier NXY |

# @importcarnet

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom du fichier d'importation |
| type | STANDARD | code donnant le type de format de carnet (de 0 à ...) parmi ceux enregistrés dans le système. |
| strict | STANDARD | indique si l'échec de l'importation renvoie une erreur (strict=1) ou la valeur 0 (strict=0) |

## Action:

importation dans le document (création d'observations) d'un fichier electronique externe dans un format prédéfini et enregistré dans le système

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Echec à l'importation de <filename>. | ERR | impossible d'importer |
| 0 ou 1 | NUM | importation réalisée avec succès (1) ou non (0) |

## Exemples:

| | |
|---|---|
| @importcarnet(@file,1,1) | "file" contient un nom de fichier DAT (leica = 2° format de carnet prédéfini) |

# @inf

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| term1 | STANDARD | premier terme |
| term2 | STANDARD | second terme |

## Action:

renvoie [term1 < term2] opérateur "strictement inférieur à"
la valeur stockées sont comparées suivant leur nature :
elles seront comparées comme étant des valeurs numériques décimales seulement si ce sont deux valeurs de type numérique.
elles seront comparées comme étant des angles au format TopoCad (ex: 125.12dv) seulement si ce sont des valeurs de type angle
Si une variable numérique contient une valeur octale ou hexadécimale, la comparaison peut ne pas être celle désirée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [term1 < term2] | BOOL | |

## Exemples:

| | |
|---|---|
| @inf(@var1,10) | renvoie 1 si le contenu de la variable de nom "var1" est inférieur à 10, renvoie 0 sinon |

# @infe

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| term1 | STANDARD | premier terme |
| term2 | STANDARD | second terme |

## Action:

renvoie [term1 <= term2] opérateur "inférieur ou égal à"
la valeur stockées sont comparées suivant leur nature :
elles seront comparées comme étant des valeurs numériques décimales seulement si ce sont deux valeurs de type numérique.
elles seront comparées comme étant des angles au format TopoCad (ex: 125.12dv) seulement si ce sont des valeurs de type angle
Si une variable numérique contient une valeur octale ou hexadécimale, la comparaison peut ne pas être celle désirée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [term1 <= term2] | BOOL | |

## Exemples:

| | |
|---|---|
| @inf(@var1,10) | renvoie 1 si le contenu de la variable de nom "var1" est inférieur ou égal à 10, renvoie 0 sinon |

# @inv

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val | STANDARD | valeur |

## Action:

1/val

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| inverse d'un nombre nul | ERR | tentative de division par 0 |
| 1/val | NUM | résultat |

## Exemples:

| | |
|---|---|
| @inv(@mavar) | fournit <1/[mavar]> |

# @inversesensli

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| li | LI | liaison dont il faut inverser le sens |

## Action:

inverse le sens de la liaison. Attention : si la liaison supporte un signe de mitoyenneté, celui ci sera inversé également.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | liaison inversée |

## Exemples:

| | |
|---|---|
| @inversesensli(@vli) | inverse la liaison dans la variable vli |

# @invvar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |

## Action:

[nomvar] <== [1/nomvar]

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| inverse d'un nombre nul | ERR | tentative de division par 0 |
| impossible de calculer l'inverse d'une variable pointeur | ERR | si la variable n'est pas standard |
| [nomvar] | NUM | résultat (valeur de la variable après négation) |

## Exemples:

| | |
|---|---|
| @invvar(mavar) | fournit <1/mavar> |

# @ivar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de variable à retourner |

## Action:

permet d'accéder à une variable de manière indirecte.

l'accès ordinaire à une variable TED est effectué sous forme : @nomvar
il n'est pas possible cependant d'avoir quelque chose du genre @@nomvar, c'est à dire d'accéder à une variable dont le nom est fourni par une variable. C'est le role de @ivar(@nomvar) fournissant la valeur de la variable dont le nom est donné par la variable "nomvar".

Il est ainsi possible de créer des tableaux.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| value | ??? | valeur de la variable |

## Exemples:

```
@setvar(nbvar,7,NUM),
@for(@setvar(i,0,NUM),
   @inf(@i,@nbvar),
   @list(
      @setvar(nomvar,"mavar",STR),
      @concatvar(nomvar,@i,0),
      @setvar(@nomvar,
         @concat("variable n°",@add(@i,1)),
         STR)
         ),
   @next(i,-1)
   )
```

allocation d'un tableau de 7 variables nommées "mavar0", "mavar1", ... contenant "variable n°1", "variable n°2"...

```
@setvar(i,5,NUM),
@setvar(nomvar,"mavar",STR),
@concatvar(nomvar,@i),
@return(@ivar(@nomvar))
```

accès à la variable indexée : renvoie la valeur "variable n°5"

# @layer

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom | STANDARD | nom de la couche à rechercher |

## Action:

Recherche la première couche dont le nom est 'nom' et renvoie son numéro (de 0 à ...).
Si le nom de couche est inexistant dans le document courant, renvoie alors la valeur (−1)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <no_couche> | NUM | le numéro de couche ou (−1) si non trouvé |

## Exemples:

| | |
|---|---|
| @layer("AB_ajouts") | renvoie le numéro de la couche de nom "AB_ajouts" |

# @layeradd

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| name | STANDARD | nom de la couche à ajoutter au document |

## Action:

Ajoutte une couche vide de nom "name" au document
Cette couche nouvelle voit son système de coordonnées se caler sur le système de la couche de travail (au niveau du document et au niveau de l'affichage de la vue)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'ajoutter une couche | ERR | erreur à l'ajout de la couche dans le document |
| N | NUM | N = indice de la nouvelle couche créée (0 étant la première couche) |

## Exemples:

| | |
|---|---|
| @layeradd("nouvelle couche") | ajoutte une couche de nom "nouvelle couche" |

# @layercolor

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| no | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| color | STANDARD | couleur sous forme d'entier 0xBBGGRR |
| back | STANDARD | 0 pour la couleur de la couche<br>1 pour la couleur de fond |

## Action:

change ou fournit la couleur de la couche pour le document et la fenêtre courante
cette fonction peut obliger la fenêtre à reconstituer le bitmap pour la couche concernée

NB:
−il est possible d'affecter une couleur "None" c'est à dire pas d'affichage en fournissant la valeur −1 : la couleur stockée "None" a pour valeur 0xFF000000 mais ne peut etre affectée directement.
−si on fournit la valeur −2 comme couleur, cela est interprété comme une demande de renvoie de la couleur actuelle sans modification

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | la coucleur de la couche a été modifiée par la valeur fournie |
| <couleur> | NUM | couleur de la couche demandée |

## Exemples:

| | |
|---|---|
| @layercolor(0,0xFF,0) | fixe la couleur rouge à la couche 0 |
| @layercolor(0,−2,0) | fournit la couleur de la couche 0 |

# @layerinit

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| no | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| inittoset | STANDARD | valeur des options à forcer à ON sous forme de champ de bit avec chaque bit signifiant :<br>• bit 1 = 0x0002 = Visible<br>• bit 2 = 0x0004 = Monochrome<br>• bit 5 = 0x0020 = Actif<br>• bit 6 = 0x0040 = Bitmap<br>• bit 7 = 0x0080 = BmpTransparent<br>• bit 8 = 0x0100 = BmpAll<br>• bit 9 = 0x0200 = BmpBkTransparent<br>• bit 10 = 0x0400 = BmpMonoNoPreserv |
| inittoclear | STANDARD | valeur des options à forcer à OFF sous forme de champ de bit avec chaque bit signifiant :<br>• bit 1 = 0x0002 = Visible<br>• bit 2 = 0x0004 = Monochrome<br>• bit 5 = 0x0020 = Actif<br>• bit 6 = 0x0040 = Bitmap<br>• bit 7 = 0x0080 = BmpTransparent<br>• bit 8 = 0x0100 = BmpAll<br>• bit 9 = 0x0200 = BmpBkTransparent<br>• bit 10 = 0x0400 = BmpMonoNoPreserv |

## Action:

Permet d'activer ou/et désactiver des options d'affichage et traitement de la couche pour le document et la vue en cours.
Cette fonction oblige la fenètre à reconstituer le bitmap pour la couche concernée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @layerinit(0,0x02,0x20) | rend la couche 0 visible mais non active |

# @layername

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| no | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| newname | STANDARD | nouveau nom de la couche |

## Action:

renomme la couche d'indice "no"
renvoie le nom de la couche si newname=""

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <layername> | STR | nom de la couche |

## Exemples:

| | |
|---|---|
| @layername(0,plan) | renomme la couche 0 en lui donnant le nom "plan" |
| @layername(0,"") | renvoie le nom de la couche 0 sans modifier ce dernier |

# @leftstr

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| texte | STANDARD | texte dont il faut extraire la chaine |
| nbcar | STANDARD | nombre de caractères à extraire à partir de la gauche |

## Action:

extrait une chaine de "nbcar" caractères à partir du début de la chaine "texte". Si le nombre de carcatères est plus important que le nombre de caractères de la chaine, alors renvoie la chaine entière.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <leftstring> | STR | renvoie la chaine extraite |

## Exemples:

| | |
|---|---|
| @leftstr(commenter,4) | renvoie "comm" |
| @leftstr(commenter,−4) | renvoie "comme" |
| @leftstr(commenter,20) | renvoie "commenter" |

# @len

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | chaine dont on veut connaitre la longueur |

## Action:

renvoie la longueur de la chaine de caractère fournie (sans le 0 final)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | longueur de la chaine |

## Exemples:

| | |
|---|---|
| @len(@var1) | renvoie la longueur de la chaine contenue dans la variable |
| @len("coucou me voilou") | renvoie 16 |

# @list

*Nb paramètres*
*indéterminé*

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| liste de commandes | ??? | liste de commandes séparées par la virgule. Le nombre de commande peut être quelconque |

## Action:

permet de regrouper une série d'actions, de commandes en une seule action, commande

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| résultat de la dernière commande de la liste | ??? | le type varie suivant la commande |

## Exemples:

| | |
|---|---|
| @if(@var,<br>@list(*@commande1*,*@commande2*,*@commande3*),<br>@list(*@commande4*, *@commande5*)) | listes de commandes effectuées à la suite quand une fonction attends en fait une commande |

# @loaddoc

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| file | STANDARD | nom du fichier MAP à charger |
| opt | STANDARD | 1 ou 0 : indique si le document chargé doit devenir le document par défaut (1) ou non (0) |

## Action:

Charge un fichier map et le prend comme document courant si opt=1

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| 1 | NUM | document chargé |
| 0 | NUM | document non chargé |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Document ... déjà ouvert | ERR | document déjà présent dans l'application |

## Exemples:

@loaddoc(c:\topocad\doc\document.map,1)

# @loadformatpage

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| fname | STANDARD | nom du fichier INI comprenant une section FORMAT_PAGE à exploiter |

## Action:

charge de nouveaux formats de page pour l'application (qui remplaceront ceux existant) à partir d'un fichier INI désigné.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de modifier les formats de page (occupés) | ERR | L'application utilise les formats de pages qui ne peuvent donc être modifiés (mode prévisualisation ou impression en cours) |
| Impossible d'ouvrir le fichier des formats de pages | ERR | nom de fichier fourni erroné ou impossibilité d'ouvrir le fichier ou le lire |
| Fichier des formats de pages invalide | ERR | un format de page est invalide dans le fichier INI ou le fichier INI est incorrect |
| <nb> | NUM | formats chargés : renvoie le nombre de formats de pages chargés |

## Exemples:

@loadformatpage("c:\topocad\topocad.ini") recharge les formats de page par défaut

# @loadmasques

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ini | STANDARD | nom du fichier INI à charger |

## Action:

Charge de nouveaux masques pour le modèle courant. Le nombre de masques fournis dans la section MASQUES du fichier INI donné doit être identique au nombre de masques actuellement gérés dans le modèle.
Cette commande évite de recharger le modèle entièrement pour des modifications mineures (épaisseurs, couleurs...)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Cette table des masques n'est pas valide pour le modèle en cours (nombre de masques inadapté) | ERR | nombre de masques inadapté au modèle |
| fichier corrompu | ERR | le fichier n'est pas un fichier INI ou le format n'est pas reconnu. |
| Impossible de charger le fichier | ERR | fichier non trouvé ou inaccessible |
| OK | STR | tout s'est déroulé correctement et la nouvelle table des masques est active |

## Exemples:

| | |
|---|---|
| @loadmasques(c:\topocad\doc\topocad.ini) | charge le modèle fourni dans le fichier de config TOPOCAD.INI. |

# @loadmodele

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom du fichier INI |

## Action:

Charge un modele SCD c'est à dire les masques de création, modification, recherche, les types d'objets du modèle et les couches par défaut. Ce chargement dynamique n'est possible que si aucun document n'existe dans l'application ou dans le cas où un document non vide existe, si le nombre de classes du nouveau modèle est au moins égal au nombre de classes du modèle à substituer, le nombre de couches du document en cours est au moins égal au nombre de couches que nécessitte le modèle, et le nombre de type de relations sémantiques du nouveau modèle est au moins égal au nombre de types de relations de cet ancien modèle .
Le SCD est alors chargé ainsi que tous les masques dans le fichier INI.

Le fichier INI doit comporter les sections suivantes: MASQUES, OBJETS, COUCHES, RELATIONS qui doivent être cohérentes entre elles (références aux masques, aux couches ...), sinon le chargement est abandonné.

La section objet contient la manière de calculer de l'identifiant pour chaque type d'objet qui est donc rechargée, cependant l'allocation d'une base de donnée pour les propriétés de ces types d'objet n'est pas réalisé comme au démarrage de l'application. On peut donc avoir à l'issue de ce chargement pour un type d'objet une base de donnée allouée mais sans methode pour calculer l'identifiant (il est alors neccessaire d'appeler @deldatabase pour fermer et supprimer la base de donnée plus utilisée) et aucune base de donnée allouée pour un type d'objet ayant désormais une méthode de calcul de l'identifiant et donc succeptible de gérer des propriétés des objets (il est alors nécessaire d'appeler @adddatabase pour créer une nouvelle base de données).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de charger le fichier INI | ERR | modèle incohérent, fichier corrompu ou inadapté... |
| OK | STR | loadmodele réalisé avec succès |

## Exemples:

| | |
|---|---|
| @loadmodele(c:\temp\pci2.ini) | charge un nouveau SCD |

# @loadtrf

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| trffile | STANDARD | nom du fichier TRF à charger |

## Action:

Charge une transformation dans la fenêtre courante.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'ouvrir le fichier de transformation | ERR | le fichier TRF n'existe pas ou n'a pas été trouvé. |
| Transformation invalide !!! | STR | la transformation a mal été chargée ou n'est pas inversible |
| OK | STR | tout s'est déroulé correctement |

## Exemples:

| | |
|---|---|
| @loadtrf(c:\topocad\doc\matransf.trf) | charge la transformation MATRANSF.TRF. |

# @loadtxt

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| file | STANDARD | nom du fichier TXT à charger |
| opt | STANDARD | 1 ou 0 : indique si le document chargé doit devenir le document par défaut (1) ou non (0) |

## Action:

Charge un fichier texte et le prend comme document courant si opt=1
ATTENTION: les documents textes ne sont pas manipulables par TED : seul le chargement en fenetre texte est possible, à l'issue du chargement, aucune fenetre plan n'étant active, les fonction TED suivantes ne peuvent manipuler les documents/vues plan, il est donc nécessaire si besoin de repasser sur une fenetre plan par @setdocvue ou par l'option 1 de la commande

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| 1 | NUM | document chargé |
| 0 | NUM | document non chargé |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Document ... déjà ouvert | ERR | document déjà présent dans l'application |

## Exemples:

@loadtxt(c:\topocad\doc\rapport.txt,1)

# @local

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable locale à créer |
| value | ??? | valeur à affecter (peut être 0 pour les variables pointeurs non initialisées)<br>Dans le cas ou on fourni un paramètre standard à une variable pointeur la signification de la valeur est alors la suivante:<br>PT : numéro de point (pointeur nul si non trouvé)<br>LI : dans tous les cas initialisé avec le pointeur nul<br>FC : idem<br>EC : idem<br>OBJ : numéro d'objet (pointeur nul si non trouvé)<br>MASK... : classe du masque de classe à aller chercher (pointeur nul si non trouvé ou hors classe)<br>RELSEM : numero de relation sémantique (la première n'est pas accessible)<br>Dans tous les cas si 0 est fourni, c'est le pointeur NULL qui est affecté à la variable |
| type | STANDARD | chaine représentant le type de variable<br>• STR: Type chaine de caractères<br>• NUM: Type numérique<br>• ANG: Type Angle<br>• BOOL: Type Booléen<br>• PT : type élément point<br>• LI :type élément liaison<br>• FC: type élément face<br>• EC: type élément écriture<br>• OBJ: type objet<br>• MASKCRPT: type masque de création de point<br>• MASKCRLI: type masque de création de liaison<br>• MASKCRFC: type masque de création de face<br>• MASKCREC: type masque de création d'écriture<br>• MASKMDPT: type masque de modification de point<br>• MASKMDLI: type masque de modification de liaison<br>• MASKMDFC: type masque de modification de face<br>• MASKMDEC: type masque de modification d'écriture<br>• MASKCHPT: type masque de recherche de point<br>• MASKCHLI: type masque de recherche de liaison<br>• MASKCHFC: type masque de recherche de face<br>• MASKCHEC: type masque de recherche d'écriture<br>• RELSEM : type relation sémantique<br>• MAPDOC : type document MAP<br>• PLANVIEW : type vue (fenêtre de visualisation) fenêtre Plan |

## Action:

création d'une variable locale au fichier TED en cours
(la valeur de la variable et la variable est perdue une fois le fichier TED terminé)
La modification d'une variable locale se fait par @setvar ou @setlvar

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | ??? | renvoie la variable (après affectation ou modification) |

## Exemples:

| | |
|---|---|
| @local(mask,5,MASKMDFC) | affecte à la variable "mask" le masque de modification des faces de la classe parcelle |

# @maillage

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche dans laquelle rechercher le maillage des liaisons (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des liaisons devant réaliser un maillage dans la couche |
| nivsel | STANDARD | niveau de sélection à positionner |
| estmaille | STANDARD | indique si on doit sélectionner toutes les liaisons qui constituent un maillage ou si on doit sélectionner les liasons (polylignes) orphelines. |

## Action:

détecte et sélectionne les polylignes constituant un maillage pour les liaisons de classe "classe" de la couche "couche" ou détecte celles ne constituant pas un maillage (suivant "estmaille").
une polyligne ne constitue pas un maillage si une de ses extrémités n'est pas rattachée à deux autres polylignes (il s'agit donc là d'une fonction réseau et non topologique)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N liaisons ont été sélectionnées | NUM | N = Nombre de liaisons sélectionnées |

## Exemples:

| | |
|---|---|
| @maillage(0,5,0,0) | sélectionne toutes les liaisons orphelines dans le maillage des parcelles (ne pouvant constituer l'appui d'une face) |

# @maillage2faces

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classefc | STANDARD | classe des faces à créer (de 1 à NbTObj−1) |
| nivsel | STANDARD | niveau de sélection des liaisons représentant le maillage (de 0 à 31) |

## Action:

Scrute toutes les liaisons de la couche "couche" qui sont sélectionnées au niveau "nivsel" et qui constituent un maillage et crée à l'intérieur de chaque maille une face de classe "classefc" si il n'en existe pas déjà une.
A l'issue de l'opération, seul le contour externe reste sélectionné ainsi que les éventuelles branches orphelines ne constituant pas de maillage. Tous le maillage ayant une face à droite et à gauche est déselectionné.
Si une face existe regroupant plusieurs mailles, alors la face de chacune des maille n'est pas créée.
Si une face existe et recouvre une maille n'ayant aucune liaison commune avec cette face alors la face de la maille est créée.
Si une face comporte un circuit d'arc à l'intérieur (face à trou), une face ordinaire recouvrant le trou sera créée ainsi qu'une face pour le trou (le traitement des faces à trous peut être réalisé par @makefacesatrou)

voir également @CreateFcWithLi

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Maillage2Faces : Erreur maillage incorrect ! | ERR | si anomalie dans le maillage (liaisons du maillage se recoupant) |
| N faces ont été créées | NUM | N = Nombre de faces créées |

## Exemples:

| | |
|---|---|
| @maillage2faces(0,5,0) | crée les faces internes de parcelles à l'intéreur du maillage sélectionné au niveau 0 |

# @maj

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des éléments à modifier (de 0 à 31) |
| test | STANDARD | 0x001 = TESTMAJ_ASUPRIMER<br>0x002 = TESTMAJ_AAJOUTER<br>0x004 = TESTMAJ_NOTOUCH<br>0x010 = TESTMAJ_ACTIF<br>0x020 = TESTMAJ_NONACTIF<br>0x040 = TESTMAJ_SELNOTOUCH<br>0x100 = TESTMAJ_SELSUPANO<br>0x200 = TESTMAJ_SELAJTANO<br>0x400 = TESTMAJ_SELSUPDEPEND<br>0x800 = TESTMAJ_SELAJTDEPEND |

## Action:

*Si test est non nul* : Effectue les tests de mise à jour pour voir si les attributs ASupprimer et AAjouter sont cohérents.

TESTMAJ_ASUPRIMER : vérifie que chaque élément concerné marqué ASupprimer ne supprime pas un élément qui n'est pas ASupprimer. Les éléments concernés sont les éléments faisant partie d'une classe active si TESTMAJ_ACTIF ou/et d'une classe non active si TESTMAJ_NONACTIF. Autrement dit, controle la cohérence de l'attribut ASupprimer.

TESTMAJ_NOTOUCH : vérifie qu'on ne demande pas la suppression ou modification d'un élément de classe active si TESTMAJ_ACTIF ou de classe non active si TESTMAJ_NONACTIF (même si il y a une cohérence dans l'attribut ASupprimer et AAjouter).

TESTMAJ_AAJOUTER : vérifie que chaque élément marqué AAjouter ne soit pas obligé d'être créé pour qu'un autre élément de classe active (si TESTMAJ_ACTIF) ou non active (si TESTMAJ_NONACTIF) se crée, Autrement dit controle la cohérence de l'attribut AAjouter pour les classes données.

TESTMAJ_SELNOTOUCH : sélectionne au niveau de sélection 'sel' les éléments dont on demande la modification ou suppression et qui ne doivent pas l'être (cf TESTMAJ_NOTOUCH).

TESTMAJ_SELSUPANO : sélectionne au niveau de sélection 'sel' les éléments ASupprimer qui entrainent la suppression d'éléments qui ne sont pas ASupprimer. (cf TESTMAJ_ASUPRIMER)

TESTMAJ_SELSUPDEPEND : sélectionne au niveau de sélection 'sel' les éléments supprimés par suppression d'éléments ASupprimer et qui ne devraient pas être supprimés. (cf TESTMAJ_ASUPRIMER)

TESTMAJ_SELAJTANO : sélectionne au niveau de sélection 'sel' les éléments AAjouter qui ne sont pas cohérents avec leurs dépendances (ex: liaisons AAjouter alors que les faces s'y appuyant ne sont pas AAjouter. cf TESTMAJ_AAJOUTER)

TESTMAJ_SELAJTDEPEND : sélectionne au niveau de sélection 'sel' les éléments dépendants d'éléments AAjouter et qui ne sont pas AAjouter (ex: faces non marquées AAjouter alors que les liaisons la constituant sont AAjouter. cf TESTMAJ_AAJOUTER)

*Si test =0* : la mise à jour est faite, c'est à dire Tous les éléments marqués ASupprimer sont supprimés (avec d'éventuels éléments et objets dépendants) et tous les éléments marqués AAjoutter voient leur attribut AAjoutter effacé.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de faire la mise à jour, des éléments sont vérouillés | ERR | erreur si test==0 et des éléments sont verrouillés |
| Impossible de faire la mise à jour, les éléments ne sont pas en cohérence | STR | des erreurs sont trouvées (et éventuellement sélectionnées au niveau "nivsel") |
| OK | STR | tout est OK |

## Exemples:

| | |
|---|---|
| @setinit(15,0), | désactive la classe piscine |
| @setinit(5,0x22), | active la classe parcelles (autres classes supposées actives) |
| @Maj(0,0,0x21), | Seul le contour des piscine est marqué en principe AAjouter donc il ne faut pas controler l'intégriter de AAjouter pour cette classe, teste seulement si l'intégrité pour suppression des piscines est réalisée |
| @setinit(15,0x22), | active la classe piscine |
| @setinit(5,0), | désactive les parcelles (autres classes supposées actives) |
| @Maj(0,0,0x27), | teste si on touche aux parcelles |
| @setinit(15,0), | désactive les piscines |
| @setinit(5,0), | désactive les parcelles |
| @Maj(0,0,0x13), | teste l'intégrité des classes actives (le reste des classes en ajout et suppression) |
| @setinit(15,0x22), | réactive l'ensemble |
| @setinit(5,0x22), | |
| @Maj(0,0,0) | et fais la mise à jour réellement |

# @majdiff

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche source (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche destination (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des éléments à transférer |

## Action:

les élements sélectionnés au niveau "nivsel" (et obj) de "couchesrce" sont copiés dans "couchedest" avec l'attribut AAjouter (ceux ci doivent former un bloc intègre)
les éléments sélectionnés au niveau "nivsel" (et obj) de "couchedest" ont leur attributs positionnés à ASupprimer

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Intégrité des objets ou éléments non respectée ou impossible de transférer les éléments! | ERR | si les éléments de la couche source à copier ne forme pas blocs (une face sélectionnée sans sa liaison par exemple) |
| OK | STR | mise à jour faite |

## Exemples:

| | |
|---|---|
| @majdiff(0,1,0) | mise à jour de la couche 1 par la couche 0 grace aux éléments sélectionnés au niveau 0 |

# @majdispocouches

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | liste d'indices ordonné indiquant le nouvel ordre de disposition des couches existantes |

## Action:

cette commande permet de redéfinir l'ordre d'affichage des couches du document.
une liste d'entiers indique les positions de chacune des couches du document, chaque couche étant désignée par son indice dans l'ancienne disposition (de 0 à NbCouches−1).
la liste doit donc comporter "NbCouches" entiers de 0 à NbCouches−1 dans l'ordre nouveau que l'on désire

La chaine de caractères transmise ne peut excéder 1024 caractères (pas plus de 250 couches env)

Une commande @objintegrity doit suivre pour recalculer la couche des objets (si on en a besoin)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés ou si la liste ne comprend pas 'NbCouches' entiers. |
| Liste d'indices invalide | ERR | si un indice fourni est invalide ou si un indice (de 0 à NbCouches−1) est absent dans la liste |
| Ok | STR | les couches ont été réorganisées |

## Exemples:

| | |
|---|---|
| @majdispocouches("1 2 3 0") | Dans un document à 4 couches<br>la couche en position 1 vient en 1° position (position 0)<br>la couche en position 2 vient en 2° position<br>la couche en position 3 vient en 3° position<br>la couche en position 0 vient en 4° position<br>autrement dit on amène la première couche en dernière position, la dernière position étant la première a être affichée, elle s'affichera par dessous les autres. |
| @majdispocouches(1 2 3 0) | commande invalide |
| @setvar(vtab,1,STR),<br>@concatvar(vtab,0,1),<br>@majdispocouches(@vtab) | dans un document à deux couches, inverse la position des deux couches (équivalent à @majdispocouches("1 0")) |
| @majdispocouches("3 1 0 2") | commande invalide si le document n'a pas 4 couches |
| @majdispocouches("3 1 0 2") | Dans un document à 4 couches<br>la couche en position 3 vient en 1° position (position 0)<br>la couche en position 1 vient en 2° position<br>la couche en position 0 vient en 3° position<br>la couche en position 2 vient en 4° position |

# @makeallrelsem

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | couche des objets ou éléments source (ou −1 si déterminée automatiquement par le SCD) |
| couchedest | STANDARD | couche des objets ou éléments destination (ou −1 si déterminée automatiquement par le SCD) |
| type | STANDARD | type de relations à créer |
| checkcard | STANDARD | indique si la création de relations sémantiques se fait jusqu'à atteindre la cardinalité maximale de ce type de relation (1) ou si crée toutes les relations semantiques possibles sans se soucier de la cardinalité (0). |

## Action:

recherche et crée toutes les relations sémantiques de type donné. Si des relations sémantiques existent déjà, elles ne sont pas effacée même si elle sont incorrectes, de plus en aucun cas, TopoCad ne créera plus de relations sémantique que permis par la cardinalité directe de la relation si "checkcard"=1.

IMPORTANT : Cette fonction suppose que les informations concernant les objets sont mises à jour (ces informations sont mises à jour par @objintegrity)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N relations ont été créées | NUM | renvoie le nombre de relations créées |

## Exemples:

| | |
|---|---|
| @makeallrelsem(0,1,7,0) | crée toutes les relations Parcelle−>Subd de section |

# @makedeport

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des objets à scruter |
| nivsel | STANDARD | niveau de sélection des écritures à scruter (de 0 à 31) |

## Action:

crée une flèche de rattachement (déport d'écriture) entre les écritures sélectionnées de la couche "couche" faisant partie d'un objet de classe "classe" et la face de cet objet.

- si un seul objet de classe "classe" à face unique a été trouvé pour cette écriture l'écriture est déselectionnée et le déport repositionné ou créé.
- si aucun objet de classe "classe" à face unique a été trouvé mais un seul à face multiple alors l'écriture est déselectionnée, le déport repositionné ou crée et sélectionné.
- dans les autres cas (écriture sur plusieurs objets ou un objet d'une autre classe ou aucun), aucun déport n'est ajoutté pour l'écriture qui est laissée sélectionnée.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @makedeport(0,5,0) | crée les déports d'écritures des objets de classe parcelle dont les écritures sont sélectionnées |

# @makefacesatrou

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivsel | STANDARD | niveau de sélection des faces à régulariser (de 0 à 31) |

## Action:

Inspecte toutes les faces sélectionnées au niveau "nivsel" pour en faire un milieu topologique et transforme faces contenant entièrement d'autres faces (sélectionnées) en faces à trous (en rajoutant le trous de cette face). Il s'agit donc d'une normalisation des faces sélectionnées l'une envers l'autre.

