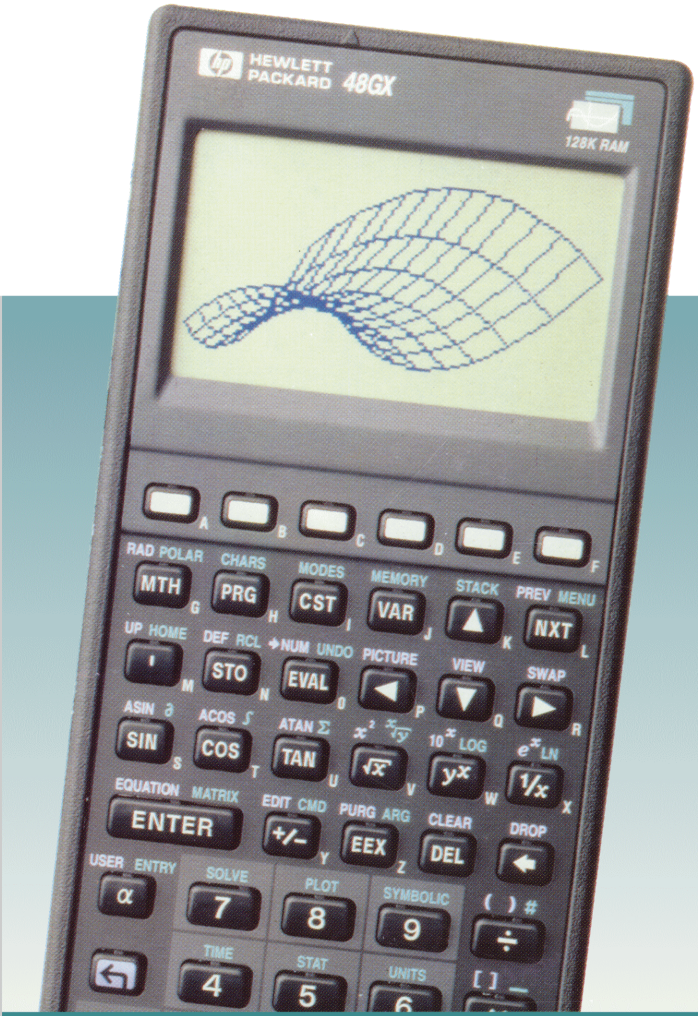
LA BIBLE de la HP 48 G/GX

Manuel de référence avancé
Version française



LA BIBLE de la **HP 48 G/GX**

Manuel de référence avancé
Version française

ÉDITIONS POLE

Avertissement

En raison de la complexité des techniques informatiques, ce document est présenté au lecteur dans le seul but de faciliter sa compréhension du produit dont il traite. **Hewlett Packard France décline en conséquence toute responsabilité pour tout dommage pouvant résulter des informations et des programmes contenus dans ce document**. HPF ne garantit ni la fiabilité, ni les conséquences d'utilisation de ses produits logiciels lorsqu'ils sont utilisés sur du matériel dont elle n'a pas assuré la fourniture.

Les informations contenues dans ce manuel sont originales. Elles ont été conçues et mises au point par Hewlett Packard. L'acheteur s'interdit en conséquence, sauf accord écrit de HPF :

- de les divulguer ou d'en faciliter la divulgation ;
- de les copier ou de les reproduire en tout ou partie ;
- de les traduire dans toute autre langue

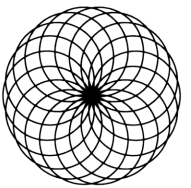
Remerciements

Hewlett-Packard tient à remercier les membres de l'Education Advisory Committee (Dr Thomas Dick, Dr Lynn Garner, Dr John Kenelly, Dr Don LaTorre, Dr Jerold Mathews et Dr Gil Proctor) pour l'aide apportée lors de la mise au point de ce produit. Sont également associés à ces remerciements Messieurs Donald R. Asmus, Scott Burke et Bhushan Gupta, ainsi que les étudiants de l'Institute of Technology de l'Oregon, de même que Carla Randall et ses étudiants en mathématiques pour leur précieuse collaboration.

POLE tient à remercier Messieurs Gilles Cohen, Jean-Michel Ferrard et Robert Pulluard pour avoir relu et amélioré la version française.

LA BIBLE de la HP 48 G/GX

Manuel de référence avancé
Version française



POLE

(Juillet 1993 : Edition 1, Janvier 1994 : Edition 2, Mai 1994 : Edition 3).

Traduction de l'édition 3 : Hewlett Packard

I.S.B.N. 2-909737-11-X

Achevé d'imprimer en août 1995 à Mouscron (Belgique) par Imprim'Tout SA.

CHEZ LE MEME ÉDITEUR

Le magazine des calculatrices Hewlett Packard
Performance Calcul
(ex-*HAUTE PERFORMANCE*)
Voir conditions d'abonnement en dernière page de ce volume

Les guides *Performance Calcul*
Programmation sur HP 48G/GX (Hors Série 1)
Le graphisme sur HP 48G/GX (Hors Série 2)
L'assembleur de la HP 48G/GX (Hors Série 3)

Les disquettes *Performance Calcul*
PERF 1 (160 programmes pour HP 48S/SX)
PERF 2 (180 programmes pour HP 48S/SX/G/GX)

Le minitel *Performance Calcul*
3615 HPERF

Table des matières

3. Commandes

4. Equations

1

Programmation

Si vous avez déjà utilisé un calculateur ou un ordinateur, le concept de *programme* ne vous est certainement pas inconnu. Plus puissant qu'une commande intégrée, le programme, d'une manière générale, permet au calculateur ou à l'ordinateur d'effectuer automatiquement certaines tâches définies par vos soins. Pour le HP 48, un programme est un *objet* qui remplit cet office.

Introduction à la programmation

Pour le HP 48, un programme est un objet entouré de délimiteurs « » contenant une suite de nombres, de commandes et d'autres objets à exécuter automatiquement pour effectuer une tâche.

Par exemple, un programme prenant un nombre dans la pile, calculant sa factorielle et divisant le résultat par 2 se présenterait comme suit :

```
«! 2 / »
```

ou encore :

```
«
  !
  2
  /
»
```

Contenu d'un programme

Un programme contient une suite d'objets qu'il traite en fonction de leur type comme le résume le tableau ci-dessous.

Comportement des objets dans les programmes

Objet	Action
Commande	*Exécutée.*
Nombre	Placé dans la pile.
Expression algébrique	Placée dans la pile.
Chaîne	Placée dans la pile.
Liste	Placée dans la pile.
Programme	Placé dans la pile.
Nom global (avec délimiteurs)	Placé dans la pile.
Nom global (sans délimiteurs)	■ Programme *exécuté.* ■ Nom évalué. ■ Répertoire : devient le répertoire en cours. ■ Autre objet placé dans la pile.
Nom local (avec délimiteurs)	Placé dans la pile.
Nom local (sans délimiteurs)	Contenu placé dans la pile.

Comme vous pouvez le constater, la plupart des types d'objets sont simplement placés dans la pile, mais les commandes intégrées et les programmes nommés sont *exécutés*. Les exemples suivants montrent le résultat de l'exécution de programmes contenant différentes suites d'objets.

Exemples d'exécution de programmes

Programme	Résultat
« 1 2 »	2: 1 1: 2
« "Bonjour" { A B } »	2: "Bonjour" 1: { A B }
« '1+2' »	1: '1+2'
« '1+2' →NUM »	1: 3
« « 1 2 + » »	1: « 1 2 + »
« « 1 2 + » EVAL »	1: 3

Les programmes peuvent aussi contenir des *structures*. Une structure est un segment de programme ayant une organisation bien définie. Il existe deux sortes de structures :

- **Structures de variables locales.** La commande → définit des noms de variables locales et une expression algébrique ou un objet-programme correspondant qui est évalué à l'aide de ces variables.
- **Structures avec branchements.** Des instructions spéciales de structure (comme DO ... UNTIL ... END) définissent des structures conditionnelles ou en boucle, déterminant l'ordre d'exécution des instructions au sein d'un programme.

Une *structure de variable locale* est organisée de l'une des deux manières suivantes au sein d'un programme :

« → nom_1 ... nom_n '*expression algébrique*' »
« → nom_1 ... nom_n « *programme* » »

La commande → retire *n* objets de la pile et les stocke dans les variables locales nommées. L'expression algébrique ou le programme est *automatiquement évalué* parce qu'il est un élément de la structure, même si, par ailleurs, ces objets sont placés dans la pile. Chaque fois qu'un nom de variable locale apparaît dans l'expression algébrique ou dans l'objet-programme, il est remplacé par le contenu de la variable.

Le programme suivant prend deux nombres dans la pile et renvoie un résultat numérique :

```
« → a b 'ABS(a-b)' »
```

Calculs dans un programme

Bon nombre de calculs dans les programmes prennent leurs données dans la pile. Voici deux façons caractéristiques de manipuler ces données :

- **Commandes de la pile.** Elles opèrent directement sur les objets de la pile.
- **Structures de variables locales.** Elles stockent les objets de la pile dans des variables locales temporaires, puis utilisent les noms des variables pour représenter les données dans l'expression algébrique ou l'objet-programme suivant.

Les calculs numériques offrent des exemples typiques de ces méthodes. Les trois programmes suivants utilisent deux nombres de la pile pour calculer l'hypoténuse d'un triangle rectangle avec la formule $\sqrt{x^2+y^2}$.

```
« SQ SWAP SQ + √ »
« → x y « x SQ y SQ + √ » »
« → x y '√(x^2+y^2)' »
```

Le premier programme utilise des commandes de la pile pour manipuler les nombres qui y figurent ; le calcul utilise la syntaxe de la pile. Le deuxième programme se sert d'une structure de variable locale pour stocker et récupérer les nombres ; le calcul utilise la syntaxe de la pile. Le troisième emploie aussi une structure de variable locale, mais l'opération est exprimée en syntaxe algébrique. Notez que la formule sous-jacente est plus évidente dans ce dernier programme. La troisième méthode offre le maximum de facilité de rédaction, de lecture et de mise au point.

Saisie et exécution de programmes

Un programme est un objet, il occupe un niveau dans la pile et vous pouvez le stocker dans une variable.

Pour saisir un programme :

1. Appuyez sur ◀ « ». Le témoin PRG s'affiche, indiquant que le mode de saisie de programme est actif.
2. Saisissez les commandes et les autres objets (avec les délimiteurs appropriés) dans l'ordre correct d'exécution des opérations.
   - Appuyez sur SPC pour séparer les nombres consécutifs.
   - Appuyez sur ▶ pour aller au-delà des délimiteurs.
3. *Facultatif :* appuyez sur ▶ ↵ (à la ligne) pour commencer une nouvelle ligne dans la ligne de commande.
4. Appuyez sur ENTER pour placer le programme dans la pile.

En mode saisie de programme (témoin PRG affiché), les touches de commande ne sont pas exécutées, elles sont simplement saisies dans la ligne de commande. Seules les opérations non programmables (comme ◆ et VAR) sont exécutées.

Les ruptures de lignes sont ignorées lorsque vous appuyez sur ENTER.

Pour introduire des commandes et d'autres objets dans un programme :

- Appuyez sur la touche de clavier ou de menu de la commande ou de l'objet.
  ou
- Tapez les caractères avec le clavier alpha.

Pour stocker ou nommer un programme :

1. Saisissez le programme dans la pile.
2. Saisissez le nom de la variable (avec délimiteurs ') et appuyez sur STO.

Vous pouvez doter les programmes de noms les décrivant, en vous basant, par exemple, sur :

- Le calcul ou l'opération en question. Exemples : *SPH* (volume d'une calotte sphérique), *SORT* (tri d'une liste).
- L'entrée et la sortie. Exemples : $X \rightarrow FX$ (x en $f(x)$), $RH \rightarrow V$ (rayon et hauteur en volume).
- La technique. Exemple : *SPHLV* (volume d'une calotte sphérique avec des variables locales).

Pour exécuter un programme :

- Appuyez sur [VAR], puis sur la touche de menu correspondant au nom du programme.
  ou
- Saisissez le nom du programme (*sans* délimiteurs) et appuyez sur [ENTER].
  ou
- Placez le nom du programme au niveau 1 et appuyez sur [EVAL].
  ou
- Placez l'objet-programme au niveau 1 et appuyez sur [EVAL].

Pour arrêter l'exécution d'un programme :

- Appuyez sur [CANCEL].

Exemple : Saisissez un programme qui prend dans la pile la valeur d'un rayon et calcule le volume d'une sphère de rayon r en utilisant :

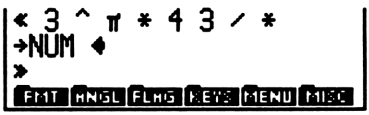
$$V = \frac{4}{3}\pi r^3$$

Pour calculer ce volume manuellement, après avoir saisi le rayon dans la pile, il faut appuyer sur :

3 [y^x] [←] [π] [×] 4 [ENTER] 3 [÷] [×] [←] [→NUM]

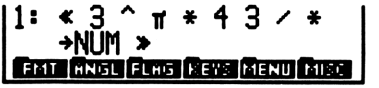
Saisissez cette séquence de touches pour en faire un programme. ([→] [↵] commence une nouvelle ligne.)

[←] [« »]
3 [y^x] [←] [π] [×] 4 [SPC] 3 [÷] [×]
[→] [↵] [←] [→NUM]



Placez le programme dans la pile.

ENTER



Stockez le programme dans la variable *VOL*. Saisissez ensuite 4 pour le rayon et exécutez le programme *VOL*.

' VOL STO
4 VAR VOL

Le programme est le suivant :

```
« 3 ^ π * 4 3 / * →NUM »
```

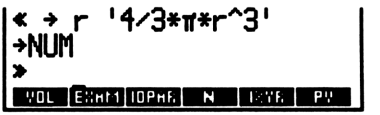
Exemple : Remplacez le programme de l'exemple précédent par un autre plus facile à lire. Saisissez un programme qui utilise une structure de variable locale pour calculer le volume d'une sphère. Ce programme est :

```
« → r '4/3*π*r^3' →NUM »
```

(Il est nécessaire d'inclure →NUM car π génère un résultat symbolique.)

Saisissez le programme. (▶ ↵ commence une nouvelle ligne.)

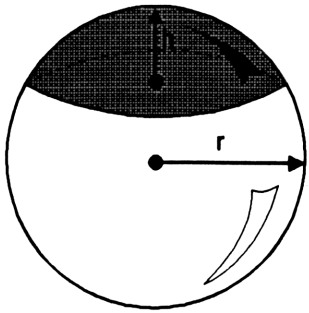
◀ «»
▶ ▶ r SPC
' 4 ÷ 3 × ◀ π ×
r y^x 3 ▶ ▶ ↵
◀ →NUM



Placez le programme dans la pile, stockez-le dans *VOL* et calculez le volume pour un rayon de 4.

4 VOL

Exemple : Saisissez un programme *SPH* calculant le volume d'une calotte sphérique de hauteur h dans une sphère de rayon R en utilisant les valeurs stockées dans les variables H et R.



$$V = \frac{1}{3}\pi h^2(3r - h)$$

Dans les chapitres relatifs à la programmation, les "diagrammes de la pile" montrent les arguments devant se trouver dans la pile préalablement à l'exécution d'un programme et les résultats placés dans la pile par le programme. Voici le diagramme de la pile pour le programme *SPH*.

Niveau 1	→	**Niveau 1**
	→	*volume*

Le diagramme indique que le programme *SPH* ne prend aucun argument dans la pile et renvoie le volume de la calotte sphérique au niveau 1. (vous êtes censé avoir stocké la valeur numérique du rayon dans la variable *R* et celle de la hauteur dans la variable *H*. Il s'agit de variables *globales*, qui existent en dehors de ce programme.)

Les listages de programme comportent les étapes du programme dans la colonne de gauche et les commentaires correspondants dans la colonne de droite. Souvenez-vous que pour saisir le programme, vous pouvez soit appuyer sur les touches de commande, soit tapez les noms de commande. Dans ce premier listage, les séquences de touches sont également indiquées.

Programme :	Touches :	Commentaires :
«	◀ « »	Commence le programme.
'1/3	' 1 ÷ 3	Commence l'expression algébrique pour calculer le volume.
*π*H^2	× ◀ π × H y^x 2	Multiplie par πh^2.
*(3*R-H)'	× ◀ ( ) 3 × R − H ▶ ▶	Multiplie par $3r - h$, achevant le calcul et terminant l'expression.
→NUM	◀ →NUM	Convertit l'expression comportant π en un nombre.
»		Termine le programme.
	ENTER	Place le programme dans la pile.
	' SPH STO	Stocke le programme dans la variable *SPH*.

Le programme est le suivant :

```
« '1/3*π*H^2*(3*R-H)' →NUM »
```

A présent, utilisez *SPH* pour calculer le volume d'une calotte sphérique de rayon $r = 10$ et de hauteur $h = 3$.

D'abord, stockez les données dans les variables appropriées. Puis, sélectionnez le menu VAR et exécutez le programme. Le résultat est renvoyé au niveau 1 de la pile.

10 ' R STO
3 ' H STO
VAR SPH

Affichage et modification des programmes

Pour afficher et modifier des programmes, utilisez la ligne de commande, comme pour les autres objets.

Pour afficher ou modifier un programme :

1. Affichez le programme :
   - Si le programme est au niveau 1, appuyez sur ◀ EDIT, ou ▼.
   - Si le programme est stocké dans une variable, affichez cette dernière au moyen du gestionnaire de mémoire en appuyant sur (▶ MEMORY), ou sur VAR ▶ et sur la touche de menu de la variable, puis sur ▼.
2. *Facultatif* : Faites les modifications souhaitées.
3. Appuyez sur ENTER pour les sauvegarder (ou sur CANCEL pour les annuler) et revenez dans la pile.

Le gestionnaire de mémoire permet de modifier un programme stocké sans avoir à effectuer une opération de stockage. ◀ EDIT permet de modifier le programme et de stocker ensuite la nouvelle version dans une variable différente.

Lors de la modification d'un programme, il se peut qu'il soit nécessaire de changer de mode de saisie sur la ligne de commande, pour passer du mode de saisie de programme (pour la modification de la plupart des objets) au mode de saisie algébrique/de programme (pour modifier des objets algébriques). Les témoins PRG et ALG indiquent le mode en cours.

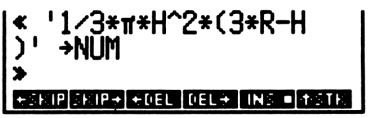
Pour changer de mode de saisie :

- Appuyez sur ▶ ENTRY.

Example: Modifiez le programme *SPH* de l'exemple précédent de sorte qu'il stocke le nombre du niveau 1 dans la variable *H* et le nombre du niveau 2 dans la variable *R*.

Utilisez EDIT pour commencer la modification de *SPH*.

VAR
▶ SPH ▼



Placez le curseur au-delà du premier délimiteur du programme et insérez les nouvelles étapes.