A l'issue de l'opération, seules les faces ayant contribuées à la création de trous dans d'autres faces sont sélectionnées.

NB: cette procédure n'est pas en capacité de traiter plus d'un "niveau" de trous à la fois, c'est à dire de créer des faces à trou dont les trous sont des faces à trou qu'il faut créer (3 cercles concentriques par exemple) et ne créera dans ce cas qu'une seule face à trou alors que deux devraient être créées. Il faut alors soit réitérer la procédure soit utiliser la procedure manuelle.
Dans ces cas un message est envoyé à ERRORxxx.LOG indiquant que TopoCad n'a pu créer la face à trou et le trou concerné est sélectionné.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N faces à trou créées | STR | nombre de trous intégrés dans des faces |

## Exemples:

| | |
|---|---|
| @makefacesatrou(0,0) | régularise les faces de la couche 0 sélectionnées au niveau 0 |

# @makeobjext

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des objets concernés |
| extradataname | STANDARD | nom de la donnée extra unifiant les éléments |

## Action:

crée les objets de classe "classe" à partir d'éléments ne faisant partie d'aucun objet, de classe "classe" et sur la couche "couche" et dont la donnée extra de nom "extradataname" est le critère de reconnaissance (même valeur).
Une donnée extra vide n'est pas considérée comme une donnée extra valide (pas d'objet)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | N = Nombre d'objets créés |

## Exemples:

| | |
|---|---|
| @makeobjext(0,5,idu) | crée les objets parcelles à partir des données extra de nom "idu" |

# @makeobjlbl

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des objets concernés |

## Action:

crée les objets de classe "classe" à partir d'éléments ne faisant partie d'aucun objet, de classe "classe" et sur la couche "couche" et dont l'étiquette est le critère de reconnaissance (même étiquette).
Une étiquette vide n'est pas considérée comme une étiquette valide (pas d'objet)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | N = Nombre d'objets créés |

## Exemples:

| | |
|---|---|
| @makeobjlbl(0,5) | crée les objets parcelles à partir des étiquettes |

# @makerattacht

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des faces à scruter et celle des liaisons adjacentes devant avoir une flèche de rattachement |
| nivsel | STANDARD | niveau de sélection correspondant au réseau de liaisons à considérer (par ex: bati dur+ parcelles) |
| forme | STANDARD | forme du rattachement qui est :<br>0 Mur,<br>1 Fosse,<br>2 Cloture,<br>3 Haie,<br>4 RattachtAL,<br>5 FlecheRattacht,<br>6 Halte,<br>7 Arret,<br>8 Station |
| flags | STANDARD | indicateurs binaires pour le type de comportement à adopter<br>1 = UNIQUE = un seul rattachement imposé par polyligne (efface l'existant)<br>2 = USEEX = Utilise le rattachement le plus probable existant<br>4 = NOCLOSE = ignore les polylignes fermées<br>8 = ONEPERFACE = un seul rattacht par face |

## Action:

création automatique de flèches de rattachement sur les liaisons de classe "classe" de la couche "couche" qui sont sélectionnées au niveau "nivsel" :

cette procédure execute les taches suivantes :
1) marque toutes les liaisons autour des faces de classe "classe" pour traitement
2) si "UNIQUE" marque comme traitées toutes les liaisons autour des faces qui possèdent déjà un signe de nature "forme"
3) parmi ces liaisons restantes, constitue toutes les polylignes possibles en considérant qu'une polyligne est constituée de liaisons concomittantes sélectionnées au niveau sel (sinon chaque liaison est une polyligne)
4) si "NOCLOSE" , ne traite pas les polylignes fermées
5) crée le signe de ratacht de nature "forme" en prenant en considération les options 1 et 2

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

```
@setvar(lali,0,LI),                                  initialise variable pour scruter les Liaisons
@setselecttravail(31),                               niveau de sélection 31
@setvar(masque,8,MASKCHLI),                          masque de recherche des liaisons bati leger dans la variable
@firstwithmask(lali,masque),
@while(@lali,
   @list(
      @select(lali),                                 sélectionne toutes les liaisons bati leger
      @nextwithmask(lali,masque)
      )
   ),
@makerattacht(@couchetravail,8,31,5,13),             crée les flèche de rattachement (5) en fonction des liaisons bati leger (sélectionné au niveau 31) qui ont une face bati léger (8) avec une fleche par face et une par polyligne (efface si plusieurs en utilisant le plus probable)
@deselectall,                                        déselectionne tout
@setvar(masque,7,MASKCHLI),                          masque de recherche des liaisons bati dur
@firstwithmask(lali,masque),                         sélectionne toutes les liaisons bati dur
@while(@lali,
   @list(
      @select(lali),
```

```
    @nextwithmask(lali,masque)
    )
  ),

@makeratacht(@couchetravail,7,31,5,5),

@deselectall,
@setselecttravail(0),
@param(terminé !!!)
```

crée des flèches de rattachement sur toutes les polylignes sélectionnées qui bordent une face bati dur mais pas sur des polylignes fermées et une par face

déselectionne tout

remet le niveau de sélection courant à 0

affiche "terminé!!!"

# @makerelsem

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eltobj | STANDARD<br>PT<br>LI<br>FC<br>EC<br>OBJ | element, objet ou variable contenant ce dernier qui doit faire l'objet de la relation (source) |
| type | STANDARD | type de relation(s) à créer |
| couchedest | STANDARD | couche destination à scruter pour rechercher la destination de la relation (élement ou objet) |
| checkcard | STANDARD | indique si la création de relations sémantiques se fait jusqu'à atteindre la cardinalité maximale de ce type de relation (1) ou si crée toutes les relations semantiques possibles sans se soucier de la cardinalité (0). |

## Action:

recherche et crée toutes les relations sémantiques de type donné dont la source est *eltobj*. Recherche la destination dans la couche *couchedest* uniquement. Ne crée pas plus de relations que la cardinalité directe de la relation ne le permet si "checkcard"=1.

IMPORTANT : Cette fonction suppose que les informations concernant les objets sont mises à jour (ces informations sont mises à jour par @objintegrity)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N relations ont été créées | NUM | renvoie le nombre de relations créées |

## Exemples:

| | |
|---|---|
| @makerelsem(@obj,7,1,1) | crée une relation Parcelle−>Subd de section de l'objet parcelle désigné par la variable "obj" si elle n'existe pas déjà car la cardinalité maxi est de ce type de relation est 1. La subdiv de section est recherchée uniquement sur la couche 1. |

# @mask2obj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un masque de modification |
| | ou | ou |
| mask | MASKMDPT<br>MASKMDLI<br>MASKMDFC<br>MASKMDEC | masque de modification |
| | STANDARD | nom d'une variable contenant un objet à modifier |
| obj | ou | ou |
| | OBJ | objet à modifier |

## Action:

modifie les éléments d'un objet grace au masque de modification fourni
(on ne peut pas modifier à la fois par ex les liaisons et les faces de l'objet, mais seulement les faces ou seulement les liaisons de l'objet, pour modifier tous les éléments d'un objet cf @modifie)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @mask2obj(maskfc,monobj) | modifie les faces de l'objet contenu dans la variable "monobj" par le masque de modification de faces contenu dans la variable "maskfc" |

# @mdimove

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| vue | PLANVIEW | fenêtre concernée par le déplacement ou dimensionnement |
| x | STANDARD | coordonnée X du coin supérieur gauche de la fenêtre |
| y | STANDARD | coordonnée Y du coin supérieur gauche de la fenêtre |
| w | STANDARD | largeur de la fenêtre |
| h | STANDARD | hauteur de la fenêtre |

## Action:

replace et/ou redimensionne une fenêtre vue plan de l'application.

le système de coordonnée utilisé est celui de la fenêtre client de l'application (0,0 correspond au coin supérieur gauche et les X croissent vers la droite et les Y vers le bas.

si −1 ou une valeur supérieure à la fenêtre client est fournie, alors la coordonnée n'est pas modifiée (position ou taille)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre a un types ou valeur non approprié |
| 1 | STR | fenêtre réorganisée |
| 0 | STR | impossible de réorganiser la fenêtre (coordonnées non valide, taille trop petite...) |

## Exemples:

| | |
|---|---|
| @mdimove(@mavue,0,0,−1,−1) | déplace la vue au coin supérieur gauche sans redimensionner la fenêtre |
| @mdimove(@mavue,−1,−1,500,−1) | donne une largeur de 500 pixels à la fenêtre |

# @mdiorg

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| code | STANDARD | indique l'arrangement à opérer sur les fenêtres :<br>0 = cascade<br>1 = mosaique verticale<br>2 = mosaique horizontale<br>3 = fenetre courante plein ecran |

## Action:

réorganise les fenètres MDI dans la fenêtre client de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | fenêtres réorganisées |

## Exemples:

| | |
|---|---|
| @mdiorg(1) | équivalent au menu *fenêtre*\|*mosaïque* |

# @modeincorp

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la propriété "mode d'incorporation au SIG" du document (chaine de caractère)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <mode incorp> | STR | |

## Exemples:

| | |
|---|---|
| @modeincorp | renvoie "Numérisation manuelle" |

# @modifie

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eltobj | STANDARD | nom de la variable contenant l'élément ou objet à modifier |
| | PT | point à modifier avec le masque de modification de point de la classe donnée ou d'indice donné |
| | LI | liaison à modifier avec le masque de modification de liaison de la classe donnée ou d'indice donné |
| | FC | face à modifier avec le masque de modification de face de la classe donnée ou d'indice donné |
| | EC | écriture à modifier avec le masque de modification d'écriture de la classe donnée ou d'indice donné |
| | OBJ | objet à modifier avec les masques de modifications de point, liaison, face, écriture de la classe donnée |
| classe | STANDARD | Cn = classe n à considérer<br>n = indice n du tableau des masques de modification à considérer |

## Action:

modifie l'élément ou les éléments de l'objet fournis avec le(s) masque(s) de modification(s) de la classe "classe" ou d'indice "classe"
pour modifier tous les éléments de type donné d'un objet cf @mask2obj

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @modifie(@monobj,C5) | modifie les éléments de l'objet contenu dans la variable "monobj" par les masques de modification de classe parcelle (classe 5) |
| @modifie(@monpt,5) | modifie le point dans la variable "monpt" par le masque de modification de point d'indice 5 (le 6° masque) |
| @modifie(@monobj,5) | erreur : un objet ne peut être modifié que par les masques de modification de classe, un indice ne fournit pas les données suffisantes pour modifier tous les éléments d'un objet qui peuvent être de différente nature. |

# @modulo

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | valeur1 |
| val2 | STANDARD | valeur2 |

## Action:

fournit val1 modulo val2
la fonction attends des valeurs entières comme paramètres sous forme décimale, octale ou hexadécimale

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <val1/val2> | NUM | founit la valeur val1 après division par la valeur val2 |

## Exemples:

| | |
|---|---|
| @modulo(7,3) | renvoie 1 |
| @modulo(0x11,8) | renvoie 1 |
| @modulo(7,−3) | renvoie 1 |

# @msgbox

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| message | STANDARD | texte du message à transmettre à l'utilisateur (maxi 2048 car) |
| boutons | STANDARD | code des boutons à faire apparaitre:<br>1 = MB_OKCANCEL<br>2 = MB_YESNO<br>3 = MB_YESNOCANCEL<br>autre = MB_OK |

## Action:

déclenche l'ouverture d'une boite de message avec les boutons standard

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <choix> | NUM | numéro du bouton cliqué (1 ou 2), tout autre choix (CANCEL, abandon) renvoyant 0 |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |

## Exemples:

| | |
|---|---|
| @msgbox("Voulez vous sauvegarder le document?",2) | ouvre une boite de dialogue : voulez vous sauvegarder le document ? avec comme choix : OUI et NON |

# @msgdlg

*Nb paramètres*
indéfini

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| prompt | STANDARD | texte du prompt (phrase interrogative) |
| msg1 | STANDARD | texte du choix n°1 |
| msg2 | STANDARD | texte du choix n°2 |
| msg3 | STANDARD | texte du choix n°3 |
| ...etc | | |

## Action:

déclenche l'ouverture d'une boite de dialogue de choix de TopoCad avec des boutons de choix au bas de la fenètre et un prompt dans la barre de statut.
Bien que cette commande puisse être lancée sans la présence du contexte document/vue, il est cependant préférable que ce dernier soit présent, le prompt étant succeptible d'être écrasé par un autre message si le document/vue n'est pas présent.

Le nombre de paramètres peut varier de 2 à 6

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <choix> | NUM | numéro du choix effectué ou 0 si ESC a été frappé. |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |

## Exemples:

```
@msgdlg("Voulez vous sauvegarder le
document?","OUI","NON")
```

# @mul

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | valeur 1 |
| val2 | STANDARD | valeur 2 |

## Action:

val1*val2

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <val1*val2> | NUM | résultat |

## Exemples:

| | |
|---|---|
| @mul(2,@fact) | multiplie le contenu de la variable "fact" par 2 |

# @multiprintbegin

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

Commande d'impression multiple :
Il est possible d'imprimer plusieurs plans dans un même Job d'imprimante par une série de commandes de multiimpression valides que par TED.
L'utilisateur utilise des séquences d'impression pour imprimer une page à la fois : pour cela l'utilisateur dispose de plusieurs commandes qui doivent se suivre

1) Commencer la tache de multi−impression par *@multiprintbegin*. Cela a pour effet de vérouiller la fenêtre principale de l'application et de faire apparaitre une boite de dialogue permettant de mettre à tout moment fin à la tache d'impression

2) Effectuer les impressions de page de plan par *@multiprintmap* ou *@multiprintmapat* ou des impressions d'images par *@multiprintimage*

3) Terminer la tache de multi−impression par *@multiprintend*. cela a pour effet de valider a nouveau la fenêtre principale de l'application et de fermer la boite de dialogue qui permettait d'interrompre la tache d'impression, et enfin d'envoyer le travail (Job) à faire à l'imprimante.

L'impression multiple permet d'automatiser et surtout de considérablement accélérer les impressions multiples de plans et surtout de pouvoir interrompre l'ensemble de l'impression, ce qui n'est pas possible si on lance une série de @print car une interruption n'interrompt que le @print en cours car chaque @print est un Job distinct pour l'imprimante.

En cas de séquencement incorrect (deux @multiprintbegin à la suite par exemple), un message d'erreur est envoyé et le programme TED est arrété.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| OK | STR | impression réalisée |
| Impossible d'imprimer | ERR | impression non réalisée |

## Exemples:

```
@multiprintbegin,
@multiprintmap(@objet1),
@multiprintimage(D:\BAL\IMAGE_OBJET1.TIF,"Objet 1 − Détail",1),
@multiprintmap(@objet2),
@multiprintimage(D:\BAL\IMAGE_OBJET2.BMP,"Objet 2 − Détail",0),
@multiprintend,
@return(ok)
```

exemple de programme utilisant les séquences d'impression multiple

# @multiprintend

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

termine une séquence de multi−impression et rend la main à l'utilisateur.
cf @*multiprintbegin* pour une explication du séquencement des impressions multiples

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| OK | STR | impression réalisée |
| Erreur d'impression | ERR | |

## Exemples:

@multiprintend

# @multiprintimage

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| file | STD | chemin et nom complet d'un fichier image à imprimer sur une page |
| info | STD | chaine de caractère à imprimer en début de page (cadré en haut à gauche). Peut être nulle si impression d'entete non désiré. |
| type | STD | 0 = fichier BMP<br>1 = fichier TIF |

## Action:

imprime une image provenant d'un fichier BMP ou TIF au centre d'une page (en redimensionnant cette dernière pour tenir dans la page sans déformer les proportions) et optionnellement une chaine de caractère en entête de cette page.
cf *@multiprintbegin* pour une explication du séquencement des impressions multiples

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | impression réalisée |
| Impossible d'imprimer | STR | impression non réalisée |

## Exemples:

@multiprintimage(D:\BAL\IMAGE.BMP,"",0) imprime le fichier image IMAGE.BMP

# @multiprintmap

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | OBJ | |
| | PT | |
| eltobj | LI | objet ou élément à imprimer |
| | FC | |
| | EC | |

## Action:

repositionne la fenêtre en centrant sur l'élément ou l'objet fourni et imprime avec les paramètres d'impression courant.
si eltobj est NULL (pointeur nul) ou si eltobj="0", aucune erreur n'est transmise et la procédure se contente d'imprimer à la position courante de la fenetre
cf @*multiprintbegin* pour une explication du séquencement des impressions multiples

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | impression réalisée |
| Impossible d'imprimer | STR | impression non réalisée |

## Exemples:

| | |
|---|---|
| @mutiprintmap(@vobj) | imprime l'objet contenu dans "vobj" |
| @setvar(vdummy,0,OBJ),<br>@mutiprintmap(@vdummy) | imprime la fenêtre courante |
| @multiprintmap(0) | imprime la fenetre courante |

# @multiprintmapat

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| X | STANDARD | coordonnée X du milieu de la fenêtre plan d'impression ( en coordonnées réelles et vis à vis de la couche de travail) |
| Y | STANDARD | coordonnée Y du milieu de la fenêtre plan d'impression. |

## Action:

repositionne la fenêtre en centrant aux coordonnées fournies et imprime avec les paramètres d'impression courant.
cf *@multiprintbegin* pour une explication du séquencement des impressions multiples

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | impression réalisée |
| Impossible d'imprimer | STR | impression non réalisée |

## Exemples:

@multiprintmapat(796616.16,340581.62,1) imprime aux coord X=796616.16 et Y=340581.62

# Aide non implémentée:

ce fichier vient d'être créé par TopoCad

# @mulvar

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |
| value | STANDARD | valeur à multiplier |

## Action:

[nomvar] <== [nomvar] * value

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| impossible de multiplier une variable pointeur | ERR | si la variable n'est pas standard |
| [nomvar] | NUM | résultat (valeur de la variable après multiplication) |

## Exemples:

| | |
|---|---|
| @mulvar(produit,@fact) | multiplie le contenu de la variable "produit" par le contenu de la variable "fact" et l'energistre dans la variable "produit" et le renvoie |

# @nbcadreformatpage

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le nombre de cadres du format de page courant

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <NbCadres> | NUM | nombre de cadres du format de page courant |

## Exemples:

| | |
|---|---|
| @nbcadreformatpage | renvoie 5, le nombre de cadre pour le format de page courant (croquis A4) |

# @nbclasses

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le nombre de classes (ou types d'objet) du modèle (appelé aussi SCD). Ce nombre comprend la "non classe".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [nbclasses] | NUM | |

## Exemples:

| | |
|---|---|
| @nbclasses | renvoie 25 sur un document neuf et le modèle "PCI" par défaut |

# @nbcouches

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le nombre de couches du document.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [nbcouches] | NUM | |

## Exemples:

| | |
|---|---|
| @nbcouches | renvoie 3 sur un document neuf |

# @nbeltforelt

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | PT, LI, FC, ou EC | élement à tester |
| telt | STANDARD | Type d'élément à rechercher<br>1 = points<br>2 = liaisons<br>4 = écritures<br>16 = faces<br>256 = signe de mitoyenneté<br>512 = déport d'écriture |
| classe | STANDARD | classe de l'élément à rechercher ou −1 si toute classe est à considérer (pour comptage) |
| nivsel | STANDARD | niveau de sélection à considérer pour l'élément à rechercher (pour comptage) ou −1 si tout est à rechercher |

## Action:

examine l'élément et compte le nombre d'éléments de type "telt" en relation avec celui−ci : seuls sont considérés éventuellement les éléments sélectionnés au niveau "nivsel" et qui sont de classe "classe"
renvoie le nombre d'éléments qui répondent aux conditions

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nb éléments] | NUM | nombre d'éléments en relation |

## Exemples:

| | |
|---|---|
| @nbeltforelt(@lali,1,−1,−1), | renvoie toujours 2 quel que soit la liaison "lali" car une liaison est en relation toujours avec 2 points. |
| @nbeltforelt(@lepoint,2,5,0), | renvoie le nombre de liaisons de classe parcelle sélectionnées au niveau 0 attachées au point "lepoint". |

# @nbeltforobj

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'objet à tester |
| obj | ou | ou |
| | OBJ | objet à tester |
| classe | STANDARD | classe des éléments concernée par la sélection (0 à NbTObj−1). Mettre (−1) pour considérer toutes classes. |
| nivsel | STANDARD | niveau de sélection à positionner pour les éléments de l'objet qui correspondent. Mettre (−1) si aucune sélection ne doit être faite. |
| telt | STANDARD | Type d'élément de l'objet à considérer : combinaison binaire des valeurs :<br>1 = points<br>2 = liaisons<br>4 = écritures<br>16 = faces |

## Action:

examine les éléments de type "telt" de l'objet fourni et éventuellement les sélectionne au niveau "nivsel" s'ils sont de classe "classe"
renvoie le nombre d'éléments qui répondent aux conditions (qui sont sélectionnés dans le cas ou on choisit de sélectionner ces éléments)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nb éléments] | NUM | nombre d'éléments (sélectionnés) |

## Exemples:

| | |
|---|---|
| @nbeltforobj(varobj,−1,−1,16), | donne le nombre de faces de l'objet contenu dans la variable "varobj" |
| @nbeltforobj(varobj,−1,0,23), | sélectionne tous les éléments de l'objet au niveau 0 et renvoie le nombre d'éléments total de l'objet |

# @nbformesface

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit le nombre de formes de faces de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de formes de faces |

## Exemples:

| | |
|---|---|
| @nbformefaces | renvoie le nombre de formes de faces |

# @nbformesliaison

*Nb paramètres*
0

## Entrée:

*Nom* *Type* *Commentaire*

## Action:

fournit le nombre de formes de liaisons de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de formes de liaisons |

## Exemples:

| | |
|---|---|
| @nbformeliaisons | renvoie le nombre de formes de liaisons |

# @nbformespoint

*Nb paramètres*
0

## Entrée:

*Nom* *Type* *Commentaire*

## Action:

fournit le nombre de formes de points de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de formes de points |

## Exemples:

| | |
|---|---|
| @nbformespoint | renvoie le nombre de formes de points |

# @nbobjforelt

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément à tester |
| elt | ou | ou |
| | PT, LI, FC, EC | élément à tester |
| classe | STANDARD | classe (type) des objets concerné par la sélection (0 à NbTObj−1). Mettre (−1) pour considérer toutes classes. |
| nivsel | STANDARD | niveau de sélection à positionner pour les objets de l' élément qui correspondent. Mettre (−1) si aucune sélection ne doit être faite. |

## Action:

examine les objets de type "classe" de l'élément fourni et éventuellement les sélectionne au niveau "nivsel" (c'est à dire sélectionne les éléments des objets).
renvoie le nombre d'objets qui répondent aux conditions (qui sont sélectionnés dans le cas ou on choisit de sélectionner ces objets)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nb objets] | NUM | nombre d'objets (sélectionnés) |

## Exemples:

| | |
|---|---|
| @nbobjforelt(varelt,−1,−1), | donne le nombre d'objets qui contienne l'élément contenu dans la variable "varelt" |
| @nbobjforelt(varelt,−1,0), | sélectionne tous les éléments des objets qui contiennent l'élément "varelt" au niveau 0 et renvoie le nombre d'objets total de l'élément |

# @nbpatterns

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit le nombre de motifs courant de l'application

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de motifs |

## Exemples:

| | |
|---|---|
| @nbpatterns | renvoie le nombre de rmotifs |

# @nbrelsem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de relation ou −1 si on désire tout compter |

## Action:

fournit le nombre de relations d'un type donné ou non du document.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N | NUM | renvoie le nombre de relations |

## Exemples:

| | |
|---|---|
| @nbrelsem(7) | renvoie le nombre de relations Parcelle−>Subd de section |

# @nbscriptcmd

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

Renvoie le nombre de commandes de scripts TED de l'application.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| N | NUM | nombre de commandes scripts |

## Exemples:

| | |
|---|---|
| @nbscriptcmd | nombre de commandes disponibles |

# @nbtrel

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie le nombre de types de relation sémantique du modèle (appelé aussi SCD). Ce nombre comprend la "correspondance".

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [nbtrel] | NUM | |

## Exemples:

| | |
|---|---|
| @nbtrel | renvoie 10 sur un document neuf et le modèle "PCI" par défaut |

# @neg

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val | STANDARD | valeur |

## Action:

−nomvar
Attention: comportement particulier si la valeur est de type angulaire : 10gv => neg => 390gv

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <−val> | NUM | résultat |

## Exemples:

| | |
|---|---|
| @neg(@mavar) | fournit <−mavar> |

# @negvar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |

## Action:

[nomvar] <== [−nomvar]
Attention: comportement particulier si la valeur est de type angulaire : 10gv => neg => 390gv

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| impossible de calculer l'opposé d'une variable pointeur | ERR | si la variable n'est pas standard |
| [nomvar] | NUM | résultat (valeur de la variable après négation) |

## Exemples:

| | |
|---|---|
| @negvar(mavar) | fournit <−mavar> |

# @newdoc

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

Crée un nouveau document vide, ouvre une fenetre plan de ce dernier. Ce document devient le document courant du programme TED, la vue plan devient la vue courante du programme TED.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| 1 | NUM | document et vue créés |
| 0 | NUM | erreur |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Erreur systeme | ERR | nb de templates incorrect |

## Exemples:

@newdoc

# @next

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | Nom d'une variable de type PT, LI, FC,EC,MAPDOC,PLANVIEW,NUM ou STR |
| couche | STANDARD | couche concernée (0 à NbCouches) ou −1 si ttes les couches sont concernées (donc élément suivant du document) |

## Action:

incrémente la variable donnée, s'il s'agit d'une variable contenant un élément, fournit l'élément suivant dans la même couche ou dans le document (si couche ==−1)
Il n'est pas controlé si "couche" est bien la couche de l'élément dont il faut prendre le suivant en cas de recherche sur la même couche

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | ??? | renvoie la variable incrémentée |

## Exemples:

| | |
|---|---|
| @next(var,−1) | fournit l'élément suivant celui se trouvant dans la variable "var" |

# @nextobjet

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable contenant l' objet |
| classe | STANDARD | classe de l'objet qu'il faut choisir ou 0 si toutes les classes sont acceptées et −1 si seulement objets composés d'éléments de différentes classes |
| couche | STANDARD | couche de l'objet qu'il faut choisir ou −1 si toutes les couches sont acceptées |

## Action:

fournit l'objet suivant (au niveau du document entier) de classe "classe" (si classe différent de 0) et de couche "couche" (si couche différent de −1), si aucun objet n'est trouvé (fin de liste), alors renvoie un pointeur nul.
la variable est incrémentée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | OBJ | renvoie la variable après incrémentation |

## Exemples:

| | |
|---|---|
| @nextobjet(varobj,0,−1) | renvoie le prochain objet du document |

# @nextobservation

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation courante |
| num | STD | numero d'observation suivante à rechercher ou 0 si l'observation suivante de numero quelconque est à fournir |

## Action:

recherche la prochaine observation dont le résultat a pour numero "num", ou recherche simplement l'observation suivante si "num"==0

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <obs> | OBS | l'observation ou NULL si aucune trouvée |

## Exemples:

| | |
|---|---|
| @nextobservation(@curobs,127) | fournit la prochaine observation de numero 127 |
| @nextobservation(@curobs,0) | fournit l'observation suivante |

# @nextrelsem

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD ou PT | 0 |
| eltobj | LI FC EC OBJ | ou élement ou objet dont il faut trouver la prochaine relation (de type donné) |
| type | STANDARD | type de relation recherchée (si type==−1 alors toutes les relations sont considérées) |
| rel | RELSEM | relation au dela de laquelle rechercher (dans toutes les relations de l'élément ou objet) |

## Action:

si eltobj= "0" alors fournit la prochaine relation du document après celle désignée par "rel"
sinon fournit la prochaine relation d'un élément ou objet après celle désignée par "rel".
Si type est différent de −1, alors fournira la prochaine du type donné uniquement

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <relation> | RELSEM | renvoie la relation ou NULL si non trouvé |

## Exemples:

| | |
|---|---|
| @nextrelsem(@obj,7,@currentrel) | renvoie la prochaine relation de type parcelle−>subdiv de sect de l'objet désigné par la variable "obj" après la relation désignée par currentrel. Renvoie une erreur si la variable "obj" ne contient pas un objet et NULL si la relation "currentrel" n'est pas relative à l'objet. |

# @nextwithelt

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à incrémenter qui peut être de type PT, LI, FC,OBJ |
| nomelt | PT,LI,FC,EC | élément |

## Action:

fournit l'élément suivant celui pointé par "nomvar" et qui est en relation avec l'élément "nomelt"

si nomvar est PT, nomelt ne peut être que LI
si nomvar est LI, nomelt ne peut être que PT ou FC
si nomvar est FC, nomelt ne peut être que LI
si nomvar est OBJ, nomelt ne peut etre que PT, LI, FC, ou EC

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | PT,LI,FC,OBJ | élément ou objet suivant (valeur de la variable après "incrémentation") |

## Exemples:

@nextwithelt(var,@elt)

# @nextwithmask

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à incrémenter qui peut être de type PT, LI, FC ou EC |
| mask | STANDARD | nom de la variable masque qui peut être de type MASKCHPT, MASKCHLI, MASKCHFC, ou MASKCHEC |
| couche | STANDARD | couche à scruter ou toutes si −1 |

## Action:

fournit l'élément suivant de la même couche ou de toutes les couches répondant au masque de recherche donné
Il n'est pas controlé si "couche" est bien la couche de l'élément dont il faut prendre le suivant en cas de recherche sur la même couche.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | PT,LI,FC ou EC | élément suivant (valeur de la variable après incrémentation) |

## Exemples:

@nextwithmask(var,monmasque)

# @nextwithobj

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à incrémenter qui peut être de type PT, LI, FC ou EC |
| nomobj | OBJ | objet à scruter |

## Action:

fournit l'élément suivant celui pointé par "nomvar" et qui appartient à l'objet "nomobj"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | PT,LI,FC ou EC | élément suivant (valeur de la variable après "incrémentation") |

## Exemples:

| | |
|---|---|
| @nextwithobj(var,@obj) | var est le nom d'une variable de type EC contenant une écriture de l'objet.<br>obj est le nom d'une variable objet<br>Cette fonction recherche l'écriture suivante de l'objet |

# @not

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | NUM, PT, LI, FC, EC ou OBJ | fournit la négation du paramètre (opérateur booléen inverse) |

## Action:

opérateur booléen NOT (logique)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| NOT parm1 | BOOL | renvoie 1 ou 0 |

## Exemples:

| | |
|---|---|
| @not(@var) | renvoie la négation de la variable |

# @num

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| value | STANDARD | renvoie la valeur fournie sous forme numérique |

## Action:

permet de convertir une valeur chaine ou standard explicitement en valeur numérique.
une chaine de caractère (ex: "coucou") convertie en valeur numérique et utilisée comme tel sera équivalente à la valeur 0
une chaine de caractère (ex: "coucou") convertie en valeur numérique puis reconvertie en chaine de caractère par affectation sur une variable de type STR redonnera la même chaine ("coucou")

attention: une chaine représentant un entier devrait toujours être convertie avec @numi au lieu de @num car on peut fournir un entier sous forme octale ou hexadécimale.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| value | NUM | conversion effectuée |

## Exemples:

| | |
|---|---|
| @num(3) | renvoie la valeur numérique '3' |
| @setvar(machaine,75,STR),<br>@num(@machaine) | renvoie la valeur numérique '75' |
| @setvar(machaine,"67.879",STR),<br>@setvar(var1,@machaine,NUM), | var1 est la conversion en numérique de 'machaine' |
| @setvar(machaine,"coucou",STR),<br>@setvar(var1,@machaine,NUM),<br>@out(@equ(@var1,@num(0))),<br>@setvar(var2,@var1,STR),<br>@out(@var2) | var1 est la conversion en numérique de 'machaine'<br>la première sortie renvoie '1' car la conversion numérique de "coucou" est égale à 0.<br>la seconde sortie (de var2) renvoie "coucou" |

# @numi

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| value | STANDARD | renvoie la valeur fournie sous forme numiérique |

## Action:

permet de convertir une valeur chaine ou standard explicitement en valeur numiérique décimale. Contrairement à la foinction @num, la chaine désigne ici un entier sous forme décimal, octal (0...) ou hexadécimal (0x...)
une chaine de caractère (ex: "coucou") convertie en valeur numiérique et utilisée comme tel sera équivalente à la valeur 0
une chaine de caractère (ex: "coucou") convertie en valeur numiérique puis reconvertie en chaine de caractère par affectation sur une variable de type STR donnera la chaine ("0")

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| value | numi | conversion effectuée |

## Exemples:

| | |
|---|---|
| @numi(3) | renvoie la valeur numiérique '3' |
| @numi(0x0000FF) | renvoie la valeur numérique '255' |
| @numi(coucou) | renvoie la valeur numerique '0' |

# @objintegrity

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

Recalcule les informations concernant l'objet dont notamment la couche en fonction de ses éléments. Cette information n'étant pas mise à jour en permanence, il convient lorsque on l' utilise de lancer au préalable cette procédure. Un objet dont les éléments appartiennent à des couches différentes est un objet non intègre : le numero de couche renvoyé alors de l'objet est −1 indiquant un objet non intègre.
La procédure renvoie le nombre d'objets non intègres.