▶ ' H ▶ STO
' R ▶ STO

Sauvegardez la version modifiée de *SPH* dans la variable. Puis, pour vérifier que les modifications ont été enregistrées, affichez *SPH* dans la ligne de commande.

ENTER VAR ↶ SPH
↷ SPH ▼

Appuyez sur CANCEL pour annuler l'affichage.

Création de programmes sur un ordinateur

Il est pratique de créer des programmes et autres objets sur un ordinateur, puis de les transférer au HP 48 via le port série de ce dernier.

Lorsque vous créez les programmes sur ordinateur, vous pouvez y inclure des "commentaires".

Pour inclure des commentaires dans un programme :

- Placez un caractère @ au début et à la fin du texte de commentaires.
  ou
- Placez un caractère @ au début du texte, la fin de ce dernier correspondant à la fin de la ligne.

Lorsque le HP 48 traite du texte dans la ligne de commande, qu'il ait été saisi à partir du clavier ou transféré d'un ordinateur, il annule les caractères @ ainsi que le texte qu'ils contiennent. Cependant, ce n'est pas le cas si ces caractères se trouvent à l'intérieur d'une chaîne.

Utilisation des variables locales

Le programme *SPH* de l'exemple précédent utilise des variables globales pour le stockage et le rappel des données. Or, il existe certains inconvénients à utiliser des variables globales dans les programmes :

- Après l'exécution du programme, les variables globales dont vous n'avez plus besoin doivent être supprimées si vous désirez effacer le menu VAR et libérer de la mémoire.
- Vous devez explicitement stocker les données dans des variables globales avant l'exécution du programme, ou faire exécuter STO par le programme.

Les *variables locales* remédient à ces inconvénients. Ce sont des variables temporaires *créées par un programme*. Elles n'existent qu'au moment de l'exécution du programme et ne sont pas utilisables en dehors de celui-ci. Elles n'apparaissent jamais dans le menu VAR. De plus, elles sont plus rapidement accessibles que les variables globales. (Par convention dans ce manuel, nous utilisons des minuscules pour les noms des variables locales.) Une variable compilée est une forme de variable locale utilisable en dehors du programme pour lequel elle a été créée. Pour de plus amples informations, consultez "Variables locales compilées", page 1-16.

Création de variables locales

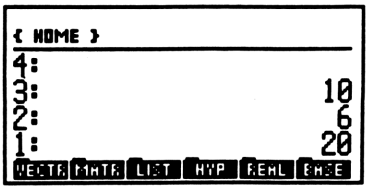
Dans un programme, les variables locales sont créées dans une *structure*.

Pour saisir une structure de variable locale dans un programme :

1. Saisissez la commande → (appuyez sur ▶ →).
2. Saisissez un ou plusieurs noms de variables.
3. Entrez une *procédure de définition* (une expression algébrique ou un objet-programme) qui utilise ces noms.

« → nom_1 nom_2 ... nom_n ' *expression algébrique* ' »
ou
« → nom_1 nom_2 ... nom_n « *programme* » »

Lorsque la commande → est exécutée dans un programme, *n* valeurs sont prises dans la pile et affectées aux variables nom_1, nom_2, ... nom_n. Par exemple, si la pile contient :



dans ce cas :

» → a crée la variable locale $a = 20$.

→ a b crée les variables locales $a = 6$ et $b = 20$.

→ a b c crée les variables locales $a = 10$, $b = 6$ et $c = 20$.

La procédure de définition utilise les variables locales pour faire des calculs.

Les structures de variables locales présentent les avantages suivants :

- La commande → stocke les valeurs de la pile dans les variables correspondantes. Vous n'avez pas besoin d'exécuter explicitement STO.
- Les variables locales disparaissent automatiquement lorsque la procédure de définition, pour laquelle elles ont été créées, a terminé son exécution. En conséquence, elles n'apparaissent pas dans le menu VAR et n'occupent la mémoire que le temps de l'exécution du programme.
- Les variables locales existent uniquement à l'intérieur de leur procédure de définition : des structures de variables locales différentes peuvent donc utiliser les mêmes noms sans qu'il y ait conflit.

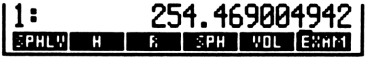
Exemple : Le programme *SPHLV* suivant calcule le volume d'une calotte sphérique avec des variables locales. La procédure de définition est une expression algébrique.

Niveau 2	Niveau 1	→	Niveau 1
r	*h*	→	*volume*

Programme :	**Commentaires :**
«	
→ r h	Crée des variables locales r et h pour le rayon de la sphère et la hauteur du segment.
'1/3*π*h^2*(3*r-h)'	Exprime la procédure de définition. Dans ce programme, la procédure de définition de la structure de la variable locale est une expression algébrique.
→NUM	Convertit l'expression en nombre.
»	
ENTER ' SPHLV STO	Stocke le programme dans la variable *SPHLV*.

A présent, utilisez le programme *SPHLV* pour calculer le volume d'une calotte sphérique de rayon $r = 10$ et de hauteur $h = 3$. Saisissez les données dans la pile dans l'ordre approprié, puis exécutez le programme.

10 ENTER 3
VAR SPHLV



Evaluation des noms locaux

Les noms locaux sont évalués différemment des noms globaux. Lorsqu'un nom global est évalué, l'objet stocké dans la variable correspondante est lui-même évalué. (Vous avez vu comment les programmes stockés dans les variables globales sont automatiquement évalués lorsque leur nom est évalué.)

Lorsqu'un nom local est évalué, l'objet stocké dans la variable correspondante est renvoyé dans la pile mais il n'est *pas* évalué. Lorsqu'une variable locale contient un nombre, l'effet est identique à celui de l'évaluation d'un nom global, puisque le fait de placer un nombre dans la pile équivaut à l'évaluer. En revanche, si une variable locale contient un programme, une expression algébrique ou un nom de variable globale, cet objet *doit être explicitement évalué* (par exécution de EVAL) après son renvoi dans la pile.

Définition de la portée des variables locales

Les variables locales existent ***uniquement*** à l'intérieur de leur procédure de définition.

Exemple : L'extrait de programme suivant montre la disponibilité des variables locales dans les procédures de définition *imbriquées* (procédures au sein de procédures). Les variables locales *a*, *b* et *c* existant déjà lors de l'exécution de la procédure de définition des variables locales *d*, *e* et *f*, elles sont disponibles dans cette procédure.

Programme :	**Commentaires :**
`«`	
⋮	Aucune variable locale disponible.
`→ a b c`	Définit les variables locales *a*, *b*, *c*.
`«`	Les variables locales *a*, *b*, *c* sont
`a b + c +`	disponibles dans cette procédure.
`→ d e f`	Définit les variables locales *d*, *e*, *f*.
`'a/(d*e+f)'`	Les variables locales *a*, *b*, *c* et *d*, *e*, *f* sont disponibles dans cette procédure.
`a c / -`	Seules les variables locales *a*, *b*, *c*
`»`	sont disponibles.
⋮	Aucune variable locale disponible.
`»`	

Exemple : Dans l'extrait de programme suivant, la procédure de définition des variables locales *d*, *e* et *f* appelle un programme préalablement créé et stocké par vos soins dans la variable globale *P1*.

Programme :	Commentaires :
`«`	
`⋮`	
`→ a b c`	
`«`	
`a b + c +`	
`→ d e f`	Définit les variables locales *d*, *e*, *f*.
`'P1+a/(d*e+f)'`	Les variables locales *a*, *b*, *c* et *d*, *e*, *f* sont disponibles dans cette procédure. La procédure de définition exécute le programme stocké dans la variable *P1*.
`a c / -`	
`»`	
`⋮`	
`»`	

Les six variables locales *ne sont pas* disponibles dans le programme *P1* car elle n'existaient pas lors de la création de *P1*. Les objets stockés dans les variables locales ne sont disponibles pour *P1* que si vous les placer dans la pile afin que *P1* les utilise et les stocke dans des variables globales.

Réciproquement, le programme *P1* peut créer sa propre structure de variables locales (avec des noms quelconques, tels que *a*, *c* et *f*, par exemple) sans qu'il y ait conflit avec les variables locales du même nom dans la procédure qui appelle *P1*. Il est possible de créer un type spécial de variable locale utilisable dans d'autres programmes ou sous-programmes. Ce type de variable locale est dite *compilée*.

Variables locales compilées

Les variables globales encombrent la mémoire et les variables locales sont inutilisables en dehors du programme qui les a créées. Les variables locales compilées font la liaison entre ces deux types de variables. Pour les programmes, les variables locales compilées sont prises en compte comme des variables globales et pour le calculateur, elles opèrent comme des variables locales. Vous pouvez donc créer une variable locale compilée dans une structure de variables locales,

l'utiliser dans tout autre programme appelé au sein de cette structure, et, lorsque le programme se termine, cette variable est supprimée.

Les variables locales compilées sont dotées d'une convention d'écriture spécifique : elles doivent débuter par une ←. Exemple :

```
«
 → ←y
'IFTE(←y<0,BELOW,ABOVE)'
»
```

La variable ←y est une variable locale compilée utilisable par les deux programmes BELOW et ABOVE.

Création de fonctions-utilisateur comme programmes

La procédure de définition d'une structure de variables locales peut être soit une expression algébrique, soit un objet-programme.

Un programme comprenant uniquement une structure de variables locales dont la procédure de définition est une expression algébrique est une fonction-utilisateur.

Si un programme commence par une structure de variables locales et s'il a un programme comme procédure de définition, la totalité du programme se comporte comme une fonction-utilisateur, et ce de deux façons : il prend des arguments numériques ou symboliques ; il prend ces arguments soit dans la pile, soit en syntaxe algébrique. Cependant, il *n'a pas* de dérivée. (Le programme de définition doit, comme les procédures de définition des expressions algébriques, renvoyer uniquement *un* résultat dans la pile.)

L'utilisation d'un programme comme procédure de définition d'une structure de variables locales présente un avantage : il peut contenir des commandes non admises dans les expressions algébriques, des structures en boucle par exemple.

Utilisation des tests et des structures conditionnelles

Vous pouvez utiliser des commandes et des structures de branchement permettant aux programmes de poser des questions et de prendre des décisions. Des *fonctions de comparaison* et des *fonctions logiques* testent si les conditions spécifiées sont réunies. Les *structures et les commandes conditionnelles* utilisent les résultats des tests pour prendre les décisions en conséquence.

Conditions de test

Un test est une expression algébrique ou une suite de commande renvoyant un *résultat* dans la pile. Ce résultat est soit *vrai* (indiqué par la valeur 1), soit *faux* (indiqué par la valeur 0).

Pour inclure un test dans un programme :

- Pour utiliser la syntaxe de la pile, saisissez les deux arguments, puis la commande de test.
- Pour utiliser la syntaxe algébrique, saisissez l'expression de test (avec délimiteurs ').

Les résultats de tests sont souvent utilisés dans les structures conditionnelles pour déterminer la clause de la structure à exécuter. (Voir "Utilisation des structures et commandes conditionnelles", page 1-21.)

Exemple : Testez si X est inférieur à Y. Pour utiliser la syntaxe de la pile, saisissez `X Y <`. Pour utiliser la syntaxe algébrique, saisissez `'X<Y'`. (Dans les deux cas, si X contient 5 et Y contient 10, le test est vrai et la valeur 1 est renvoyée dans la pile.)

Utilisation des fonctions de comparaison

Ces fonctions comparent deux objets, en utilisant la syntaxe de la pile ou la syntaxe algébrique.

Fonctions de comparaison

Touche	Commande programmable	Description
PRG TEST (pages 1 et 2):		
==	==	Teste l'égalité de deux objets.
≠	≠	Différent.
<	<	Inférieur.
>	>	Supérieur.
≤	≤	Inférieur ou égal.
≥	≥	Supérieur ou égal.
SAME	SAME	Identique. Comme ==, mais ne permet pas la comparaison entre la valeur numérique d'une expression algébrique (ou d'un nom) et un nombre. Prend en compte également la taille de mot d'un entier binaire.

Les commandes de comparaison renvoient 1 (vrai) ou 0 (faux) en fonction du résultat de la comparaison, ou une expression pouvant être évaluée à la valeur 1 ou 0. L'ordre de comparaison est "niveau 2 *test* niveau 1," où *test* correspond à la fonction de comparaison.

Toutes les commandes de comparaison, à l'exception de SAME, renvoient ce qui suit :

- Si aucun des objets n'est une expression algébrique ou un nom, elles renvoient 1 dans le cas où les deux objets sont de même type et ont la même valeur. Sinon, elles renvoient 0. Par exemple, si 6 est stocké dans *X*, `X 5 <` place 6 et 5 dans la pile, puis les supprime et renvoie 0. (Les listes et programmes sont considérés comme ayant la même valeur si les objets qu'ils contiennent sont identiques. Dans le cas de chaînes, "inférieur" correspond à "antérieur dans l'ordre alphabétique.")
- Si un objet est une expression algébrique (ou un nom) et l'autre une expression algébrique (ou nom) ou un nombre, elles renvoient une expression qui doit être évaluée pour obtenir un résultat de test basé sur des valeurs numériques. Par exemple, si 6 est stocké dans *X*, `'X' 5 <` renvoie `'X<5'`, puis →NUM renvoie 0.

(Notez que == est utilisé pour des comparaisons, tandis que = sépare les deux membres d'une équation.)

SAME renvoie 1 (vrai) si deux objets sont identiques. Par exemple, 'X+3' 4 SAME renvoie 0 quelle que soit la valeur de *X* car l'expression algébrique 'X+3' n'est pas identique au nombre réel 4. Pour être identiques, les entiers binaires doivent avoir la même taille de mot et la même valeur. Pour tous les types d'objets autres que les expressions algébriques, les noms et les entiers binaires, SAME fonctionne comme ==.

Vous pouvez utiliser toutes les fonctions de comparaison (sauf SAME) dans une expression algébrique en les plaçant *entre* les deux arguments de cette dernière. Par exemple, si 6 est stocké dans *X*, 'X<5' →NUM renvoie 0.

Utilisation des fonctions logiques

Les fonctions logiques renvoient un résultat de test basé sur les résultats de deux tests exécutés antérieurement. Notez que ces quatre fonctions interprètent *tout argument non nul* comme un résultat vrai.

Fonctions logiques

Touche	Commande programmable	Description
(PRG) TEST (page 2):		
AND	ET	Renvoie 1 (vrai) uniquement si les deux arguments sont vrais.
OR	OU	Renvoie 1 (vrai) si au moins un argument est vrai.
XOR	OU exclusif	Renvoie 1 (vrai) uniquement si l'un des deux arguments est vrai.
NOT	NON	Renvoie 1 (vrai) si l'argument est 0 (faux) ; sinon, renvoie 0 (faux).

AND, OR et XOR combinent deux résultats de test. Par exemple, si 4 est stocké dans *Y*, Y 8 < 5 AND renvoie 1. D'abord, Y 8 < renvoie 1 dans la pile. Puis, AND supprime 1 et 5 de la pile, les interprétant tous deux comme des résultats vrais, et renvoie 1 dans la pile.

NOT renvoie l'inverse logique du résultat du test. Par exemple, si 1 est stocké dans X et 2 dans Y, X Y < NOT renvoie 0.

Vous pouvez utiliser AND, OR et XOR comme des fonctions *infixes* dans les expressions algébriques. Par exemple, '3<5 XOR 4>7' →NUM renvoie 1.

Vous pouvez utiliser NOT comme fonction *préfixe* dans les expressions algébriques. Par exemple, 'NOT Z≤4' →NUM renvoie 0 si $Z = 2$.

Test des types d'objets

La commande TYPE ([PRG] TEST [NXT] TYPE) prend tout objet comme argument et renvoie le numéro identifiant son type. Par exemple, "BONJOUR" TYPE renvoie la valeur 2, qui correspond au type d'objet "chaîne". Pour connaître les objets du HP 48 et leur numéro de type, consultez le tableau des types d'objets au chapitre 3, dans la rubrique consacrée à la commande TYPE.

Test de structures linéaires

La commande LININ ([PRG] [NXT] TEST [→] [PREV] LININ) prend une équation algébrique au niveau 2 et une variable au niveau 1 comme arguments et renvoie 1 si l'équation est linéaire pour cette variable, ou 0 dans le cas contraire. Par exemple, 'H+Y^2' 'H' LININ renvoie 1 car l'équation présente une structure linéaire pour H. Pour de plus amples informations, reportez-vous à la commande LININ au chapitre 3.

Utilisation de structures et de commandes conditionnelles

Les *structures conditionnelles* permettent à un programme de prendre une décision en fonction du résultat des tests.

Les *commandes conditionnelles* permettent d'exécuter une clause vraie ou fausse (chacune d'elle constituant une commande ou un objet *unique*).

Ces structures et commandes conditionnelles se trouvent dans le menu PRG BRCH ([PRG] BRCH) :

- Structure IF ... THEN ... END.
- Structure IF ... THEN ... ELSE ... END.
- Structure CASE ... END.

- Commande IFT (IF-THEN).
- Fonction IFTE (IF-THEN-ELSE).

Structure IF ... THEN ... END

La syntaxe est la suivante :

« ... IF *clause-de-test* THEN *clause-vraie* END ... »

IF ... THEN ... END exécute la suite de commandes placée dans la *clause-vraie* uniquement si un test (*clause-de-test*) a la valeur "vrai". La clause de test peut être une suite de commandes (par exemple, A B ≤) ou une expression algébrique (par exemple, 'A≤B'). Si la clause de test est une expression algébrique, elle est *automatiquement évaluée* à un nombre (→NUM ou EVAL ne sont pas nécessaires.

IF débute la clause de test, qui laisse un résultat de test dans la pile. THEN retire ce résultat de la pile. Si la valeur est différente de zéro, la clause vraie est exécutée. Sinon, l'exécution du programme reprend après END. Voir "Exemples de structures conditionnelles", page 1-24.

Pour saisir IF ... THEN ... END dans un programme :

- Appuyez sur PRG BRCH ◀ IF .

Commande IFT

La commande IFT prend deux arguments : un *résultat-de-test* au niveau 2 et un objet *clause-vraie* au niveau 1. Si le résultat de test est vrai, l'objet de la clause vraie est exécutée. Sinon, les deux arguments sont retirés de la pile. Voir "Exemples de structures conditionnelles", page 1-24.

Pour saisir IFT dans un programme :

- Appuyez sur PRG BRCH NEXT ◀ IFT .

Structure IF ... THEN ... ELSE ... END

La syntaxe est la suivante :

« ... IF *clause-de-test*
THEN *clause-vraie* ELSE *clause-fausse* END ... »

IF ... THEN ... ELSE ... END exécute soit la séquence de commandes de *clause-vraie* si la *clause-de-test* est vraie, soit la

séquence de commandes de *clause-fausse* si la *clause-de-test* est fausse. Si la clause de test est une expression algébrique, elle est automatiquement évaluée à un nombre (→NUM ou EVAL ne sont pas nécessaires).

IF débute la clause de test qui laisse un résultat dans la pile. THEN retire ce résultat. Si la valeur est différente de zéro, la clause vraie est exécutée. Sinon, la clause fausse est exécutée. Après exécution de la clause appropriée, l'exécution reprend après END. Voir "Exemples de structures conditionnelles", page 1-24.

Pour saisir IF ... THEN ... ELSE ... END dans un programme :

- Appuyez sur (PRG) BRCH (▶) IF .

Fonction IFTE

La syntaxe algébrique de cette fonction est la suivante :

'IFTE(*test*, *clause-vraie*, *clause-fausse*)'

Si *test* est vrai, l'expression algébrique *clause-vraie* est évaluée. Sinon, la *clause-fausse* est évaluée.

Vous ne pouvez pas utiliser la fonction IFTE avec la syntaxe de la pile. Cette fonction prend trois arguments : un *résultat-de-test* au niveau 3, un objet *clause-vraie* au niveau 2 et un objet *clause-fausse* au niveau 1. Voir "Exemples de structures conditionnelles", page 1-24.

Pour saisir IFTE dans un programme ou dans une expression algébrique :

- Appuyez sur (PRG) BRCH (NEXT) (◀) IFTE .

Structure CASE ... END

La syntaxe est la suivante :

« ... CASE
clause-de-test$_1$ THEN *clause-vraie*$_1$ END
clause-de-test$_2$ THEN *clause-vraie*$_2$ END
⋮
clause-de-test$_n$ THEN *clause-vraie*$_n$ END
clause-par-défaut (facultative)
END ... »

La structure CASE ... END permet d'exécuter une série de commandes de *clause-de-test*, puis d'exécuter la séquence de commandes de *clause_vraie* appropriée. Le premier test qui renvoie un résultat vrai provoque l'exécution de la clause vraie corrrespondante, mettant fin à la structure CASE ... END. Après le dernier test, vous pouvez inclure une *clause-par-défaut* qui sera exécutée si tous les tests sont évalués comme étant faux. Si une clause de test est une expression algébrique, elle est automatiquement évaluée comme nombre (→NUM ou EVAL ne sont pas nécessaires).

Lorsque CASE est exécutée, la clause de $test_1$ est évaluée. Si le test est vrai, la clause $vraie_1$ est exécutée et le programme passe directement à END. Si la clause de $test_1$ est fausse, le programme passe à la clause de $test_2$. L'exécution au sein de la structure CASE continue jusqu'à ce qu'une clause vraie soit exécutée ou que toutes les clauses de tests soient évaluées comme étant fausses. Si une clause par défaut est incluse, elle est exécutée si toutes les clauses de test ont été évaluées comme étant fausses. Voir "Exemples de structures conditionnelles" ci-après.

Pour saisir CASE ... END dans un programme :

1. Appuyez sur PRG BRCH ◀ CASE pour saisir CASE ... THEN ... END ... END.
2. Pour chaque clause de test supplémentaire, placez le curseur après le END d'une clause de test et appuyez sur ▶ CASE pour saisir THEN ... END.

Exemples de structures conditionnelles

Ci-après figurent des exemples de structures conditionnelles dans des programmes.

Exemple : une action conditionnelle Le programme ci-après teste la valeur se trouvant au niveau 1. Si elle est positive, elle est rendue négative. Le premier programme utilise une séquence de commandes comme clause de test :

```
« DUP IF 0 > THEN NEG END »
```

La valeur dans la pile doit être dupliquée car la commande > retire deux arguments de la pile (0 et la copie de la valeur produite par DUP).

La version suivante utilise une expression algébrique comme clause de test :

```
« → x « x IF 'x>0' THEN NEG END » »
```

La version suivante utilise la commande IFT :

```
« DUP 0 > « NEG » IFT »
```

Exemple : une action conditionnelle. Ce programme multiplie deux nombres s'ils sont différents de zéro.

Programme :	**Commentaires :**
«	
→ x y	Crée deux variables locales x et y contenant les deux nombres de la pile.
«	
IF	Débute la clause de test.
'x≠0'	Teste l'un des nombres et laisse un résultat de test dans la pile.
'y≠0'	Teste l'autre nombre, laissant un autre résultat de test dans la pile.
AND	Contrôle si les deux tests sont vrais.
THEN	Met fin à la clause de test et exécute la clause vraie.
x y *	Multiplie les deux nombres uniquement si AND renvoie une valeur vraie.
END	Met fin à la clause vraie.
»	
»	

Le programme suivant accomplit la même tâche que le programme précédent :

```
« → x y « IF 'x AND y' THEN x y * END » »
```

La clause de test 'x AND y' renvoie "vrai" si les deux nombres sont différents de zéro.

La version suivante utilise la commande IFT :

```
« → x y « 'x AND y' 'x*y' IFT » »
```

Exemple : deux actions conditionnelles. Ce programme prend une valeur x dans la pile et calcule $(\sin x)/x$. Pour $x = 0$, la division serait erronée. Dans ce cas, le programme renvoie donc la valeur limite 1.

```
« → x « IF 'x≠0' THEN x SIN x / ELSE 1 END » »
```

La version suivante utilise la syntaxe algébrique de IFTE :

```
« → x 'IFTE(x≠0,SIN(x)/x,1)' »
```

Exemple : deux actions conditionnelles. Ce programme multiplie deux nombres s'ils sont différents de zéro, sinon il renvoie la chaîne "ZERO".

Programme :	**Commentaires :**
«	
→ n1 n2	Crée les variables locales.
«	Débute la procédure de définition.
IF	Débute la clause de test.
'n1≠0 AND n2≠0'	Teste *n1* et *n2*.
THEN	Si les deux nombres sont
n1 n2 *	différents de zéro, multiplie les deux valeurs.
ELSE	Sinon, renvoie la chaîne ZERO.
"ZERO"	
END	Met fin à la structure conditionnelle.
»	Met fin à la procédure de définition.
»	

Exemple : deux actions conditionnelles. Ce programme teste si deux nombres de la pile ont la même valeur. Si c'est le cas, il élimine l'un d'eux et stocke l'autre dans la variable V1. Sinon, il stocke le nombre du niveau 1 dans V1 et le nombre du niveau 2 dans V2.

Programme :	**Commentaires :**
«	
IF	Pour la clause de test, copie les
DUP2	nombres dans les niveaux 1 et 2
SAME	et teste s'ils ont la même valeur.
THEN	Pour la clause de test, élimine
DROP	l'un des nombres et stocke l'autre
'V1' STO	dans *V1*.
ELSE	Pour la clause fausse, stocke le
'V1' STO	nombre du niveau 1 dans *V1* et le
'V2' STO	nombre de niveau 2 dans *V2*.
END	Met fin à la structure conditionnelle.
»	
ENTER	Place le programme dans la pile.
' TST STO	Stocke le programme dans *TST*.

Saisissez les nombres 26 et 52, puis exécutez *TST* pour les comparer. Comme ces nombres ne sont pas égaux, le menu VAR contient à présent deux nouvelles variables, *V1* et *V2*.

26 ENTER 52
VAR TST

Exemple : plusieurs actions conditionnelles. Le programme suivant stocke l'argument de niveau 1 dans une variable s'il s'agit d'une chaîne, d'une liste ou d'un programme.

Programme :	**Commentaires :**
«	
→ y	Définit la variable locale *y*.
«	Débute la procédure de définition.
CASE	Débute la structure "CASE".
y TYPE 2 SAME	1er cas : si l'argument est
THEN y 'STR' STO END	une chaîne, elle est stockée dans *STR*.
y TYPE 5 SAME	2e cas : si l'argument est
THEN y 'LIST' STO END	une liste, elle est stockée dans *LIST*.
y TYPE 8 SAME	3e cas : si l'argument est
THEN y 'PROG' STO END	un programme, il est stocké dans *PROG*.
END	Met fin à la structure "CASE".
»	Met fin à la procédure de définition.
»	

Utilisation des structures en boucle

Vous pouvez utiliser les structures en boucle pour exécuter plusieurs fois une partie de programme. Pour spécifier à l'avance combien de fois la boucle doit se répéter, utilisez une *boucle finie*. Pour lancer un test qui déterminera si la boucle doit ou non être répétée, utilisez une *boucle infinie*.

Les *structures en boucle* permettent à un programme d'exécuter plusieurs fois une séquence de commandes. Les structures en boucle sont créées au moyen de commandes, dites mots de structure, qui ne fonctionnent que si elles sont combinées de manière appropriée. Ces commandes se trouvent dans le menu PRG BRCH ([PRG] BRCH) :

- START ... NEXT et START ... STEP.
- FOR ... NEXT et FOR ... STEP.
- DO ... UNTIL ... END.
- WHILE ... REPEAT ... END.

En outre, la fonction Σ fournit une solution de remplacement aux structures en boucle finie pour les sommations.

Utilisation des structures en boucle finie

Les compteurs des structures en boucle finie s'incrémentent de manière différente :

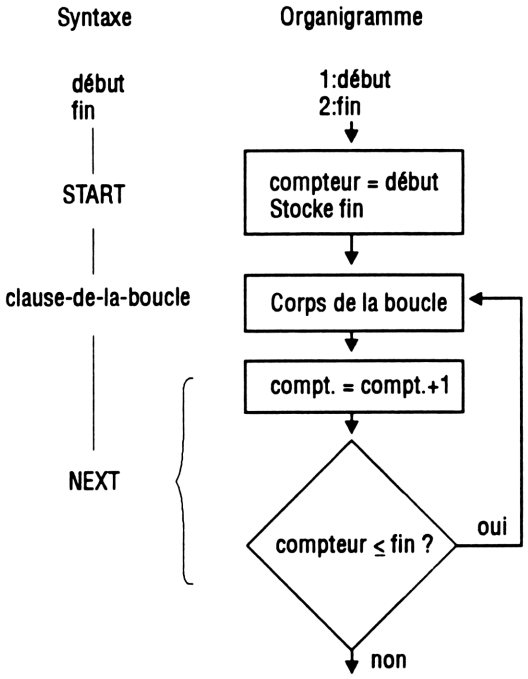
- NEXT incrémente le compteur de 1 à chaque boucle.
- STEP incrémente ou décrémente le compteur d'une valeur spécifiée à chaque boucle.

Structure START ... NEXT

La syntaxe est la suivante :

« ... *début fin* START *clause-de-la-boucle* NEXT ... »

START ... NEXT exécute la séquence de commandes de *clause-de-la-boucle* autant de fois que l'indique la fourchette *début* à *fin*. La clause de la boucle est toujours exécutée au moins une fois.



Structure START ... NEXT

START prend deux nombres (*début* et *fin*) dans la pile et les stocke comme valeurs initiale et finale d'un compteur de boucle. Puis la clause de la boucle est exécutée. NEXT incrémente le compteur de 1 et vérifie si sa valeur est inférieure ou égale à *fin*. Si c'est le cas, la clause de la boucle est réexécutée. Sinon, l'exécution reprend après NEXT.

Pour saisir START ... NEXT dans un programme :

- Appuyez sur PRG BRCH ◀START.

Exemple : Le programme suivant crée une liste contenant 10 fois la chaîne "ABC" :

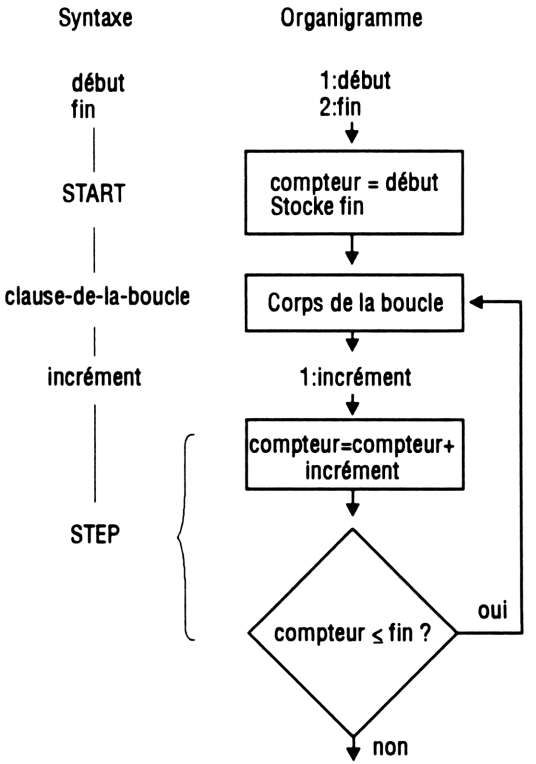
```
« 1 10 START "ABC" NEXT 10 →LIST »
```

Structure START ... STEP

La syntaxe est la suivante :

« ... *début fin* START *clause-de-la-boucle incrément* STEP ... »

START ... STEP exécute la *clause-de-la-boucle* comme le fait START ... NEXT, à ceci près que le programme spécifie la valeur d'incrément du compteur au lieu de l'augmenter d'une unité. La clause de la boucle est toujours exécutée au moins une fois.



Structure START ... STEP

START prend deux nombres (*début* et *fin*) dans la pile et les stocke comme valeurs initiale et finale du compteur de boucle. Ensuite, la clause de la boucle est exécutée. STEP prend la valeur de l'incrément dans la pile et incrémente le compteur en conséquence. Si

l'argument de STEP est une expression algébrique ou un nom, il est automatiquement évalué à un nombre.

La valeur d'incrément peut être positive ou négative. Si elle est positive, la boucle est exécutée à nouveau tant que le compteur est inférieur ou égal à *fin*. Si l'incrément est négatif, la boucle est exécutée tant que le compteur est supérieur ou égal à *fin*. Dans les autres cas, l'exécution reprend après STEP. Dans l'organigramme précédent, l'incrément est positif.

Pour saisir START ... STEP dans un programme :

- Appuyez sur PRG BRCH ▶ START.

Exemple : Le programme suivant prend un nombre x dans la pile et calcule plusieurs fois le carré de ce nombre ($x/3$ fois) :

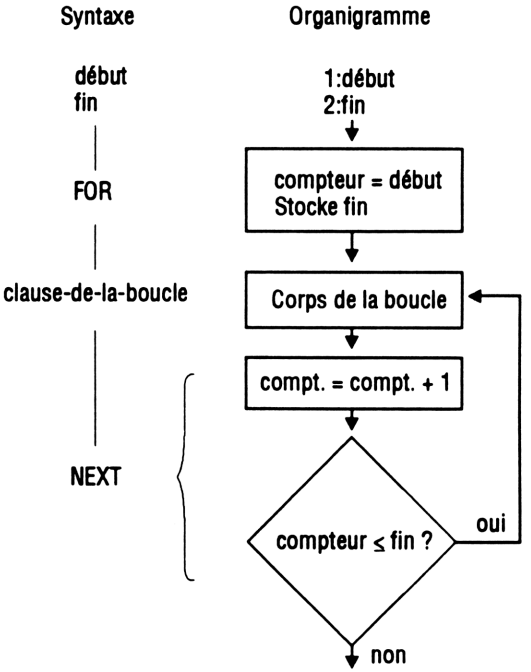
```
« DUP → × « × 1 START × SQ -3 STEP » »
```

Structure FOR ... NEXT

La syntaxe est la suivante :

« ... *début fin* FOR *compteur clause-de-la-boucle* NEXT ... »

FOR ... NEXT exécute le segment de programme *clause-de-la-boucle* autant de fois que l'indique la différence entre *début* et *fin*, en utilisant la variable locale *compteur* comme compteur de boucle. Cette variable est utilisable dans la clause de la boucle. La clause de la boucle est toujours exécutée au moins une fois.



Structure FOR ... NEXT

FOR prend *début* et *fin* dans la pile comme valeurs initiale et finale du compteur de boucle, puis crée la variable locale *compteur* comme compteur de boucle. La clause de la boucle est ensuite exécutée (*compteur* peut figurer dans cette clause). NEXT incrémente de un le *nom_de_compteur*, puis vérifie si sa valeur est inférieure ou égale à *fin*. Si c'est le cas, la clause de la boucle est répétée (avec la nouvelle

valeur de *compteur*). Sinon, l'exécution reprend après NEXT. A la fin de la boucle, la valeur de *compteur* est supprimée.

Pour saisir FOR ... NEXT dans un programme :

- Appuyez sur PRG BRCH ◀ FOR .

Exemple : Le programme suivant place les carrés des entiers 1 à 5 dans la pile :

```
« 1 5 FOR j j SQ NEXT »
```

Exemple : Le programme suivant prend la valeur x dans la pile et les puissances entières i de x. Par exemple, si $x = 12$ et *début* et *fin* respectivement 3 et 5, le programme renvoie 12^3, 12^4 et 12^5. Il nécessite en entrée *début* et *fin* aux niveaux 3 et 2 et x au niveau 1. (→ x retire x de la pile, y laissant *début* et *fin* comme arguments de FOR.)

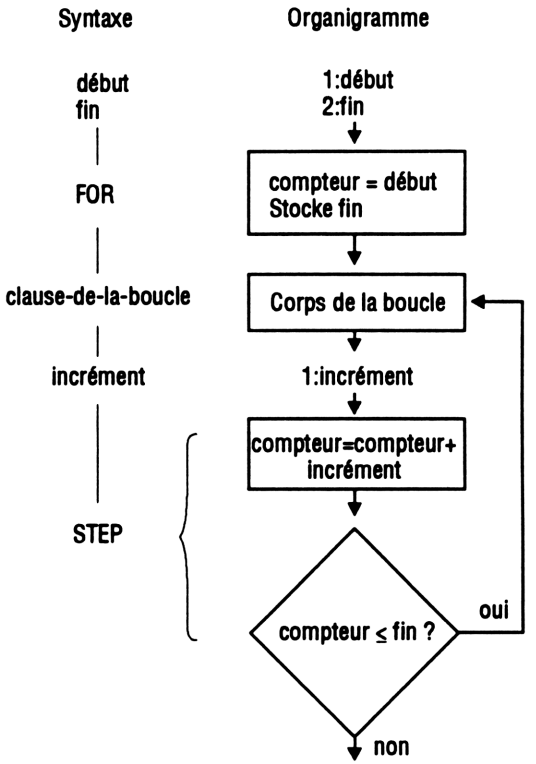
```
« → x « FOR n 'x^n' EVAL NEXT » »
```

Structure FOR ... STEP

La syntaxe est la suivante :

« ... *début fin* FOR *compteur clause-de-la-boucle incrément* STEP ... »

FOR ... STEP exécute la séquence *clause-de-la-boucle* comme le fait FOR ... NEXT, à ceci près que le programme spécifie la valeur d'incrément pour *compteur* au lieu de l'augmenter d'une unité. La clause de la boucle est toujours exécutée au moins une fois.



Structure FOR ... STEP

FOR prend *début* et *fin* dans la pile comme valeurs initiale et finale du compteur de boucle, puis crée la variable locale *compteur* comme compteur de boucle. La clause de la boucle est ensuite exécutée (*compteur* peut figurer à l'intérieur de la clause). STEP prend

la valeur d'incrément dans la pile et augmente le compteur en conséquence. Si l'argument de STEP est une expression algébrique ou un nom, il est automatiquement évalué à un nombre.