NB: les informations de sélection Select, User, et Attributs qui ne sont pas non plus mises à jour en permanence sont également mises à jour en fonction des éléments lors de cette opération : Un objet est sélectionné si tous ses élements le constituant sont sélectionnés

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <nb> | NUM | nombre d'objets non intègres |

## Exemples:

@objintegrity

# @objsansclasse

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe concernée à scruter (de 1 à NbTObj−1) |
| nivsel | STANDARD | niveau de sélection à positionner (de 0 à 31) |
| strict | STANDARD | indique si on doit tester si la classe de l'élément est égale (1) ou de niveau supérieur ou égal (0) par rapport au type d'élément (face, liaison...) |

## Action:

recherche tous les eléments appartenant à un objet et qui ne respectent pas la classe de l'objet (ne sont pas de même classe ou de classe supérieure) ou la nature de l'objet (une face pour un objet linéaire par ex) et les sélectionne.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N éléments ont été sélectionnés | NUM | N = Nombre d'éléments en anomalie |

## Exemples:

| | |
|---|---|
| @objsansclasse(0,5,0,1) | recherche les éléments des objets parcelles sur la couche 0 n'étant pas de classe parcelle et les sélectionne au niveau 0 |

# @observationcalcule

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation à calculer |
| dest | PT | point destination |

## Action:

Effectue un calcul de point à partir de l'observation "obs".
Si "dest" est NUL, alors le point résultat de l'observation est le point à modifier s'il existe déjà ou à créer s'il n'existe pas.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| −3 | NUM | l'observation est non déterminative |
| −2 | NUM | un point nécessaire au calcul de l'observation n'a pas été trouvé |
| −1 | NUM | calcul impossible (ex: droites parallèles) |
| 0 | NUM | calcul effectué et résultat dans "dest" |

## Exemples:

```
@setvar(curobs,@addobservation( "Rayonnt
37320,A=37319,B=37318,C= 37318,Distance CX=
5.800,Angle/AB=200.0000gv,"),OBS),
@setvar(monpt,@getpoint(1),PT),
@observationcalcule(@monobs, @monpt)
```

calcule l'observation 37320 et transmet le résultat du calcul dans les coordonnées de "monpt" de numéro 1.

```
@setvar(monpt,0,PT),
@observationcalcule(@monobs, @monpt)
```

recherche si le point 37320 existe, s'il existe, modifie ses coordonnées avec le résultat du calcul de l'observation, sinon crée un point de numéro 37320 résultat de l'observation. "monpt" est toujours initialisé à 0.

# @observationconcerne

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation à calculer |
| num | STD | numero de point à tester |

## Action:

Détermine si un numero de point rentre en compte dans le calcul d'une observation..

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 3 | NUM | constitue le point résultat et un des points utilisé pour le calcul (observation récursive) |
| 2 | NUM | constitue le point résultat du calcul |
| 1 | NUM | constitue un des points utilisés pour le calcul |
| 0 | NUM | ne rentre pas en ligne de compte pour le calcul |

## Exemples:

| | |
|---|---|
| @observationconcerne(@monobs,127) | indique si le point 127 est utilisé dans l'observation "monobs" |

# @observationtype

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS | observation concernée |

## Action:

Détermine le type d'une observation

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 | NUM | PtAligne |
| 1 | NUM | Perpe |
| 2 | NUM | Proportion |
| 3 | NUM | Projection |
| 4 | NUM | Rayonnt |
| 5 | NUM | Bilateration |
| 6 | NUM | Intersection |
| 7 | NUM | Relevement |
| 8 | NUM | CercleDroite |
| 9 | NUM | TriangleDDA |
| 10 | NUM | Paralleles |
| 11 | NUM | Visee |
| 12 | NUM | Mesurage |

## Exemples:

| | |
|---|---|
| @observationtype(@monobs) | renvoie le type d'observation de "monobs" |

# @or

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm1 | STANDARD | premier paramètre |
| parm2 | STANDARD | second paramètre |

## Action:

renvoie (parm1 OU parm2)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [parm1 OU parm2] | BOOL | renvoie 0 ou 1 |

## Exemples:

| | |
|---|---|
| @or(5,0) | renvoie 1 (valeur logique TRUE) |
| @or(test,0) | renvoie 0 (FALSE) car "test" equivaut à FALSE et 0 = FALSE |

# @ordonne

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche à ordonner |
| type | STANDARD | type d'élément à ordonner<br>EltPoint =1<br>EltLiaison = 2<br>EltFace = 16<br>EltEcriture = 4 |
| name | STANDARD | nom de l'Xdata donnant l'ordonancement |

## Action:

cette fonction ordonne les éléments en fonction des données de chaque élément de nom "name". Si name n'est pas fourni (chaine vide), c'est l'étiquette de l'élément qui servira à ordonner les éléments.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | les éléments ont été ordonnés |

## Exemples:

| | |
|---|---|
| @ordonne(0,1,"") | ordonne les points de la couche 0 d'après leur étiquette |
| @ordonne(0,1,ORDRE) | ordonne les points de la couche 0 d'après les données XData des éléments de nom "ORDRE" |

# @orior

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie l'orientation d'origine du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OriOr | ANGLE | renvoie dans le sens et unité courante sans les suffixes (ex : "234.7624") |

## Exemples:

@orior

# @out

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | chaine de caractère à écrire dans le fichier ERRORxxx.LOG |
| | ou | |
| parm | PT, LI, FC, OBJ, MASK... | ou valeur d'une variable pointeur à écrire dans le fichier ERRORxxx.LOG |

## Action:

ecrit une chaine de caractère dans le fichier des messages et erreurs en cours suivi de CR+LF

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| parm | ??? | le paramètre fourni est renvoyé |

## Exemples:

| | |
|---|---|
| @out(@var) | sortie de la valeur de var dans le fichier des erreurs |

# @param

*Nb paramètres*
indéfini

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| param1 | STD | nom du paramètre 1 |
| param2 | STD | nom du paramètre 2 |
| ....etc | | |

## Action:

Cette fonction affecte des noms aux paramètres utilisés dans une sous procédure TED.
La fonction doit être la première d'une sous procédure afin de déclarer ces noms avant toute affectation de variable, elle controle ainsi la bonne liaison entre les procédures entre elles (que leurs nombre de paramètres soit identique).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Nombre de paramètres incorrect | ERR | le nombre de paramètres transmis diffère ou bien @param n'est pas utilisé comme première instruction |
| Ok | STD | affectation réussie |

## Exemples:

```
; programme1
@exec(programme2.ted,"ligne de  texte")
;programme2
@param(nom_interne),
@out(@nominterne),
@out(@nominterne),
@out(@nominterne)
```

exemple de deux programmes : un principal (1) appelant un autre (2) chargé d'afficher 3 fois ce qu'on lui donne en paramètre

# @paste

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STD | couche destinatrice |
| type | STD | 0 = transfert brut<br>1 = transfert par la transformation courante<br>2 = transfert par l'inverse de la transformation courante |

## Action:

colle le contenu du presse papier dans la couche désignée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| couche vérouillée | ERR | si la couche réceptrice n'accepte pas de données |
| OK | STR | collage réussi |

## Exemples:

| | |
|---|---|
| @paste(@couchetravail,0) | colle le presse papier dans la couche courante |

# @popline

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom d'un fichier texte |
| varname | STANDARD | nom de la variable à initialiser avec la ligne recherchée |
| varindex | STANDARD | nom de la variable servant d'index de position du fichier |

## Action:

extrait la ligne à partir de la position donnée par "varindex" d'un fichier de nom "filename". 0 est la première position, 1 le second caractère du fichier ...etc.
la ligne est copiée dans la variable de nom "varname" si la fonction n'échoue pas.La variable de nom "varindex" est incrémentée du nombre de caractères lus jusque à la fin de ligne ou est mise à −1 si c'est la fin de fichier.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| −1 ou −2 ou N | NUM | N => nouvel index du fichier<br>−1 =>fin de fichier<br>−2 =>pas de possibilité de positionner le pointeur de fichier |

## Exemples:

| | |
|---|---|
| @popline(c:\batch.txt,mavar,ind) | fournit la ligne du fichier c:\batch.txt qui est en position donnée par la variable de nom "ind" |

# @print

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | OBJ<br>PT | |
| eltobj | LI<br>FC<br>EC | objet ou élément à imprimer |
| dlg | STD | indique si une boite de dialogue d'impression doit préalablement apparaitre ou bien si les critères d'impression par défaut sont pris pour imprimer directement. |

## Action:

repositionne la fenêtre en centrant sur l'élément ou l'objet fourni et imprime avec les paramètres d'impression courant.
si eltobj est NULL (pointeur nul) ou si eltobj="0", aucune erreur n'est transmise et la procédure se contente d'imprimer à la position courante de la fenetre

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | impression réalisée |
| Impossible d'imprimer | STR | impression non réalisée (pas de document...) |

## Exemples:

| | |
|---|---|
| @print(@vobj,1) | imprime l'objet contenu dans "vobj" |
| @setvar(vdummy,0,OBJ),<br>@print(@vdummy,1) | imprime la fenêtre courante |
| @print(0,0) | imprime la fenetre courante |

# @printat

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| X | STANDARD | coordonnée X du milieu de la fenêtre plan d'impression ( en coordonnées réelles et vis à vis de la couche de travail) |
| Y | STANDARD | coordonnée Y du milieu de la fenêtre plan d'impression. |
| dlg | STD | indique si une boite de dialogue d'impression doit préalablement apparaitre ou bien si les critères d'impression par défaut sont pris pour imprimer directement. |

## Action:

repositionne la fenêtre en centrant aux coordonnées fournies et imprime avec les paramètres d'impression courant.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | impression réalisée |
| Impossible d'imprimer | STR | impression non réalisée (pas de document...) |

## Exemples:

@printat(796616.16,340581.62,1) imprime aux coord X=796616.16 et Y=340581.62

# @projapplygridshift

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| grid | STANDARD | nom de la grille ou des grilles à charger séparés par le caractère '\|' (barre verticale). ce caractère est convertit en virgule avant l'appel à la librairie proj4. Les noms des fichiers doivent être fournis complets avec le chemin. |
| inverse | STANDARD | valeur booleenne indiquant si la grille représente une transformation inverse (source et destination sont alors inversés) |
| X | STANDARD | nom de la variable de type NUM accueillant la coordonnée X du point devant à transformer |
| Y | STANDARD | nom de la variable de type NUM accueillant la coordonnée Y du point devant à transformer |
| Z | STANDARD | nom de la variable de type NUM accueillant la coordonnée Z du point devant à transformer |
| a_srce | STANDARD | valeur du demi grand axe de l'ellipsoide source (en cas de grille utilisant un calcul sur l'ellipsoide) |
| es_srce | STANDARD | valeur de $e^2 = (a^2 - b^2)/a^2$ de l'ellipsoide source (en cas de grille utilisant un calcul sur l'ellipsoide) |
| a_dest | STANDARD | valeur du demi grand axe de l'ellipsoide destination (en cas de grille utilisant un calcul sur l'ellipsoide) |
| es_dest | STANDARD | valeur de $e^2 = (a^2 - b^2)/a^2$ de l'ellipsoide destination (en cas de grille utilisant un calcul sur l'ellipsoide) |

## Action:

transforme les coordonnées du point de coordonnées données par X, Y et Z suivant la grille de transformation donnée
les coordonnées fournies sont données avec la précision de 21 décimales. Il est donc possible d'utiliser la fonction pour effectuer une transformation par grille sur toute sorte de valeurs (coordonnées géographiques, coordonnées cartésiennes, coordonnées en projection.... tout dépendant de la composition de la grille).
La grille est un fichier binaire de format spécifique à TopoCad et ne correspond pas aux grilles de la librairie Proj4. Un utilitaire extérieur permet de créer une grille TopoCad ou de convertir ces différentes types de grille (NTv1,NTv2,IGN...) en une grille TopoCad.
Le type d'interpolation effectué par grille est identique à celle effectuée par la lib proj4 ainsi que par l'IGN, à savoir une interpolation bilinéaire dans la plupart des types de grille.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| variable NUM attendue | ERR | X, Y ou Z ne sont pas des variables de type NUM |
| projection source invalide | ERR | Les paramètres de définition du système source sont invalides ou incomplets |
| projection destination invalide | ERR | Les paramètres de définition du système destination sont invalides ou incomplets |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées (par ex parce que les coordonnées sont en dehors de la zone de définition de la grille...ou pour une autre raison) |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
@setvar(L,@dms2r(3d),NUM),
@setvar(P,@dms2r(45d),NUM),
@setvar(H,0,NUM),
@projapplygridshift(c:\topocad\RGF93.dat,1,L,P,H,@rgf_a,@rgf_es,@ntf_a,@ntf_es)
```

corrige des coordonnees geographiques NTF afin de les avoir dans le système RGF93 à partir d'une grille de transformation (RGF93=>NTF)..

```
@setvar(X,999001.08,NUM),
@setvar(Y,40256.77,NUM),
@setvar(Z,0,NUM),
@projapplygridshift(c:\topocad\Sausheim1.dat,0,L,P,H,0,0,0,0)
```

corrige des coordonnées transformées pour les adapter à partir d'une grille de transformation à un autre système de coordonnées.

# @projdatumtransform

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| projsrce | STANDARD | definition du système géodésique source suivant le standard proj4 |
| projdest | STANDARD | definition du système géodésique destination suivant le standard proj4 |
| L | STANDARD | nom de la variable de type NUM contenant la longitude sous forme d'une valeur exprimée en radian |
| P | STANDARD | nom de la variable de type NUM contenant la latitude sous forme d'une valeur exprimée en radian |
| H | STANDARD | nom de la variable de type NUM contenant l'altitude ellipsoïdale. |

## Action:

transforme un point défini par ses coordonnées géographiques dans le système géodésique "projsrce" en son homologue dans le système géodésique "projdest"
la définition du système géodésique ne doit pas comporter de caractère guillemet " si celle ci est entre guillemets. Les paramètres de projection n'interviennent évidemment pas dans les calculs, seules les différences entre les ellipsoïdes comptent.
les longitudes et latitudes sont fournies en retour sous forme de valeurs en radians avec 21 décimales

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Variable NUM attendue | ERR | L, P ou H ne sont pas des variables de type NUM |
| projection source invalide | ERR | Les paramètres de définition du système source sont invalides ou incomplets |
| projection destination invalide | ERR | Les paramètres de définition du système destination sont invalides ou incomplets |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées (par ex parce que la transformation fait appel à une grille et que les coordonnées sont en dehors de la zone de définition de la grille...ou pour une autre raison) |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
;conversion geo=>cartesienne=>geo
@setvar(L,@dms2r(0d),NUM),
@setvar(P,@dms2r(45d),NUM),
@setvar(H,0,NUM),
@out(DEPART : CLARKE 80 IGN),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
@out(TRANSFORME : GRS80),
@projdatumtransform("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773","+proj=lcc +ellps=GRS80
+towgs84=0,0,0 +x_0=700000 +y_0=6600000 +lon_0=3d
+lat_0=46d30' +lat_1=44d +lat_2=49d",L,P,H),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
@out(RETRANSFORME : CLARKE 80 IGN),
@projdatumtransform("+proj=lcc +ellps=GRS80
+towgs84=0,0,0 +x_0=700000 +y_0=6600000 +lon_0=3d
+lat_0=46d30' +lat_1=44d +lat_2=49d","+proj=lcc
+ellps=clrk80IGN +towgs84=−168,−60,+320 +x_0=600000
+y_0=200000 +lon_0=2d20'14.025 +lat_0=44d06'
+lat_1=43d11'57.44859 +lat_2=44d59'45.93773",L,P,H),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
```

initialise les valeurs d'un point géographique 0dE et 45dN et transforme ce point du système "Clarke 80 IGN" vers "GRS80".

renvoie :

```
DEPART : CLARKE 80 IGN
L= 0dE
P= 45dN
H= 0
TRANSFORME : GRS80
L= 0d0'2.73947"W
P= 44d59'59.85617"N
H= 45.22402352108156265
RETRANSFORME : CLARKE 80 IGN
L= 0dE
P= 45d0'0.00033"N
H= 0.010090210420457879081
```

# @projfwd

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| proj | STANDARD | definition de la projection suivant le standard proj4 |
| L | STANDARD | nom de la variable de type NUM donnant la longitude sous forme de réel en radian |
| P | STANDARD | nom de la variable de type NUM donnant la latitude sous forme de réel en radian |

## Action:

transforme des coordonnées géographiques en coordonnées projetées suivant la définition de la projection fournie.
la définition de la projection ne doit pas comporter de caractère guillemet " si celle ci est entre guillemets.
les coordonnées fournies sont données avec la précision des coordonnees courante.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Ok | STR | La variable donnée par L contient X et la variable donnée par P contient Y en coordonnées projetées |

## Exemples:

```
@setvar(vx,@dms2r(3d),NUM),
@setvar(vy,@dms2r(45d30'),NUM),
@projfwd("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",vx,vy),
@out(@concat("X=",@vx,1)),
@out(@concat("Y=",@vy,1)),
```

affiche les coordonnées X et Y en projection Lambert III du point de coordonnées géographiques longitude 3° E, latitude 45°30' N. soit
X = 651811.871
Y = 355782.357

# @projgeocentric2geodetic

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| a | STANDARD | valeur du demi grand axe de l'ellipsoïde |
| es | STANDARD | valeur de $e^2=(a^2-b^2)/a^2$ |
| X | STANDARD | nom d'une variable de type NUM contenant la coordonnée cartésienne X |
| Y | STANDARD | nom d'une variable de type NUM contenant la coordonnée cartésienne Y |
| Z | STANDARD | nom d'une variable de type NUM contenant la coordonnée cartésienne Z |

## Action:

transforme les coordonnées tridimensionnelles cartésiennes en coordonnées géographiques
Les variables X,Y,Z contiennent à l'issu de la fonction les valeurs L,P,H (longitude, latitude, hauteur ellipsoidale)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Variable NUM attendue | ERR | X, Y ou Z ne sont pas les noms d'une variable de type NUM |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
@setvar(L,@dms2r(3d),NUM),
@setvar(P,@dms2r(45d),NUM),
@setvar(H,0,NUM),
@out(DEPART : COORD GEOGRAPHIQUES),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
@out(TRANSFORME : COORD TRIDIM CARTESIENNES),
@projgeocentric2geodetic(6378249.2,0.006803487646299877488880022769287 4,L,P,H),
@out(@concat("X=",@L,1)),
@out(@concat("Y=",@P,1)),
@out(@concat("Z=",@H,1)),
@out(RETRANSFORME : COORD GEOGRAPHIQUES),
@projgeocentric2geodetic(6378249.2,0.006803487646299877488880022769287 4,L,P,H),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
```

transforme des coordonnées géographiques en coordonnées cartésiennes puis inversement en affichant les résultats.

# @projgeocentric2wgs

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| proj | STANDARD | paramètres de définition de système géographique source selon les règles de proj4 |
| pt | PT | point contenant les coordonnées à transformer |

## Action:

transforme les coordonnées tridimensionnelles cartésiennes du système géodésique donnée vers le système WGS84 ou vers le système pivot si le système pivot n'est pas le WGS84

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Projection invalide | ERR | Les paramètres fournis du système géodésique sont erronés |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

| | |
|---|---|
| @setvar(pt,@creatept(0,0,4511602.51,236443.06,4487057.24),PT), | transforme des coordonnées du point |
| @ProjGeocentric2WGS("+proj=lcc +ellps=clrk80IGN | X=4511602.51 |
| +topivot=−168,−60,+320 +x_0=600000 +y_0=200000 | Y=236443.06 |
| +lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859 | Z=4487057.24 |
| +lat_2=44d59'45.93773",@pt), | renvoie: |
| @out(@concat("X=",@getx(@pt),1)), | X= 4511434.510 |
| @out(@concat("Y=",@gety(@pt),1)), | Y= 236383.060 |
| @out(@concat("Z=",@getz(@pt),1)), | Z= 4487377.240 |

# @projgeod

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| proj | STANDARD | definition de la projection suivant le standard proj4 |
| tab | STANDARD | nom des variables de type NUM servant de tableau d'entrée et de sortie de la fonction<br>En entrée:<br>lam1 = longitude point initial 1 (en radian)<br>phi1 = latitude point initial 1 (en radian)<br>al12 = azimuth 1−>2 (en radian)<br>dist = distance orthodromique 1 −> 2<br>En sortie:<br>lam1 = longitude point initial 1<br>phi1= latitude point initial 1<br>lam2 = longitude point final 2<br>phi2= latitude point final 2<br>al12 = azimuth 1−>2 (en radian)<br>al21 = azimuth 2−>1 (en radian)<br>dist = distance orthodromique 1 −> 2 |

## Action:

Calcul d'une géodésique.
Donne longitude, latitude et azimuth arrière d'un point à partir d'une longitude, latitude, azimuth avant, et distance d'un premier point
'Tab' représente le nom à partir duquel la fonction ira chercher les paramètres d'entrée (4) et fournira ceux de sortie (7).
Par exemple si on fournit 'mavar' à tab, la fonction ira chercher les valeurs dans les variables mavar1,mavar2, mavar3 et mavar4 et rendra le calcul dans les variables mavar1 ;;; à ...mavar7.
Les variables mavar1 à mavar7 doivent exister.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié ou si la chaine de définition de la projection est invalide |
| 4 variables NUM 'n&meN' attendue | ERR | X, Y ou Z ne sont pas des variables de type NUM |
| Impossible d'enregistrer la variable... | ERR | collision de noms de variables |
| Ok | STR | Les variables ont été initialisées avec les résultats |

## Exemples:

```
@setvar(tab1,@dms2r(3d),NUM),
@setvar(tab2,@dms2r(45d),NUM),
@out(@concat(Longitude=,@r2dms(@tab1,WE),1)),
@out(@concat(Latitude=,@r2dms(@tab2,NS),1)),
@setvar(tab3,@dms2r(0d),NUM),
@setvar(tab4,1852.18,NUM),
@setvar(tab5,0,NUM),
@setvar(tab6,0,NUM),
@setvar(tab7,0,NUM),
@out(@concat(Azimuth=,@r2dms(@tab3," "),1)),
@out(@concat(Distance=,@tab4,1)),
@projgeod("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",tab),
@out(@concat(Longitude=,@r2dms(@tab3,WE),1)),
@out(@concat(Latitude=,@r2dms(@tab4,NS),1)),
@return(ok)
```

Calcule la longitude et latitude du point à une distance 1852.18m en direction du nord

```
@setvar(tab1,@dms2r(3d),NUM),
@setvar(tab2,@dms2r(45d),NUM),
@out(@concat(Longitude=,@r2dms(@tab1,WE),1)),
```

Calcule la longitude et latitude du point situé à l'est à une distance de 25 km, où l'on voit que si l'on va tout droit en étant dans l'hémisphère nord dans une direction initiale Est,

```
@out(@concat(Latitude=,@r2dms(@tab2,NS),1)),
@setvar(tab3,@dms2r(90d),NUM),
@setvar(tab4,25000,NUM),
@setvar(tab5,0,NUM),
@setvar(tab6,0,NUM),
@setvar(tab7,0,NUM),
@out(@concat(Azimuth=,@r2dms(@tab3," "),1)),
@out(@concat(Distance=,@tab4,1)),
@projgeod("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",tab),
@out(@concat(Longitude=,@r2dms(@tab3,WE),1)),
@out(@concat(Latitude=,@r2dms(@tab4,NS),1)),
@return(ok)
```

on se retrouve plus au Sud (le plan dirigé Est coupe l'équateur)

# @projgeodetic2geocentric

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| a | STANDARD | valeur du demi grand axe de l'ellipsoïde |
| es | STANDARD | valeur de $e^2=(a^2-b^2)/a^2$ |
| L | STANDARD | nom d'une variable de type NUM contenant la longitude |
| P | STANDARD | nom d'une variable de type NUM contenant la latitude |
| H | STANDARD | nom d'une variable de type NUM contenant la hauteur ellipsoidale |

## Action:

transforme les coordonnées géographiques en coordonnées tridimensionnelles cartésiennes
Les variables L,P,H contiennent à l'issu de la fonction les valeurs X,Y,Z cartésiennes.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Variable NUM attendue | ERR | L, P ou H ne sont pas les noms d'une variable de type NUM |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
@setvar(L,@dms2r(3d),NUM),
@setvar(P,@dms2r(45d),NUM),
@setvar(H,0,NUM),
@out(DEPART : COORD GEOGRAPHIQUES),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
@out(TRANSFORME : COORD TRIDIM CARTESIENNES),
@projgeodetic2geocentric(6378249.2,0.006803487646299877488880022769287 4,L,P,H),
@out(@concat("X=",@L,1)),
@out(@concat("Y=",@P,1)),
@out(@concat("Z=",@H,1)),
@out(RETRANSFORME : COORD GEOGRAPHIQUES),
@projgeocentric2geodetic(6378249.2,0.006803487646299877488880022769287 4,L,P,H),
@out(@concat("L=",@r2dms(@L,"EW"),1)),
@out(@concat("P=",@r2dms(@P,"NS"),1)),
@out(@concat("H=",@H,1)),
```

transforme des coordonnées géographiques en coordonnées cartésiennes en affichant les résultats.