La valeur d'incrément peut être positive ou négative. Si elle est positive, la boucle est exécutée à nouveau tant que le compteur est inférieur ou égal à *fin*. Si l'incrément est négatif, la boucle est exécutée tant que le compteur est supérieur ou égal à *fin*. Sinon, la valeur de *compteur* est supprimée et l'exécution reprend après STEP. Dans l'organigramme précrédent, l'incrément est positif.

Pour saisir FOR ... STEP dans un programme :

- Appuyez sur PRG BRCH ▶ FOR .

Exemple : Le programme suivant place le carré des entiers 1, 3, 5, 7 et 9 dans la pile :

```
« 1 9 FOR x x SQ 2 STEP »
```

Exemple : Le programme suivant prend *n* dans la pile et renvoie une suite de nombres 1, 2, 4, 8, 16, ... , *n*. Si *n* ne figure pas dans la suite, le programme s'arrête à la valeur immédiatement inférieure à *n*.

```
« 1 SWAP FOR n n n STEP »
```

Le premier n correspond à la déclaration de la variable locale pour la boucle FOR. Le deuxième n est placé dans la pile à chaque itération de la boucle. Le troisième est utilisé par STEP comme incrément.

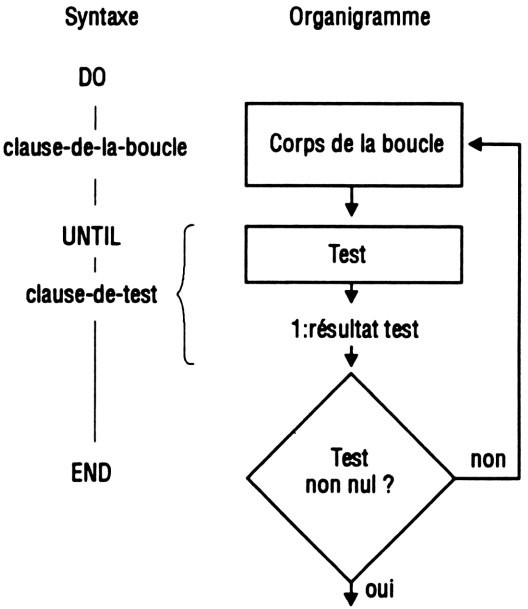
Utilisation des structures en boucle infinie

Structure DO ... UNTIL ... END

La syntaxe est la suivante :

« ... DO *clause-de-la-boucle* UNTIL *clause-de-test* END ... »

DO ... UNTIL ... END exécute la séquence *clause-de-la-boucle* de façon répétée jusqu'à ce que *clause_de_test* renvoie un résultat vrai (différent de zéro). La clause de test étant exécutée *après* la clause de la boucle, cette dernière est toujours exécutée au moins une fois.



Structure DO ... UNTIL ... END

DO commence l'exécution de la clause de la boucle. UNTIL la termine et commence la clause de test. Celle-ci laisse un résultat de test dans la pile. END retire ce résultat. Si sa valeur est zéro, la clause de la boucle est à nouveau exécutée. Sinon, l'exécution reprend après END. Si l'argument de END est une expression algébrique ou un nom, il est automatiquement évalué à un nombre.

Pour saisir DO ... UNTIL ... END dans un programme :

- Appuyez sur PRG BRCH ↩ DO

Exemple : Le programme suivant calcule $n + 2n + 3n + \ldots$ pour une valeur de n. Le programme s'arrête lorsque la somme excède 1000 et renvoie la somme, ainsi que le coefficient de n.

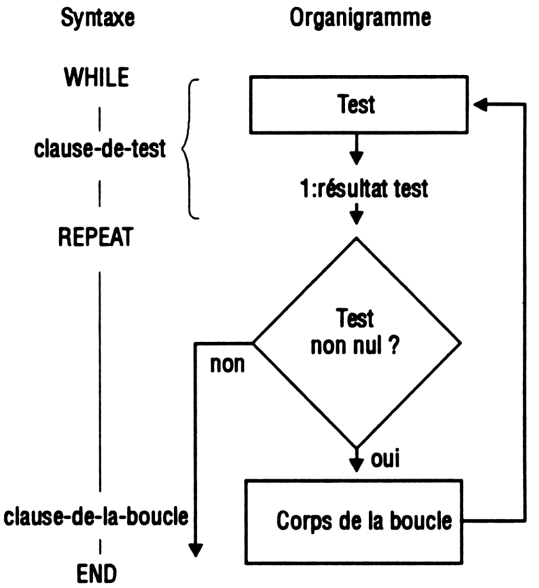
Programme :	**Commentaires :**
«	
DUP 1	Duplique n, stocke la valeur dans
→ n s c	n et s, et initialise c à 1.
«	Débute la procédure de définition.
DO	Débute la clause de la boucle.
'c' INCR	Incrémente le compteur d'une unité. (Voir "Utilisation des compteurs de boucle", page 1-40.)
n * 's' STO+	Calcule $c \times n$ et ajoute le produit à s.
UNTIL	Débute la clause de test.
s 1000 >	Répète la boucle jusqu'à ce que $s > 1000$.
END	Met fin à la clause de test.
s c	Place s et c dans la pile.
»	Met fin à la procédure de définition.
»	

Structure WHILE ... REPEAT ... END

La syntaxe est la suivante :

« ... WHILE *clause-de-test* REPEAT *clause-de-la-boucle* END ... »

WHILE ... REPEAT ... END évalue de manière répétitive un test (*clause-de-test*) et exécute la séquence *clause-de-la-boucle* si le test est vrai. La clause de test étant exécutée *avant* la clause de la boucle, cette dernière n'est pas exécutée si le test est faux.



Structure WHILE ... REPEAT ... END

WHILE commence l'exécution du test, qui renvoie un résultat dans la pile. REPEAT prend la valeur dans la pile. Si elle est différente de zéro, l'exécution se poursuit par la clause de la boucle. Sinon elle reprend après END. Si l'argument de REPEAT est une expression algébrique ou un nom, il est automatiquement évalué à un nombre.

Pour saisir WHILE ... REPEAT ... END dans un programme :

- Appuyez sur PRG BRCH ◀ WHILE

Exemple : Le programme suivant débute avec un nombre dans la pile et effectue de manière répétitive une division par 2 tant que le résultat reste divisible par 2. Par exemple, si le nombre est 24, le programme calcule 12, puis 6 et enfin, 3.

```
« WHILE DUP 2 MOD 0 == REPEAT 2 / DUP END DROP »
```

Exemple : Le programme suivant prend un nombre quelconque de vecteurs ou de tableaux dans la pile et les ajoute à la matrice statistique. (Les vecteurs et tableaux doivent avoir le même nombre de colonnes.) WHILE ... REPEAT ... END est utilisé plutôt que DO ... UNTIL ... END car le test doit être effectué *avant* l'addition. (S'il y a *seulement* des vecteurs ou des tableaux contenant le même nombre de colonnes dans la pile, le programme s'arrête en produisant une erreur une fois le dernier vecteur ou tableau ajouté à la matrice statistique.)

```
« WHILE DUP TYPE 3 == REPEAT Σ+ END »
```

Utilisation des compteurs de boucle

Pour certains problèmes, il peut s'avérer nécessaire de disposer d'un compteur à l'intérieur d'une structure en boucle afin de suivre le nombre de boucles. (Ce compteur n'est pas lié à la variable compteur des structures FOR ... NEXT/STEP.) Toute variable globale ou locale peut faire office de compteur. Les commandes INCR et DECR servent respectivement à incrémenter et décrémenter la valeur du compteur *et* à placer la nouvelle valeur obtenue dans la pile.

La syntaxe est la suivante :

« ... '*variable*' INCR ... »
ou
« ... '*variable*' DECR ... »

Pour saisir INCR ou DECR dans un programme :

- Appuyez sur ↰ (MEMORY) ARITH INCR ou DECR .

Les commandes INCR et DECR prennent un nom de variable globale ou locale (avec délimiteurs ') comme argument. Cette variable doit contenir un nombre réel. Ces commandes opèrent comme suit :

1. Elles modifient la valeur stockée dans la variable de +1 ou −1.
2. Elles renvoient la nouvelle valeur dans la pile.

Exemples : Si *c* contient la valeur 5, `'c' INCR` stocke 6 dans *c* et renvoie 6 dans la pile.

Le programme suivant prend un maximum de cinq vecteurs dans la pile et les ajoute à la matrice statistique en cours.

Programme :	**Commentaires :**
`«`	
`0 → c`	Stocke 0 dans la variable locale *c*.
`«`	Débute la procédure de définition.
`WHILE`	Débute la clause de test.
`DUP TYPE 3 ==`	Renvoie la valeur vraie si le niveau 1 contient un vecteur.
`'c' INCR`	Incrémente le compteur et renvoie la valeur dans *c*.
`5 ≤`	Renvoie la valeur vraie si le compteur est $c \leq 5$.
`AND`	Renvoie la valeur vraie si les deux résultats de test précédents sont vrais.
`REPEAT`	
`Σ+`	Ajoute le vecteur à ΣDAT.
`END`	Met fin à la structure.
`»`	Met fin à la procédure de définition.
`»`	

Utilisation de sommations au lieu de boucles

Pour certains calculs impliquant des sommations, vous pouvez utiliser la fonction Σ au lieu des boucles. Σ s'utilise en syntaxe de la pile ou en syntaxe algébrique. Cette fonction répète automatiquement l'opération d'addition pour la fourchette spécifiée de la variable d'indice et ce, sans recourir à une structure en boucle.

Exemple : Les programmes suivants prennent une limite supérieure entière *n* dans la pile, puis calculent la sommation :

$$\sum_{j=1}^{n} j^2$$

L'un des programmes utilise une structure FOR ... NEXT, l'autre la fonction Σ.

Programme :	**Commentaires :**
«	
0 1 ROT	Initialise la sommation et place les limites.
FOR j	Effectue le calcul en boucle.
j SQ +	
NEXT	
»	

Programme :	**Commentaires :**
«	
→ n	Utilise Σ pour effectuer la
'Σ(j=1,n,j^2)'	sommation.
»	

Exemple : Le programme suivant utilise ΣLIST pour calculer la somme de tous les éléments d'un vecteur ou d'une matrice. Le programme prend, dans la pile, un tableau ou un nom s'évaluant en tableau et renvoie la sommation.

Programme :	**Commentaires :**
«	
OBJ→	Calcule les dimensions du tableau et place ce dernier dans une liste au niveau 1.
1	Ajoute 1 à la liste. (Si le tableau
+	est un vecteur, la liste au niveau 1 ne comporte qu'un élément. ΠLIST produit une erreur si la liste comporte moins de deux éléments.)
ΠLIST	Multiplie tous les éléments de la liste.
→LIST	Convertit les éléments du tableau
ΣLIST	en une liste et en effectue la
»	somme.

Utilisation des indicateurs

Vous pouvez utiliser des indicateurs pour commander le fonctionnement du calculateur ou l'exécution d'un programme. Un indicateur est une sorte de commutateur qui est soit activé (*armé*) ou désactivé (*désarmé*). Le test de l'état d'un indicateur dans une structure conditionnelle ou en boucle permet au programme de prendre une décision en fonction du résultat. Certains indicateurs ayant une signification unique pour le calculateur, les tests de ces derniers étendent les possibilités de prise de décision du programme au-delà de ce que permettent les fonctions logiques et de comparaison.

Types d'indicateurs

Le HP 48 dispose de deux types d'indicateurs :

- **Indicateurs système.** Indicateurs −1 à −64. Ces indicateurs ont une signification prédéfinie pour le calculateur.
- **Indicateurs utilisateur.** Indicateurs 1 à 64. Les indicateurs utilisateur ne sont utilisés par aucune des opérations intégrées. Leur signification dépend entièrement de la manière dont ils sont employés par le *programme*.

Vous trouverez en annexe C la liste des 64 indicateurs système et leur définition. Par exemple, l'indicateur système −40 commande l'affichage de l'horloge. Lorsqu'il est *désarmé* (état par défaut) l'horloge n'est pas affichée et lorsqu'il est *armé*, l'horloge est affichée. (Appuyez sur CLK dans le menu ◀ MODES MISC, pour armer ou désarmer l'indicateur −40.)

Lorsque vous armez les indicateurs utilisateur 1 à 5, le témoin correspondant s'allume. Certaines cartes enfichables peuvent utiliser les indicateurs utilisateur 31 à 64.

Armement, désarmement et test des indicateurs

Les commandes des indicateurs prennent un numéro dans la pile : un entier de 1 à 64 (indicateurs utilisateur) ou de −1 à −64 (indicateurs système).

Pour armer, désarmer ou tester un indicateur :

1. Saisissez le numéro de l'indicateur (positif ou négatif).
2. Exécutez la commande voulue (voir le tableau ci-après).

Commandes des indicateurs

Touche	Commande programmable	Description
[PRG] TEST [NXT] [NXT] ou [↰] [MODES] FLAG :		
SF	SF	Arme l'indicateur.
CF	CF	Désarme l'indicateur.
FS?	FS?	Renvoie 1 (vrai) si l'indicateur est armé et 0 (faux) s'il est désarmé.
FC?	FC?	Renvoie 1 (vrai) si l'indicateur est désarmé et 0 (faux) s'il est armé.
FS?C	FS?C	Teste l'indicateur (renvoie vrai s'il est armé), puis désarme l'indicateur.
FC?C	FC?C	Teste l'indicateur (renvoie vrai s'il est désarmé), puis désarme l'indicateur.

Exemple : indicateur système. Le programme suivant définit une alarme pour le 15 septembre, 1994 à 17 heures 05. Il teste d'abord l'état de l'indicateur système −42 (format de la date) dans une structure conditionnelle pour fournir la date au format de date en cours.

Programme :	Commentaires :
«	
IF	Teste l'état de l'indicateur −42
−42 FC?	(format de date).
THEN	Si l'indicateur −42 est désarmé,
9.151994	fournit la date au format *mois/jour/année*.
ELSE	Si l'indicateur −42 est armé,
15.091994	fournit la date au format *jour.mois.année*.
END	Met fin à la structure conditionnelle.
17.05 "TEST COMPLETE"	Définit l'alarme : 17.05
3 →LIST STOALARM	correspond à l'heure de l'alarme et "TEST COMPLETE" au message associé.
»	

Exemple : Indicateur utilisateur. Le programme suivant renvoie soit la partie décimale soit la partie entière d'un nombre situé au niveau 1 en fonction de l'état de l'indicateur utilisateur 10.

Programme :	Commentaires :
«	
IF	Débute la structure conditionnelle.
10 FS?	Teste l'état de l'indicateur utilisateur 10.
THEN	Si l'indicateur 10 est armé,
IP	renvoie la partie entière.
ELSE	Si l'indicateur 10 est désarmé,
FP	renvoie la partie décimale.
END	Met fin à la structure conditionnelle.
»	

Pour utiliser ce programme, saisissez un nombre, armez l'indicateur 10 (pour obtenir la partie entière) ou désarmez-le (pour obtenir la partie décimale), puis exécutez le programme.

Rappel et stockage de l'état des indicateurs

Lorsqu'un programme modifie l'état d'un indicateur lors de l'exécution, vous pouvez, le cas échéant, faire en sorte qu'il en sauvegarde et restitue ensuite l'état d'origine.

Les commandes RCLF (rappel des indicateurs) et STOF (stockage des indicateurs) permettent le rappel et le stockage de l'état des indicateurs du HP 48. Pour ces commandes, un entier binaire de 64 bits représente l'état des 64 indicateurs. Les bits 0 correspondent à un indicateur désarmé, et les bits 1 à un indicateur armé. Le bit le plus à droite (le moins significatif) correspond à l'indicateur système −1 ou à l'indicateur utilisateur 1.

Pour rappeler l'état en cours des indicateurs :

- Exécutez RCLF (◀ MODES FLAG NXT RCLF).

RCLF renvoie une liste contenant deux entiers binaires de 64 bits représentant l'état en cours des indicateurs système et utilisateur :

{ #n_s #n_u }

Pour modifier l'état en cours des indicateurs :

1. Saisissez l'argument d'état des indicateurs (voir plus loin).
2. Exécutez STOF (◀ MODES FLAG NXT STOF).

STOF définit l'état en cours des indicateurs en fonction de l'argument d'état spécifié :

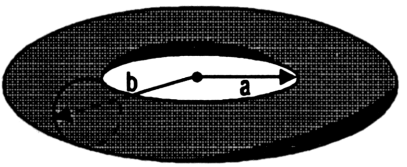
#n_s	Modifie uniquement l'état des indicateurs système.
{ #n_s #n_u }	Modifie l'état des indicateurs système et utilisateur.

Exemple : Le programme *PRESERVE*, page 2-9, utilise les commandes RCLF et STOF.

Utilisation de sous-programmes

Le programme étant un objet, il peut être utilisé comme sous-programme d'un autre programme. Le programme B, utilisé par le programme A, est *appelé* par ce dernier dont il constitue un sous-programme.

Exemple : Le programme *TORSA* calcule l'aire de la surface d'un tore de rayon interne a et de rayon externe b. *TORSA* est un sous-programme de *TORSV* qui calcule le volume d'un tore.



L'aire de la surface et le volume sont calculés par :

$$A = \pi^2(b^2 - a^2) \qquad V = \frac{1}{4}\pi^2(b^2 - a^2)(b - a)$$

(La grandeur $\pi^2(b^2 - a^2)$ dans la deuxième équation correspond à l'aire de la surface d'un tore calculé par *TORSA*.)

Diagramme de la pile et listage du programme *TORSA* :

Niveau 2	Niveau 1	→	Niveau 1
a	*b*	→	*aire de la surface*

Programme :	Commentaires :
«	
→ a b	Crée les variables locales *a* et *b*.
'π^2*(b^2-a^2)'	Calcule l'aire de la surface.
→NUM	Convertit une expression algébrique en un nombre.
»	
ENTER	Place le programme dans la pile.
' TORSA STO	Stocke le programme dans *TORSA*.

Diagramme de la pile et listage du programme *TORSV* :

Niveau 2	Niveau 1	→	Niveau 1
a	*b*	→	*volume*

Programme :	Commentaires :
«	
→ a b	Crée les variables locales *a* et *b*.
«	Débute un programme comme procédure de définition.
a b TORSA	Place dans la pile les nombres stockés dans *a* et *b* comme arguments, puis appelle *TORSA*.
b a - * 4 /	Termine le calcul du volume au moyen de l'aire de la surface.
»	Met fin à la procédure de définition.
»	
ENTER	Place le programme dans la pile.
' TORSV STO	Stocke le programme dans *TORSV*.

A présent, utilisez *TORSV* pour calculer le volume d'un tore de rayon interne $a = 6$ et de rayon externe $b = 8$.

6 ENTER 8
VAR TORSV

Exécution pas à pas d'un programme

Il est plus facile de comprendre le fonctionnement d'un programme si vous l'exécutez pas à pas. Cette façon de procéder peut vous aider à déboguer vos programmes ou à comprendre des programmes écrits par d'autres personnes.

Pour exécuter un programme pas à pas à partir du début :

1. Placez le programme ou son nom au niveau 1 (ou ligne de commande).
2. Appuyez sur PRG NXT RUN DBUG pour le lancer et arrêter immédiatement son exécution. Le témoin HALT s'affiche dans la zone d'état.
3. Effectuez l'une des opérations suivantes :
   - Pour afficher l'étape suivante du programme dans la zone d'état et l'exécuter, appuyez sur SST .
   - Pour afficher l'étape suivante ou les deux étapes suivantes sans les exécuter, appuyez sur NEXT .
   - Pour poursuivre l'exécution normale, appuyez sur ← CONT .
   - Pour abandonner l'exécution, appuyez sur KILL .
4. Répétez l'étape précédente autant de fois que nécessaire.

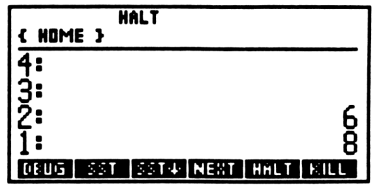
Pour désactiver le témoin HALT à tout moment :

- Appuyez sur PRG NXT RUN KILL .

Exemple : Exécutez le programme *TORSV* pas à pas avec $a = 6$ et $b = 8$.

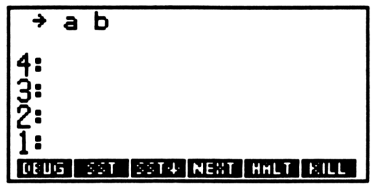
Sélectionnez le menu VAR et saisissez les données nécessaires. Saisissez le nom du programme et lancez le débogage. HALT indique que l'exécution du programme est interrompue.

◀ CLEAR
VAR
6 ENTER 8 ENTER
' TORSV
PRG NXT RUN DBUG



Affichez et exécutez la première étape du programme. Notez qu'il prend les deux arguments dans la pile et les stocke dans les variables locales *a* et *b*.

SST



Poursuivez l'exécution pas à pas jusqu'à ce que le répertoire en cours s'affiche dans la zone d'état. Observez la pile et la zone d'état tout au long de l'exécution pas à pas du programme.

SST ... SST

Pour exécuter un programme pas à pas à partir du milieu :

1. Insérez une commande HALT dans le programme, là où vous désirez commencer l'exécution pas à pas.
2. Exécutez le programme normalement. Celui-ci s'arrête après exécution de la commande HALT et le témoin HALT s'affiche.
3. Effectuez l'une des opérations suivantes :
   - Pour afficher l'étape suivante dans la zone d'état et l'exécuter, appuyez sur SST .
   - Pour afficher l'étape suivante ou les deux étapes suivantes sans les exécuter, appuyez sur NEXT .
   - Pour poursuivre l'exécution normale, appuyez sur ◀ CONT .
   - * Pour abandonner l'exécution, appuyez sur KILL .
4. Répétez l'étape précédente autant de fois que nécessaire.

Lorsque vous voulez exécuter le programme normalement, supprimez la commande HALT.

Pour exécuter un programme pas à pas lorsque l'étape suivante est un sous-programme :

- Pour exécuter le sous-programme en une seule étape, appuyez sur SST .
- Pour exécuter le sous-programme pas à pas, appuyez sur SST↓ .

SST exécute l'étape suivante d'un programme. Si celle-ci est un sous-programme, SST l'exécute en une seule étape. SST↓ fonctionne comme SST , mais si l'étape suivante est un sous-programme, il passe à la première étape du sous-programme, l'exécutant pas à pas.

Exemple : Dans l'exemple précédent, vous avez utilisé SST pour exécuter le sous-programme *TORSA* en une seule étape. A présent, exécutez le program *TORSV* pas à pas pour calculer le volume d'un tore de rayons $a = 10$ et $b = 12$. Lorsque vous accédez au sous-programme *TORSA*, exécutez-le pas à pas.

Sélectionnez le menu VAR et saisissez les données nécessaires. Saisissez le nom de programme et lancez le débogage. Exécutez les quatre premières étapes du programme, puis vérifiez l'étape suivante.

(←) CLEAR VAR
10 ENTER 12
' TORSV
PRG NXT RUN DBUG
SST↓ (4 fois)
NEXT

L'étape suivante est TORSA. Exécutez-le pas à pas, puis vérifiez que vous en êtes à la première étape de *TORSA*.

SST↓
NEXT

Appuyez sur ↩ CONT ↩ CONT pour terminer l'exécution du sous-programme et du programme.

Le tableau suivant récapitule les commandes d'exécution pas à pas d'un programme.

Commandes d'exécution pas à pas

Touche	Commande Programmable	Description
PRG NXT	RUN :	
DBUG		Lance l'exécution du programme, puis l'interrompt comme si HALT constituait la première commande du programme. Prend comme argument le programme ou le nom de programme dans le niveau 1.
SST		Exécute l'objet suivant, ou la commande, du programme interrompu.
SST↓		Identique à SST , sauf si l'étape suivante du programme est un sous-programme. Dans ce cas, procède à l'exécution pas à pas jusqu'à la première étape du sous-programme.
NEXT		Affiche l'objet ou les deux objets suivants, sans les exécuter. Ils restent affichés jusqu'à la séquence de touches suivante.
HALT	HALT	Interrompt l'exécution du programme au niveau de la commande HALT dans le programme.
KILL	KILL	Annule tous les programmes interrompus et désactive le témoin HALT.
↩ CONT	CONT	Reprend l'exécution d'un programme interrompu.

Interception d'erreurs

Si vous effectuez une opération incorrecte au clavier, elle n'est pas exécutée et un message d'erreur s'affiche. Par exemple, si vous exécutez + avec un vecteur et un nombre réel de la pile, le HP 48 affiche le message + Error: Bad Argument Type et renvoie les arguments dans la pile (si la commande Last est activée).

Il en va de même dans un programme, dont l'exécution est en outre stoppée. Si vous anticipez les conditions d'erreur, le programme est en mesure de les traiter sans interruption de l'exécution.

Dans le cas des programmes simples, il est facile de les relancer en cas d'arrêt dû à une erreur. Dans d'autres cas, vous pouvez concevoir des *pièges* à erreur afin de poursuivre l'exécution. Vous pouvez également créer des conditions d'erreurs utilisateur pour qu'elles soient interceptées par les programmes. Les commandes d'interception d'erreurs se trouvent dans le menu PRG ERROR.

Création et analyse d'erreurs

Nombre de conditions sont automatiquement détectées par le HP 48 comme des erreurs, et elles sont automatiquement traitées comme telles dans les programmes.

En outre, vous pouvez définir des conditions d'erreur. Vous pouvez définir la cause d'une *erreur-utilisateur* (ainsi que le message correspondant) ou d'une erreur intégrée. En règle générale, vous incluez une structure conditionnelle ou en boucle comportant un test de condition d'erreur. Si la condition d'erreur est constatée, l'erreur-utilisateur ou intégrée correspondante est signalée.

Pour provoquer une erreur-utilisateur dans un programme :

1. Saisissez une chaîne (avec délimiteurs " ") contenant le message d'erreur voulu.
2. Exécutez la commande DOERR (menu PRG ERROR).

Pour provoquer arbitrairement une erreur intégrée dans un programme :

1. Saisissez le numéro de l'erreur (sous forme d'entier binaire ou de nombre réel).
2. Exécutez la commande DOERR (menu PRG ERROR).

Si DOERR est intercepté dans une structure IFERR (voir plus loin), l'exécution se poursuit. Sinon, l'exécution est arrêtée à la commande DOERR et le message d'erreur s'affiche.

Pour analyser une erreur dans un programme :

- Pour obtenir le numéro de la dernière erreur, exécutez ERRN (menu PRG ERROR).
- Pour obtenir le message d'erreur de la dernière erreur, exécutez ERRM (menu PRG ERROR).
- Pour supprimer les informations relatives à la dernière erreur, exécutez ERR0 (menu PRG ERROR).

Le numéro d'une erreur-utilisateur est #70000h. Pour de plus amples informations sur les numéros des erreurs intégrées, consultez l'annexe B du Manuel d'utilisation.

Exemple : Le programme suivant arrête prématurément l'exécution du programme si la liste au niveau 1 contient trois objets.

```
«
 OBJ→
 IF 3 SAME
 THEN "3 OBJECTS IN LIST" DOERR
 END
»
```

Le tableau suivant récapitule les commandes d'interception d'erreurs.

Commandes d'interception d'erreurs

Touche	Commande programmable	Description
(PRG) (NXT) ERROR:		
DOERR	DOERR	Provoque une erreur. Pour une chaîne au niveau 1, provoque une erreur-utilisateur : le calculateur se comporte comme dans le cas d'une erreur ordinaire. Pour un entier binaire ou un nombre réel au niveau 1, provoque l'erreur intégrée correspondante. Si elle n'est pas interceptée dans une structure IFERR, DOERR affiche le message d'erreur correspondant et arrête l'exécution du programme. (Pour 0 au niveau 1, il y a arrêt de l'exécution sans mise à jour du numéro ou du message d'erreur, comme avec (CANCEL).)
ERRN	ERRN	Renvoie le numéro de la dernière erreur sous forme d'entier binaire. Renvoie #0 si ne numéro d'erreur a été supprimé au moyen de ERR0.
ERRM	ERRM	Renvoie le message de la dernière erreur (une chaîne de caractères). Renvoie une chaîne vide si le numéro d'erreur a été supprimé au moyen de ERR0.
ERR0	ERR0	Supprime le dernier numéro d'erreur et le message associé.

Création d'une structure d'interception d'erreurs

Les structures conditionnelles suivantes permettent d'intercepter les erreurs :

- IFERR ... THEN ... END.
- IFERR ... THEN ... ELSE ... END.

Structure IFERR ... THEN ... END

La syntaxe est la suivante :

« ... IFERR *clause-d'interception* THEN *clause-d'erreur* END ... »

Les commandes de la *clause d'erreur* ne sont exécutées que si une erreur se produit durant l'exécution de la *clause d'interception*. Si une erreur se produit dans la clause d'interception, elle est ignorée, le reste de la clause d'interception n'est pas pris en compte et l'exécution du programme passe directement à la clause d'erreur. Si *aucune* erreur ne se produit dans la clause d'interception, la clause d'erreur est ignorée et l'exécution reprend après la commande END.

Pour saisir IFERR ... THEN ... END dans un programme :

- Appuyez sur PRG NXT ERROR ◀ IFERR.

Exemple : Le programme suivant prend un nombre quelconque de vecteurs ou de tableaux dans la pile et les ajoute à la matrice statistique. Cependant, le programme s'arrête en signalant une erreur s'il détecte un vecteur ou un tableau comportant un nombre différent de colonnes. En outre, si *seuls* des vecteurs ou des tableaux comportant un nombre identique de colonnes se trouvent dans la pile, le programme s'arrête en signalant une erreur une fois le dernier vecteur ou tableau retiré de la pile.

```
« WHILE DUP TYPE 3 == REPEAT Σ+ END »
```

Dans la version suivante, le programme ajoute l'objet de niveau 1 à la matrice statistique tant qu'aucune erreur ne se produit. Lorsque c'est le cas, il s'arrête et affiche le message DONE.

Programme :	**Commentaires :**
«	
IFERR	Débute la clause d'interception.
WHILE	La structure WHILE répète
1	l'opération d'ajout de vecteurs et
REPEAT	de tableaux à la matrice
Σ+	statistique jusqu'à ce qu'une
END	erreur se produise.
THEN	Débute la clause d'erreur. Si une
"DONE" 1 DISP	erreur se produit dans la structure
1 FREEZE	WHILE, affiche le message DONE dans la zone d'état.
END	Met fin à la structure d'erreur.
»	

Structure IFERR ... THEN ... ELSE ... END

La syntaxe est la suivante :

« ... IFERR *clause-d'interception*
THEN *clause-d'erreur* ELSE *clause-de-normalité* END ... »

Les commandes de la *clause d'erreur* ne sont exécutées que si une erreur se produit durant l'exécution de la *clause d'interception*. Si une erreur se produit dans la clause d'interception, elle est ignorée, le reste de la clause d'interception n'est pas pris en compte et l'exécution du programme passe directement à la clause d'erreur. Si *aucune* erreur ne se produit dans la clause d'interception, l'exécution passe à la *clause de normalité* à la fin de la clause d'interception.

Pour saisir IFERR ... THEN ... ELSE ... END dans un programme :

- Appuyez sur PRG NXT ERROR ▶ IFERR.

Exemple : Le programme suivant demande deux nombres, puis les additionne. S'il en manque un, le programme affiche un message d'erreur et renouvelle sa demande.

Programme :	Commentaires :
«	
DO	Débute la boucle principale.
"KEY IN a AND b" " "	Demande deux nombres.
INPUT OBJ→	
UNTIL	Débute la clause de test de la boucle.
IFERR	La clause d'interception contient
+	seulement la commande +.
THEN	Si une erreur se produit, affiche le
ERRM 5 DISP	message Too Few Arguments
2 WAIT	pendant 2 secondes, puis place 0
0	(faux) dans la pile pour la boucle principale.
ELSE	S'il n'y a pas d'erreur, place 1
1	(vrai) dans la pile pour la boucle principale.
END	Met fin à la clause d'interception d'erreurs.
END	Met fin à la boucle principale. Si 0 (faux) a été placé dans la pile, la boucle principale se répète, sinon le programme est terminé.
»	

Saisie

Pour effectuer une saisie en cours de programme, vous pouvez faire en sorte que ce dernier s'arrête ou qu'il utilise des listes de sélection ou des masques de saisie (boîtes de dialogue), avant de reprendre son exécution. Les commandes nécessaires à cet effet sont les suivantes :

- PROMPT ([←][CONT] pour reprendre).
- DISP FREEZE HALT ([←][CONT] pour reprendre).
- INPUT ([ENTER] pour reprendre).
- INFORM
- CHOOSE

Commandes de saisie de données

Touche	Commande	Description
PRG NXT	IN :	
INFOR	INFORM	Crée un masque de saisie utilisateur.
NOVA	NOVAL	Place une valeur de garde pour la commande INFORM. Renvoyée lorsqu'une valeur manque dans un champ du masque de saisie.
CHOOS	CHOOSE	Crée un liste de sélection utilisateur.
KEY	KEY	Renvoie un résultat de test au niveau 1 et, s'il y a eu pression sur une touche, l'emplacement de cette dernière (niveau 2).
WAIT	WAIT	Interrompt l'exécution du programme pendant la durée spécifié (en secondes, niveau 1).
INPUT	INPUT	Interrompt l'exécution du programme pour la saisie des données.
PROM	PROMPT	Interrompt l'exécution du programme pour la saisie des données.

Utilisation de PROMPT ... CONT

PROMPT utilise la zone d'état pour demander la saisie et permet à l'utilisateur d'employer normalement le clavier à cet effet.

Pour saisir PROMPT dans un programme :

1. Saisissez la chaîne de demande de saisie (avec délimiteurs " ") devant s'afficher dans la zone d'état.
2. Saisissez la commande PROMPT (menu PRG IN).

« ... "*chaîne-de-demande*" PROMPT ... »

PROMPT prend l'argument chaîne dans le niveau 1, l'affiche (sans les délimiteurs " ") dans la zone d'état, puis interrompt l'exécution du programme, rendant la main au clavier du calculateur.

Lorsque l'exécution reprend, les données saisies sont laissées telles quelles dans la pile.

Pour répondre à PROMPT lors de l'exécution d'un programme :

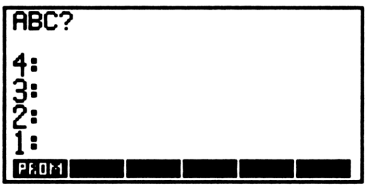
1. Saisissez les données. Vous pouvez utiliser les opérations du clavier pour effectuer les calculs nécessaires.
2. Appuyez sur ◀ CONT.

Le message reste affiché jusqu'à ce que vous appuyiez sur ENTER ou sur CANCEL, ou que vous mettiez à jour la zone d'état.

Exemple : Si vous exécutez le segment de programme suivant :

```
« "ABC?" PROMPT »
```

l'affichage se présente ainsi :



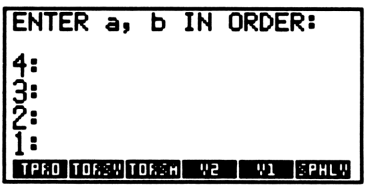
Exemple : Le programme suivant, *TPROMPT*, vous demande de saisir les dimensions d'un tore, puis appelle le programme *TORSA* (page 1-47) pour en calculer l'aire de la surface. Nul besoin de saisir des données dans la pile avant l'exécution du programme.

Programme :	**Commentaires :**
«	
"ENTER a, b IN ORDER:"	Place la chaîne de demande de saisie dans la pile.
PROMPT	Affiche la chaîne dans la zone d'état, interrompt l'exécution du programme et rend la main au clavier du calculateur.
TORSA	Exécute *TORSA* avec les arguments qui viennent d'être saisis dans la pile.
»	
ENTER ' TPROMPT STO	Stocke le programme dans *TPROMPT*.

Exécutez *TPROMPT* pour calculer le volume d'un tore de rayon interne $a = 8$ et de rayon externe $b = 10$.

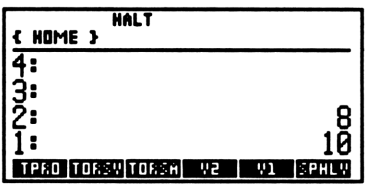
Exécutez *TPROMPT*. Le programme demande les données nécessaires.

← CLEAR
VAR TPRO



Saisissez les rayons interne et externe. Une fois que vous avez appuyé sur ENTER, le message est supprimé de la zone d'état.

8 ENTER 10



Relancez l'exécution du programme.

← CONT

Notez que lorsque le programme est interrompu par PROMPT, vous pouvez effectuer des opérations avec le calculateur comme avant le

lancement du programme. Si le rayon externe *b* du tore de l'exemple précédent mesure 0.83 pied, vous pouvez le convertir en pouces *pendant l'interruption du programme* en appuyant sur .83 ENTER 12 ×, puis sur ◀ CONT.

Utilisation de DISP FREEZE HALT ... CONT

DISP FREEZE HALT permet de commander l'affichage pendant la saisie et à l'utilisateur d'employer normalement le clavier.

Pour saisir DISP FREEZE HALT dans un programme :

1. Saisissez une chaîne ou un autre objet devant s'afficher en tant que demande de saisie.
2. Saisissez le numéro de la ligne à laquelle l'affichage doit se produire.
3. Saisissez la commande DISP (menu PRG OUT).
4. Saisissez un nombre spécifiant les zones de l'affichage à "geler".
5. Saisissez la commande FREEZE (menu PRG OUT).
6. Saisissez la commande HALT (menu PRG OUT).