# @projgetgridshift

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| grid | STANDARD | nom de la grille ou des grilles à charger séparés par le caractère '\|' (barre verticale). ce caractère est convertit en virgule avant l'appel à la librairie proj4. Les noms des fichiers doivent être fournis complets avec le chemin. |
| X | STANDARD | nom de la variable de type NUM contenant la coordonnée X du point intérieur à la grille |
| Y | STANDARD | nom de la variable de type NUM contenant la coordonnée Y du point intérieur à la grille |
| V | STANDARD | substantif des noms des variables recevant les paramètres recherchés de la grille |

## Action:

cette commande recherche dans la grille les paramètres aux coordonnées X,Y.
les paramètres sont enregistrés dans les variables de noms formés du nom fourni suivi d'un nombre de 1 à N
Ces variables doivent exister : si V="param" alors les variables "param1", "param2"...doivent exister

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| variable NUM attendue | ERR | X, Y ou Z ne sont pas des variables de type NUM |
| projection destination invalide | ERR | Les paramètres de définition du système destination sont invalides ou incomplets |
| ERREUR TRANSFORMATION SYSGEOD: coordonnées hors grille. | ERR | Les coordonnées sont en dehors de la zone de définition de la grille. |
| Impossible d'enregistrer la variable... | ERR | collision de noms de variables |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
; 3d0E / 45d0N
@setvar(L,@dms2r("3d0'0.0E"),NUM),
@setvar(P,@dms2r("45d0'0.0N"),NUM),
@setvar(H,0,NUM),

@setvar(ntf_a,6378249.2,NUM),
@setvar(ntf_es,0.006803487647360856976,NUM),
@setvar(wgs_a,6378137,NUM),
@setvar(wgs_es,0.006694380035531086854 4,NUM),

@out("TEST NTF => WGS84"),
@out("================="),
@setvar(vX,@L,NUM),
@setvar(vY,@P,NUM),
@setvar(vZ,@H,NUM),
@setvar(v1,0,NUM),
@setvar(v2,0,NUM),
@setvar(v3,0,NUM),
@projgeodetic2geocentric(@ntf_a,@ntf_es,vX,vY,vZ),
@projgetgridshift(c:\test\ign.dat,L,P,V),
@subvar(vX,@V1),
@subvar(vY,@V2),
@subvar(vZ,@V3),
@projgeocentric2geodetic(@wgs_a,@wgs_es,vX,vY,vZ),
@out(@concat(X=,@r2dms(@vX,"EW"),1)),
@out(@concat(Y=,@r2dms(@vY,"NS"),1)),
@out(@concat(Z=,@vZ,1)),
```

corrige des coordonnees geographiques NTF afin de les avoir dans le système RGF93 à partir de la grille de transformation de l'IGN.

resultat obtenu
2d59'57.65692"
44d59'59.90220"

# @projinv

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| proj | STANDARD | definition de la projection suivant le standard proj4 |
| X | STANDARD | nom de la variable de type NUM donnant la coordonnée X en projection |
| Y | STANDARD | nom de la variable de type NUM donnant la coordonnée Y en projection |

## Action:

transforme des coordonnées projetées en coordonnées géographiques suivant la définition de la projection fournie.
la définition de la projection ne doit pas comporter de caractère guillemet " si celle ci est entre guillemets.
les angles en radians fournis sont donnés avec 21 décimales

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Ok | STR | La variable donnée par X contient la longitude L et la variable donnée par Y contient la latitude P |

## Exemples:

```
@setvar(vl,651811.871,NUM),
@setvar(vp,355782.357,NUM),
@projinv("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",vl,vp),
@out(@concat("L=",@r2dms(@vl),1)),
@out(@concat("P=",@r2dms(@vp),1)),
```

affiche les longitude et latitude du point dont les coordonnées X et Y en projection Lambert III sont 651811.871 et 355782.357 soit
L=3° E
P=45°30' N.

# @projinvgeod

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| proj | STANDARD | definition de la projection suivant le standard proj4 |
| tab | STANDARD | nom des variables de type NUM servant de tableau d'entrée et de sortie de la fonction<br>En entrée:<br>lam1 = longitude point initial 1 (en radian)<br>phi1 = latitude point initial 1 (en radian)<br>lam2 = longitude point final 2 (en radian)<br>phi2 = latitude point final 2 (en radian)<br>En sortie:<br>lam1 = longitude point initial 1<br>phi1= latitude point initial 1<br>lam2 = longitude point final 2<br>phi2= latitude point final 2<br>al12 = azimuth 1−>2 (en radian)<br>al21 = azimuth 2−>1 (en radian)<br>dist = distance orthodromique 1 −> 2 |

## Action:

Calcul d'une géodésique.
Donne l'azimuth avant et arrière et la distance entre deux points dont les longitudes et latitudes sont données.
'Tab' représente le nom à partir duquel la fonction ira chercher les paramètres d'entrée (4) et fournira ceux de sortie (7).
Par exemple si on fournit 'mavar' à tab, la fonction ira chercher les valeurs dans les variables mavar1,mavar2, mavar3 et mavar4 et rendra le calcul dans les variables mavar1 ;;; à ...mavar7.
Les variables mavar1 à mavar7 doivent exister.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié ou si la chaine de définition de la projection est invalide |
| 4 variables NUM 'n&meN' attendue | ERR | X, Y ou Z ne sont pas des variables de type NUM |
| Impossible d'enregistrer la variable... | ERR | collision de noms de variables |
| Ok | STR | Les variables ont été initialisées avec les résultats |

## Exemples:

```
@setvar(tab1,@dms2r(3d),NUM),
@setvar(tab2,@dms2r(45d),NUM),
@out(@concat(Longitude=,@r2dms(@tab1,WE),1)),
@out(@concat(Latitude=,@r2dms(@tab2,NS),1)),
@setvar(tab3,@dms2r(3d),NUM),
@setvar(tab4,@dms2r(45d1'),NUM),
@setvar(tab5,0,NUM),
@setvar(tab6,0,NUM),
@setvar(tab7,0,NUM),
@out(@concat(Longitude=,@r2dms(@tab3,WE),1)),
@out(@concat(Latitude=,@r2dms(@tab4,NS),1)),

@projinvgeod("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",tab),

@out(@concat(Azimuth 12=,@r2dms(@tab5," "),1)),
@out(@concat(Azimuth 21=,@r2dms(@tab6," "),1)),
@out(@concat(Distance=,@tab7,1)),
@return(ok)
```

Calcule la distance orthodromique pour une minute au Nord du point de longitude 3d et latitude 45d

# @projtransform

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| projsrce | STANDARD | definition du système de coordonnées source suivant le standard proj4 |
| projdest | STANDARD | definition du système de coordonnées destination suivant le standard proj4 |
| pt | PT | point devant être transformé |

## Action:

transforme les coordonnées du point "pt" du système "projsrce" au système "projdest"
la définition de la projection ne doit pas comporter de caractère guillemet " si celle ci est entre guillemets.
les coordonnées fournies sont données avec la précision des coordonnees courantes.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| projection source invalide | ERR | Les paramètres de définition du système source sont invalides ou incomplets |
| projection destination invalide | ERR | Les paramètres de définition du système destination sont invalides ou incomplets |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées (par ex parce que la transformation fait appel à une grille et que les coordonnées sont en dehors de la zone de définition de la grille...ou pour une autre raison) |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
@setvar(pt,@creatept(0,0,800000,350000,0),PT),
@projtransform("+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773","+proj=lcc +ellps=clrk80IGN
+towgs84=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=46d48' +lat_1=45d53'56.108
+lat_2=47d41'45.652",@pt),
@out(@concat("X=",@getx(@pt),1)),
@out(@concat("Y=",@gety(@pt),1)),
```

crée un point de coordonnées 800000,350000 puis transforme ce point de Lambert III en Lambert II et affiche les nouvelles coordonnées X et Y.

# @projwgs2geocentric

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| proj | STANDARD | paramètres de définition de système géographique destination selon les règles de proj4 |
| pt | PT | point contenant les coordonnées à transformer |

## Action:

transforme les coordonnées tridimensionnelles cartésiennes de WGS84 (ou du système pivot si le système pivot n'est pas le WGS84) vers le système géodésique donné

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est de type inaproprié |
| Projection invalide | ERR | Les paramètres fournis du système géodésique sont erronés |
| ERREUR TRANSFORMATION... | ERR | Les coordonnées n'ont pu être transformées |
| Ok | STR | Les coordonnées du point ont été transformées |

## Exemples:

```
@setvar(pt,@creatept(0,0,4511602.51,236443.06,4487057.24),PT),
@out(POINT),
@out(@concat("X=",@getx(@pt),1)),
@out(@concat("Y=",@gety(@pt),1)),
@out(@concat("Z=",@getz(@pt),1)),
@out(POINT TRANSFORME => WGS),
@ProjGeocentric2WGS("+proj=lcc +ellps=clrk80IGN
+topivot=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",@pt),
@out(@concat("X=",@getx(@pt),1)),
@out(@concat("Y=",@gety(@pt),1)),
@out(@concat("Z=",@getz(@pt),1)),
@out(POINT RETABLI <= WGS),
@ProjWGS2Geocentric("+proj=lcc +ellps=clrk80IGN
+topivot=−168,−60,+320 +x_0=600000 +y_0=200000
+lon_0=2d20'14.025 +lat_0=44d06' +lat_1=43d11'57.44859
+lat_2=44d59'45.93773",@pt),
@out(@concat("X=",@getx(@pt),1)),
@out(@concat("Y=",@gety(@pt),1)),
@out(@concat("Z=",@getz(@pt),1)),
```

transforme des coordonnées du point vers WGS puis retransforme dans le système source en affichant les différentes étapes :

```
POINT
X= 4511602.510
Y= 236443.060
Z= 4487057.240
POINT TRANSFORME => WGS
X= 4511434.510
Y= 236383.060
Z= 4487377.240
POINT RETABLI <= WGS
X= 4511602.510
Y= 236443.060
Z= 4487057.240
```

# @propdlg

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| idu | STANDARD | identifiant complet (avec préfixe de 4 caractères) de l'objet graphique ou non graphique à editer |

## Action:

ouvre une boite de dialogue de saisie des propriétés d'un objet identifiable.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | en cas de sortie par OK |
| Abandon | STR | en cas de sortie par Abandon ou ESC |

## Exemples:

| | |
|---|---|
| @propdlg("PARC0030000A1024") | ouverture et edition des propriétés de l'objet parcelle PARC0030000A1024 |

# @purgelis

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si ttes les couches |
| classe | STANDARD | numéro de la classe des points ou −1 si tous les points |
| selsrce | STANDARD | niveau de sélection des points à considérer (de 0 à 31) ou −1 si tous les points sont à considérer |
| seldest | STANDARD | niveau de sélection à positionner en cas d'action de sélection |
| action | STANDARD | Combinaison binaire d'indicateurs:<br>PURGELIS_SUPPRIMER:<br>bit 0 = 1 => supprimer<br>bit 0 = 0 => sélectionner au niveau sel<br>PURGELIS_TOUS:<br>bit 1 = 1 => ttes les liaisons<br>bit 1 = 0 => seulement les liaisons de forme "trait" simple |

## Action:

Effectue un certain nombre d'actions indiquées relatives à la purge des liaisons inutiles du document, c'est à dire les liaisons ne supportant aucune face et ne faisant partie d'aucun objet.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N liaisons ont été sélectionnées/supprimées | NUM | |

## Exemples:

| | |
|---|---|
| @purgelis(0,−1,−1,0,0) | sélectionne les liaisons simples isolés de la couche 0 au niveau 0 de sélection |

# @purgepts

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si ttes les couches |
| classe | STANDARD | numéro de la classe des points ou −1 si tous les points |
| selsrce | STANDARD | niveau de sélection des points à considérer (de 0 à 31) ou −1 si tous les points sont à considérer (dans ce cas PURGEPTS_GLOBAL est invalide) |
| seldest | STANDARD | niveau de sélection à positionner en cas d'action de sélection |
| eps | STANDARD | valeur de la perpe au dela de laquelle un point n'est pas aligné entre deux points |
| larg | STANDARD | largeur entre les points de part et d'autre du point à supprimer/sélectionner au dela de laquelle le point est à considérer comme devant être traité |
| action | STANDARD | Combinaison binaire d'indicateurs:<br>PURGEPTS_SUPPRIMER:<br>bit 0 = 1 => supprimer<br>bit 0 = 0 => sélectionner au niveau sel<br>PURGEPTS_ALIGNES:<br>bit 1 = 1 => pts alignés<br>bit 1 = 0 => pts isolés<br>PURGEPTS_TOUS:<br>bit 2 = 1 => ts les points<br>bit 2 = 0 => seulement les points de forme générique<br>PURGEPTS_NOJUXT:<br>bit 3 = 1 => les suppressions successives ne sont pas admises (par ex: sur un arc un point sur 2 est supprimé)<br>bit 3 = 0 => tous les points entre deux pts quelconques sont considérés<br>PURGEPTS_LARGMINI:<br>bit 4 = 0 => tous les points sont à considérer quelle que soit la distance entre les points encadrant ce point.<br>bit 4 = 1 => seuls les points encadrés par des points distants de plus de "larg" seront traités.<br>PURGEPTS_LARGMAXI:<br>bit 5 = 0 => tous les points sont à considérer quelle que soit la distance entre les points encadrant ce point.<br>bit 5 = 1 => seuls les points encadrés par des points distants de moins de "larg" seront traités.<br>PURGEPTS_GLOBAL:<br>bit 6 = 0 => considère la largeur et la hauteur par rapport aux points de la chaine les plus proches<br>bit 6 = 1 => considère la largeur et la hauteur par rapport aux plus proches points de la chaine qui ne sont pas candidats à la suppression |

## Action:

Effectue un certain nombre d'actions indiquées relatives à la purge des points inutiles du document, c'est à dire les points isolés ou les points alignés entre deux points.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N points ont été sélectionnés/supprimés | NUM | |

## Exemples:

| | |
|---|---|
| @purgepts(0,−1,−1,0,0,0,1) | supprime les points simples isolés de la couche 0 |

# @pushline

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| filename | STANDARD | nom d'un fichier texte |
| chaine | STANDARD | chaine à insérer en fin du fichier |
| crlf | STANDARD | indique si un couple CR+LF est inséré en fin de fichier |

## Action:

ajoutte une chaine de caractère à la fin d'un fichier.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| impossible d'ouvrir le fichier... | STR | erreur |
| Ecriture dans le fichier impossible | STR | erreur |
| OK | STR | chaine ajouttée |

## Exemples:

| | |
|---|---|
| @pushline(c:\batch.bat,"cls",1) | ajoutte la ligne "cls" en fin du fichier BATCH.BAT |

# @puzzle

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivselsrce | STANDARD | niveau de sélection des points et liaisons à recoller source (de 0 à 31) |
| nivseldest | STANDARD | niveau de sélection des points et liaisons à recoller destination (de 0 à 31) |
| tol1 | STANDARD | tolérance maxi de déplacement que peut supporter la source. |
| tol2 | STANDARD | tolérance maxi de déplacement que peut supporter la destination |
| action | STANDARD | combinaison des valeurs suivantes<br><br>0x4000 = PUZZLE_INTSRCE<br>0x8000 = PUZZLE_INTDEST<br>0x0100 = PUZZLE_CREEREL<br>0x0200 = PUZZLE_FROMREL<br>0x1000 = PUZZLE_FROMPOS<br>0x0001 = PUZZLE_PTPT<br>0x0002 = PUZZLE_LILI<br>0x0010 = PUZZLE_PTLI_SRCEDEST<br>0x0020 = PUZZLE_PTLI_DESTSRCE<br>0x2000 = PUZZLE_INTERSECTE<br>0x0400 = PUZZLE_NOCROSSREL<br>0x0004 = PUZZLE_EXCLUSSELECT |

## Action:

cette fonction effectue une opération de recollement avec fusion de points et liaisons entre des élémensts sources et des élémensts destination

Les opérations sont faites dans l'ordre suivant

INTSRCE = réunit au préalable en un seul point les points de la source à une distance inférieure à tol1

INTDEST = réunit au préalable en un seul point les points de la destination à une distance inférieure à tol2

INTERSECTE = crée les points à l'intersection de la source et de la destination et les écarte du reste du traitement

Si CREEREL

PTPT = crée une correspondance entre les points les plus proche de la source vers la destination (et les écarte du reste du tritement)

PTLI_SRCEDEST = crée une correspondance entre les points de la source vers les liaisons de la destination (en créant le point sur la destination)

PTLI_SRCEDEST = crée une correspondance entre les points de la destination vers les liaisons de la source (en créant le point sur la source)

NOCROSSREL = evite que les correspondances créées ne se croisent

Sinon si FROMPOS

PTPT = réunit les points les plus proches entre eux (et les écarte du reste du tritement)

LILI = fusionne les liaisons entre elles

PTLI_SRCEDEST = crée et fusionne les points de la source sur les liaisons de la destination

PTLI_SRCEDEST = crée et fusionne les points de la destination sur les liaisons de la source

LILI = fusionne à nouveau les liaisons entre elles

les points traités sont exclus de la sélection si EXCLUSSELECT

FROMREL: effectue les rapprochements avec les correspondances créées (avec CREEREL) ou existantes (dans ce dernier cas les correspondances de l'ensemble du document sont considérées)

PTPT = effectue la réunion entre les points

LILI = effectue la fusion des liaisons

la position des points réunis est fonction des tolérances tol1 et tol2. Si tol1==tol2 le point résultant sera excatement au milieu des deux points, si tol1=0 et tol1=1, le point résultant sera le point source, si tol1=1 et tol1=0, le point résultant sera le point destination.

la fonction échoue si "couchesrce"=="couchedest" et "nivselsrce"=="nivseldest" (ne fait rien).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N points ... ont été déplacés/insérés ... | NUM | N = Nombre de points déplacés |

## Exemples:

| | |
|---|---|
| @puzzle(1,0,0,0.35,0.35,0xC033) | effectue un traitement de débord de feuille sur la sélection |

# @puzzleobj

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| objsrce | OBJ | objet source |
| objdest | OBJ | objet destination |
| sel | STANDARD | niveau de sélection des points et liaisons à considérer dans les faces de l'objet ou −1 si tous les contours de l'objet sont à considérer |
| tol1 | STANDARD | tolérance maxi de déplacement que peut supporter la source. |
| tol2 | STANDARD | tolérance maxi de déplacement que peut supporter la destination |
| action | STANDARD | combinaison des valeurs suivantes<br><br>0x4000 = PUZZLE_INTSRCE<br>0x8000 = PUZZLE_INTDEST<br>0x0100 = PUZZLE_CREEREL<br>0x0200 = PUZZLE_FROMREL<br>0x1000 = PUZZLE_FROMPOS<br>0x0001 = PUZZLE_PTPT<br>0x0002 = PUZZLE_LILI<br>0x0010 = PUZZLE_PTLI_SRCEDEST<br>0x0020 = PUZZLE_PTLI_DESTSRCE<br>0x2000 = PUZZLE_INTERSECTE<br>0x0400 = PUZZLE_NOCROSSREL<br>0x0004 = PUZZLE_EXCLUSSELECT |

## Action:

cette fonction effectue une opération de recollement entre points et liaisons entre les éléments de l'objet source et les éléments de l'objet destination sélectionnés au niveau 'sel'

Les opérations sont faites dans l'ordre suivant

INTSRCE = réunit au préalable en un seul point les points de la source à une distance inférieure à tol1

INTDEST = réunit au préalable en un seul point les points de la destination à une distance inférieure à tol2

INTERSECTE = crée les points à l'intersection de la source et de la destination et les écarte du reste du traitement

Si CREEREL

PTPT = crée une correspondance entre les points les plus proche de la source vers la destination (et les écarte du reste du tritement)

PTLI_SRCEDEST = crée une correspondance entre les points de la source vers les liaisons de la destination (en créant le point sur la destination)

PTLI_SRCEDEST = crée une correspondance entre les points de la destination vers les liaisons de la source (en créant le point sur la source)

NOCROSSREL = evite que les correspondances créées ne se croisent

Sinon si FROMPOS

PTPT = réunit les points les plus proches entre eux (et les écarte du reste du tritement)

LILI = fusionne les liaisons entre elles

PTLI_SRCEDEST = crée et fusionne les points de la source sur les liaisons de la destination

PTLI_SRCEDEST = crée et fusionne les points de la destination sur les liaisons de la source

LILI = fusionne à nouveau les liaisons entre elles

les points traités sont exclus de la sélection si EXCLUSSELECT

FROMREL: effectue les rapprochements avec les correspondances créées (avec CREEREL) ou existantes (dans ce dernier cas les correspondances de l'ensemble du document sont considérées)

PTPT = effectue la réunion entre les points

LILI = effectue la fusion des liaisons

la position des points réunis est fonction des tolérances tol1 et tol2. Si tol1==tol2 le point résultant sera excatement au milieu des deux points, si tol1=0 et tol1=1, le point résultant sera le point source, si tol1=1 et tol1=0, le point résultant sera le point destination.

La fonction échoue si les objets sont non intègres

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 ou 1 | BOOL | indique si le traitement a été possible ou si un objet est non integre ou non surfacique |

## Exemples:

| | |
|---|---|
| @puzzleobj(@obj1,@obj2,0.35,0.35,0xC033) | amène les points de l'objet 1 sur les points ou liaisons de l'objet 2 à mi distance si la distance à ces points ou liaisons est inférieure à 0.70 m |

# @puzzlerel

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| extradataname | STANDARD | nom de la donnée extra des points indiquant la tolérance de déplacement du point (poids) |

## Action:

Effectue un déplacement des points les uns vers les autres à partir des relations sémantiques de type correspondance existantes entre eux et avec un poids dont la valeur est stockée dans la donnée extra du point de nom 'extradataname'

Si un ensemble de N relations existe entre le point PT1 et les points PT2 à PTN, chacun ayant une donnée extra de nom 'extradataname' de valeur p1,p2 à pN, les coordonnées résultantes de tous ces points seront (p1*PT1 + p2*PT2 + p3*PT3 +...+ pN*PTN)/(p1+p2+p3+...+pN)

Un poids nul peut être fourni. Si le bloc à traiter est composé de points tous de poids nuls, la position résultante sera celle calculée comme si tous les poids étaient à 1.
Un poids infini peut être fourni en fournissant une valeur négative à la donnée "extradataname". Toute valeur négative est considérée comme étant un poids infini.
Si le bloc à traiter comprend deux points de poids infini, le bloc ne sera pas traité et un message est fourni à ERRORxxx.LOG

La procedure srcutte tous les points et traite chaque point ainsi :

−si a une donnée extra de nom "extradataname" et au moins une correspondance alors traite le point sinon passe au suivant
−recherche toutes les correspondances de ce point aux autres points possédant une donnée extra de nom "extradataname"
−établit la position pondérée de ce groupe de points comme indiqué ci−dessus et modifie les coordonnées de ces points (ne modifie pas les coordonnées des points en correspondance mais n'ayant pas de donnée extra de nom "extradataname")
−supprime toutes les correspondances de ce point et des points en correspondance avec ce point (y compris les correspondances avec des points n'ayant pas de données extra de nom "extradataname")

La procédure se comporte donc comme si elle traitait des groupes de points indépendants

La donnée extra doit être de type 'R' (réelle) sinon sera considérée absente

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <N> | NUM | N = Nombre de points déplacés |

## Exemples:

| | |
|---|---|
| @puzzlerel(tolerance) | repositionne tous les points en relation |

# @qupl

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

renvoie l'information de "qualité du plan" du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| qupl | STR | chaine de caractère (ex : "plan remembré") |

## Exemples:

@qupl

# @r2dms

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| r | ANG<br>ou<br>STANDARD (autre que ANG) | chaine représentant un angle ( ex: "12.6754gv")<br>ou<br>valeur numérique exprimant un angle en radian |
| postfixe | STANDARD | chaine de 2 caractères représentant le caractère postfixé en cas de signe positif et négatif. généralement on fournit "EW" pour une longitude et "NS" pour une latitude. Dans ces cas la chaine n'aura jamais de signe négatif comme préfixe mais sera postfixé par un des deux caractères fournis.<br>Si on fournit une chaine ou une valeur qui n'a pas deux caractères, par exemple 0, la chaine ne sera pas postfixé et donc signée. |

## Action:

transforme une valeur numérique représentant un angle en radian en une chaine de caractère exprimant une valeur degrés, minutes, secondes

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si le paramètre est invalide |
| <dms> | STR | chaine DMS de l'angle |

## Exemples:

| | |
|---|---|
| @r2dms(0.7890585066898254674,"EW") | renvoie 45d12'35"E |
| @r2dms(−0.7890585066898254674,"EW") | renvoie 45d12'35"W |
| @r2dms(−0.7890585066898254674,0) | renvoie −45d12'35" |
| @r2dms(3.499385150248631504,"EW") | renvoie 200d30'E |
| @setvar(varang,40.5dv,ANG)<br>@r2dms(@varang,0) | renvoie 40d30' |

# @razscriptcmd

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

Cette commande supprime toutes les commandes du menu *script* : si un bouton de commande est également associé, il sera supprimé.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Ok | STR | commandes supprimées |

## Exemples:

| | |
|---|---|
| @razscriptcmd | supprime les commandes |

# @razselect

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| select | STANDARD | niveau de sélection concerné ou −1 si tous les niveaux désirés |
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si toutes les couches désirées |
| telt | STANDARD | type d'éléments à considérer : combinaison des valeurs suivantes:<br>1 =0x01= EltPoint<br>2 = 0x02 =EltLiaison<br>4 = 0x04 = EltEcriture<br>16 = 0x10 = EltFace<br>256 = 0x100 = EltSigne<br>512= 0x200 = EltDeport<br>1024=0x400 = EltRelation (les relations ne sont pas des éléments mais ce drapeau est utilisé pour les relations du document) |

## Action:

déselectionne tous les points, liaisons, faces, écritures,signes, déports, relations de la couche "couche" au niveau de sélection "select"
NB: la sélection des objets n'est pas modifiée : Elle peut être mise à jour en fonction de ses élements par @objintegrity

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @razselect(0,2,23) | déselectionne du niveau 0 les points, liaisons, faces et écritures de la couche 2 |
| @razselect(@selecttravail,−1,0x317) | déselectionne du niveau courant tous les élements de toutes les couches |

# @recentrage

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| options | STANDARD | champs de bits indiquant :<br>bit 0 :<br>1=les systèmes de coordonnées de toutes les couches sont ajustés pour être identique (un point de coord X,Y terrain sur une couche A et un point de même coordonnées sur une couche B seront confondus sur l'écran)<br>0 = les couches sont translatées de manière à ce que le centre de l'écran correspondant aux coord X,Y soit confondu sur l'écran aux points de mêmes coordonnées des différentes couches.<br>bit 4 :<br>1 = un zoom écran est effectué à l'issue de l'opération afin de rendre l'ensemble des couches non gelées visibles.<br>bit 8 :<br>0 =l'action intervient sur toutes les couches.<br>1 = l'action n'intervient que sur les couches actives |

## Action:

Recale les couches les unes par rapport aux autres en effectuant éventuellement un zoom d'ensemble de manière à amener tous les éléments dans la fenêtre.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @recentrage(0x10) | zoom d'ensemble |
| @recentrage(0x111) | recale le système de coordonnées de toutes les couches actives sur la couche de travail puis effectue un zoom d'ensemble |

# @refvar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable dont la référence est à fournir |

## Action:

Cette fonction ne peut être fournie que comme paramètre à la fonction @exec et fournit une référence à une variable plutot que la valeur de la variable

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés ou utilisée pour être affectée à une variable |
| [nomvar] | VARREF | renvoie la reference de la variable |

## Exemples:

| | |
|---|---|
| @exec("cube.ted",@val) | passage par valeur : la fonction cube ne peut modifier "val" |
| @exec("cube.ted",@refvar(val)) | passage par reference : la fonction cube peut modifier "val" |

```
;fonction cube
@param(valeur),
@setvar(valeur,@mul(@mul(@valeur,@valeur),@valeur),NUM)
```

# @regexmatch

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| chaine | STANDARD | chaine à tester |
| regexp | STANDARD | expression régulière |
| casesensitive | STANDARD | indique si sensible à la casse (1) ou non (0) |

## Action:

effectue un test d'une chaine à partir d'une expression régulière (ECMAScript)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Expression régulière invalide | ERR | la chaine fournie comme expression régulière n'est pas une expression régulière valide |
| résultat du test | BOOL | 0 ou 1 |

## Exemples:

| | |
|---|---|
| @regexmatch("test de coucou me voilou","coucou.*me.*voilou",0) | renvoie 1 |

# @rem

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit le texte de la remarque du document (texte libre attaché au document que l'on peut modifier par le menu Edition|Annotation Plan)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Le texte de la remarque est trop long | ERR | |
| [rem] | STR | texte libre attaché au document |

## Exemples:

@rem

# @replacestr

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| text | STANDARD | chaine dont on veut remplacer des occurences |
| srch | STANDARD | chaine à rechercher |
| repl | STANDARD | chaine de remplacement |

## Action:

remplacement de la chaine "srch" par la schaine "repl" dans la chaine "text" et renvoie de la chaine ainsi modifiée.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <chaine> | STR | renvoie la chaine modifiée |

## Exemples:

| | |
|---|---|
| @replacestr("coucouetmeetvoilou","cou","et") | renvoie la chaine "etetetmeetvoilou" |
| @replacestr("coucouetmeetvoilou","cou","") | renvoie la chaine "etmeetvoilou" |

# @return

*Nb returnètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | ??? | renvoie le paramètre fourni en provoquant une rupture de sous−programme TED |

## Action:

renvoie simplement le paramètre et termine le fichier TED en cours d'execution

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| parm | ??? | type du paramètre (standard) |

## Exemples:

| | |
|---|---|
| @return(OK) | renvoie OK et quitte le fichier TED en cours d'éxecution |

# @reservesel

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| sel | STANDARD | niveau de sélection à réserver (de 0 à 31) |
| raz | STANDARD | indique si une remise à zéro du niveau (déselection de tous les éléments, symboles et relations) avec interrogation à l'écran si nécessaire est souhaitée. |

## Action:

inspecte pour voir si le niveau de sélection donné est libre de toute sélection : si c'est le cas renvoie 1, sinon interroge l'utilisateur pour savoir si le niveau de sélection doit être remis à 0 (déselection de tout pour le niveau donné). Renvoie alors 1 si l'utilisateur a accepté de remettre à 0 sinon renvoie 0.
NB: si la procédure renvoie 1, la sélection des objets est également actualisée.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 1 | STR | reserve de sélection effectuée : tous les éléments, symboles et relations ont alors leurs sélection libérée |
| 0 | STR | le niveau de sélection n'est pas libéré soit parce que l'utilisateur n'a pas voulu le libérer (raz à 0) soit parce qu'en présence d'une sélection, une confirmation d'effacement de la sélection n'a pas été donnée. |