« ... *objet-demande ligne-d'affichage* DISP
zone-à-geler FREEZE HALT ... »

DISP affiche l'objet à la ligne spécifiée de l'affichage. DISP prend deux arguments dans la pile : un objet au niveau 2 et un numéro de ligne d'affichage (de 1 à 7) au niveau 1. Si l'objet est une chaîne, elle est affichée sans délimiteurs " ". L'affichage créé par DISP dure le temps de l'exécution du programme. A la fin de ce dernier, ou s'il est interrompu par HALT, le calculateur récupère l'environnement normal de la pile et met à jour l'affichage. Cependant, vous pouvez utiliser FREEZE pour conserver l'affichage de la demande de saisie.

FREEZE "gèle" les zones d'affichage afin d'en empêcher toute mise à jour tant qu'il n'y a pas eu de *pression sur une touche*. L'argument *n* au niveau 1 est la somme des codes correspondant aux zones à geler : 1 pour la zone d'état, 2 pour la zone de la pile et de la ligne de commande, 4 pour la zone de menu.

L'exécution du programme est interrompue à l'emplacement de la commande HALT, le témoin HALT est activé et le calculateur reprend la main pour l'exécution normale des opérations au clavier.

Lorsque l'exécution reprend, les données saisies sont laissées telles quelles dans la pile.

Pour répondre à HALT lors de l'exécution d'un programme :

1. Saisissez les données. Vous pouvez utiliser les opérations du clavier pour effectuer les calculs nécessaires.
2. Appuyez sur ◀ CONT.

Exemple : Si vous exécutez le segment de programme suivant :

```
« "ABC■DEF■GHI" CLLCD 1 DISP 3 FREEZE HALT »
```

l'affichage se présente ainsi :

(Le symbole ■ représente le caractère de retour à la ligne ↵ du calculateur une fois le programme entré dans la pile.)

Utilisation de INPUT ... ENTER

INPUT permet d'utiliser la zone de la pile pour une demande de saisie, de fournir des données à saisir par défaut et d'empêcher l'utilisation des opérations normales dans la pile ou de modifier les données qui s'y trouvent.

Pour saisir INPUT dans un programme :

1. Saisissez une chaîne de demande de saisie (avec délimiteurs " ") devant s'afficher en haut de la zone de la pile.
2. Saisissez une chaîne ou une liste (avec délimiteurs) spécifiant le contenu et le comportement de la ligne de commande (voir plus loin).
3. Saisissez la commande INPUT (menu PRG IN).
4. Saisissez OBJ→ (menu PRG TYPE) ou tout autre commande traitant la saisie en tant que chaîne.

```
« ... "chaîne-de-demande" "ligne-de-commande" INPUT OBJ→ ...
»
```

ou

« ... "*chaîne-de-demande*" {*ligne-de-commande*} INPUT OBJ→ ...
»

INPUT, dans sa forme la plus simple, prend deux chaînes comme arguments (voir plus loin la liste des options). INPUT efface la zone de la pile, affiche la chaîne du niveau 2 en haut de cette zone et la chaîne du niveau 1 dans la ligne de commande. Ensuite, INPUT active le mode de saisie de programme, place le curseur d'insertion après la chaîne dans la ligne de commande et interrompt l'exécution du programme.

A la reprise de l'exécution, les données saisies sont renvoyées au niveau 1 sous forme de chaîne, dite *résultante*.

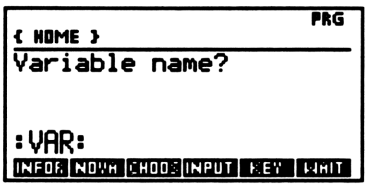
Pour répondre à INPUT lors de l'exécution d'un programme :

1. Saisissez les données. (Vous ne pouvez pas exécuter de commandes : elles sont simplement copiées dans la ligne de commande.)
2. *Facultatif* : pour effacer la ligne de commande et recommencer, appuyez sur CANCEL.
3. Appuyez sur ENTER.

Exemple : Si vous exécutez le segment de programme suivant :

```
« "Variable name?" ":VAR:" INPUT »
```

l'affichage se présente ainsi :

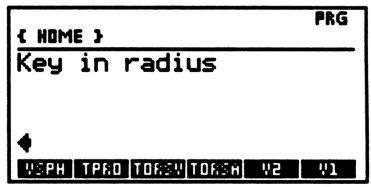


Exemple : Le programme suivant, *VSPH*, calcule le volume d'une sphère. *VSPH* demande le rayon de la sphère, l'élève au cube et multiplie par $^4/_3\ \pi$. *VSPH* exécute INPUT pour demander le rayon. INPUT active le mode de saisie de programme lorsque le programme s'interrompt pour la saisie des données.

Programme :	Commentaires :
«	
"Key in radius"	Spécifie la chaîne de demande de saisie.
""	Spécifie la chaîne de la ligne de commande. Dans le cas présent, la ligne de commande sera vide.
INPUT	Affiche la demande, place le curseur au début de la ligne de commande et interrompt le programme pour la saisie des données (ici, le rayon de la sphère).
OBJ→	Convertit la chaîne résultante en objet approprié à ses composants, ici un nombre réel.
3 ^	Elève le rayon au cube.
4 * 3 / π * →NUM	Termine le calcul.
»	
ENTER ' VSPH STO	Stocke le programme dans *VSPH*.

Exécutez *VSPH* pour calculer le volume d'une sphère de rayon 2.5.

VAR VSPH



Saisissez le rayon et reprenez l'exécution du programme.

2.5 ENTER

Pour inclure des options INPUT :

- Utilisez une liste (avec délimiteurs { }) comme argument de la ligne de commande pour INPUT. Cette liste peut contenir plusieurs des éléments suivants :
  - chaîne de la ligne de commande (avec délimiteurs " ").

- Position du curseur sous forme de nombre réel ou de liste contenant deux nombres réels.
- Options de fonctionnement ALG, α ou V.

Sous sa forme courante, l'argument de niveau 1 de INPUT est une liste qui spécifie le contenu et l'interprétation de la ligne de commande. Cette liste peut contenir plusieurs des paramètres suivants, quel qu'en soit l'ordre :

{ "*ligne-de-commande*" *position-du-curseur options-de-fonctionnement* }

"*ligne-de-commande*"	Spécifie le contenu de la ligne de commande lorsque le programme s'interrompt. Les caractères imbriqués de retour à la ligne multiplient les lignes dans l'affichage. (Sinon, la ligne de commande est vierge.)
position-du-curseur	Spécifie la position et le type du curseur dans la ligne de commande. (Sinon, il y a un curseur d'insertion à la fin de la ligne de commande.) ■ Un *nombre réel n* spécifie le *n*ième caractère de la première ligne de la ligne de commande. Zéro spécifie la fin de la chaîne de la ligne de commande. Un nombre positif spécifie le curseur d'*insertion* ; un nombre négatif, le curseur de *remplacement*. ■ Une *liste* {*ligne caractère*} spécifie la position de la ligne et du caractère. La ligne 1 correspond à la première ligne de la ligne de commande. Les caractères se décomptent à partir de l'extrémité gauche de chaque ligne : le caractère 0 spécifie l'extrémité de la ligne. Un nombre positif spécifie le curseur d'*insertion*, un nombre négatif le curseur de *remplacement*.
options-de-fonctionnement	Spécifie la configuration de la saisie et le traitement au moyen des noms sans apostrophes suivants : ■ ALG active le mode de saisie algébrique/de programme (pour la syntaxe algébrique). (Sinon, le mode de saisie de programme est actif.)

- α (α →A) spécifie le verrouillage en mode de saisie alphabétique. (Sinon, ce mode est inactif.)
- V vérifie si la chaîne résultante (sans délimiteurs " ") est un objet ou une séquence d'objets autorisés. Si la chaîne est incorrecte, INPUT affiche le message `Invalid Syntax` et renouvelle la demande de saisie de données. (Si l'option n'est pas spécifiée, la syntaxe n'est pas contrôlée.)

Pour concevoir la chaîne de la ligne de commande

pour INPUT :

- Pour une saisie simple, utilisez une chaîne produisant un objet correct :
  - Utilisez une chaîne vide.
  - Utilisez un indentificateur : :*identificateur*:.
  - Utilisez un commentaire : @*texte*@.
- Pour une saisie spécifique, utilisez une chaîne produisant un cadre reconnaissable.

Une fois que vous avez saisi les données dans la ligne de commande et appuyé sur ENTER pour reprendre l'exécution, le contenu de la ligne de commande est renvoyé au niveau 1 sous forme de chaîne résultante. En principe, la chaîne résultante contient également la chaîne d'origine de la ligne de commande. Si vous concevez avec soin la chaîne de la ligne de commande, vous pouvez faciliter le processus d'extraction des données saisies.

Pour traiter la chaîne résultante de INPUT :

- Pour une saisie simple, utilisez OBJ→ pour convertir la chaîne en objets correspondants.
- Pour une saisie complexe, utilisez l'option V pour que INPUT vérifie l'exactitude des objets, puis OBJ→ pour convertir la chaîne en objets correspondants.
- Pour une saisie spécifique, traitez-la comme un objet de type chaîne, si possible en extrayant les données sous forme de sous-chaînes.

Exemple :

Le programme *VSPH*, page 1-64 utilise une chaîne de ligne de commande vide.

Exemple : Le programme *SSEC*, page 1-70 utilise une chaîne de ligne de commande dont les caractères forment un cadre. Le programme extrait les sous-chaînes de la chaîne résultante.

Exemple : La chaîne de la ligne de commande "@UPPER LIMIT@" affiche @UPPER LIMIT@◆ dans la ligne de commande. Si vous appuyez sur 200 (ENTER), la chaîne résultante est "@UPPER LIMIT@200". Lorsque la commande OBJ→ extrait le texte de la chaîne, elle supprime les caractères @, ainsi que le texte à l'intérieur, et renvoie le nombre 200. (Pour plus d'informations sur les commentaires introduits par @, consultez "Création de programmes sur un ordinateur", page 1-11.)

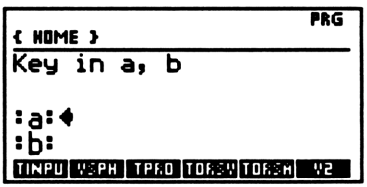
Exemple : Le programme suivant, *TINPUT*, exécute INPUT pour demander les rayons interne et externe d'un tore, puis appelle le programme *TORSA* (page 1-47) pour en calculer l'aire de la surface. *TINPUT* demande les valeurs de *a* et *b* dans une ligne de commande dédoublée. L'argument de niveau 1 de INPUT est une liste contenant :

- La chaîne de la ligne de commande déterminant les identificateurs et les délimiteurs de deux objets nommés.
- Une liste imbriquée spécifiant la position initiale du curseur.
- Le paramètre V pour vérifier l'exactitude de la syntaxe dans la chaîne résultante.

Programme :	**Commentaires :**
«	
"Key in a, b"	La chaîne de niveau 2, affichée en haut de la zone de la pile.
{ ":a:■:b:" {1 0} V }	La liste de niveau 1 contient une chaîne, une liste et l'option de vérification. (Pour saisir la chaîne, appuyez sur ▶ " " ▶ :: a ▶ ▶ ↵ ▶ :: b. Après avoir appuyé sur ENTER pour placer le programme terminé dans la pile, la chaîne s'affiche sur une ligne avec un symbole ■ mis pour le caractère de retour de ligne.) La liste imbriquée place le curseur d'insertion à la fin de la ligne 1.
INPUT	Affiche les chaînes de la pile et de la ligne de commande, positionne le curseur, active le mode de saisie de programme et interrompt l'exécution du programme pour la saisie des données.
OBJ→	Convertit la chaîne en objets correspondants (en l'occurrence deux objets nommés).
TORSA	Appelle *TORSA* pour calculer l'aire de la surface.
»	
ENTER ' TINPUT STO	Stocke le programme dans *TINPUT*.

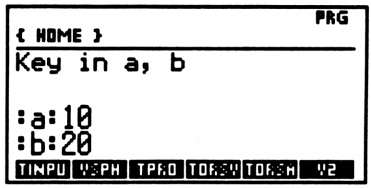
Exécutez *TINPUT* pour calculer l'aire de la surface d'un tore de rayon interne $a = 10$ et d'un rayon externe $b = 20$.

VAR TINPU



Saisissez la valeur de a, appuyez sur ▼ pour placer le curseur sur la zone de saisie suivante, puis saisissez la valeur de b.

10 ▼ 20



Reprenez l'exécution du programme.

ENTER

Exemple : Le programme suivant exécute INPUT pour demander le numéro de sécurité sociale, puis extrait deux chaînes : les trois premiers chiffres et les quatre derniers. L'argument de niveau 1 pour INPUT spécifie :

- Une chaîne de ligne de commande comportant des tirets.
- Le curseur en mode *remplacement* positionné au début de la chaîne de demande de saisie (−1). Ceci permet à l'utilisateur de "remplir" la chaîne de la ligne de commande en utilisant ▶ pour passer d'un tiret au suivant.
- Par défaut, aucune vérification de la syntaxe de l'objet n'est effectuée : les tirets rendent le contenu incorrect en tant qu'objet.

Niveau 1 →	Niveau 2	Niveau 1
→	*"quatre derniers chiffres"*	*"trois premiers chiffres"*

Programme :	Commentaires :
«	
"Key in S.S. #"	Chaîne de demande de saisie.
{ " - - " -1 }	Chaîne de la ligne de commande (3 espaces avant le premier –, deux entre les tirets et 4 après le dernier –).
INPUT	Interrompt le programme pour la saisie.
DUP 1 3 SUB SWAP 8 11 SUB	Copie la chaîne résultante, puis extrait les trois premiers chiffres et les quatre derniers chiffres sous forme de chaîne.
»	
ENTER ' SSEC STO	Stocke le programme dans *SSEC*.

Utilisation de INFORM et CHOOSE

Vous pouvez utiliser des masques de saisie (boîtes de dialogue) et des listes de sélection pour la saisie en cours d'exécution des programmes. Lorsque ces derniers contiennent des masques de saisie ou des listes de sélection, ils attendent que vous les confirmiez (OK ou les annuliez CANCEL) avant de poursuivre l'exécution.

Si vous appuyez sur OK, CHOOSE renvoie l'élément sélectionné (ou sa valeur correspondante) au niveau 2 et 1 au niveau 1. INFORM renvoie une liste de valeur de champ au niveau 2 et un 1 au niveau 1.

Les commandes INFORM et CHOOSE renvoient toutes deux 0 si vous appuyez sur CANCEL.

Pour définir un masque de saisie :

1. Saisissez une chaîne de titre pour le masque de saisie (utilisez → " ").
2. Saisissez une liste de spécifications de champ.
3. Saisissez une liste d'options de format.
4. Saisissez une liste de valeurs de réinitialisation (qui s'affichent lorsque vous appuyez sur RESET).
5. Saisissez une liste de valeurs par défaut.
6. Exécutez la commande INFORM.

Exemple : Saisissez un titre "FIRST ONE" (ENTER).
Spécifiez un champ { "Name:" } (ENTER).
Saisissez les options de format (une colonne, position de tabulation cinq) { 1 5 } (ENTER).
Saisissez la valeur de réinitialisation du champ { "THERESA" } (ENTER).
Saisissez la valeur par défaut { "WENDY" } (ENTER).
Exécutez INFORM ((PRG) (NXT) IN INFOR).
L'écran de gauche s'affiche. Appuyez sur (NXT) RESET OK et l'écran de droite s'affiche.

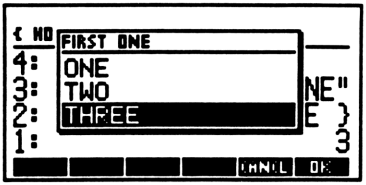
Vous pouvez définir un message d'aide et le type de données à saisir dans un champ en saisissant des spécifications sous forme de listes. Par exemple, { {"Name:" "Enter your name" 2} } définit le champ Name, affiche Enter your name au bas du masque de saisie et n'admet que des objets de type 2 (chaînes).

Pour définir une liste de sélection :

1. Saisissez une chaîne de titre.
2. Saisissez une liste d'éléments. S'il s'agit d'une liste comportant deux listes contenant chacune un élément, le premier est affiché dans la liste de sélection et le second est renvoyé au niveau 2 lorsque l'on appuie sur OK.
3. Saisissez un numéro de position correspondant à l'élément mis en valeur par défaut. (Dans le cas de 0, la liste de sélection est uniquement consultable.)
4. Exécutez la commande CHOOSE.

Exemple : Saisissez un titre "FIRST ONE" (ENTER).
Saisissez une liste d'éléments { ONE TWO THREE } (ENTER).
Saisissez le numéro de position de l'élément mis en valeur par défaut 3 (ENTER)

Exécutez CHOOSE (PRG NXT IN CHOOS).
La liste de sélection suivante s'affiche :



Exemple : Le programme suivant utilise des masques de saisie, des listes de sélection et des boîtes de messages pour créer une base de données simple contenant un répertoire téléphonique.

Programme :

```
«
'NAMES' VTYPE
  IF -1 ==
  THEN { } 'NAMES' STO
  END
 WHILE
"PHONELIST OPTIONS:"
{
{ "ADD A NAME" 1}
{ "VIEW A NUMBER" 2 }
 } 1 CHOOSE


  REPEAT → c «

CASE c 1 ==
        THEN
          WHILE
```

Commentaires :

Vérifie si la liste de noms (NAMES) existe, si ce n'est pas le cas, en crée une vide.

Sans pression sur CANCEL, crée une liste de sélection récapitulant les options de la base de données. Lorsque l'on appuie sur OK, le second élément d'un couple de la liste est renvoyé dans la pile.

Stocke la valeur renvoyée dans *c*.

1er cas (ajouter un nom).

Sans pression sur CANCEL, effectue les opérations suivantes :

Programme :

```
"ADD A NAME"
 {
 { "NAME:" "ENTER NAME" 2 }
 { "PHONE:" "ENTER A
PHONE NUMBER" 2 } }
 { } { } { } INFORM
 REPEAT
DUP
   IF { NOVAL } HEAD POS
   THEN
    DROP
    "Complete both fields
    before pressing OK"
    MSGBOX
 ELSE 1
    →LIST NAMES + SORT
    'NAMES' STO
    END
  END
 END

c 2 ==
 THEN
   IF { } NAMES SAME
   THEN
    "YOU MUST ADD A
      NAME FIRST"
     MSGBOX
```

Commentaires :

Crée un masque de saisie qui prend le nom et le numéro de téléphone. Les deux champs acceptent uniquement des chaînes (objets de type 2).

Vérifie si l'un des champs est vierge.

Si c'est le cas, affiche un message.

Sinon, ajoute la liste à NAMES, effectue le tri et la stocke à nouveau dans NAMES.

Met fin à la structure IF, la boucle WHILE et l'instruction CASE.