## Exemples:

| | |
|---|---|
| @reservesel(0,2), | efface le niveau de sélection 0 sans interrogation (équivalent à @razselect). renvoie toujours 1 |
| @reservesel(0,1) | verifie que le niveau de sélection ne soit pas utilisé, si c'est le cas interroge pour savoir si on doit le liberer : si oui renvoie alors 1 sinon renvoie 0 |
| @reservesel(0,0) | teste si le niveau de sélection 0 est libre de toute sélection : si oui renvoie 1, si non renvoie 0 |

# @rightstr

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| texte | STANDARD | texte dont il faut extraire la chaine |
| nb | STANDARD | nombre de carcatères à extraire depuis le dernier caractère de la chaine |

## Action:

extrait une chaine de "nb" caractères à partir de la fin de la chaine "texte". Si le nombre fourni est plus important que le nombre de carctères de la chaine, alors fournit toute la chaine.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <rightstring> | STR | renvoie la chaine extraite |

## Exemples:

| | |
|---|---|
| @rightstr(commenter,4) | renvoie "nter" |
| @rightstr(commenter,−3) | renvoie "menter" |
| @rightstr(commenter,20) | renvoie "commenter" |

# @roundval

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | STANDARD | valeur (ou texte) à arrondir |
| nbdec | STANDARD | nombre de décimales voulues (−1 = arondi à dixaine, −2 = arrondi à centaine, 3= arrondi au millième...) |
| sens | STANDARD | arrondi : −1 = à la valeur inférieure, 0 = à la valeur la plus proche, 1 = à la valeur supérieure |

## Action:

founit la valeur arrondie à "nbdec" décimales

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| valeur | STR | chaine représentant la valeur arrondie |

## Exemples:

| | |
|---|---|
| @roundval(0.126,2,0) | renvoie 0.13 |
| @roundval(0.122,2,1) | renvoie 0.13 |
| @roundval(112.47,−1,0) | renvoie 110 |

# @run

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom | STANDARD | nom complet du fichier à exécuter<br>−2 = appel au shell |
| timeout | STANDARD | sinon indique si le processus doit être lancé de manière synchrone (TopoCad attend la fin du processus) ou asynchrone (TopoCad lance le processus et continue à travailler). Les valeurs possibles sont alors:<br>−1 = processus assynchrone<br>0 à N = processus synchrone. N représente alors en seconde le temps au delà duquel TopoCad reprend la main en considérant que le processus lancé est en erreur. Une valeur de 0 indique une attente infinie. |
| show | STANDARD | 0 = fenêtre du processus cachée<br>1 = fenêtre du processus normal |

## Action:

lance un processus à partir de TopoCad

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 | NUM | time out écoulé : reprise en main de TopoCad |
| 1 | NUM | processus terminé dans les temps (pour processus synchrone) ou processus lancé correctement (pour processus asynchrone ou appel au shell). |
| Fichier de commande inexistant XXXXXX | ERR | impossible de trouver le fichier exécutable |
| Impossible de lancer XXXXX | ERR | le processus n'a pu être lancé correctement |
| Erreur de synchronisation des processus | ERR | le processus s'est terminé pour une cause inconnue ou est en erreur |
| <motif erreur> | ERR | erreur dans ShellExecute (renvoie le motif de l'erreur) |

## Exemples:

@run("c:\comini\comini.exe",600,1)

@run("w:\topocad\doc\listeoge.ods",−2,1)

@setlvar(shellcmd,"w:\cdif\controle_da\rejet_DA.odt
macro:///TopoCad.Util.PrepareLettre(\x22CODEINSEE|NUMERODA|ECHELLE|PARCELLE|CLEDA|REFGEO\x22,\x22066|1234|2000|AB01
@run(@shellcmd,−2,1)

# @savedoc

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| name | STANDARD | chemin et nom complet du fichier MAP de sauvegarde<br>si le paramètre est vide, alors sauvegarde avec le nom courant du document |

## Action:

enregistre le document courant:

- si un nom est fournit en paramètre:
  - s'il s'agit d'un nouveau document, l'enregistre avec ce nouveau nom qui devient le nom du document.
  - s'il s'agit d'un document déjà nommé, l'enregistre sous ce nouveau nom, qui devient alors le document courant.
- si aucun nom n'est fournit en paramètre:
  - s'il s'agit d'un nouveau document, demande le nom, l'affecte au document courant et l'enregistre tel quel.
  - s'il s'agit d'un document déja nommé, le sauvegarde sous son nom.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| OK | STR | |

## Exemples:

| | |
|---|---|
| @savedoc("") | sauvegarde le document en cours |

# @savetrf

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| trffile | STANDARD | nom du fichier TRF à sauvegarder |

## Action:

Sauvegarde la transformation de la fenêtre courante dans le fichier spécifié.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de sauvegarder le fichier de transformation | ERR | erreur en écriture |
| OK | STR | tout s'est déroulé correctement |

## Exemples:

| | |
|---|---|
| @savetrf(c:\topocad\doc\matransf.trf) | sauvegarde la transformation dans MATRANSF.TRF. |

# @scr2tp

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche concernée (source ou destination) |
| inverse | STANDARD | 0 = coord écran vers coord terrain/papier<br>1 = coord terrain/papier vers écran |
| varx | STANDARD | nom de la variable recevant la coordonnée X du point |
| pt | ou<br>PT | ou<br>point dont les coordonnées sont à modifier |
| vary | STANDARD | nom de la variable recevant la coordonnée Y du point |

## Action:

transforme les coordonnées relatives à la fenêtre plan en coordonnées réelles relatives à la couche 'couche' (terrain ou papier) ou l'inverse si 'inverse' = 1.
les variables varx et vary contiennent en entrée les valeurs à transformer et en sortie les valeurs transformées.
on peut également fournir un point à transformer :, dans ce cas le paramètre vary peut être quelconque.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | les coordonnées ont été transmises aux variables |

## Exemples:

```
@setvar(varx,0,NUM),
@setvar(vary,0,NUM),
@scr2tp(2,0,varx,vary)
```

fournit les coordonnées terrain sur la couche 2 du coin supérieur gauche de la fenêtre plan (qui a des ccordonnées fenetre 0,0).

```
@setvar(monpt,@creatept(0,0,0,0,""),
@scr2tp(2,0,@monpt,0)
```

modifie les coordonnées du point qui est sur la couche 0 par la matrice de transformation des coordonnées écran vers terrain de la couche 2

# @scriptcmdstate

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom/indice | STR ou NUM | nom de la commande script ou indice de la commande script (de 0 à NBMAXSCRIPTCMD) |
| etat | STANDARD | état à fixer ou rechercher |

## Action:

Cette commande permet de modifier l'état d'un script se présentant sous forme d'une option de menu ou d'un bouton de commande dans la boite des scripts

Si la commande est une commande multi−état (en principe avec des images provenants de fichiers BMP différents), alors la commande change l'image courante du bouton (le menu n'est pas impacté) afin de se conformer à l'indice de l'image fournie dans l'état.
si −1 est fourni pour etat, alors la commande ne fais rien et renvoie simplement l'etat de la commande script c'est à dire en principe l'image courante

S'il s'agit d'une commande possédant une ou pas d'image (de menu uniquement), alors
1 pour etat modifie l'état pour que la commande de menu soit cochée ou le bouton enfoncé.
0 pour etat modifie l'état pour que la commande de menu ne soit pas cochée ou le bouton non enfoncé.
si −1 est fourni pour etat, alors la commande ne fais rien et renvoie simplement l'etat de la commande script :
1 pour une commande cochée ou un bouton enfoncé
0 pour une commande non cochée ou un bouton non enfoncé

la désignation de la commande se fait par son intitulé ou par son indice dans le tableau des scripts, ajouttés en principe par la commande @addscriptcmd

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <ind> | NUM | renvoie l'etat de la commande après eventuelle modification |

## Exemples:

| | |
|---|---|
| @scriptcmdstate("Faces des parcelles visibles",−1) | renvoie 1 si la commande "Faces des parcelles visibles" est cochée ou 0 sinon |
| @scriptcmdstate("Faces des parcelles visibles",0) | décoche la commande "Faces des parcelles visibles" |
| @scriptcmdstate("Faces des parcelles visibles",1) | cochei la commande "Faces des parcelles visibles" |

```
@addscriptcmd("Controles
parcellaires","c:\topocad\ted\controle1.ted","",0),
@addscriptcmd("Controles subdiv
fisc","c:\topocad\ted\controle2.ted","",0),
@addscriptcmd("Controles
sections","c:\topocad\ted\controle3.ted","",0),
....
```

interrogation par l'indice 1 de la commande "Controles subdiv fisc" c.a.d la seconde commande

```
@if(@scriptcmdstate(1,−1),
  "",
  @list(
    @hintext("le controle des subdiv fisc n'a pas été fait :
traitement en cours..."),
    @exec("c:\topocad\ted\controle2.ted")
    )
  ),
...
```

# @section

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la section du document (information sous forme de chaine de caractère faisant partie des propriétés du document)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [section] | STR | section du document |

## Exemples:

@section

# @selcrossrel

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de relation sémantique (0 pour les correspondances) |
| nivsel | STANDARD | niveau de sélection à positionner |

## Action:

sélectionne au niveau "nivsel" toutes les relations croisées de type "type" du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <N> | NUM | renvoie le nombre de relations séectionnées par l'opération |

## Exemples:

@selcrossrel(0,0)

# @seldblfaces

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivselsrce1 | STANDARD | niveau de sélection 1 des faces en doubles à scruter (de 0 à 31, −1 si toutes les faces sont à scruter) |
| nivselsrce2 | STANDARD | niveau de sélection 2 des faces en doubles à scruter (de 0 à 31, −1 si toutes les faces sont à scruter) |
| nivseldest | STANDARD | niveau de sélection des faces en doubles à positionner(de 0 à 31) |
| tol | STANDARD | tolérance approximative de recouvrement de faces (au dela de laquelle 2 faces ne sont plus considérées identiques) |

## Action:

Selectionne au niveau "nivseldest" les faces identiques qui font doublon.
Une face est considérée identique à une autre si tous ses points sont à la même position que tous les points de l'autre face à "tol" près.
Pour un ensemble de faces de mêmes positions, elles seront alors sélectionnées au niveau "nivseldest" si au moins une face de cet ensemble est sélectionnée au niveau 1 et une autre au niveau 2
Le niveau de sélection source doit être différent du niveau de sélection destination.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Controle des faces doublons : N faces ont été sélectionnées | STR | N = Nombre de faces sélectionnées |

## Exemples:

| | |
|---|---|
| @seldblfaces(0,0,0,1,0.20) | sélectionne au niveau 1 les faces dont les points sont identiques à 20 cm près |

# @select

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | niveau de sélection à positionner |
| | STANDARD | nom de la variable contenant l'élément ou objet à sélectionner |
| eltobj | PT, LI, FC, EC,RELSEM ou OBJ | élément, relation ou objet à sélectionner |

## Action:

sélectionne l'élément ou l'objet fourni au niveau demandé

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <elt/obj> | ??? | renvoie l'élément ou l'objet |

## Exemples:

@select(0,mavar)

# @selectcou

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| select | STANDARD | niveau de sélection (de 0 à 31) |
| couche | STANDARD | numéro de la couche à sélectionner (de 0 à NbCouches−1) ou −1 si toutes les couches sont concernées |

## Action:

sélectionne la couche (au niveau de sélection "select") dont le numero d'ordre est donné mais ne sélectionne pas les éléments composant cette couche

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @selectcou(0,0) | selectionne la couche 0au niveau de sélection 0 |

# @selectextclass

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | couche des éléments à considérer ou −1 si tous les éléments sont à considérer |
| classe | STANDARD | classe des éléments à considérer ou −1 si tous les éléments sont à considérer |
| selsrce | STANDARD | niveau de sélection des éléments à considérer (de 0 à 31) ou −1 si on ne tient pas compte du niveau de sélection |
| seldest | STANDARD | niveau de sélection à positionner pour les éléments à marquer |
| telt | STANDARD | type d'éléments à scruter : combinaison des valeurs suivantes:<br>1 = EltPoint<br>2 = EltLiaison<br>4 = EltEcriture<br>16 = EltFace |
| action | STANDARD | sous forme de champ de bit<br><br>bit 4 = recherche l'objet (1) ou l'élément (0)<br>bit 5 = mode selection d'objet : détermine comment on considère un objet comme sélectionné ou non<br>0 = normal = l'objet lui même est sélectionné<br>1 = sélection d'un élément au moins<br><br>bit 0 et 1 = mode de sélection<br>0 = pas d'effet de bord (uniquement les élements de l'objet)<br>1 = effet de bord (les liaisons des faces et les points des liaisons avec)<br>2 = effet de bord si l'élément ne fait pas partie d'un autre objet (sans action si bit 4 est à 0) |

## Action:

si action sur objets:

sélectionne au niveau "seldest" de sélection tous les éléments de type donné par "telt" des objets
Si "couche" est différent de −1, les objets doivent appartenir à la couche "couche"
Si "classe" est différent de −1, les objets doivent appartenir à la classe "classe"
Si "selsrce" est différent de −1 les objets doivent être sélectionnés au niveau "selsrce"
(il est donc conseillé de lancer @objintegrity avant de lancer cette fonction dans ce cas)

si action sur elements:

sélectionne au niveau "seldest" de sélection tous les éléments satisfaisant aux critères suivants:
Eléments de type donné par "telt" (peut etre une combinaison de types)
Si "couche" est différent de −1, les éléments doivent appartenir à la couche "couche"
Si "classe" est différent de −1, les éléments doivent appartenir à la classe "classe"
Si "selsrce" est différent de −1 les éléments doivent être sélectionnés au niveau "selsrce"
Les éléments rattachés qui sont nécessaires à la construction de ces premiers éléments (les points et liaisons pour une face, et les points pour une liaison) sont également sélectionnés si le bit 0 de "action"==1 (effet de bord)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | nombre d'éléments ou objets sélectionnés |

## Exemples:

| | |
|---|---|
| @selectextclass(0,5,−1,0,1) | sélectionne les éléments de classe parcelle et les éléments dépendants de la couche 0 |

# @selecttravail

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit le niveau de sélection courant

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| SelectTravail | NUM | niveau de sélection courant |

## Exemples:

@selecttravail

# @selectwithmask

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| telt | STANDARD | type d'éléments concernés par la sélection |
| option | STANDARD | options sous forme de champ de bits<br>• bit 0 = SELWMASK_SELALLCLA<br>♦ 1 = normal<br>♦ 0 = ne sélectionne pas les éléments ayant déjà une classe<br>• bit 1 = SELWMASK_CHECKCLA<br>♦ 1 = la recherche (le masque) tient compte de la classe<br>♦ 0 = la recherche (le masque) ne tient pas compte de la classe<br>• bit 2 = SELWMASK_CHECKALLMASK<br>♦ 1 = tient compte des masques 0 (neutres)<br>♦ 0 = ne tient pas compte des masques 0 (neutres) |

## Action:

sélectionne au niveau de sélection courant tous les éléments de la couche courante qui répondent au masque de recherche de la classe courante.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | nombre d'éléments sélectionnés |

## Exemples:

| | |
|---|---|
| @selectwithmask(23,1) | sélectionne tous les éléments répondant au masque de la classe courante |

# @selext

*Nb paramètres*
7

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | indice de la couche à traiter (de 0 à ...) ou −1 pour toutes les couches |
| selsrce | STANDARD | niveau de sélection source à tester |
| seldest | STANDARD | niveau de sélection destination à positionner |
| isselsrce | STANDARD | indique si on doit tester une sélection (1) ou une absence de sélection (0) |
| isseldest | STANDARD | indique si on doit sélectionner (1) la destination ou la déselectioner (0) |
| eltsrce | STANDARD | type d'élément source (1=Point,2=Liaison,4=Ecriture,16=Face,256=Signe,512=Deport) |
| eltdest | STANDARD | type d'élément destination (1=Point,2=Liaison,4=Ecriture,16=Face,256=Signe,512=Deport) |

## Action:

Cette fonction effectue une propagation de sélection sur des éléments Point, Liaison, Face...

Si eltsrce=Point et eltdest=Liaison, les liaisons comprenant un point sélectionné seront sélectionnées
Si eltsrce=Point et eltdest=Face, les faces ayant un point sélectionné seront sélectionnées
Si eltsrce=Liaison et eltdest=Point, les points des liaisons sélectionnées seront sélectionnés
Si eltsrce=Liaison et eltdest=Face, les faces bordées par des liaisons sélectionnées seront sélectionnées
Si eltsrce=Face et eltdest=Point, les points des faces sélectionnées seront sélectionnés
Si eltsrce=Face et eltdest=Liaison, les liaisons bordant les faces sélectionnées seront sélectionnées
Si eltsrce=Ecriture et eltdest=Deport, les deports d'une écriture sélectionnée sont sélectionnés
Si eltsrce=Liaison et eltdest=Signe, les signes d'une liaison sélectionnée sont sélectionnés
Si eltsrce=Deport et eltdest=Ecriture, les écritures des deports sélectionnés sont sélectionnées
Si eltsrce=Signe et eltdest=Liaison, les liaisons ayant un signe sélectionné sont sélectionnées

Tout autre cas provoque une erreur.
Pour une sélection d'éléments de mêmes type à des niveaux de sélection différent cf @selop

La fonction renvoie le nombre d'élements sélectionnés (passant de l'état non sélectionné à l'état sélectionné ou vice versa)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <nbsel> | NUM | renvoie le nombre d'éléments sélectionnés ou déselectionnés |

## Exemples:

| | |
|---|---|
| @selext(−1,0,0,1,1,1,16) | sélectionne au niveau 0 toutes les faces dont un point est sélectionné |

# @selextremity

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si ttes les couches |
| selsrce | STANDARD | niveau de sélection des liaisons dont on doit déterminer les extrémités (de 0 à 31 ou −1 si aucun niveau de sélection n'est à considérer) |
| seldest | STANDARD | niveau de sélection des points extremités à positionner (de 0 à 31) |

## Action:

sélectionne les points extrémités des polylignes formées par les liaisons sélectionnées.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| <N> | STR | N = Nombre de points sélectionnés |

## Exemples:

| | |
|---|---|
| @selextremity(0,1,2) | sélectionne au niveau 2 les points extrémités des liaisons sélectionnées au niveau 1 de la couche 0 |

# @selliwithsm

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche dans laquelle rechercher les polylignes avec un signe (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des liaisons constituant les polylignes à rechercher |
| nivsel | STANDARD | niveau de sélection à positionner |
| nature | STANDARD | indique quelle nature de signe on doit rechercher et qui ordonne à la polyligne de se sélectionner |

## Action:

détecte et sélectionne les polylignes dont au moins une liaison à au moins un signe de nature "nature" Ne considère que les polylignes constituées de liaisons de classe "classe" dans la couche "couche"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| N liaisons ont été sélectionnées | NUM | N = Nombre de liaisons sélectionnées |

## Exemples:

| | |
|---|---|
| @selliwithsm(0,7,0,5) | sélectionne toutes les polylignes de batis durs (7) ayant une flèche de rattachement (5) |

# @selmultli

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivselsrce | STANDARD | niveau de sélection des liaisons à scruter (de 0 à 31 et −1 si toutes les liaisons sont à scruter) |
| nivseldest | STANDARD | niveau de sélection à positionner pour les liaisons multiples (de 0 à 31) |

## Action:

sélectionne au niveau "nivsel" les liaisons multiples de la couche "couche"
nivselsrce ne peut être égal à nivseldest

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Détection des liaisons multiples : N liaisons multiples ont été sélectionnées | STR | N = Nombre de liaisons sélectionnées |

## Exemples:

| | |
|---|---|
| @selmultli(0,−1,0) | sélectionne au niveau 0 toutes les liaisons multiples de la couche 0 |

# @selneighbours

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des faces à rechercher (−1 si toutes classes) |
| selsrce | STANDARD | niveau de sélection des faces dont il faut trouver les faces "voisines" (de 0 à 31) |
| seldest | STANDARD | niveau de sélection des faces trouvées à positionner (de 0 à 31) |
| opt | STANDARD | option:<br>0 = sélectionne les faces qui jouxtent ces faces à l'extérieur (les autres faces ayant un coté commun)<br>1 = sélectionne les faces qui jouxtent ces faces à l'intérieur (les faces en elles mêmes)<br>2 = sélectionne les faces qui jouxtent ces faces à l'extérieur et à l'intérieur (les faces et leurs voisines) |

## Action:

sélectionne au niveau "seldest" les faces qui bordent les faces sélectionnées au niveau "selsrce"

si selsrce=seldest la sélection finale est imprévisible et dépend de l'ordre dans lequel sont les faces : on doit donc en principe fournir selsrce<>seldest.

renvoie le nombre de faces sélectionnés par l'opération

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <nbfc> | NUM | nombre de faces sélectionnées |

## Exemples:

| | |
|---|---|
| @selneighbours(0,5,0,1,0) | sélectionne au niveau 1 les parcelles qui bordent les faces sélectionnées au niveau 0 de la couche 0 |
| @while(@selneighbours(@couchetravail,5,@selectravail,@selectravail,2) | sélectionne tous les blocs de parcelles jusqu'aux confins de ces dernières contenant une parcelle au moins de sélectionnée. |

# @selop

*Nb paramètres*
9

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| seldest | STANDARD | niveau de sélection destinataire (de 0 à 31) |
| selsrce1 | STANDARD | niveau de sélection N1 (de 0 à 31) |
| selsrce2 | STANDARD | niveau de sélection N2 (de 0 à 31) |
| seldestbool | STANDARD | indique si on doit sélectionner (1) ou déselectionner (0) les éléments du niveau de sélection destinataire |
| selsrce1bool | STANDARD | indique si on doit considérer les éléments sélectionnés (1) ou déselectionnés (0) du niveau de sélection source 1 |
| selsrce2bool | STANDARD | indique si on doit considérer les éléments sélectionnés (1) ou déselectionnés (0) du niveau de sélection source 2 |
| op | STANDARD | opérateur<br>0 = OU<br>1 = ET<br>2 = transfert de srce 1 sur dest |
| couche | STANDARD | couche des éléments concernés par l'opération (−1 si toutes les couches) |
| telt | STANDARD | type d'élément concerné<br>1 = points<br>2 = liaisons<br>4 = écritures<br>16 = faces.....etc |

## Action:

transferts de niveaux de sélection
pour une selection hierarchique (par ex les points des liaisons), cf @selext

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @selop(0,1,2,1,1,1,1,−1,23) | sélectionne au niveau 0 tous les éléments (−1) de tous type (23) qui sont sélectionnés au niveau 1 et sélectionnés au niveau 2 |

# @selproxptli

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivselsrce | STANDARD | niveau de sélection des liaisons à considérer (de 0 à 31). Si toutes les liaisons de la couches sont à scruter, alors nivselsrce=−1 |
| nivseldest | STANDARD | niveau de sélection à positionner pour les points trouvés proches de liaisons et les liaisons concernées (de 0 à 31) |
| tol | STANDARD | tolérance au dela de laquelle un point n'est plus considéré comme proche d'une liaison (en mètre) |

## Action:

sélectionne au niveau "nivseldest" tous les points qui sont à proximité d'une liaison à considérer c'est a dire à moins de "tol" mètres et sélectionne au niveau "nivseldest" les liaisons parmi celles à considérer qui ont un point proche à fusionner.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Controle des points proches de liaisons : N points ont été sélectionnés | STR | N = Nombre de points sélectionnés |

## Exemples:

| | |
|---|---|
| @selproxptli(0,0,0.20) | sélectionne au niveau 0 les points de la couche 0 qui sont à moins de 20 cm d'une liaison |

# @selproxptpt

*Nb paramètres*
5

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivselsrce | STANDARD | niveau de sélection des points à considérer (de 0 à 31) ou −1 si tous les points de la couche sont à considérer |
| nivseldest | STANDARD | niveau de sélection à positionner pour les points trouvés proches de points (de 0 à 31) |
| tol | STANDARD | tolérance au dela de laquelle un point n'est plus considéré comme proche d'un autre point (en mètre) |

## Action:

sélectionne au niveau "nivseldest" tous les points à considérer qui sont à proximité d'autres points c'est a dire à moins de "tol" mètres

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Controle des points doublons : N points ont été sélectionnés | STR | N = Nombre de points sélectionnés |

## Exemples:

| | |
|---|---|
| @selproxptpt(0,0,0.20) | sélectionne au niveau 0 les points de la couche 0 qui sont à moins de 20 cm entre eux |

# @selptobservation

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| obs | OBS<br>ou<br>NUM | observation dont le(les) point(s) est(sont) à sélectionner<br>ou<br>numéro d'observation dont le(les) point(s) est(sont) à sélectionner |
| opt | STD | options: sous forme de champ binaire :<br>0x01 = le point résultat de l'observation est à considérer<br>0x02 = tous les points dont la station (ptC) a même numéro dans les Rayonnements sont sélectionnés<br>0x04 = tous les points dont la station (ptC) a même numéro dans les Rayonnements ainsi que dans les observations non déterminatives Visée et Mesurage (ptA) sont sélectionnés<br>0x08 = tous les points références des observations (visee et rayonnt) dont la station a même numéro sont sélectionnés |

## Action:

sélectionne tous les points relatifs aux observations de numero "num"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | indique le nombre de points qui sont passés de l'état non sélectionné à l'état sélectionné |

## Exemples:

| | |
|---|---|
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37318,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @selptobservation(@curobs,1) | sélectionne le point 37320 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37317,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @selptobservation(@curobs,2) | sélectionne tous les points rayonnés dont la station est 37318 |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37317,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @selptobservation(@curobs,4) | sélectionne tous les points rayonnés, les visées et les mesurages dont la station est 37318. |
| @setvar(curobs,@addobservation( "Rayonnt 37320,A=37319,B=37317,C= 37318,Distance CX= 5.800,Angle/AB=200.0000gv,"),OBS), @selptobservation(@curobs,8) | sélectionne toutes les références (rayonnement et visées) dont la station est 37318. |
| @selptobservation(37319,1) | sélectionne le point 37319 |
| @selptobservation(37319,4) | sélectionne tous les points résultats des observations dont le point 37319 est la station pour les observations de type Rayonnt, Visee ou Mesurage |
| @selptobservation(37319,8) | sélectionne toutes les références des observations dont le point 37319 est la station pour les observations de type Rayonnt ou Visee |

# @setatt

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément, un objet ou un masque |
| | ou | ou |
| eltobjmask | PT, LI, FC, EC, OBJ | élément dont les attributs sont à positionner |
| | ou | ou |
| | MASK... | masque dont les attributs sont à positionner |
| att | STANDARD | attributs à positionner à l'élément, l'objet ou le masque sous forme de champs binaires. Les valeurs sont une combinaisons de :<br><br>1= Numbered (numéroté, présentation)<br>2= HiddenMono<br>4= HiddenCoul<br>8= Simple<br>0x10= Locked<br>0x100= Italique<br>0x200= Gras<br>0x400= Barre<br>0x800= Souligne<br>0x1000= Opaque<br>0x1000000= AAjouter<br>0x2000000= ASupprimer |
| withsymb | STANDARD | agit sur les symboles de l'élément ou non (en cas de liaison ou écriture) ou sur les éléments en cas d'objet |

## Action:

positionne les attributs d'un élément, un objet ou un masque. Pour un masque de modification, fait en sorte que ce dernier positionne l'attribut, pour un masque de recherche fait en sorte qu'une recherche de la présence de l'attribut se fasse. positionne éventuellement les symboles associés à l'élément (liaison ou écriture)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setatt(@pt,1,0) | positionne l'attribut "numéroté" au point "pt" |

# @setbmp2tp

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de couche (de 0 à NbCouches−1) |
| mat | STANDARD | série de 6 réel représentant respectivement a,b,c,d,p,q coefficients de la matrice de transfert des coordonnées Bitmap vers coordonnées terrain (ou papier)<br>NB: les coordonnees Bitmap sont S−E |

## Action:

modifie la matrice de transformation des coordonnees Bitmap vers coordonnées terrain/papier

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | propriété traduite et écrite |

## Exemples:

| | |
|---|---|
| @setbmp2tp(2, 1 0 0 −1 0 0) | 1 pixel du bitmap = 1 mètre |

# @setbmpname

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de couche (de 0 à NbCouches−1) |
| nom | STANDARD | chemin et nom complet du fichier BMP à associer à la couche |

## Action:

associe un nom de fichier à la couche "couche". Il est nécessaire que la couche soit visible et l'attribut Bitmap positionné pour que l'image s'affiche à l'écran.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | fichier associé |

## Exemples:

@setbmpname(2, "c:\topocad\doc\essai.bmp")

# @setcadreformatpage

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice du cadre dans le format de page. Cet indice doit être un indice valide pour le format de page courant. |
| nom | STANDARD | nom du cadre. Ce nom peut être composé de texte formattés (ex: "Section @section" indiquera d'inscrire "Section AB" dans le cadre, cf "textes formattés" pour plus d'infos). |
| x0 | STANDARD | coordonnée X du point supérieur gauche du cadre |
| y0 | STANDARD | coordonnée Y du point supérieur gauche du cadre |
| x1 | STANDARD | coordonnée X du point inférieur droite du cadre |
| y1 | STANDARD | coordonnée Y du point  inférieur droite du cadre |

## Action:

modifie un cadre du format de page courant.. (ce format de  page modifié n'est pas sauvegardé dans la configuration de TopoCad).
le nom du cadre n'est pas modifié si une chaine vide est fournie.
les coordonnées du cadre ne sont pas modifiées si x1<x0 ou y1<y0
ATTENTION: Le cadre "@plan" ne peut pas être renommé, mais seulement modifié dans ses coordonnées

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | cadre de format de page modifié |

## Exemples:

| | |
|---|---|
| @setcadreformatpage(2,"SANS ECHELLE",0,0,−1,−1) | modifie le cadre du format de page courant anciennement celui indiquant l'échelle par un texte fixe "SANS ECHELLE". La position du cadre n'est pas modifiée. |
| @setcadreformatpage(4,"X",50,50,600,600) | Modifie le cadre du format de page courant anciennement le cadre Plan en lui affectant une nouvelle taille. "X" est ici ignoré. |
| @setcadreformatpage(0,"@plan",50,50,600,600) | Modifie le cadre du format de page courant anciennement le cadre indiquant la commune en lui affectant une nouvelle taille. "@plan" est ici ignoré car on ne peut modifier le nom du cadre plan |
| @setcadreformatpage(0,"@plan",0,0,−1,−1) | Ne modifie rien : concerne le cadre d'indice 0 soit le cadre de commune mais ne peut modifier l'intitulé (invalide) ni les dimensions (invalides). Renvoie Ok. |

# @setclasse

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément, un objet ou un masque |
| | ou | ou |
| eltobjmask | PT, LI, FC, EC | élément dont la classe est à modifier |
| | ou | ou |
| | MASK... | masque dont la classe est à modifier |
| classe | STANDARD | classe à affecter à l'élément, l'objet ou le masque |

## Action:

affecte une classe à un élément, un objet ou un masque

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setclasse(@obj,5) | met la classe parcelle à l'objet contenu dans la variable "obj" |

# @setclassetravail

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe à affecter à la classe de travail |

## Action:

modifie la classe de travail courante

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Classe par défaut positionnée à N | STR | N = nouvelle classe de travail |

## Exemples:

| | |
|---|---|
| @setclassetravail(5) | la classe parcelle devient la classe courante |

# @setcodeinsee

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| codeinsee | STANDARD | nouveau code INSEE du document |

## Action:

modifie le code INSEE du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setcodeinsee(298) | modifie le code INSEE du document |

# @setcolor

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément ou masque |
| | ou | ou |
| eltobjmask | | |
| | PT, LI, FC,EC, MASK... | élément ou masque dont la couleur est à modifier |
| color | STANDARD | couleur à affecter (sous forme RGB : valeur 0xBBGGRR) |

## Action:

affecte une couleur à un élément ou masque
NB: il est possible d'affecter une couleur "None" c'est à dire pas d'affichage en fournissant la valeur −1 : la couleur stockée "None" a pour valeur 0xFF000000 mais ne peut etre affectée directement.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setcolor(@varelt,255) | met la couleur rouge vif à l'élément contenu dans la variable"varelt" |

# @setcommune

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| commune | STANDARD | nom de la commune à attribuer au document |

## Action:

modifie le nom de la commune du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setcommune(COUR ET BUIS) | affecte "COUR ET BUIS" au nom de commune du document |

# @setcopl

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| copl | STANDARD | chaine de caractère (normalisée ou non) indiquant le mode de confection du plan<br>pour un export Edigéo les chaines doivent être l'une de celles−ci<br><br>"Ancien plan"<br>"Plan rénové par voie de mise à jour"<br>"Plan rénové par voie de renouvellement"<br>"Plan rénové par voie de réfection"<br>"Plan remanié (réfection)"<br>"Plan obtenu après remembrement"<br>"Plan obtenu par exploitation de plans d'arpentages" |

## Action:

modifie la propriété COPL du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setcopl("Ancien plan")

# @setcouchetravail

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche qui doit devenir la couche de travail |

## Action:

modifie la couche de travail courante

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Couche de travail positionnée à N | STR | N = nouvelle couche de travail courante |

## Exemples:

| | |
|---|---|
| @setcouchetravail(0) | la couche 0 devient la couche de travail |

# @setcurrenttrelsem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STD | type de relation sémantique (de 0 à NbTRelSem−1) |

## Action:

renvoie le type de relation sémantique courant de la fenetre, sous forme d'indice de 0 à NbTRelSem−1 (0 étant le type des correspondances pt à pt)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| OK | STR | type courant modifié |
| Paramètre invalide | ERR | si le paramètre a une valeur non approprié |

## Exemples:

| | |
|---|---|
| @setcurrenttrelsem(0) | fixe la correspondance comme relation sémantique courante |

# @setdata

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom de l'option | STANDARD | nom de l'option à modifier |
| value | STANDARD | valeur de l'option/nom du fichier de config DXF, SHP ou MIF |

## Action:

modifie une option globale de l'application en lui affectant une nouvelle valeur qui doit être exprimée en décimal si il s'agit d'une valeur numérique.

Il peut s'agir du nom des variables globales présentes dans la section DATA du fichier TOPOCAD.INI. Les variables disponibles sont alors:

- PrecisionCoord
- PrecisionAltitude
- PrecisionDistance
- PrecisionAngle
- AngleUnit
- AngleSens
- BmpPath
- DocPath
- AuxPath
- DbaPath
- HlpPath
- CarPath
- PreExec
- PostExec
- HauteurNumero
- HauteurDistance
- HauteurSurface
- UnitSurface
- TailleFlecheNord
- CouleurFlecheNord
- ManipCoucheTransf
- ManipCoucheCopie
- ModeSelect
- PtEcVisible
- PtSimpleVisible
- FcVisible :
  Cette option est une combinaison de plusieurs options afin d'activer la vision des éléments Faces de toutes les classes en mode visions polychromes et désactiver HideClassMode afin de voir les faces suivant leurs critères intrinsèques en mode vison élement si FcVisible est à 1. Elle réalise l'inverse si FcVisible est à 0 : dans ce cas toutes les faces sont invisibles dans tous les types de visions (sauf si elles sont sélectionnées ou font l'objet d'une opération ou sont marquées comme AAjouter ou ASupprimer en mode vision element)
- HideClassMode
- SigneMitoyOn
- DeportEcOn
- DrawPolyline
- FormatDates
- ImprHasFNord
- CouleurQuadrillage
- Quadrillage
- HauteurCoordQuadr
- SelVisibleOnPrint
- DefFormatPage
- PointCache
- ModeReelVision
- EspactHachMonochrome
- AngleHachMonochrome
- CAPointProlLi

- CopieSymForCut
- CAInserePtDsLi
- PrecisionSurface
- NXYUser
- NXYWithLabel
- LOCUser
- VisionRelSem
- OrdreInterpolGravitaire
- NoAutoAddField
- TedDebug
- Tmw (TopoCad Mouse Wheel : cf fonctions de zoom)
- MethodeCodif

Il peut s'agir des noms suivants (qui parlent d'eux−mêmes):

- LargeurIdent
- LargeurCoord
- LargeurAltitude
- LargeurDistance
- LargeurAngle
- PrecisionAngle
- PCIConfigLot
- EcrNumero
- IncrEcrNumero
- LastNum (propre au document)
- UserHelp (nom du fichier d'aide utilisateur − interne − @setdata(userhelp," ") invalide l'aide utilisateur)

Il peut s'agir de "FormatPoints" qui modifie alors le format utilisateur de sortie des points (6° format dans la section FORMATS_POINTS du fichier de configuration TOPOCAD.INI). Dans ce cas la chaine de format est attendue en paramètre.

Il peut s'agir de "FormatEcritures" qui modifie alors le format utilisateur de sortie des écritures (6° format dans la section FORMATS_ECRITURES du fichier de configuration TOPOCAD.INI). Dans ce cas la chaine de format est attendue en paramètre.

Il peut s'agir de "DaNumParam" qui modifie les paramètres de transfert des Das numériques. Dans ce cas une chaine est fournie représentant les 3 entiers fournissant respectivement le groupe, le type, et enfin la valeur du parametre. (cf DaNumParam du fichier de configuration pour une information de ces paramètres).

Il peut s'agir de "DaNumEntete" qui modifie l'entete à l'export dans un export de da numérique. Les 3 lignes sont accessibles en faisant préceder la chaine à attribuer d'un chiffre de 1 à 3 pour respectivement les lignes d'entete 1 à 3. Si le premier caractère de la chaine n'est pas 1 ou 2 ou 3, la totalité de la chaine est attribué à la 3° ligne de l'entete.

Il peut s'agir de "InfoLabel" : cette variable interne indique si l'affichage des numeros de points, des distances de laisons, de surfaces de faces est substitué par l'affichage des étiquettes de ceux ci. Cette variable interne est toujours à 0 au démarrage de l'application (pas d'affichage d'étiquettes). L'accélérateur CTL+ALT+I permet de switcher d'un mode à l'autre pour un type d'élément en fonction du mode courant (Pt,Li,Fc).

Il peut aussi s'agir des noms suivants :

DxfConfig, MifConfig, ShpConfig, PCIConfig, EdigeoConfig, ApicConfig, DaNumConfig,OsmConfig,KmlConfig.
dans ces cas là, "value" représente le nom d'un fichier INI dans lequel une section respectivement ACADDATA, MIFDATA, SHPDATA, PCIDATA, EDIGEODATA, APICDATA, DANUMDATA, OSMDATA, KMLDATA existe et dont les paramètres remplaceront ceux existant.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | propriété traduite et écrite |

## Exemples:

| | |
|---|---|
| @setdata(PrecisionCoordonnee,4) | fixe la precision des coord à 4 décimales |
| @setdata(MifConfig,"c:\topocad\doc\ma_config_mif.ini") | lis les configurations d'import/export MIF dans le fichier "ma_config_mif.ini" |
| @setdata(InfoLabel,17), | force l'affichage des étiquettes pour les points (1) à la place du numéro de point, et pour les faces (16) à la place de la |

surface.

# @setdateedi

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| dateedi | STANDARD | date d'édition (sous format conforme au format des dates. par défaut : JJ/MM/AAAA) |

## Action:

modifie la date d'édition , propriété du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setdateedi("01/01/1996")

# @setdateincorp

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| dateincorp | STANDARD | date d'incorporation au SIG (sous format conforme au format des dates. par défaut : JJ/MM/AAAA) |

## Action:

modifie la date d'incorporation au SIG , propriété du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setdateincorp("01/01/1996")

# @setdateredi

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| dateredi | STANDARD | date de réédition (sous format conforme au format des dates. par défaut : JJ/MM/AAAA) |

## Action:

modifie la date de réédition , propriété du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setdateredi("01/01/1996")

# @setdbprop

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom de la variable contenant l'objet |
| obj | ou | ou |
| | OBJ | objet dont il faut affecter la propriété |
| nomprop | STANDARD | nom de la propriété ou créer |
| value | STANDARD | valeur de la propriété |

## Action:

modifie ou affecte une propriété à l'objet "obj". Une base standard ou de ce type d'objet doit être ouverte et l'objet doit être en mesure de calculer son identifiant afin de se connecter à la base de donnée de TopoCad qui sera donc modifiée à l'issue de l'opération.

lorsqu'on ajoutte une propriété à une base de classe, le nom de la propriété étant un nom de champ de cette base, si le champ n'existe pas déjà, TopoCad rajouttera un champ à cette base de classe ce qui peut s'avérer plus lent que prévu. Si *NoAutoAddField* est à 1 alors cette opération n'est pas réalisée et la fonction renvoie une erreur.

Cette fonction ne gère pas les multipropriétés (la capacité de la base standard d'avoir plusieurs propriétés de même nom et de différentes valeurs)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible de calculer l'identifiant de l'objet | ERR | si l'objet ne peut calculer l'identifiant |
| Impossible d'écrire la propriété de l'objet | ERR | si problème dans l'écriture de la propriété (base non ouverte ou non accesisible...) |
| OK | STR | propriété traduite et écrite |

## Exemples:

| | |
|---|---|
| @setdbprop(obj,SURFACE,@surfobj(@obj)) | calcule la surface de l'objet "obj" et écrit une propriété pour cet objet de nom "SURFACE" contenant le résultat |

# @setdbpropwithid

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| idu | STANDARD | identifiant complet de l'objet dont il faut modifier la propriété (avec préfixe) |
| nomprop | STANDARD | nom de la propriété à modifier ( modifier ou créer pour une base standard) |
| value | STANDARD | valeur de la propriété |

## Action:

modifie ou affecte une propriété à l'objet d'identifiant "idu". Une base standard ou de ce type d'objet doit être ouverte.

lorsqu'on ajoutte une propriété à une base de classe, le nom de la propriété étant un nom de champ de cette base, si le champ n'existe pas déjà, TopoCad rajouttera un champ à cette base de classe ce qui peut s'avérer plus lent que prévu. Si *NoAutoAddField* est à 1 alors cette opération n'est pas réalisée et la fonction renvoie une erreur.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Impossible d'écrire la propriété de l'objet d'IDU xxxx | ERR | si problème dans l'écriture de la propriété (base non ouverte ou non accesisible...) |
| OK | STR | propriété traduite et écrite |

## Exemples:

| | |
|---|---|
| @setdbpropwithid(@idu,SECTION,@section) | fournit le nom de la section comme propriété de l'objet dont l'identifiant est dans la variable "idu" |

# @setdebug

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| value | STANDARD | modifie la variable TedDebug |

## Action:

active ou désactive le mode debug en modifiant la variable système TedDebug.
Cette variable est constituée de deux champs : un champ haut (bits 8 à 15) et bas (bits 0 à 7)
Le champ bas est destiné à l'utilisateur qui peut s'en servir dans un programme TED pour valider ou non des sorties permettant de déboguer ou scrutter le comportement d'un programme.
Le champ haut, active le mode debug standard et quelques options supplémentaires.
Le bit de valeur 0x100 valide le mode debug avec sortie dans le fichier des messages ERRORxxx.LOG des résultats de chacune des commandes du programme TED.
Le bit de valeur 0x200 valide le mode debug avec les infos de position du curseur (dans le fichier) de chacune des commandes.
Le bit de valeur 0x400 invalide le comportement de création de la fonction @setvar
Le bit de valeur 0x800 invalide le comportement de création de la fonction @setgvar
Le bit de valeur 0x1000 valide l'émission de message et d'un bip lorsqu'une fonction demande la création d'une variable générale c'est à dire une variable non locale qui sera visible par tout le programme.
@setdebug(ON) est équivalent à TedDebug|=0x100, validation du mode debug
@setdebug(OFF) est équivalent à TedDebug&=~0x100, dévalidation du mode debug

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setdebug(ON) | active le mode debug |
| @if(@binand(@getdata(TedDebug),1),<br>  @out("Attention: fichier non trouvé !"),<br>  ""<br>  ), | sortie seulement si mode deug utilisateur (1) est validé |

# @setdocvue

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| doc | MAPDOC | document devant devenir le document courant |
| vue | PLANVIEW<br>STANDARD | vue devant devenir la visualisation courante du document courant ou NULL ou 0 si on veut la première vue |

## Action:

change la vue et le document courant pour le programme TED.
toutes les variables concernant des entités propres à d'autres documents (PT, LI, FC, EC, OBJ, RELSEM) deviennent alors inaccessibles.
si le paramètre 'vue' est une vue n'appartenant pas au document 'doc' fourni, alors une erreur 'paramètre invalide' est déclenchée.
si le paramètre 'vue' est un pointeur NUL ou "0", la première vue du document est choisie.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | vue et document courants ont été changés |

## Exemples:

@setdocvue(@mondoc,@mavue)

# @setechelle

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eor | STANDARD | échelle du plan (nombre supérieur à 0) |

## Action:

modifie l'échelle du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setechelle(1000)

# @setechorigin

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| eor | STANDARD | échelle d'origine du plan (nombre supérieur à 0) |

## Action:

modifie la propriété EOR du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setechorigin(1000)

# @seteltdata

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD | niveau de sélection des éléments (de tout le document) dont la donnée est à fixer |
| | ou | ou |
| | PT, LI, FC, EC | élément dont une donnée est à fixer |
| | ou | ou |
| | MASKCHPT, MASKCHLI, MASKCHFC, MASKCHEC | masque de recherche des éléments dont la donnée est à fixer |
| name | STANDARD | nom de la donnée à fixer (composé de caractères alphanumériques et du signe '_' (underscore) |
| val | STANDARD | valeur de la donnée (attendue en décimal en cas de valeur numérique) |
| type | STANDARD | indique comment doit être interprétée la donnée<br>S = sous forme de chaine (255 caractères maxi)<br>R = sous forme de réel<br>I = sous forme d'entier décimal |

## Action:

fixe la donnée propre à un élément.
chaque élément possède une collection de données qui lui sont propre et qui meurent et vivent avec lui. Une fusion de deux éléments entraine irrémédiablement la disparition des données d'un des deux éléments fusionné. Ces données sont donc utilisées soit de manière temporaire soit dans le cadre d'une gestion statique du plan (ex: navigateur pour lequel les données ne changent pas).
Les données ont des types divers mais TED ne pourra enregistrer que des données de type numérique (réel) ou des chaines de caractères de faible taille (max 255 caractères).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <data> | STR | donnée de l'élémént |

## Exemples:

| | |
|---|---|
| @seteltdata(elt,debit,10,R) | écrit une donnée d'un élément |
| @seteltdata(0,ce_apic,1,I) | fixe la donnée xdata entière "1" à tout élément sélectionné au niveau 0 (y compris les symboles) |

# @setenv

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nom | STANDARD | nom de la variable d'environnement à modifier |
| val | STANDARD | valeur de la variable d'environnement |

## Action:

fixe une variable d'environnement.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| ECHEC SetEnv | ERR | si la variable ne peut être stockée |
| <OK> | STR | renvoie OK si l'opération a réussie |

## Exemples:

| | |
|---|---|
| @setenv(communespath,d:\commmunes) | renvoie "Ok" |
| @setenv(ignpath,d:\ign) | renvoie "Ok" |

# @setepaisseur

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément ou masque |
| | ou | ou |
| eltobjmask | | |
| | PT, LI, FC,EC, MASK... | élément ou masque dont l'épaisseur est à modifier |
| epaisseur | STANDARD | épaisseur à affecter (en 1/100° de mm) |

## Action:

affecte une épaisseur à un élément ou masque

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setepaisseur(@varelt,20) | met l'épaisseur 0.2 à l'élément contenu dans la variable"varelt" |

# @setfeuille

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomfeuille | STANDARD | nom de la feuille (ou subdivision de section) |

## Action:

modifie le nom de la feuille du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setfeuille(1) | le nom de feuille devient "1" (pour une section ayant plusieurs feuilles par ex) |

# @setformatpage

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice du format de page à prendre dans la table des formats de page (de 0 à ...) |

## Action:

fixe le format de page par défaut à utiliser parmi les formats de pages disponibles

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés (supérieur au nombre de formats de pages) |
| OK | STR | format de page par défaut initialisé |

## Exemples:

@setformatpage(2)

# @setforme

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt | STANDARD ou PT, LI, FC, EC, MASK... | nom de la variable contenant un élément ou masque<br>ou<br>élément ou masque dont il faut modifier la forme |
| forme | STANDARD | forme à affecter à l'élément ou au masque<br>Forme des points<br>==========<br>0 Simple ==> une petite croix<br>1 Tete ==> un point entouré d'espace<br>2 Piquet ==> cercle avec queue<br>3 Borne ==> cercle de 1 mm<br>4 Croix ==> croix de 3 mm de diam.<br>5 Clou ==> représentation du clou (orienté)<br>6 BornePolygo ==> 2 cercles concentriques<br>7 ClouPolygo ==> clou cerclé<br>8 PtTriangu ==> point de triangu cadastrale non borné (triangle)<br>9 BorneTriangu ==> point de triangu cadastrale borné<br>10 PtGeodesique ==> point géodésique non borné<br>11 BorneGeodesique ==> point géodésique borné<br>12 PtPolygo ==> détail topo ponctuel ou point polygonation repéré<br>13 BorneCommune ==> borne limite de commune<br>14 PyloneHT ==> pylone (ligne de transport de force)<br>15 PyloneTelepherique==> pylone de téléphérique<br>16 Egout ==> plaque d'égout<br>17 BorneNGF ==> borne NGF<br>18 RepereNGF ==> repère NGF<br>19 RepereMRL ==> repère nivellement MRL<br>20 RepereNiv ==> repère de nivellement<br>21 PoteauEDF ==> poteau EDF<br>22 PoteauPTT ==> poteau PTT<br>23 RepereTopo ==> repère topographique (calvairepuits...)<br>24 Lampadaire ==> lampadaire<br>25 Arbre ==> arbre isolé<br>26 Puits ==> symbole puits (PCI version 1.7 et ultérieures)<br>Forme des liaisons<br>==========<br>0 Pointilles ==> pour les cotations (petits points)<br>1 Trait ==> trait de parcelles et batiments<br>2 DetTopo ==> trait de détail topo 1−1<br>3 Subdiv ==> trait de subdivision fiscale 4−2<br>4 Chemin ==> trait de chemin ou trotoir 3−2<br>5 Sentier ==> trait de sentier 2−1<br>6 Charge ==> limite de charge<br>7 Teleph ==> téléphérique<br>8 Acqueduc ==> acqueduc<br>9 Gazoduc ==> gazoduc oléoduc<br>10 HteTens ==> ligne de transport de force (haute tension)<br>11 Lieudit<br>12 SubdivSection<br>13 Section<br>14 Commune<br>15 Departement<br>16 Etat<br>17 FlecheCrsEau<br>18 ParkTerr<br>19 Corresp ==> correspondance |

Formes Vectorielles des motifs (pattern)
==========
0 SansMotif
1 Hachures50
2 Hachures150
3 Grille
4 Grille50
5 HachuresHorz
6 HachuresVert
7 HachEtang

## Action:

modifie la forme de l'élément ou du masque

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | forme rectifiée |

## Exemples:

| | |
|---|---|
| @setforme(@elt,3) | met la forme "borne" (3) à l'élément contenu dans la variable "elt" |

# @setformeface

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice de la forme (de 0 à NbFormeface−1) |
| chaine forme | STANDARD | chaine de description de la forme |

## Action:

modifie la forme de face d'indice "ind".
l'indice dans la chaine de description doit être le même que celui fourni par "ind" sinon la fonction échoue.
la chaine de description de la forme est la même que celle fournie dans le fichier TopoCad.ini mais les guillemets sont remplacés par les caractères "\x22"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | forme de face modifiée |

## Exemples:

@setformeface(3,"0 \x22Cimetière Chrétien\x22 R 0 0 0") modifie la quatrième forme des faces (d'indice 3)

# @setformeliaison

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice de la forme (de 0 à NbFormeliaison−1) |
| chaine forme | STANDARD | chaine de description de la forme |

## Action:

modifie la forme de liaison d'indice "ind".
l'indice dans la chaine de description doit être le même que celui fourni par "ind" sinon la fonction échoue.
la chaine de description de la forme est la même que celle fournie dans le fichier TopoCad.ini mais les guillemets sont remplacés par les caractères "\x22"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | forme de liaison modifiée |

## Exemples:

| | |
|---|---|
| @setformeliaison(0,"0 \x22Mes Pointillés\x22 0 Trait 7 Vide 7 0") | modifie la première forme des liaisons (d'indice 0) |

# @setformepoint

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice de la forme (de 0 à NbFormePoint−1) |
| chaine forme | STANDARD | chaine de description de la forme |

## Action:

modifie la forme de point d'indice "ind".
l'indice dans la chaine de description doit être le même que celui fourni par "ind" sinon la fonction échoue.
la chaine de description de la forme est la même que celle fournie dans le fichier TopoCad.ini mais les guillemets sont remplacés par les caractères "\x22"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | forme de point modifiée |

## Exemples:

| | |
|---|---|
| @setformepoint(3,"3 \x22Borne\x22 \x22BO\x22 PenC BrushE EllipseE −50 0 −50 0 50 1 50 1 0") | modifie la quatrième forme des points (d'indice 3) |

# @setgvar

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier ou créer |
| value | STANDARD | valeur à affecter |

## Action:

création ou modification d'une variable globale au système.
cette variable continue d'exister tant que l'application TopoCad n'est pas fermée. La valeur d'initialisation de cette variable est fixée par cette commande.
tous les programmes TED lancés ultérieurement connaitrons cette variable avec cette valeur fixée ici, par contre un appel ultérieur dans la même procédure TED de @setvar modifie la variable qui aura alors une nouvelle valeur tant que la procédure n'est pas terminée.
les variables globales ne peuvent être que de type STR (chaine)
Cette fonction ne peut créer de variable si l'option du mode debug est activée (cf @setdebug)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | STR | renvoie la variable (après affectation ou modification) |

## Exemples:

| | |
|---|---|
| @setgvar(ignpath,"c:\ign"), | la variable "ignpath" est fixée comme variable globale avec une valeur actuelle et d'initialisation pour les futurs lancements de programme TED à "c:\ign" |
| @setvar(ignpath,"d:\ign") | pour le programme en cours et uniquement pour celui−la, la variable prend la valeur "d:\ign" |

# @setidalgo

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |
| idalgo | STANDARD | algorithme de calcul de l'identifiant (entier codé binaire)<br>• bit 0 = Utilise le code INSEE de la commune<br>• bit 1 = Utilise le préfixe de section (code commune ratachée) du document<br>• bit 2 = Utilise la section contenant l'objet<br>• bit 3 = Utilise la subdiv de section contenant l'objet<br>• bit 4 = Utilise la parcelle contenant l'objet<br>• bit 5 = Utilise le toponyme de l'objet<br>• bit 6 = Utilise l'étiquette de l'objet<br>• bit 7 = Utilise le numéro (Id) de l'objet<br>• bit 8 = Utilise la donnée XData de nom IDENT de l'élément directeur de l'objet<br>Ces 4 dernières options s'excluent mutuellement.<br>idalgo =0 signifie que le type d'objet ne supporte pas les identifiants |

## Action:

modifie la manière de calculer son identifiant des objets de classe "classe"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | IdAlgo modifié ! |

## Exemples:

| | |
|---|---|
| @setidalgo(12,0x80) | fait en sorte que les objets "voies publiques" (objets de classe 12) puissent calculer leur identifiant constitiué par le numéro d'objet : l'identifiant aura une forme du style "VPUB543" et les objets voies publiques accepteront des propriétés |

# @setidparm

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |
| iudparm | STANDARD | nouvelle valeur pour IdParm : |

Ce paramètre indique la taille en nombre de caractère que doit avoir le texte, étiquette, ou le numéro d'objet

- Si la taille est négative (ex: −4) alors l'identifiant sera considéré valide si et seulement si la taille de l'étiquette....etc est égale à l'opposédu paramètre (ex: 4 caractères)
- Si la taille est positive (ex 4) alors l'identifiant sera considéré valide si la taille de l'étiquette ....etc est ingférieure ou égale au paramètre et dans ce cas sera formatté à cette taille cadré à droite en complètant avec des 0 (ex: 125 donnera 0125)

Pour les classes Commune, Section, Subdivision de section, Parcelle, ces tailles sont imposées à respectivement (3,5,2,4) et le paramètre signifie alors comment sont récupérés les informations d'appartenance d'un objet à respectivement la commune, la section, la subdiv. de section, la parcelle (la valeur 0 et 1 pour cette dernière n'est pas permise)

- 0 = l'information est prise dans les caractéristiques du document (Code Insee pour la commune, Nom section pour la section, Nom de feuille pour la subdivision de section) que l'on peut modifier dans le menu Edition|Infos plan.
- 1 = l'information est prise dans le nom de la couche auquel appartient l'objet. Suivant si ce choix est pris pour la commune(C), la section(S), la subdiv de section(F),la parcelle (P) le nom d'une couche peut alors être formaté suivant les manières suivantes :

  Le nom de la couche est formatté avec comme séparateur "_"

| ***Commune*** | ***Section*** | ***Subdiv. de section*** | ***Parcelle*** | ***Nom de la couche (commencant par...et suivi éventuellement de "_" et autres caractères)*** |
|---|---|---|---|---|
| . | . | . | . | <non utilisé> |
| . | . | 1 | . | F ou FF |
| . | 1 | . | . | S ou SS ou SSSSS |
| . | 1 | 1 | . | SSSSS_F ou SSSSS_FF |
| 1 | . | . | . | CCC |
| 1 | . | 1 | . | CCC_F ou CCC_FF |
| 1 | 1 | . | . | CCC_S ou CCC_SS ou CCC_SSSSS |
| 1 | 1 | 1 | . | CCC_SSSSS_F ou CCC_SSSSS_FF |
| . | . | . | 1 | PPPP |
| . | . | 1 | 1 | F_PPPP ou FF_PPPP |
| . | 1 | . | 1 | S_PPPP ou SS_PPPP ou SSSSS_PPPP |
| . | 1 | 1 | 1 | SSSSS_F_PPPP ou SSSSS_FF_PPPP |
| 1 | . | . | 1 | CCC_PPPP |
| 1 | . | 1 | 1 | CCC_F_PPPP ou CCC_FF_PPPP |
| 1 | 1 | . | 1 | CCC_S_PPPP ou CCC_SS_PPPP ou CCC_SSSSS_PPPP |
| 1 | 1 | 1 | 1 | CCC_SSSSS_F_PPPP ou CCC_SSSSS_FF_PPPP |

*ex: un nom de couche 435_AB_macouche ou 435_A_macouche peut donc donner comme information la commune, et la section. si IdParm de (Commune,Sect,Feuille,Parc) est (1,1,x,x) mais ne peut donner l'information si est (0,1,x,x) car alors la la section est considéré comme étant "435" de taille 3 qui est invalide pour un code section (un code section ne peut avoir que 1, 2 ou 5 caarctères)*

- 2 = l'information est récupérée en scrutant tous les objets de type respectivement commune, section, subdiv de section et en fournissant le texte du premier objet contenant graphiquement l'objet dont on cherche l'appartenance. ( un objet appartient à un autre si l'intersection entre eux est supérieure à EpsilonAppFace)
- 3 = l'information est récupérée en scrutant tous les objets de type respectivement commune, section, subdiv de section et en fournissant l'étiquette du premier objet contenant graphiquement l'objet dont on cherche l'appartenance. ( un objet appartient à un autre si l'intersection entre eux est supérieure à EpsilonAppFace)
- 4 = l'information est récupérée en scrutant tous les objets de type respectivement commune, section, subdiv de section et en fournissant le texte du premier objet en relation (sémantique quelconque) avec l'objet dont on cherche l'identifiant
- 5 = l'information est récupérée en scrutant tous les objets de type respectivement commune, section, subdiv de section et en fournissant l'étiquette du premier objet en relation (sémantique quelconque) avec l'objet dont on cherche l'identifiant