2e cas (Visualiser un numéro).

Vérifie si NAMES est une liste vide.

Si c'est le cas, affiche un message.

Programme :	Commentaires :
ELSE	Si NAMES n'est pas vide,
WHILE	crée une liste de sélection en
"VIEW A NUMBER"	utilisant les éléments de
NAMES 1 CHOOSE	NAMES comme critère de choix.
REPEAT	En appuyant sur OK, le
→STR MSGBOX	second élément d'un couple de liste de NAMES (numéro de téléphone) est renvoyé sous forme de chaîne et affiché.
END	Met fin à la boucle WHILE,
END	la structure IF et
END	l'instruction CASE.
END	Met fin à la structure
»	CASE, à la procédure de
END	définition de la variable
»	locale, à la boucle WHILE et au programme.
ENTER ' PHONES STO	Stocke le programme dans *PHONES*.

Vous pouvez supprimer des noms et des numéros en modifiant la variable NAMES. Pour améliorer ce programme, vous pouvez créer un programme de suppression.

Avertisseur sonore

Pour saisir BEEP dans un programme :

1. Saisissez un nombre spécifiant la fréquence du son en hertz.
2. Saisissez un nombre spécifiant la durée d'émission en seconde.
3. Saisissez la commande BEEP (menu PRG NXT OUT).

« ... *fréquence durée* BEEP ... »

BEEP prend deux arguments dans la pile : la fréquence au niveau 2 et la durée au niveau 1.

Exemple : La version modifiée ci-après de *TPROMPT* émet un signal sonore de 440 hertz, pendant une demi-seconde avant l'affichage de l'invite de saisie de données.

Programme :	**Commentaires :**
`«`	
`"ENTER a, b IN ORDER:"`	
`440 .5 BEEP`	Emet un son juste avant l'affichage de l'invite de saisie de données.
`PROMPT`	
`TORSA`	
`»`	

Arrêt d'un programme pour une saisie de séquence de touches

Le programme peut s'interrompre et attendre que l'utilisateur saisisse une séquence de touches. A cet effet, utilisez les commandes WAIT et KEY.

Utilisation de WAIT

En principe, la commande WAIT interrompt l'exécution pendant un nombre de secondes donné, mais vous pouvez spécifier que l'interruption soit liée à la la pression sur une touche.

Pour saisir WAIT dans un programme :

- Pour interrompre le programme sans modifier l'affichage, saisissez 0 et la commande WAIT (menu PRG IN).
- Pour interrompre le programme et afficher le menu en cours, saisissez −1 et la commande WAIT (menu PRG IN).

WAIT prend 0 ou −1 dans le niveau 1, puis interrompt l'exécution jusqu'à ce que l'utilisateur appuie sur la séquence de touches appropriée.

Dans le cas de −1, WAIT affiche le menu spécifié en cours. Cela vous permet de créer et d'afficher un menu d'options utilisateur pendant

que le programme est interrompu. (Un menu créé avec MENU ou TMENU ne s'affiche en principe pas avant que le programme se termine ou soit interrompu.)

A la reprise de l'exécution, le numéro à trois chiffres indiquant l'emplacement de la touche qui a été utilisée est laissé dans la pile. Ce numéro indique la ligne, la colonne et le mode shift de la touche.

Pour répondre à WAIT lors de l'exécution d'un programme :

- Appuyez sur une quelconque séquence de touches correcte. (↰ ou α seules ne constituent pas une séquence correcte.)

Utilisation de KEY

Vous pouvez utiliser KEY dans une boucle infinie pour interrompre l'exécution jusqu'à la pression sur une touche quelconque ou une touche donnée.

Pour saisir une boucle contenant KEY dans un programme :

1. Saisissez la structure en boucle.
2. Dans la séquence de clause de test, saisissez la commande KEY (menu PRG IN) ainsi que toutes commandes de touche nécessaires.
3. Dans la clause de la boucle, ne saisissez *aucune* commande pour donner l'impression d'une condition de "pause".

KEY renvoie 0 au niveau 1 lorsque la boucle commence. La commande continue de renvoyer 0 jusqu'à ce que l'on appuie sur une touche. Elle renvoie alors 1 au niveau 1 et le numéro d'emplacement à deux chiffres de la touche au niveau 2. Par exemple, ENTER renvoie 51 et ↰ 71.

En principe, la clause de test entraîne la répétition de la boucle jusqu'à ce que l'on appuie sur une touche. Lorsque c'est le cas, vous pouvez utiliser des tests de comparaison pour vérifier la valeur du numéro de touche.

(Voir "Utilisation des structures en boucle infinie", page 1-37 et "Utilisation des fonctions de comparaison", page 1-18.)

Pour répondre à une boucle KEY lors de l'exécution d'un programme :

- Appuyez sur une touche quelconque. (◀ ou α *compte* parmi les touches correctes.)

Exemple : Le segment de programme suivant renvoie 1 au niveau 1 si l'on appuie sur + et 0 au niveau 1 si l'on appuie sur n'importe quelle autre touche :

```
« ... DO UNTIL KEY END 95 SAME ... »
```

Résultats

Vous pouvez définir la manière dont un programme présente ses résultats et les rendre plus lisibles grâce aux techniques exposées ci-après.

Commandes de sortie des données

Touche	Commande	Description
PRG NXT	OUT :	
PVIEW	PVIEW	Affiche PICT en commençant aux coordonnées spécifiées.
TEXT	TEXT	Affiche la pile.
CLLCD	CLLCD	Efface l'affichage de la pile.
DISP	DISP	Affiche un objet à une ligne spécifiée.
FREEZ	FREEZE	Gèle une zone spécifiée de l'affichage jusqu'à ce que l'on appuie sur une touche.
MSGB	MSGBOX	Crée une boîte de messages utilisateur.
BEEP	BEEP	Emet un signal sonore à la fréquence (en hertz, niveau 2) et de la durée (en secondes, niveau 1) spécifiées.

Identification des résultats

Pour identifier un résultat :

1. Placez l'objet résultant dans la pile.
2. Saisissez un identificateur : une chaîne, un nom entre apostrophes ou un numéro.
3. Saisissez la commande →TAG (menu PRG TYPE).

« ... *objet identificateur* →TAG ... »

→TAG prend deux arguments (un objet et un identificateur) dans la pile et renvoie un objet identifié.

Exemple : Le programme *TTAG* suivant est identique à *TINPUT*, sauf qu'il renvoie le résultat sous la forme AREA: *valeur*.

Programme :	**Commentaires :**
«	
"Key in a, b"	
{ ":a:■:b:" {1 0} V }	
INPUT OBJ→	
TORSA	
"AREA"	Saisit l'identificateur (une chaîne).
→TAG	Utilise le résultat du programme et la chaîne pour créer un objet identifié.
»	
ENTER ' TTAG STO	Stocke le programme dans *TTAG*.

Exécutez *TTAG* pour calculer la surface d'un tore de rayon interne $a = 1.5$ et de rayon externe $b = 1.85$. Le résultat est renvoyé sous forme d'objet identifié.

VAR TTAG
1.5 ▼ 1.85
ENTER

Identification et affichage des résultats sous forme de chaînes

Pour identifier et afficher un résultat sous forme de chaîne :

1. Placez l'objet résultant dans la pile.
2. Saisissez la commande →STR (menu PRG TYPE).
3. Saisissez une chaîne pour identifier l'objet (avec délimiteurs " ").
4. Saisissez les commandes SWAP + pour permuter et concaténer les chaînes.
5. Saisissez un numéro spécifiant la ligne au niveau de laquelle afficher la chaîne.
6. Saisissez la commande DISP (menu PRG OUT).

« ... *objet* →STR *identificateur* SWAP + *ligne* DISP ... »

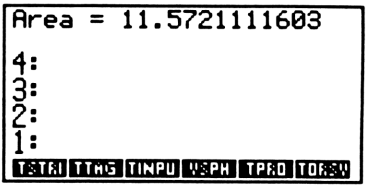
DISP affiche une chaîne sans délimiteurs " ".

Exemple : Le programme *TSTRING* suivant est identique à *TINPUT*, sauf qu'il convertit le résultat du programme en une chaîne qu'il dote d'un libellé.

Programme :	**Commentaires :**
«	
"Key in a, b"	
{ ":a:■:b:" (1 0) V }	
INPUT OBJ→	
TORSA	
→STR	Convertit le résultat en une chaîne.
"Area = "	Saisit la chaîne d'identification.
SWAP +	Permute et réunit les deux chaînes.
CLLCD 1 DISP 1 FREEZE	Affiche la chaîne résultante sans ses délimiteurs au niveau de la ligne 1 de l'affichage.
»	
ENTER ' TSTRING STO	Stocke le programme dans *TSTRING*.

Exécute *TSTRING* pour calculer l'aire d'un tore de rayons $a = 1.5$ et $b = 1.85$. Le résultat identifié s'affiche dans la zone d'état.

◀ CLEAR
VAR TSTRI
1.5 ▼ 1.85
ENTER



Pause pour afficher un résultat

Pour effectuer une pause afin d'afficher un résultat :

1. Saisissez les commandes de définition de l'affichage.
2. Saisissez la durée de la pause, en secondes.
3. Saisissez la commande WAIT (menu PRG IN).

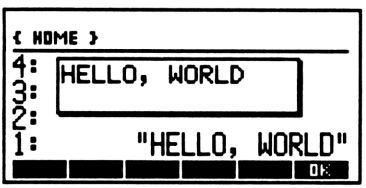
WAIT interrompt l'exécution pour le nombre de secondes au niveau 1. Vous pouvez utiliser WAIT avec DISP pour afficher des messages pendant l'exécution du programme, relatifs à des résultats intermédiaires, par exemple. (WAIT interprète les arguments 0 et −1 de manière différente. Voir "Utilisation de WAIT", page 1-76.)

Utilisation de MSGBOX pour afficher un message

Pour définir une boîte de message :

1. Saisissez une chaîne de message.
2. Exécutez la commande MSGBOX.

Exemple : Saisissez une chaîne "HELLO, WORLD" ENTER.
Exécutez MSGBOX (PRG NXT OUT MSGB).
Le message suivant s'affiche :



Vous devez accuser réception de ce message en appuyant sur OK ou sur CANCEL.

Utilisation des menus dans les programmes

Vous pouvez utiliser des menus dans les programmes pour effectuer différentes tâches :

- **Saisie de données au moyen d'un menu.** Vous pouvez définir un menu pour la saisie des données pendant l'interruption du programme, puis reprendre l'exécution du programme.
- **Application basée sur un menu.** Vous pouvez définir un menu, qui appelé par un programme, lance l'exécution d'autres programmes associés.

Pour définir un menu intégré ou un menu-bibliothèque :

1. Saisissez le numéro de menu.
2. Saisissez la commande MENU (menu MODES MENU).

Pour définir un menu personnalisé :

1. Saisissez une liste (avec délimiteurs { }) ou le nom d'une liste définissant les actions du menu. S'il s'agit d'une liste comportant deux listes d'éléments, le premier élément s'affiche dans le menu, mais c'est le *second* qui est renvoyé dans la pile lorsque l'on appuie sur la touche de menu.
2. Activez le menu :
   - Pour sauvegarder le menu comme menu CST, saisissez la commande MENU (menu MODES MENU).
   - Pour en faire un menu temporaire, saisissez la commande TMENU (menu MODES MENU).

Le menu ne s'affiche pas avant l'interruption du programme.

Les numéros des menus intégrés sont récapitulés au chapitre 3 à la rubrique de la commande MENU. Les menus-bibliothèques ont également des numéros : le numéro de la bibliothèque fait office de numéro de menu. Vous pouvez ainsi activer des menus d'applications (tels que les menus SOLVE et PLOT) et d'autres menus (tels que

VAR et CST) dans les programmes. Ces menus se comportent de la même façon que pour les opérations effectuées au clavier.

Comme son nom l'indique, vous créez un menu personnalisé en fonction de vos besoins (Voir plus loin). Vous pouvez le sauvegarder comme menu CST, pour le réutiliser en appuyant sur [CST] ; ou en faire un menu *temporaire*. Dans ce cas, il reste actif (même après l'exécution du programme) jusqu'à sélection d'un autre menu et n'a aucun impact sur le contenu de la variable *CST*.

Pour spécifier une *page* donnée du menu saisissez le numéro sous la forme *m.pp*, où *m* est le numéro de menu et *pp* le numéro de page (94.02 pour désigner la page 2 du menu TIME, par exemple). Si la valeur page *pp* n'existe pas, la page 1 est affichée (94 renvoie la page 1 du menu TIME).

Exemple : Saisissez `69 MENU` pour obtenir la page 1 du menu MODES MISC. Saisissez `69.02 MENU` pour obtenir la page 2 du menu MODES MISC.

Pour restaurer le menu précédent :

- Exécutez 0 MENU.

Pour rappeler le numéro du menu en cours :

- Exécutez la commande RCLMENU (menu MODES MENU).

Utilisation des menus pour la saisie des données

Pour afficher un menu pour la saisie de données dans un programme :

1. Définissez le menu (voir plus haut).
2. Saisissez une séquence de commandes interrompant l'exécution du programme (telle que DISP, PROMPT ou HALT).

Le programme s'interrompt jusqu'à réception d'une commande CONT (pression sur [↰][CONT]). Si vous créez un menu personnalisé de saisie de données, vous pouvez y inclure une commande CONT pour relancer automatiquement le programme en appuyant sur la touche de menu correspondante.

Exemple : Le programme suivant appelle la page 1 du menu MODES ANGL et vous demande de définir le mode d'angle. Après

avoir appuyé sur la touche de menu voulue, appuyez sur ◀ CONT pour reprendre l'exécution.

```
« 65 MENU "Select Angle Mode" PROMPT »
```

Exemple : Le programme *PIE*, page 2-57 affecte une touche à la commande CONT dans un programme temporaire.

Exemple : Le programme *MNX*, page 2-26 définit un menu temporaire incluant un programme contenant une commande CONT afin que l'exécution soit reprise automatiquement.

Utilisation de menus pour exécuter des programmes

Vous pouvez utiliser un menu personnalisé pour exécuter d'autres programmes. Il sert d'interface principale pour une application (ensemble de programmes).

Pour créer une application basée sur un menu :

1. Créez une liste de menu personnalisé pour l'application spécifiant les programmes en tant qu'objets-menu.
2. *Facultatif* : créez un programme principal définissant le menu de l'application soit en tant que menu CST, soit en tant que menu temporaire.

Exemple : Le programme *WGT* suivant calcule la masse d'un objet, en unités anglo-saxonnes ou en unités SI, en fonction du poids. *WGT* affiche un menu temporaire, à partir duquel vous pouvez lancer le programme approprié. Chaque programme vous demande de saisir le poids dans le système d'unités voulu, puis calcule la masse. Le menu reste actif jusqu'à ce que vous sélectionniez un autre menu. Vous pouvez donc effectuer autant de calculs que nécessaire.

Saisissez la liste suivante et stockez-la dans LST :

```
{
{ "ENGL" « "ENTER Wt in POUNDS" PROMPT 32.2 / » }
{ "SI" « "ENTER Wt in NEWTONS" PROMPT 9.81 / » }
}
```

' LST STO

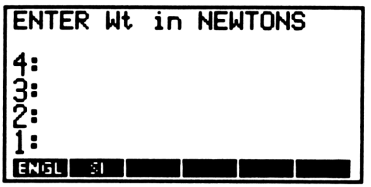
Programme :	**Commentaires :**
« LST TMENU »	Affiche le menu personnalisé stocké dans *LST*.
ENTER ' WGT STO	Stocke le programme dans *WGT*.

Utilisez *WGT* pour calculer la masse d'un objet de 12.5 N. Le programme affiche le menu, puis exécute l'opération.

VAR WGT

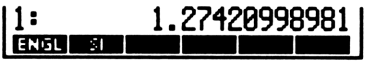
Sélectionnez le système d'unités SI, pour lancer le programme correspondant.

SI



Saisissez le poids, puis relancez le programme.

12.5 ← CONT



Exemple : Le programme *EIZ* suivant crée un menu afin d'émuler l'application HP Solve pour des calculs sur un circuit électrique capacitif. Le programme utilise l'équation $E = IZ$, dans laquelle E correspond à la tension, I au courant et Z à l'impédance.

La tension, le courant et l'impédance étant des nombres complexes, vous ne pouvez utiliser l'application HP Solve pour calculer des solutions. Le menu personnalisé de *EIZ* définit la touche de menu shiftée-gauche pour le calcul ***direct*** d'une solution pour chaque variable et définit les touches non shiftée et shiftée-droite comme fonctions de ***stockage*** et de ***rappel***. Les actions sont analogues à celles de l'application HP Solve. Le menu personnalisé est automatiquement stocké dans *CST* à la place du précédent menu personnalisé (vous pouvez appuyer sur CST pour restaurer ce menu).

Programme :	**Commentaires :**
`«`	
`DEG -15 SF -16 SF`	Spécifie le mode degrés. Arme les indicateurs −15 et −16 pour afficher les nombres complexes en mode polaire. Spécifie le mode d'affichage à 2 Fix.
`2 FIX`	
`{`	Débute la liste du menu personnalisé.
`{ "E" { « 'E' STO »`	Crée la première touche de menu pour E. Action non-shiftée : stocke l'objet dans *E*. Action shiftée-gauche : calcule *I* × *Z*, stocke le résultat dans *E* et l'affiche avec un libellé. Action shiftée-droite : rappelle l'objet dans *E*.
`« I Z * DUP 'E' STO`	
`"E: " SWAP + CLLCD`	
`1 DISP 1 FREEZE »`	
`« E » } }`	
`{ "I" { « 'I' STO »`	Crée la deuxième touche de menu.
`« E Z / DUP 'I' STO`	
`"I: " SWAP + CLLCD`	
`1 DISP 1 FREEZE »`	
`« I » } }`	
`{ "Z" { « 'Z' STO »`	Crée la troisième touche de menu.
`« E I / DUP 'Z' STO`	
`"Z: " SWAP + CLLCD`	
`1 DISP 1 FREEZE »`	
`« Z » } }`	
`}`	Met fin à la liste.
`MENU`	Affiche le menu personnalisé.
`»`	
ENTER ' EIZ STO	Stocke le programme dans *EIZ*.

Pour une alimentation de 10 volts à un angle de phase de 0°, vous mesurez un courant de 0.37 ampères à un angle de phase de 68°. Calculez l'impédance du circuit au moyen de *EIZ*.

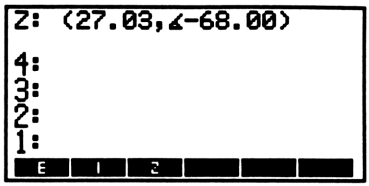
↩ CLEAR VAR EIZ

Saisissez la tension.