*ex: IdParm commune=0, IdParm section=1 et IdParm feuillet=0 alors l'identifiant d'une parcelle (IdAlgo= 0x27) sera recherché par le code INSEE du document (3 lettres) puis à partir du nom de la couche qui représente le nom de la section sous forme AB, AC, ... ZA, ZB.*

*ex: IdParm commune=1, IdParm section=1 et IdParm subdiv de sect=0 alors l'identifiant d'une parcelle (IdAlgo= 0x27 ) sera formé par le nom de la couche qui représente code INSEE de la commune et section sous forme 290_AB, 290_ZA ...*

*ex: IdParm commune=1, IdParm section=1 et IdParm subdiv de sect=0 alors l'identifiant d'une parcelle (IdAlgo= 0x27) peut être formé par le nom de la couche qui représente code INSEE de la commune et section sous forme 290_000AB_parcelles, couche contenant les parcelles du plan, 290_000AB_ca, couche contenant les circonscriptions administratives de la section AB, 290_000ZA_modifications, couche contenant les modifications sur la section ZA, ainsi que tout nom respectant ce modèle à la guise de l'utilisateur et commençant par la séquence décrite ci dessus ...*

## Action:

modifie certains paramétrages pour le calcul des identifiants concernant la classe "classe"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | IdParm de la classe modifié ! |

## Exemples:

| | |
|---|---|
| @setidparm(2,1) | modifie le paramétrage de section, les identifiants ayant besoin de la section trouveront désormais la section dans le nom de la couche et plus dans le nom de section du document |

# @setinit

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée ou −1 si toutes sont concernées |
| value | STANDARD | valeur à fournir à l'Init de la classe (des classes) concernées<br>valeur sous forme de champs de bits :<br>bit 5 = 0x20 = Actif<br>bit 4 = 0x10 = Element Face visible<br>bit 2 = 0x04 = Element Ecriture visible<br>bit 1 = 0x02 = Element Liaison visible<br>bit 0 = 0x01 = Element Point visible |

## Action:

modifie la valeur Init du type d'objet fourni (classe)
Permet d'activer ou non une classe et de rendre visible ou invisible chaque type d'élément de la classe considérée
(si l'on modifie l'élément ou le sélectionne, l'élément apparait cependant, priorité à la visualisation est alors donnée à l'élément)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | Valeur modifiée ! |

## Exemples:

| | |
|---|---|
| @setinit(5,0) | rend la classe parcelle inactive (et totalement invisible) |
| @setinit(4,0x27) | rend la classe lieudit active : ses faces seront cependant invisibles dans les visions polychtromes |
| @setinit(12,0x35) | rend la classe "voies publiques" actives mais l'axe de la voie sera invisible dans les visions polychromes. Si par erreur, on venait à créer une face de ce type, elle apparaitrait (et un message d'erreur apparaitrait indiquant un défaut de cardinalité de l'objet) |
| @setinit(3,0x37) | rend la classe subdsect (3) active et entierement visible |

# @setlabel

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant l'élément ou objet |
| eltobj | ou | ou |
| | PT, LI, FC, EC, OBJ | élément ou objet dont l'étiquette est à modifier |
| label | STANDARD | étiquette à mettre à l'élément ou l'objet |

## Action:

modifie l'étiquette d'un élément ou d'un objet

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | Etiquette modifiée ! |

## Exemples:

| | |
|---|---|
| @setlabel(elt,@noDA) | attribue comme étiquette à l'élément contenu dans la variable "elt" un numéro de DA contenu dans la variable "noDA" |

# @setlvar

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier ou créer |
| value | ??? | valeur à affecter (peut être 0 pour les variables pointeurs non initialisées)<br>Dans le cas ou on fourni un paramètre standard à une variable pointeur la signification de la valeur est alors la suivante:<br>PT : numéro de point (pointeur nul si non trouvé)<br>LI : dans tous les cas initialisé avec le pointeur nul<br>FC : idem<br>EC : idem<br>OBJ : numéro d'objet (pointeur nul si non trouvé)<br>MASK... : classe du masque de classe à aller chercher (pointeur nul si non trouvé ou hors classe)<br>RELSEM : numero de relation sémantique (la première n'est pas accessible)<br>Dans tous les cas si 0 est fourni, c'est le pointeur NULL qui est affecté à la variable |
| type | STANDARD | chaine représentant le type de variable<br>• STR: Type chaine de caractères<br>• NUM: Type numérique<br>• ANG: Type Angle<br>• BOOL: Type Booléen<br>• PT : type élément point<br>• LI :type élément liaison<br>• FC: type élément face<br>• EC: type élément écriture<br>• OBJ: type objet<br>• MASKCRPT: type masque de création de point<br>• MASKCRLI: type masque de création de liaison<br>• MASKCRFC: type masque de création de face<br>• MASKCREC: type masque de création d'écriture<br>• MASKMDPT: type masque de modification de point<br>• MASKMDLI: type masque de modification de liaison<br>• MASKMDFC: type masque de modification de face<br>• MASKMDEC: type masque de modification d'écriture<br>• MASKCHPT: type masque de recherche de point<br>• MASKCHLI: type masque de recherche de liaison<br>• MASKCHFC: type masque de recherche de face<br>• MASKCHEC: type masque de recherche d'écriture<br>• RELSEM : type relation sémantique<br>• MAPDOC : type document MAP<br>• PLANVIEW : type vue (fenêtre de visualisation) fenêtre Plan |

## Action:

Création ou modification d'une variable locale.
(la valeur de la variable et la variable sont perdue à la sortie du fichier procédure TED)
Cette fonction recherche une variable locale de même nom : si elle la trouve alors la modifie, et si elle ne la trouve pas crée la variable locale.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | ??? | renvoie la variable (après affectation ou modification) |

## Exemples:

| | |
|---|---|
| @setlvar(mask,5,MASKMDFC) | affecte à la variable "mask" le masque de modification des faces de la classe parcelle |

# @setmask

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| valeur | STANDARD | descriptif du masque. ce descriptif est le même que celui fourni dans le fichier de configuration excepté que des caractères '\|' remplacent les espaces. |
| type | STANDARD | type de masque (de point, liaison, face ou écriture, de recherche, de modification, ou de création : ex MASKCHPT) |
| classe | STANDARD | classe |

## Action:

modifie le masque de type donné pour la classe "classe" en prenant la valeur fournie.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <masque> | MASK... | renvoie le masque modifié |

## Exemples:

| | |
|---|---|
| @setvar(<br>mask,<br>@setmask(20\|0x0\|0x1000000\|0x0\|0x0\|0\|0\|0\|0\|0\|0x0\|0x1000000\|0x0\|0x0\|0x0\|0x0\|0\|0\|0,MASKCHPT,0),<br>MASKCHPT), | affecte à la variable "mask" un masque de recherche de points qui recherche les points dont l'attribut AAjouter est positionné |

# @setmaskno

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| valeur | STANDARD | descriptif du masque. ce descriptif est le même que celui fourni dans le fichier de configuration excepté que des caractères '\|' remplacent les espaces. |
| type | STANDARD | type de masque (de point, liaison, face ou écriture, de recherche, de modification, ou de création : ex MASKCHPT) |
| no | STANDARD | indice du masque (de 0 à NbMask... −1) |

## Action:

modifie le masque de type donné d'indice no en prenant la valeur fournie.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <masque> | MASK... | renvoie le masque modifié |

## Exemples:

| | |
|---|---|
| @setvar(<br>mask,<br>@setmaskno(20\|0x0\|0x1000000\|0x0\|0x0\|0\|0\|0\|0\|0\|0x0\|0x1000000\|0x0\|0x0\|0x0\|0x0\|0\|0\|0,MASKCHPT,0),<br>MASKCHPT), | affecte à la variable "mask" un masque de recherche de points qui recherche les points dont l'attribut AAjouter est positionné |

# @setmodeincorp

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| inp | STANDARD | chaine de caractère (normalisée ou non) indiquant le mode d'incorporation du plan<br>pour un export Edigéo les chaines doivent être l'une de celles−ci<br>"Numérisation manuelle"<br>"Numérisation par scanner"<br>"Incorporation directe sans numérisation préalable" |

## Action:

modifie la propriété INP du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setmodeincorp("Numérisation manuelle")

# @setmodeprogram

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| code | STANDARD | indique par un entier de 0 à 8 l'opération à effectuer<br>bit0 = SHF<br>bit1 = CTL<br>bit2 = DROITE(1) sinon GAUCHE(0)<br>bit3 = CHAINE DEFINITION MODE(1) sinon CHAINE DEFINITION COMMANDE(0) |
| chaine info | STANDARD | chaine à afficher pour la définition du mode ou de la commande |
| chaine cmd | STANDARD | nom complet du fichier TED à executer |

## Action:

Cette commande sert à programmer le mode programmable de l'application.
Ce mode permet d'accéder à huit fonctions par un clic de souris soit 4 pour le clic gauche et 4 pour le clic droite, le clic du milieu étant réservé aux opérations de zoom.
Les 4 fonctions sont accessible via les touches SHF et CTL.

On fixe d'abord le nom du mode programmé (code=8). Si aucun nom n'est fourni, c'est "MODE PROGRAMME" qui sera affiché dans ce mode (à déconseiller si plusieurs modes programmés existent).

Le nom du mode programmé servira également pour constituer le fichier HTML d'aide qui s'affichera en faisant F1 en étant dans le mode. Le fichier HTML cherché sera constitué comme suit :
"menu_mode_ted_" + (nom du mode programmé dont les espaces sont remplacés par '_') + ".html"
ex: si on fixe le nom du mode programmé à "INTERROGATION MAJIC", le fichier html appelé sera "menu_mode_ted_interrogation_majic.html"

On fournit alors également les couples de chaines représentant d'une part l'information du résultat d'une commande donné sur la barre d'état (pour clic gauche ou clic droite) et d'autre part le nom du fichier TED à exécuter lors de l'action avec la souris.

Pour entrer dans le mode utiliser la fonction TED @SetModeState avec comme paramètre la valeur MODE_TED=0x001001A0=1048992

L'action du Clic renseigne l'application avec deux ***variables globales XSCR et YSCR*** représentant les coordonnées fenêtre du clic effectué : Libre à l'utilisateur et aux procédures TED de récupérer ou non ces valeurs pour les utiliser par exemple en les convertissant en coordonnées terrain pour une recherche ou fixer un objet ou un point.
Si le clic est effectué avec un glisser−déplacer, deux autres ***variables globales XSCR2 et XSCR2*** représentant les les autres coordonnées du rectangle dessiné sont renseignées par l'application.

Pour établir un mode, il est préférable  d'établir toutes les fonctions (7) au cas ou une fonction du mode programme ait été déjà établie, et donc d'effacer toute précédente fonction en fournissant au paramètre 'chaine cmd' une chaine vide.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | affectation réalisée |

## Exemples:

```
@setmodeprogram(8,"MON MODE",""),
@setmodeprogram(0,"commande
n°1","c:\topocad\doc\comd1.ted"),
@setmodeprogram(1,"commande
n°2","c:\topocad\doc\comd2.ted"),
@setmodestate(0x001001A0),
@return(OK)
```

indique le nom du nouveau mode programmé
puis affecte au clic gauche (SHF et CTL relachés) un nom de commande "commande n°1" qui lance le fichier TED comd1.ted, puis affecte au clic gauche (SHF enfoncé) un nom de commande "commande n°2" qui lance le fichier TED comd2.ted.
Enfin se positionne dans le mode "MON MODE"

*Contenu du fichier **comd1.ted**:*

```
  @hinttext("Commande 1=",@concat(@xscr,@yscr,1))

@setmodeprogram(1,"commande n°2",""),
```

l'appui du clic gauche sur la fenêtre provoque l'affichage dans la barre d'état des coordonnées fenêtre du clic effectué.
ex: "RESULTAT=Commande1=836 123"

la commande n°2 est "effacée" et n'a donc plus d'action ni d'affichage dans la barre de statut (la valeur "commande n°2" n'est pas obligatoire).

# @setmodestate

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| mode | STANDARD | mode opératoire de l'application à initialiser sous forme d'entier (dec,octal ou hexa) |

## Action:

met TopoCad dans un mode opératoire donné.
Si mode == 0, alors la fonction ne fait rien et renvoie le mode courant.
pour le mode programmé (MODE_TED=0x001001A0=1048992), la valeur renvoyée est la chaine de caractère représentant le mode en cours et non la valeur de ce mode, la valeur renvoyée est donc de type STR sinon elle est de type NUM.
pour connaitre la valeur numérique d'un mode donné (ou le nom du mode programmé), on peut donc se mettre dans le mode et lancer la macro @setmodestate(0).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <mode> | NUM | mode courant à l'issue de la fonction ou valeur du mode courant si le paramètre est 0 |
| <nom mode> | STR | nom du mode programmé en cours si le paramètre est 0 est qu'on est en mode programmé. |

## Exemples:

| | |
|---|---|
| @setmodestate(0x233100) | met l'application en mode édition : ajout de points et liaisons |
| @setmodestate(0) | renvoie 2306304 en mode ajout de Pt+Li |
| @setmodestate(0x001001A0) | renvoie "MODE PROGRAMME" en mode programmé |
| @setmodestate(0) | |

# @setorior

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| orior | STANDARD | orientation d'origine du plan (angle) |

## Action:

modifie la propriété ICL (orientation d'origine) du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setorior(10gv)

# @setpattern

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| ind | STANDARD | indice de la forme (de 0 à Nbpattern−1) |
| fichier | STANDARD | nom du fichier BMP à remplacer de celui existant sans chemin. |

## Action:

modifie le motif d'indice "ind" en changeant le fichier BMP auquel il se réfère. Un prochain affichage affichera le nouveau motif.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| XXXX.BMP : fichier bitmap attendu inexistant! | ERR | si le fichier BMP est inexistant dans le répertoire de l'application |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | motif modifié |

## Exemples:

@setpattern(1,"CIMETISR.BMP") modifie le second motif (d'indice 1)

# @setprid

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe concernée |
| prid | STANDARD | nouvelle valeur pour PrId: doit être une chaine de 4 caractères |

## Action:

modifie le préfixe d'identifiant pour la classe donnée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | PrId de la classe modifié ! |

## Exemples:

| | |
|---|---|
| @setprid(6,"PARC") | modifie le préfixe d'identifiant des subdivisions fiscales pour qu'il soit identique aux parcelles en vue par exemple d'une gestion unique de ces 2 type d'objet. |

# @setqupl

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| qupl | STANDARD | chaine de caractère (normalisée ou non) indiquant la qualité du plan<br>pour un export Edigéo les chaines doivent être l'une de celles−ci<br><br>"Plan régulier établi avant le 20/03/1980"<br>"Plan non régulier"<br>"Plan de qualité P3"<br>"Plan de qualité P4"<br>"Plan de qualité P5" |

## Action:

modifie la propriété QUPL du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setqupl("Plan de qualité P5")

# @setr2dms

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| frac | STANDARD | valeur numérique de 0 à 8 inclus indiquant le nombre de décimales à fournir aux valeurs des secondes |
| com_w | STANDARD | valeur booleenne indiquant si une largeur standard doit être fournie c'est à dire si les valeurs des minutes et secondes doivent être complétées par des 0 à gauche. (ex: 45d5'8.734" si com_w=0 et 45d05'08.734" si com_w=1) |

## Action:

fixe le formattage de sortie des chaines DMS

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si un paramètre est invalide |
| Ok | STR | Le formattage des chaine DMS est enregistré |

## Exemples:

| | |
|---|---|
| @setr2dms(3,0) | renverra une chaine de type 45d2'5"E |
| @setr2dms(3,1) | renverra une chaine de type 45d02'05.000"E |
| @setr2dms(10,0) | renvoie une erreur (10 est hors limite) |

# @setrem

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| rem | STANDARD | texte libre à affecter à la "remarque" du document |

## Action:

modifie le texte libre qui est attaché à chaque document et appelé remarque

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setrem("levé photogrammétrique") | affecte un texte à la remarque du document (la largeur maxi de la remarque est de 200 caractères. Au dela le texte sera tronqué) |

# @setsection

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| section | STANDARD | nom de la section à affecter au document |

## Action:

modifie la section du document (propriété du document)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setsection(AB) | note "AB" comme section du document |

# @setselecttravail

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | nouveau niveau de sélection courant (de 0 à 31) |

## Action:

modifie le niveau de sélection courant

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Niveau de sélection positionné à N | STR | N niveau de sélection |

## Exemples:

| | |
|---|---|
| @setselecttravail(0) | le niveau de sélection 0 devient le niveau courant |

# @settext

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément écriture |
| eltobjmask | ou | ou |
| | EC | élément écriture dont le texte est à modifier |
| texte | STR | texte à affecter à l'écriture |

## Action:

affecte un texte à une écriture

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@settext(@varec,"125")

# @settobj

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| classe | STANDARD | classe (indice) du type d'objet à modifier |
| nom parametre | STANDARD | nom du paramètre à modifier |
| value | STANDARD | valeur du parametre |

## Action:

modifie une propriété du type d'objet de classe "classe". Cette fonction renvoie une erreur si la propriété est non valide. Les propriétés modifiables du type d'objet sont :

- Nom = nom en clair de la classe
- NbFcMin = nombre mini de face de ce type d'objet
- NbLiMin = nombre mini de liaisons de ce type d'objet
- NbPtMin = nombre mini de points de ce type d'objet
- NbEcMin = nombre mini d'écritures de ce type d'objet
- NbFcMax = nombre maxi de face de ce type d'objet
- NbLiMax = nombre maxi de liaisons de ce type d'objet
- NbPtMax = nombre maxi de points de ce type d'objet
- NbEcMax = nombre maxi d'écritures de ce type d'objet
- FcDesPriority = priorité de dessin des faces de cette classe
- LiDesPriority = priorité de dessin des liaisons de cette classe
- PtDesPriority = priorité de dessin des points de cette classe
- EcDesPriority = priorité de dessin des écritures de cette classe
- MaskCrFc = numéro d'ordre du masque de création de face pour cette classe
- MaskCrLi = numéro d'ordre du masque de création de liaison pour cette classe
- MaskCrPt = numéro d'ordre du masque de création de point pour cette classe
- MaskCrEc = numéro d'ordre du masque de création d'écriture pour cette classe
- MaskMdFc = numéro d'ordre du masque de modification de face pour cette classe
- MaskMdLi = numéro d'ordre du masque de modification de liaison pour cette classe
- MaskMdPt = numéro d'ordre du masque de modification de point pour cette classe
- MaskMdEc = numéro d'ordre du masque de modification d'écriture pour cette classe
- MaskChFc = numéro d'ordre du masque de recherche de face pour cette classe
- MaskChLi = numéro d'ordre du masque de recherche de liaison pour cette classe
- MaskChPt = numéro d'ordre du masque de recherche de point pour cette classe
- MaskChEc = numéro d'ordre du masque de recherche d'écriture pour cette classe
- Select = Elément de sélection de cette classe (1=point,2=liaison,4=écriture,16=face)
- Nature = Nature des objets de cette classe (1=point,2=liaison,4=écriture,16=face)
- AutoDetect = Indique si détection automatique de l'objet (1=oui, 0=non)
- DefCouche = couche par défaut pour ce type d'objet. Attention! si un numero de couche plus important que celui précédemment attribué est fixé, l'opération inverse ne sera pas possible tant que l'application n'est pas fermée. Cette modification peut entrainer une modification au niveau du document de telle sorte qu'il ne soit plus possible de le relire avec le SCD original (sauf à perdre les objets).
- Topology = réservé
- CouleurFc = couleur sous forme 0x00BBGGRR des faces de ce type d'objet (en visions polychromes)
- CouleurLi = couleur sous forme 0x00BBGGRR des liaisons de ce type d'objet (en visions polychromes)
- CouleurPt = couleur sous forme 0x00BBGGRR des points de ce type d'objet (en visions polychromes)
- CouleurEc = couleur sous forme 0x00BBGGRR des écritures de ce type d'objet (en visions polychromes)
  *NB: une couleur à −1 indique pas de tracé (la valeur stockée dans le fichier INI est alors 0xFF000000)*
- IdAlgo = algorithme de calcul de l'identifiant : la commande est alors identique à SetIdAlgo
- IdParm = paramètre pour le calcul de l'identifiant : la commande est alors identique à SetIdParm
- RayonDetect = rayon de détection en cas de construction automatique de l'objet
- PrId = Préfixe d'identifiant : la commande est alors identique à SetPrId. Attention! modifier cette valeur peut couper le lien avec une base de donnée déjà constituée car le lien est composé de l'identifiant complet lui même constitué de ce préfixe de 4 lettres indiquant le type d'objet (ex : PARC = préfixe pour les parcelles)

La valeur Init du type d'objet peut être modifiée par la fonction @setinit

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | type d'objet modifié |

## Exemples:

| | |
|---|---|
| @settobj(5,CouleurFc,0xFF0000) | met la couleur des faces de parcelles en bleu |
| @settobj(12,NbEcMin,0) | fixe le nombre d'écriture minimal des objets de type "voie publique" à 0, permettant ainsi de fournir des objets voies publique sans écriture et sans que soit détecté une anomalie de cardinalité sur des objets de ce type |
| @settobj(4,CouleurFc,−1) | Met la couleur des faces de la classe "Lieudits" (4) à la valeur None c'est à dire invisible |

# @settp2bmp

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de couche (de 0 à NbCouches−1) |
| mat | STANDARD | série de 6 réel représentant respectivement ai,bi,ci,di,pi,qi coefficients de la matrice de transfert coordonnées terrain (ou papier) vers les coordonnées Bitmap.<br>NB: les coordonnees Bitmap sont S−E |

## Action:

modifie la matrice de transformation des coordonnées terrain/papier vers les coordonnees Bitmap.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | propriété traduite et écrite |

## Exemples:

@settp2bmp(2, 1 0 0 −1 0 0) 1 pixel du bitmap = 1 mètre

# @settrel

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| trelsem | STANDARD | indice du type de relation sémantique à modifier |
| nom parametre | STANDARD | nom du paramètre à modifier |
| value | STANDARD | valeur du parametre |

## Action:

modifie une propriété du type de relation sémantique d'indice "trelsem" (de 0 à ...). Cette fonction renvoie une erreur si la propriété est non valide. Les propriétés modifiables du type de relation sémantique sont :

- Nom = nom en clair du type de relation sémantique
- NbRelMin = nombre mini de relations directes de ce type de relation sémantique
- NbIRelMin = nombre mini de relations inverses de ce type de relation sémantique
- NbRelMax = nombre maxi de relations directes de ce type de relation sémantique
- NbIRelMax = nombre maxi de relations inverses de ce type de relation sémantique
- ClasseSrce = classe de l'objet source ou 0 si la source est un élément. MaskChSrce est alors remis à 0 lors d'une modification.
- ClasseDest = classe de l'objet destination ou 0 si la destination est un élément. MaskChDest est alors remis à 0 lors d'une modification.
- TEltSrce = 0 si la source est un objet, sinon combinaison des types d'éléments pouvant se trouver en source (1=EltPoint,2=EltLiaison,4=EltEcriture,16=EltFace). MaskChSrce est alors remis à 0 lors d'une modification.
- TEltDest = 0 si la destination est un objet, sinon combinaison des types d'éléments pouvant se trouver en destination (1=EltPoint,2=EltLiaison,4=EltEcriture,16=EltFace). MaskChDest est alors remis à 0 lors d'une modification.
- MaskChSrce = numéro d'ordre du masque de recherche de face, liaison, écriture, point ou élément suivant la valeur dans TEltSrce
- MaskChDest = numéro d'ordre du masque de recherche de face, liaison, écriture, point ou élément suivant la valeur dans TEltDest
- Methode = Methode de détermination automatique de la relation (TEltSrce et TEltDest sont alors des valeurs simples et non des combinaisons de valeurs)
  0 = RELSEMMETHODE_MANUEL = calcul manuel
  1 = RELSEMMETHODE_APP = calcul par appartenance
  2 = RELSEMMETHODE_INCL = calcul par inclusion
  3 = RELSEMMETHODE_PROX = calcul par proximité (seuil maxi fixé par Param)
  4 = RELSEMMETHODE_DEPEND = calcul par dépendance
- Couleur = couleur sous forme 0x00BBGGRR de représentation de ce type de relation sémantique (en visions polychromes)
- Epaisseur = épaisseur de représentation de ce type de relation sémantique.
- Param = paramètre précisant la Methode (valeur maxi en mètre de la proximité)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | type de relation sémantique modifié |

## Exemples:

| | |
|---|---|
| @settrel(0,Couleur,0xFF0000) | met la couleur des "correspondances" en bleu |
| @settrel(1,NbRelMin,0) | fixe le nombre de relations minimales entre un bati dur et sa parcelle à 0 |

# @settrf

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| a | STANDARD | coefficient matriciel |
| b | STANDARD | coefficient matriciel |
| c | STANDARD | coefficient matriciel |
| d | STANDARD | coefficient matriciel |
| p | STANDARD | coefficient matriciel |
| q | STANDARD | coefficient matriciel |

## Action:

Charge une transformation dans la fenêtre courante à partir des coefficients matriciels fournis
Les coefficients sont tels que
ax+by+p=X
cx+dy+q=Y.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 | NUM | la transformatrion n'a pu être chargée (transformation non inversible, coefficients incorrects...) |
| 1 | NUM | la transformation a été chargée. |

## Exemples:

| | |
|---|---|
| @settrf(2,0,0,2,0,0) | charge une transformation permettant de doubler l'échelle |

# @setvar

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier ou créer |
| value | ??? | valeur à affecter (peut être 0 pour les variables pointeurs non initialisées)<br>Dans le cas ou on fourni un paramètre standard à une variable pointeur la signification de la valeur est alors la suivante:<br>PT : numéro de point (pointeur nul si non trouvé)<br>LI : dans tous les cas initialisé avec le pointeur nul<br>FC : idem<br>EC : idem<br>OBJ : numéro d'objet (pointeur nul si non trouvé)<br>MASK... : classe du masque de classe à aller chercher (pointeur nul si non trouvé ou hors classe)<br>RELSEM : numero de relation sémantique (la première n'est pas accessible)<br>Dans tous les cas si 0 est fourni, c'est le pointeur NULL qui est affecté à la variable |
| type | STANDARD | chaine représentant le type de variable<br>• STR: Type chaine de caractères<br>• NUM: Type numérique<br>• ANG: Type Angle<br>• BOOL: Type Booléen<br>• PT : type élément point<br>• LI :type élément liaison<br>• FC: type élément face<br>• EC: type élément écriture<br>• OBJ: type objet<br>• MASKCRPT: type masque de création de point<br>• MASKCRLI: type masque de création de liaison<br>• MASKCRFC: type masque de création de face<br>• MASKCREC: type masque de création d'écriture<br>• MASKMDPT: type masque de modification de point<br>• MASKMDLI: type masque de modification de liaison<br>• MASKMDFC: type masque de modification de face<br>• MASKMDEC: type masque de modification d'écriture<br>• MASKCHPT: type masque de recherche de point<br>• MASKCHLI: type masque de recherche de liaison<br>• MASKCHFC: type masque de recherche de face<br>• MASKCHEC: type masque de recherche d'écriture<br>• RELSEM : type relation sémantique<br>• MAPDOC : type document MAP<br>• PLANVIEW : type vue (fenêtre de visualisation) fenêtre Plan |

## Action:

Modification d'une variable ou création d'une variable utilisable dans tout le programme TED en cours
La valeur de la variable et la variable est perdue une fois la procédure TED appelante de plus haut niveau terminée (son nom et sa valeur d'initialisation peuvent rester si il s'agit d'une variable globale utilisateur).
Cette fonction ne crée pas de variable si l'option du mode debug est activé (cf @setdebug)
La fonction recherche une variable (locale ou générale) du nom donné à partir de la fin de la pile et la modifie si elle la trouve, si elle ne la trouve pas, elle crée une variable générale visible par tout le programme
Une variable générale est une variable "locale" de plus haut niveau, c'est à dire en principe générée par le programme TED qui a été lancé, et non par une sous procédure.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| [nomvar] | ??? | renvoie la variable (après affectation ou modification) |

## Exemples:

| | |
|---|---|
| @setvar(mask,5,MASKMDFC) | affecte à la variable "mask" le masque de modification des faces de la classe parcelle |

# @setvision

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| vision | STANDARD | type de vision à mettre :<br>−1 = pas de changement<br>0 = vision element<br>1 = vision classe mono<br>2 = vision classe poly<br>3 = vision objet mono<br>4 = vision objet poly |

## Action:

change la vision de la fenetre courante ou fournit la vision courante si 'vision'=−1

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <vision> | NUM | vision après changement |

## Exemples:

| | |
|---|---|
| @setvision(1) | établit la vision classe monochrome pour la fenetre |
| @setvision(−1) | fournit la vision courante |

# @setx

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément |
| elt | ou | ou |
| | PT, EC | élément dont la coordonnee est à modifier |
| X | STANDARD | coordonnée X |

## Action:

modifie la coordonnée X d'un élément

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setx(@pt,@add(@getx(pt),5)) | déplace le point de 5 m en X |

# @sety

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un élément |
| elt | ou | ou |
| | PT, EC | élément dont la coordonnee est à modifier |
| X | STANDARD | coordonnée Y |

## Action:

modifie la coordonnée Y d'un élément

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @sety(@pt,@add(@gety(pt),5)) | déplace le point de 5 m en Y |

# @setz

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un point |
| elt | ou | ou |
| | PT | point dont la coordonnee est à modifier |
| Z | STANDARD | altitude Z |

## Action:

modifie la coordonnée Z d'un point

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

| | |
|---|---|
| @setz(@pt,125.34) | fixe l'altitude du point |

# @setzonegeo

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | | chaine représentant le nom de la projection Lambert |
| zonegeo | STANDARD | cette information doit être conforme aux chaines normalisées de la nomenclature EDIGéO si on veut que l'export EDIGéO soit conforme à la norme (LAMB1, LAMB2, LAMB3, LAMB4 pour les zones Lambert) |

## Action:

modifie la zone lambert du document (uniquement l'information et non les coordonnées), propriété du document

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | |

## Exemples:

@setzonegeo("LAMB3")

# @sin

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm | ANG | fournit le sinus du paramètre exprimé sous forme normalisé des angles (ex: 125.3342gv) |

## Action:

renvoie le sinus de l'angle

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <sin(parm)> | NUM | renvoie le sinus |

## Exemples:

| | |
|---|---|
| @sin(@var) | renvoie le sinus de la variable |

# @sqrt

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| valeur | STANDARD | valeur dont il faut extraire la racine |

## Action:

sqrt[nomvar]

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| racine carrée d'un nombre négatif | ERR | la variable contient un nombre négatif |
| [sqrt(val)] | NUM | renvoie la racine carrée |

## Exemples:

| | |
|---|---|
| @sqrt(@mavar) | renvoie la racine carré de la variable |

# @sqrtvar

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |

## Action:

[nomvar] <== sqrt[nomvar]

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| racine carrée d'un nombre négatif | ERR | la variable contient un nombre négatif |
| imposible de calculer la racine carrée d'une variable pointeur | ERR | il est impossible d'extraire la racine d'une variable non standard |
| [nomvar] | ??? | renvoie la variable après opération |

## Exemples:

| | |
|---|---|
| @sqrtvar(mavar) | renvoie la racine carré de la variable elle même modifiée |

# @str2base

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| code | STANDARD | chaine codée désignant la base de données :<br>S = standard<br>Cn = base de classe "n" (de 0 à ...)<br>An = base auxiliaire de niveau "n" (de 0 à ...)<br>En = base externe de niveau "n" (de 0 à ...) |

## Action:

convertit un code sous forme de chaine de caractères désignant une base de donnée en indice dans la table des bases de données de l'applicatuion.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | indice de la base de donnée ou −1 si la chaine est invalide ou indique une base non présente |

## Exemples:

| | |
|---|---|
| @str2base(C5) | fournit l'indice de  la base des parcelles (4 en principe) |

# @sub

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| val1 | STANDARD | valeur 1 |
| val2 | STANDARD | valeur 2 |

## Action:

val1 − val2
Attention: comportement particulier si les deux valeurs sont de type angulaire : 10gv−20gv = 390gv

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <val1−val2> | ??? | résultat |

## Exemples:

| | |
|---|---|
| @sub(@mavar,17) | retranche 17 au contenu de la variable et renvoie le résultat |

# @substr

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| texte | STANDARD | texte dont il faut extraire la chaine |
| deb | STANDARD | position du premier caractère de la chaine à extraire (de 0 à ...) |
| nbcar | STANDARD | nombre de caractères à extraire |

## Action:

extrait une chaine d'une autre chaine. Si (deb+1) est supéreur au nombre de caractères de la chaine originale alors renvoie "paramètre invalide"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <substring> | STR | renvoie la chaine extraite |

## Exemples:

| | |
|---|---|
| @substr(commenter,3,4) | renvoie "ment" |

# @subvar

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable à modifier |
| value | STANDARD | valeur à retrancher de la valeur de la variable |

## Action:

[nomvar] <== [nomvar] − value
Attention: comportement particulier si les deux valeurs sont de type angulaire : 10gv−20gv = 390gv

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| variable inconnue | ERR | le nom n'est pas celui d'une variable |
| impossible de retrancher à une variable pointeur | ERR | si la variable n'est pas standard |
| [nomvar] | ??? | renvoie la variable après opération |

## Exemples:

| | |
|---|---|
| @subvar(mavar,17) | retranche 17 au contenu de la variable et la renvoie |

# @sup

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm1 | STANDARD | premier paramètre |
| parm2 | STANDARD | second paramètre |

## Action:

opérateur booleen parm1 > parm2
la valeur stockées sont comparées suivant leur nature :
elles seront comparées comme étant des valeurs numériques décimales seulement si ce sont deux valeurs de type numérique.
elles seront comparées comme étant des angles au format TopoCad (ex: 125.12dv) seulement si ce sont des valeurs de type angle
Si une variable numérique contient une valeur octale ou hexadécimale, la comparaison peut ne pas être celle désirée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| parm1>parm2 | BOOL | renvoie 0 ou 1 |

## Exemples:

| | |
|---|---|
| @sup(@var1,1000) | renvoie 1 si le contenu de la variable "var" est supérieur à 1000 |

# @supe

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| parm1 | STANDARD | premier paramètre |
| parm2 | STANDARD | second paramètre |

## Action:

opérateur booleen parm1 >= parm2
la valeur stockées sont comparées suivant leur nature :
elles seront comparées comme étant des valeurs numériques décimales seulement si ce sont deux valeurs de type numérique.
elles seront comparées comme étant des angles au format TopoCad (ex: 125.12dv) seulement si ce sont des valeurs de type angle
Si une variable numérique contient une valeur octale ou hexadécimale, la comparaison peut ne pas être celle désirée

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| parm1>=parm2 | BOOL | renvoie 0 ou 1 |

## Exemples:

| | |
|---|---|
| @supe(@var1,1000) | renvoie 1 si le contenu de la variable "var" est supeérieur ou égal à 1000 |

# @surf

*Nb paramètres indéterminé*

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| num1 | STANDARD PT | numéro du point 1 ou point 1 |
| num2 | STANDARD PT | numéro du point 2 ou point 2 |
| num3 | STANDARD PT | numéro du point 3 ou point 3 |
| num4 | STANDARD PT | numéro du point 4 ou point 4 |

...etc

## Action:

calcule et renvoie la surface du polygone décrit par les points fournis. Le dernier point peut ne pas avoir le même numéro que le premier point (le dernier point est alors implicitement le premier)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Point inconnu | ERR | un numéro de point donné est inconnu dans le document ou le numéro donné n'est pas un numéro (par ex chaine de caractères) |
| Nombre de paramètres insuffisant | ERR | si le nombre de points est insuffisant. |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <surface> | NUM | surface du polygone (avec 21 décimales) |

## Exemples:

| | |
|---|---|
| @surf(10,32,76,128) | calcule et renvoie la surface du polygone défini par les points 10−32−76−128−10 |
| @surf(1,2,@varpt) | varpt étant une variable PT contenant le point 3, calcule la surface du triangle 1−2−3 |

# @surffc

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| face | STANDARD<br>FC | nom d'une variable contenant une face<br>ou la face elle même |

## Action:

calcule et renvoie la surface de la face (qui peut être une face à trou)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [...] Variable inconnue | ERR | aucun nom de variable ne correspond |
| paramètre invalide | ERR | type inaproprié du paramètre |
| <surface> | NUM | surface de la face |

## Exemples:

@surffc(varfc)

# @surfobj

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | nom d'une variable contenant un objet |
| obj | ou | ou |
| | OBJ | objet |

## Action:

calcule et renvoie la surface des faces de l'objet

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| surface | NUM | surface de l'objet |

## Exemples:

| | |
|---|---|
| @surfobj(obj) | surface de l'objet contenu dans la variable "obj" |

# @tableload

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| type | STANDARD | type de fichier INI attendu<br><br>0 = tables de correspondance Edigéo/PCI<br>1 = tables de correspondance DXF<br>2 = tables de correspondance MIF<br>3 = tables de correspondance SHP<br>4 = tables de correspondance Edigéo<br>5 = table de correspondance Apic<br>6 = table de correspondance OSM<br>7 = table de correspondance KML |
| nomfic | STANDARD | chemin et nom complet du fichier INI contenant la ou les tables de correspondance (un fichier INI peut contenir une table de correspondance d'import et une table de correspondance d'export du même type) |

## Action:

charge une interface de traduction sous forme d'une ou deux tables de correspondance de même type contenu dans un fichier INI

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Aucune table de correspondance XXX détectée | ERR | |
| OK | STR | table(s) chargée(s) |

## Exemples:

| | |
|---|---|
| @tableload(1,"c:\dxf_cadastre.ini") | charge l'interface pour le DXF cadastre |

# @texthorsface

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nivsel | STANDARD | niveau de sélection à positionner pour les écritures en dehors de leurs face (de 0 à 31) |

## Action:

sélectionne au niveau "nivsel" toutes les écritures en dehors des faces de l'objet dont elles font partie, ceci pour les objets des classes actives.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N écritures détectées hors de faces. | NUM | N = Nombre d'écritures sélectionnées |

## Exemples:

@texthorsface(0)

# @TextObj

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| varobj | STANDARD | nom d'une variable contenant l'objet dont il faut extraire les écritures |

## Action:

fournit l'ensemble des écritures de l'objet dans une chaine composée des écritures ordonnées séparées par un espace

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Variable inconnue | ERR | nom de variable inconnu |
| <chaine> | STR | texte de l'objet |

## Exemples:

@TextObj(mavar)

# @topologie

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| nivselsrce | STANDARD | niveau de sélection des points et liaisons à recoller source (de 0 à 31) |
| nivseldest | STANDARD | niveau de sélection des points et liaisons à recoller destination (de 0 à 31) |
| tol1 | STANDARD | tolérance maxi de déplacement que peut supporter la source. |
| tol2 | STANDARD | tolérance maxi de déplacement que peut supporter la destination |
| action | STANDARD | combinaison des valeurs suivantes<br><br>0x4000 = TOPOLOGY_INTSRCE<br>0x8000 = TOPOLOGY_INTDEST<br>0x0100 = TOPOLOGY_CREEREL<br>0x0200 = TOPOLOGY_FROMREL<br>0x1000 = TOPOLOGY_FROMPOS<br>0x0001 = TOPOLOGY_PTPT<br>0x0002 = TOPOLOGY_LILI<br>0x0010 = TOPOLOGY_PTLI_SRCEDEST<br>0x0020 = TOPOLOGY_PTLI_DESTSRCE<br>0x2000 = TOPOLOGY_INTERSECTE<br>0x0400 = TOPOLOGY_NOCROSSREL<br>0x0004 = TOPOLOGY_EXCLUSSELECT |

## Action:

cette fonction effectue une opération de recollement avec fusion de points et liaisons entre des élémensts sources et des élémensts destination

Les opérations sont faites dans l'ordre suivant

INTSRCE = réunit au préalable en un seul point les points de la source à une distance inférieure à tol1

INTDEST = réunit au préalable en un seul point les points de la destination à une distance inférieure à tol2

INTERSECTE = crée les points à l'intersection de la source et de la destination et les écarte du reste du traitement

Si CREEREL

PTPT = crée une correspondance entre les points les plus proche de la source vers la destination (et les écarte du reste du tritement)

PTLI_SRCEDEST = crée une correspondance entre les points de la source vers les liaisons de la destination (en créant le point sur la destination)

PTLI_SRCEDEST = crée une correspondance entre les points de la destination vers les liaisons de la source (en créant le point sur la source)

NOCROSSREL = evite que les correspondances créées ne se croisent

Sinon si FROMPOS

PTPT = réunit les points les plus proches entre eux (et les écarte du reste du tritement)

LILI = fusionne les liaisons entre elles

PTLI_SRCEDEST = crée et fusionne les points de la source sur les liaisons de la destination

PTLI_SRCEDEST = crée et fusionne les points de la destination sur les liaisons de la source

LILI = fusionne à nouveau les liaisons entre elles

les points traités sont exclus de la sélection si EXCLUSSELECT

FROMREL: effectue les rapprochements avec les correspondances créées (avec CREEREL) ou existantes (dans ce dernier cas les correspondances de l'ensemble du document sont considérées)

PTPT = effectue la réunion entre les points

LILI = effectue la fusion des liaisons

la position des points réunis est fonction des tolérances tol1 et tol2. Si tol1==tol2 le point résultant sera excatement au milieu des deux points, si tol1=0 et tol1=1, le point résultant sera le point source, si tol1=1 et tol1=0, le point résultant sera le point destination.

La fonction échoue si la couche est non intègre

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N | NUM | N = Nombre de points déplacés |

## Exemples:

| | |
|---|---|
| @topologie(0,1,0,0.35,0.35,0xC033) | amène les points de la couche 0 sur les points ou liaisons sélectionnées de la couche 1 à mi distance si la distance à ces points ou liaisons est inférieure à 0.70 m |

# @topologiecou

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si toutes les couches doivent être traitées les une après les autres. |
| nivsel | STANDARD | niveau de sélection des éléments à considérer (de 0 à 31) ou −1 si tous les éléments de la couche sont à prendre en compte |
| epsilon | STANDARD | indique l'écart maxi au dela duquel TopoCad ne fusionne plus un point avec un point ou un point avec une liaison. |

## Action:

cette fonction regroupe les actions des fonctions TED @SelProxPtPt, @ConcatenePt, @SelMultLi, @ConcateneLi, @SelProxPtLi, @ConcatPtLi, @SelMultLi, @ConcateneLi mais sans opérer de sélection, c'est à dire effectue un premier traitement des points et liaisons en vue d'obtenir la topologie : fusion des points proches de points et de points proches de liaisons.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STR | traitement réalisé |

## Exemples:

| | |
|---|---|
| @topologie(0,0,0.01) | traitement de topologie à 1 cm près de la couche 0 |

# @topologiefc

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe des faces (toutes celles actives si −1) |
| seldest | STANDARD | niveau de sélection à positionner des faces qui s'intersectent (de 0 à 31) |

## Action:

sélectionne au niveau "nivsel" les faces de classe "classe" qui s'intersectent

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N Faces sélectionnées se recoupent | STR | N = Nombre de faces |

## Exemples:

| | |
|---|---|
| @topologiefc(0,5,0) | sélectionne les faces de classe parcelle qui se recoupent |

# @topologieli

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) ou −1 si ttoutes les couches |
| classe | STANDARD | classe des liaisons (toutes les classes actives si −1) |
| seldest | STANDARD | niveau de sélection à positionner des liaisons qui s'intersectent (de 0 à 31) |
| epsilon | STANDARD | indique l'epsilon en deca duquel l'intersection n'est pas considéré |

## Action:

sélectionne au niveau "nivsel" les liaisons de classe "classe" qui s'intersectent (afin par ex de réaliser une topologie de niveau 2)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N Liaisons sélectionnées se recoupent | STR | N = Nombre de liaisons |

## Exemples:

| | |
|---|---|
| @topologieli(0,5,0,0) | sélectionne les liaisons de classe parcelle qui se recoupent |

# @topologieobj

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| objsrce | STANDARD ou OBJ | objet ou nom de variable contenant l'objet source |
| objdest | STANDARD ou OBJ | objet ou nom de variable contenant l'objet destination |
| nivsel | STANDARD | niveau de sélection des éléments à considérer (de 0 à 31) ou −1 si tous les éléments des objets sont à prendre en compte |
| tol1 | STANDARD | tolérance maxi de déplacement que peut supporter la source. |
| tol2 | STANDARD | tolérance maxi de déplacement que peut supporter la destination |
| action | STANDARD | combinaison des valeurs suivantes<br><br>0x4000 = TOPOLOGY_INTSRCE<br>0x8000 = TOPOLOGY_INTDEST<br>0x0100 = TOPOLOGY_CREEREL<br>0x0200 = TOPOLOGY_FROMREL<br>0x1000 = TOPOLOGY_FROMPOS<br>0x0001 = TOPOLOGY_PTPT<br>0x0002 = TOPOLOGY_LILI<br>0x0010 = TOPOLOGY_PTLI_SRCEDEST<br>0x0020 = TOPOLOGY_PTLI_DESTSRCE<br>0x2000 = TOPOLOGY_INTERSECTE<br>0x0400 = TOPOLOGY_NOCROSSREL<br>0x0004 = TOPOLOGY_EXCLUSSELECT |

## Action:

cette fonction effectue une opération de recollement avec fusion de points et liaisons entre les éléments de l'objet source et les élémensts de l'objet destination sélectionnés au niveau 'nivsel'

Les opérations sont faites dans l'ordre suivant

INTSRCE = réunit au préalable en un seul point les points de la source à une distance inférieure à tol1

INTDEST = réunit au préalable en un seul point les points de la destination à une distance inférieure à tol2

INTERSECTE = crée les points à l'intersection de la source et de la destination et les écarte du reste du traitement

Si CREEREL

> PTPT = crée une correspondance entre les points les plus proche de la source vers la destination (et les écarte du reste du tritement)
>
> PTLI_SRCEDEST = crée une correspondance entre les points de la source vers les liaisons de la destination (en créant le point sur la destination)
>
> PTLI_SRCEDEST = crée une correspondance entre les points de la destination vers les liaisons de la source (en créant le point sur la source)
>
> NOCROSSREL = evite que les correspondances créées ne se croisent

Sinon si FROMPOS

> PTPT = réunit les points les plus proches entre eux (et les écarte du reste du tritement)
>
> LILI = fusionne les liaisons entre elles
>
> PTLI_SRCEDEST = crée et fusionne les points de la source sur les liaisons de la destination

PTLI_SRCEDEST = crée et fusionne les points de la destination sur les liaisons de la source

LILI = fusionne à nouveau les liaisons entre elles

les points traités sont exclus de la sélection si EXCLUSSELECT

FROMREL: effectue les rapprochements avec les correspondances créées (avec CREEREL) ou existantes (dans ce dernier cas les correspondances de l'ensemble du document sont considérées)

PTPT = effectue la réunion entre les points

LILI = effectue la fusion des liaisons

la position des points réunis est fonction des tolérances tol1 et tol2. Si tol1==tol2 le point résultant sera excatement au milieu des deux points, si tol1=0 et tol1=1, le point résultant sera le point source, si tol1=1 et tol1=0, le point résultant sera le point destination.

La fonction échoue si les objets sont non intègres

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| 0 ou 1 | BOOL | traitement réalisé (1) ou non (0) |

## Exemples:

| | |
|---|---|
| @topologie(obj1,obj2,−1,0.005,0.005,0xC033) | traitement de topologie à 1 cm près des deux objets |

# @topologierco

*Nb paramètres*
4

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche concernée (de 0 à Nb_Couches_document−1) |
| classe | STANDARD | classe dont il faut controler la topologierco |
| nivsel | STANDARD | niveau de sélection à positionner pour les cas ou la topologierco n'est pas respectée (de 0 à 31) |
| flags | STANDARD | série de drapeaux indiquant les tests à réaliser<br>0x01 = CHECKTOPOLOGY_OBJ = indique si controle aux niveau des objets (sinon des classes)<br>0x02 = CHECKTOPOLOGY_ANO = indique si sélection des anomalies<br>0x04 = CHECKTOPOLOGY_ONLYCL = considère uniquement les liaisons de la classe<br>0x08 = CHECKTOPOLOGY_MULTOBJ = indique les éléments faisant partie de plusieurs objets<br>0x70 = CHECKTOPOLOGY_EXT = indique si sélection contour externe (union des 3 drapeaux suivants)<br>0x10 = CHECKTOPOLOGY_TROUS = indique les trous<br>0x20 = CHECKTOPOLOGY_RECOUVR = indique les recouvrements<br>0x40 = CHECKTOPOLOGY_FACE0 = indique la face externe<br>0x80 = CHECKTOPOLOGY_CREATEFC= création des faces (trous, recouvrt et face0). Ce drapeau est inefficace si plus d'un des 3 drapeaux précédents sont positionnés simutanément. |

## Action:

controle des relations de construction Face par rapport aux liaisons pour la couche "couche" et la classe "classe", sélectionne les anomalies détectées au niveau "nivsel"

exemple d'utilisation : le flag CHECKTOPOLOGY_ONLYCL est utilisé en combinaison avec CHECKTOPOLOGY_EXT pour détecter les liaisons qui ne sont pas de classe parcelle collées à des faces de classe parcelle.

La détection des trous, recouvrements et faces externes ne peut etre réalisé correctement que dans le cas ou la couche est intègre et ou une partie de la topologie est réalisée : au niveau 1 (topologie de couche) et niveau 2 (création des points à l'intersection des liaisons, c'est à dire aucune liaison ne doit se couper).
La détection des trous, recouvrements, faces externes, sélectionne les contours de ces élément ainsi que les faces éventuellement créées.

Le drapeau CHECKTOPOLOGY_ANO indique une sélection des anomalies c'est à dire des liaisons ayant plusieurs faces à droite ou à gauche ou n'ayant aucune face à droite et à gauche.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| N anomalies | NUM | nombre d'anomalies détectées |

## Exemples:

| | |
|---|---|
| @topologierco(0,5,0,1) | controle des relations des objets parcelle |

# @transfertcouche

*Nb paramètres*
6

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couchesrce | STANDARD | numéro de la couche source (de 0 à Nb_Couches_document−1) |
| couchedest | STANDARD | numéro de la couche destination (de 0 à Nb_Couches_document−1) |
| sel | STANDARD | sélection à transférer (ou −1 si toute la couche est à transférer) |
| copie | STANDARD | indique si le transfert est en copie (1) ou en déplacement (0) |
| typetransf | STANDARD | indique le type de transfert<br>0 = brut<br>1 = par la transformation de la fenêtre courante<br>2 = par la transformation inverse de la fenêtre courante<br>3 = par calquage sur la fenêtre courante |
| vect_rast | STANDARD | indique ce qui est transféré de la couche<br>0x00 = demande faite à l'écran<br>0x01 = raster<br>0x10 = vectoriel<br>0x11 = raster + vectoriel |

## Action:

copie ou déplace la couche "couchesrce" sur la couche "couchedest". Cette opération concerne la partie Raster comme la partie Vectorielle de la couche. Si la couche destination possède un Raster et la couche source également, la couche source garde son Raster.

ATTENTION: Cette opération sélectionne (au niveau courant) tous les éléments de la couche source et éléments dépendants sauf les symboles (signes et deportec, prendre précautions en conséquence).

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| Les éléments sélectionnés ne forment pas un bloc indépendant! | ERR | si impossibilité de transférer |
| Erreur Interne | ERR | tentative de transfert du bitmap qui n'existe pas |
| Des éléments ou objets sont vérouillés!!! | ERR | si impossibilité de transférer |
| OK | STR | le transfert a eu lieu |

## Exemples:

| | |
|---|---|
| @transfertcouche(0,1,0,0,0x10) | déplace la couche 0 sur la couche 1 |

# @transfpt

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| pt | PT | point à transformer |
| type | STANDARD | type de transformation sous forme d'entier<br>>=0 : transformation directe courante de la fenêtre utilisée<br><0 : transformation inverse courante de la fenêtre utilisée |

## Action:

Effectue une transformation sur le point "pt" à partir de la transformation chargée dans la fenêtre courante (il existe une transformation par fenêtre) : la transformation directe ou inverse est utilisée suivant la valeur de "type"

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| Ok | STD | Transformation réalisée |

## Exemples:

| | |
|---|---|
| @transfpt(@monpt,1) | transforme le point "monpt" par la transformation de la fenêtre. |

# @vartype

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| nomvar | STANDARD | nom de la variable dont il faut chercher le type |

## Action:

renvoie le type de la variable sous forme de chaine de caractère (STR,NUM,PT,LI,FC,MASKCRFC....)

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si la variable n'existe pas |
| [type] | STR | founit le type de la variable |

## Exemples:

| | |
|---|---|
| @setvar(mavar,"test",STR),<br>@vartype(mavar) | renvoie "STR" |

# @while

*Nb paramètres*
2

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| cond | ??? | condition. (un opérateur booléen existe sur tout type) |
| commande | ??? | commande (pour une liste de commandes à executer voir @list ou @exec) |

## Action:

boucle conditionnelle
Il est possible d'interrompre la boucle par ESC

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Interruption utilisateur | ERR | boucle interrompue par l'appui de ESC de l'utilisateur |
| | ??? | renvoie le résultat de la dernière commande |

## Exemples:

```
@while(@curt,
   @list(
      @addvar(stot,@surffc(curt)),
      @NextWithMask(curt,mask)
      )
   )
```

tant que la variable "curt" n'est pas nulle, calculer la surface de "curt" la rajouter à "stot" et passer à la face suivante (curt contenant une face)

# @z

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| | STANDARD | numéro du point |
| numpt | ou | ou |
| | PT | point dont il faut extraire l'altitude |

## Action:

fournit l'altitude (Z) d'un point

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Point inconnu | ERR | si le numéro fourni n'existe pas ou si ce n'est pas un numéro de point |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| <Z> | NUM | altitude du point |

## Exemples:

| | |
|---|---|
| @z(125) | altitude du point de numéro 125 |

# @zapclasse

*Nb paramètres*
3

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| couche | STANDARD | numéro de la couche dans laquelle il faut supprimer la classe (de 0 à Nb_Couches_document−1) ou −1 si toutes les couches sont concernées |
| classe | STANDARD | classe à supprimer |
| flags | STANDARD | ensemble de drapeaux indiquant les actions à entreprendre :<br>ZAPCLASSE_OBJ = 0x01 = les éléments appartenant à l'objet sont supprimés s'ils ne font pas partie d'autres objets et si leur suppression n'entraine pas la suppression d'autres éléments devant être conservés, sinon la classe de ces éléments est recalculée en fonction de l'éventuelle appartenance à un autre type d'objet dans le cas où la précédente classe de l'élément était la classe 'classe'.<br>ZAPCLASSE_OBJBORD = 0x02 = les éléments appartenant uniquement à la bordure de l'objet sont supprimés dans les mêmes conditions que ci dessus.<br>ZAPCLASSE_CLA = 0x04 = les éléments de classe 'classe' sont supprimés s'ils ne font pas partie d'autres objets et si leur suppression n'entraine pas la suppression d'autres éléments devant être conservés.<br>ZAPCLASSE_CLABORD = 0x08 = les éléments uniquement en bordure d'éléments de classe 'classe' sont supprimés dans les mêmes conditions que ci dessus. |

## Action:

nettoie une couche (ou ttes les couches) des objets ou éléments de classe 'classe'. La méthode de suppression de tout objet ou tout élément de classe 'classe' est fixée par les drapeaux 'flags'.
cette commande peut être utilisée conjointement avec la commande @distrib qui copie les objets ou éléments d'une couche sur une autre afin de nettoyer la couche source des objets/elts distribués.
Il est nécessaire que les éléments concernés soient intègres et que tous les objets soient intègres, sinon la commande échoue.
La classe des éléments succeptibles de subir une modification d'après le modèle est recalculée et rectifiée.

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| Paramètre invalide | ERR | si les paramètres ont des types non appropriés |
| couche non intègre | ERR | des objets ou des éléments non intègres empèchent le traitement |
| OK | STD | nettoyage effectué |

## Exemples:

| | |
|---|---|
| @zapclasse(1,1,3) | supprime tous les objets communes de la couche 1 et leurs graphisme associé, mais pas les éventuelles amorces de communes (qui ne sont en principe liées à aucun objet) |
| @zapclasse(1,1,0x0F) | supprime tous les objets communes de la couche 1 et leurs graphisme associé, ainsi que les éventuelles amorces de communes et les points qui y sont attachés(y compris s'il s'agit de bornes de limites de propriété) |
| @zapclasse(1,1,0x07),<br>@purgepts(1,0,−1,−1,0,0,1) | supprime tous les objets communes de la couche 1 et leurs graphisme associé, ainsi que les éventuelles amorces de communes, puis les points simples isolés (les bornes attachées à ces amorces ne sont ici pas supprimées) |

# @zonegeo

*Nb paramètres*
0

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|

## Action:

fournit la zone géographique du document (propriété du document) sous forme de code Edigeo

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| <zon géo> | STR | zone géographique en clair |

## Exemples:

| | |
|---|---|
| @zonegeo | renvoie "LAMB3" |

# @zoom

*Nb paramètres*
1

## Entrée:

| *Nom* | *Type* | *Commentaire* |
|---|---|---|
| elt/obj/nom | STANDARD<br>PT<br>LI<br>FC<br>EC<br>OBJ | ***valeurs chaines:***<br><br>"all" : zoom d'ensemble<br>"*save nomzoom*" : sauvegarde le zoom de nom 'nomzoom' (remplace tout précédent zoom de même nom)<br>"*recall nomzoom*" : rappelle le zoom de nom 'nomzoom' avec son contexte (vision...)<br>"*nomzoom*" : rappelle le zoom de nom 'nomzoom'<br>"*set*" : zoom fenetre sur les coordonnees terrain de la couche de travail définies par les variables x1,y1,x2,y2<br><br>***valeurs pointeurs:***<br><br>objet ou élément à cadrer sur la fenêtre |

## Action:

effectue un zoom de manière à ce que l'élément ou l'objet fournit soit cadré et centré sur l'écran. Pour les éléments ou objets ponctuels, seul le centrage est effectué.
Pour un rappel de zoom, si celui−ci n'existe pas, une erreur 'paramètre invalide' est déclenchée.
Pour effectuer un zoom d'ensemble, voir la fonction @recentrage

## Sortie:

| *Valeur* | *Type* | *Commentaire* |
|---|---|---|
| [y2] variable inconnue ou de type inaproprié | ERR | en cas de paramètre "set" et si une des variables devant être présente n'existe pas ou est d'un autre type que standard |
| Paramètre invalide | ERR | si les paramètres ont des types ou valeurs non appropriés |
| OK | STR | zoom effectué |

## Exemples:

| | |
|---|---|
| @zoom(save mon zoom), | sauvegarde un zoom de nom "mon zoom" en vision classe monochrome |
| @setvision(0), | se met en vision élément |
| @zoom(zoom1), | rappelle le zoom de nom zoom1 (on est en vision élément) |
| @zoom(mon zoom), | rappelle le zoom de nom "mon zoom" (on se retrouve dans l'espace utilisé en vision monochrome mais en vision élement) |
| @zoom(recall mon zoom), | idem mais en vision monochrome |
| @zoom(recall mon zoom) | idem mais en vision monochrome |
| @zoom(@maparcelle) | recadre l'objet parcelle désigné par "maparcelle" de manière à ce qu'il soit le plus grand possible sur l'écran et centré sur ce dernier |
| @zoom(@mon_pt_de_canevas) | centre le point de canevas sur l'écran (un objet simplement ponctuel, sans écriture) ne change pas "l'échelle de l'écran" |
| @zoom(all) | zoom d'ensemble |