◀ ( ) 10 ▶ ∠ 0

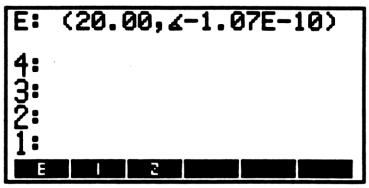
Stockez la tension. Puis, saisissez et stockez le courant. Résolvez pour obtenir l'impédance.

E
◀ ( ) .37 ▶ ∠ 68 I
◀ Z



Rappelez le courant et multipliez-le par deux. Calculez la tension.

▶ I
2 ×
I
◀ E



Appuyez sur ◀ MODES FMT STD et sur NXT MODE ANGL RECT pour restaurer les modes standard et rectangulaire.

Mise hors tension du HP 48 à partir d'un programme

Pour mettre le calculateur hors tension dans un programme :

- Exécutez la commande OFF (menu PRG RUN).

La commande OFF met le HP 48 hors tension. Si un programme exécute OFF, il reprend à la prochaine mise sous tension du calculateur.

2

Exemples de programmation

Les programmes proposés dans ces chapitres présentent des concepts de programmation de base. Ces programmes sont destinés à vous permettre d'améliorer vos compétences en matière de programmation, et à doter votre calculateur de fonctions supplémentaires.

A la fin de chaque programme, la *somme de contrôle* et la taille en octets de ce programme sont indiqués, vous permettant de vérifier si vous avez tapé le programme correctement (la somme de contrôle est un entier binaire qui identifie le programme de manière unique sur la base de son contenu). Pour vous assurer que vous avez tapé correctement le programme, stockez-le en employant son nom, placez-le nom au niveau 1, puis exécutez la commande BYTES (↩ MEMORY BYTES). La somme de contrôle du programme est ainsi renvoyée au niveau 2, et sa taille en octets au niveau 1 (si vous exécutez BYTES avec l'*objet* programme au niveau 1, vous obtenez un total d'octets différent).

Les programmes présentés dans ce chapitre apparaissent également dans les informations en ligne du logiciel Program Development Link, destiné au développement de programmes HP 48 sur des ordinateurs. Ce logiciel vous permet de charger ces programmes à partir des informations en ligne dans votre HP 48 par l'intermédiaire de son port série.

Les exemples proposés dans ce chapitre correspondent à l'hypothèse où le HP 48 se trouve dans son état initial (état par défaut), où vous n'avez donc modifié aucun de ses modes de fonctionnement (pour ramener, le cas échéant, le calculateur à cet état, reportez-vous à "Réinitialisation de la mémoire", au chapitre 5 du *Manuel d'utilisation du HP 48*). Chaque listage de programme de ce chapitre fournit les informations suivantes :

- Une brève description du programme.

- Un diagramme de syntaxe (le cas échéant), présentant les entrées requises et les résultats obtenus.
- Une présentation des techniques de programmation spécifiques utilisées.
- Les autres programmes éventuellement nécessaires.
- Le listage du programme.
- La somme de contrôle et la taille en octets du programme.

Nombres de Fibonacci

Cette section regroupe trois programmes permettant de calculer des nombres de Fibonacci :

- *FIB1* est une fonction-utilisateur, définie *récursivement* (cela signifie que sa procédure de définition contient son propre nom). *FIB1* est un programme court.
- *FIB2* est une fonction-utilisateur, présentant une boucle finie. C'est un programme plus long et plus complexe que *FIB1*, mais plus rapide.
- *FIBT* appelle à la fois *FIB1* et *FIB2*, et calcule le temps d'exécution de chaque sous-programme.

FIB1 et *FIB2* présentent une approche du calcul du *n*ème nombre de Fibonacci F_n, où :

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$$

FIB1 (Nombres de Fibonacci, version récursive)

Niveau 1	→	Niveau 1
n	→	F_n

Techniques employées dans FIB1

- **IFTE (fonction if-then-else).** La procédure de définition de *FIB1* contient la *fonction* conditionnelle IFTE, qui peut prendre son argument dans la pile ou en syntaxe algébrique.
- **Récursion.** La procédure de définition de *FIB1* est élaborée par rapport à *FIB1*, de même que F_n est défini par rapport à F_{n-1} et F_{n-2}.

Listage du programme FIB1

Programme :	**Commentaires :**
`«`	
`→ n`	Définit la variable locale *n*.
`'IFTE(n≤1,`	La procédure de définition, une
`n,`	expression algébrique. Si $n \leq 1$,
`FIB1(n-1)+FIB1(n-2))'`	alors $F_n=n$, autrement
	$F_n=F_{n-1}+F_{n-2}$.
`»`	
ENTER ' FIB1 STO	Stocke le programme dans *FIB1*.

Somme de contrôle : # 41467d (tapez ' FIB1 ↶ MEMORY BYTES)
Octets : 113.5

Exemple : Calculez F_6, puis F_{10} en employant la syntaxe algébrique.

Calculez d'abord F_6.

VAR
6 FIB1

Ensuite, calculez F_{10} en syntaxe algébrique.

' FIB1 ↶ () 10 EVAL

FIB2 (nombres de Fibonacci, version boucle)

Niveau 1	→	Niveau 1
n	→	F_n

Techniques employées dans FIB2

- **IF ... THEN ... ELSE ... END.** *FIB2* emploie la forme programmée de la structure conditionnelle (*FIB1* utilise IFTE).
- **START ... NEXT (boucle finie).** Pour calculer F_n, *FIB2* débute par F_0 et F_1, et répète une boucle pour calculer les valeurs successives de F_i.

Listage du programme FIB2

Programme :	**Commentaires :**
«	
→ n	Crée une structure de variables
«	locales.
IF n 1 ≤	Si $n \leq 1$,
THEN n	alors $F_n = n$;
ELSE	autrement ...
0 1	Place F_0 et F_1 dans la pile.
2 n	De 2 à n effectue la boucle
START	suivante :
DUP	Copie le dernier F (initialement, F_1).
ROT	Extrait le F précédent (initialement, F_0).
+	Calcule le F suivant (initialement, F_2).
NEXT	Répète la boucle.
SWAP DROP	Elimine F_{n-1}.
END	Termine la clause ELSE.
»	Termine la procédure de définition.
»	
ENTER ' FIB2 STO	Stocke le programme dans *FIB2*.

Somme de contrôle : # 51820d (tapez ' FIB2 ↩ MEMORY BYTES)
Octets : 89

Exemple : Calculez F_6 et F_{10}.

Calculez F_6.

VAR
6 FIB2

Calculez F_{10}.

10 FIB2

FIBT (Comparaison du temps d'exécution des programmes)

FIB1 calcule des valeurs intermédiaires F_i plusieurs fois, alors que *FIB2* calcule chaque F_i intermédiaire une seule fois. Par conséquent, *FIB2* est plus rapide. La différence de vitesse augmente avec la taille de *n* parce que le délai requis pour *FIB1* croît de manière exponentielle avec *n*, alors que le délai requis pour *FIB2* ne croît que de manière linéaire avec *n*.

FIBT exécute la commande TICKS afin d'enregistrer le temps d'exécution de *FIB1* et *FIB2* pour une valeur donnée de *n*.

Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
n	→	F_n	FIB1 TIME: *z*	FIB2 TIME: *z*

Techniques employées dans FIBT

- **Programmation structurée.** *FIBT* appelle les deux programmes *FIB1* et *FIB2*.
- **Utilisation par programme de l'horloge du calculateur.** *FIBT* exécute la commande TICKS afin d'enregistrer le début et la fin de chaque sous-programme.
- **Libellé du résultat.** *FIBT* libelle chaque temps d'exécution avec un message descriptif.

Programmes requis

- *FIB1* (page 2-2) calcule F_n en employant la récursion.
- *FIB2* (page 2-4) calcule F_n en employant le bouclage.

Listage du programme FIBT

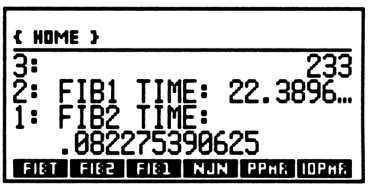
Programme :	Commentaires :
«	
DUP TICKS SWAP FIB1	Copie *n*, puis exécute *FIB1*, en
SWAP TICKS SWAP	enregistrant l'heure de début et d'arrêt.
- B→R 8192 /	Calcule le temps écoulé, le convertit en nombre réel, et convertit ce nombre en secondes. Laisse la réponse renvoyée par *FIB1* au niveau 2.
"FIB1 TIME" →TAG	Libelle le temps d'exécution.
ROT TICKS SWAP FIB2	Exécute *FIB2*, en enregistrant
TICKS	l'heure de début et de fin.
SWAP DROP SWAP	Elimine la réponse renvoyée par
- B→R 8192 /	*FIB2* (*FIB1* a renvoyé la même réponse). Calcule le temps écoulé pour *FIB2* et le convertit en secondes.
"FIB2 TIME" →TAG	Libelle le temps d'exécution.
»	
ENTER ' FIBT STO	Stocke le programme dans *FIBT*.

Somme de contrôle : # 22248d
Octets : 135

Exemple : Calculez F_{13} et comparez le temps d'exécution des deux méthodes.

Sélectionnez le menu VAR et effectuez le calcul.

VAR
13 FIBT



F_{13} est 233. *FIB2* s'exécute beaucoup plus vite que *FIB1* (d'autant plus vite que *n* est élevé). Les délais requis pour les calculs dépendant

du contenu de la mémoire et d'autres facteurs, il se peut que vous n'obteniez pas exactement les temps indiqués ci-dessus.

Affichage d'un entier binaire

Cette section comporte trois programmes :

- *PAD*, utilitaire qui convertit un objet en chaîne en fournissant un affichage justifié à droite.
- *PRESERVE*, utilitaire dont l'emploi est prévu pour des programmes qui modifient l'état du calculateur (mode d'angle, base binaire, etc.).
- *BDISP* affiche un entier binaire en bases HEX, DEC, OCT et BIN. Il appelle *PAD* pour que les nombres affichés soient justifiés à droite, et il appelle *PRESERVE* pour conserver la base binaire.

PAD (Pad avec espaces initiaux)

PAD convertit un objet en chaîne, et, si la chaîne contient moins de 22 caractères, il ajoute des espaces au début de celle-ci, afin d'atteindre ce total.

Lorsqu'une chaîne courte est affichée avec DISP, elle est *justifiée à gauche* : son premier caractère figure à l'extrémité gauche de l'affichage. En ajoutant des espaces au début d'une chaîne courte, *PAD* déplace la chaîne vers la droite. Lorsque la chaîne (espaces initiaux compris) atteint 22 caractères, elle est *justifiée à droite* : son dernier caractère figure à l'extrémité droite de l'affichage. *PAD* n'intervient pas sur les chaînes plus longues.

Niveau 1	→	Niveau 1
objet	→	" *objet*"

Techniques employées dans PAD

- **WHILE ... REPEAT ... END (boucle infinie).** La clause WHILE contient un test qui exécute la clause REPEAT et recommence le

test (si vrai), ou saute la clause REPEAT et met fin à l'exécution (si faux).

- **Operations sur les chaînes.** *PAD* illustre comment convertir un objet sous forme de chaîne, compter le nombre de caractères et combiner deux chaînes.

Listage du programme PAD

Programme :	**Commentaires :**
«	
→STR	Permet de s'assurer que l'objet est bien sous forme de chaîne (les chaînes ne sont pas affectées par cette commande).
WHILE	Est répété si la chaîne contient
DUP SIZE 22 <	moins de 22 caractères.
REPEAT	La clause de boucle ajoute un
" " SWAP +	espace initial.
END	Termine la boucle.
»	
ENTER ' PAD STO	Stocke le programme dans *PAD*.

Somme de contrôle : # 38912d
Octets : 61.5

PAD intervient dans le programme *BDISP*.

PRESERVE (sauvegarder et restaurer l'état précédent)

PRESERVE stocke l'état en cours (indicateur) du calculateur, exécute un programme dans la pile et restaure l'état antérieur.

Niveau 1	→	Niveau 1
« *programme* »	→	*résultat du programme*
'*nom programme*'	→	*résultat du programme*

Techniques employées dans PRESERVE

- **Sauvegarde de l'état de l'indicateur du calculateur.** *PRESERVE* utilise RCLF (*rappel des indicateurs*) pour enregistrer l'état en cours du calculateur dans un entier binaire et STOF (*stockage des indicateurs*) pour restaurer l'état à partir de cet entier binaire.
- **Structure de variables locales.** *PRESERVE* crée une structure de variable locale de manière à supprimer brièvement l'entier binaire de la pile. Sa procédure de définition évalue simplement l'argument programme, puis replace l'entier binaire dans la pile et exécute STOF.
- **Interception d'erreurs.** *PRESERVE* utilise IFERR pour "intercepter" une exécution incorrecte du programme dans la pile et pour restaurer des indicateurs. DOERR affiche l'erreur s'il s'en produit une.

Listage du programme PRESERVE

Programme :	**Commentaires :**
«	
RCLF	Rappelle la liste de deux entiers binaires représentant l'état des 64 indicateurs système et des 64 indicateurs utilisateur.
→ f	Stocke la liste dans la variable locale *f*.
«	Commence la procédure de définition.
IFERR	Commence l'interception d'erreurs.
EVAL	Exécute le programme placé dans la pile en tant qu'argument de niveau 1.
THEN	Si le programme a provoqué une erreur, restaure les indicateurs, affiche l'erreur et interrompt prématurément l'exécution.
f STOF ERRN DOERR	
END	Met fin au sous-programme en cause.
f STOF	Place à nouveau la liste dans la pile, puis restaure l'état de tous les indicateurs.
»	Termine la procédure de définition.
»	
ENTER ' PRESERVE STO	Stocke le programme dans *PRESERVE*.

Somme de contrôle : # 7284d
Octets : 71

PRESERVE intervient dans le programme *BDISP*.

BDISP (Affichage binaire)

BDISP affiche un nombre réel ou binaire en bases HEX, DEC, OCT et BIN.

Niveau 1	→	Niveau 1
# *n*	→	# *n*
n	→	*n*

Techniques utilisées dans BDISP

- **IFERR ... THEN ... END (interception d'erreurs).** Pour prendre en charge des nombres réels, *BDISP* comporte la commande R→B (*réel-binaire*). Toutefois, cette commande provoque une erreur si l'argument est *déjà* un entier binaire. Afin de maintenir l'exécution si une erreur a lieu, la commande R→B est placée à l'intérieur d'une clause IFERR. Aucune action n'étant requise lorsqu'une erreur se produit (étant donné qu'un nombre binaire constitue un argument acceptable), la clause THEN ne comporte pas de commandes.
- **Activation de LASTARG.** Au cas où une erreur se produit, la fonction de récupération LASTARG doit être activée, afin de renvoyer l'argument (le nombre binaire) dans la pile. A cet effet, *BDISP* désarme l'indicateur −55.
- **Boucle FOR ... NEXT (boucle finie avec compteur).** *BDISP* exécute une boucle de 1 à 4, en affichant à chaque fois *n* (le nombre) dans une base différente sur une ligne différente. Le compteur de boucle (appelé *j* dans ce programme) est une variable locale créée par la structure de programme FOR ... NEXT (et non par la commande →) et automatiquement incrémentée par NEXT.
- **Programmes non nommés en tant qu'arguments.** Un programme défini uniquement par ses délimiteurs « et » (non stocké dans une variable) n'est pas automatiquement évalué, mais est placé dans la pile et peut être employé comme argument pour un sous-programme. *BDISP* met en oeuvre cette méthode comme suit :
  - *BDISP* comporte un argument programme principal et un appel à *PRESERVE*. Cet argument programme est placé dans la pile et exécuté par *PRESERVE*.

- *BDISP* contient en outre quatre arguments programme qui "personnalisent" l'action de la boucle. Chacun d'eux comporte une commande permettant de modifier la base binaire, et chaque itération de la boucle évalue l'un de ces arguments.

Lorsque *BDISP* crée une variable locale pour *n*, la procédure de définition est un programme non nommé. Toutefois, ce programme étant une procédure de définition pour une structure de variables locales, il *est* automatiquement exécuté.

Programmes requis

- *PAD* (page 2-8) étend une chaîne à 22 caractères, de manière à ce que DISP la présente justifiée à droite.
- *PRESERVE* (page 2-9) stocke l'état en cours, exécute le programme imbriqué principal et restaure l'état.

Listage du programme BDISP

Programme :	**Commentaires :**
«	
«	Commence le programme imbriqué principal.
DUP	Crée une copie de *n*.
-55 CF	Désarme l'indicateur −55 pour activer LASTARG.
IFERR	Commence l'interception d'erreurs.
R→B	Convertit *n* en entier binaire.
THEN	Si erreur : aucune opération (pas de commande dans la clause THEN).
END	
→ n	Crée une variable locale *n* et
«	commence le programme de définition.
CLLCD	Efface l'affichage.
« BIN »	Programme imbriqué pour BIN.
« OCT »	Programme imbriqué pour OCT.
« DEC »	Programme imbriqué pour DEC.
« HEX »	Programme imbriqué pour HEX.

Programme :	**Commentaires :**
`1 4`	Définit les limites du compteur.
`FOR j`	Commence la boucle avec compteur *j*.
`EVAL`	Exécute l'un des programmes de base imbriqués (en premier celui de HEX).
`n →STR`	Crée une chaîne présentant *n* dans la base en cours.
`PAD`	Remplit la chaîne jusqu'à 22 caractères.
`j DISP`	Affiche la chaîne dans la *j*ème ligne.
`NEXT`	Incrémente *j* et répète la boucle.
`»`	Termine le programme de définition.
`3 FREEZE`	Gèle les zones d'état et de la pile.
`»`	Termine le programme imbriqué principal.
`PRESERVE`	Stocke l'état de l'indicateur en cours, exécute le programme imbriqué principal et restaure l'état.
`»`	
ENTER ' BDISP STO	Stocke le programme dans *BDISP*.

Somme de contrôle : # 18055d
Octets : 191

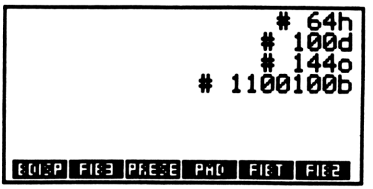
Exemple : Passez à la base DEC, afficher #100 dans toutes les bases et vérifiez si *BDISP* a restauré la base à DEC.

Effacez la pile et sélectionnez le menu MTH BASE. Vérifiez si la base en cours est DEC et entrez # 100.

← CLEAR
MTH BASE
DEC
→ # 100 ENTER

Executez *BDISP*.

(VAR) BDISP

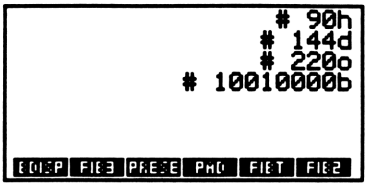


Revenez à l'affichage normal de la pile et vérifiez la base en cours.

(CANCEL)
(MTH) BASE

Même si le programme imbriqué principal a laissé le calculateur en base BIN, *PRESERVE* a toutefois restauré la base DEC. Pour vérifier si *BDISP* fonctionne également pour des nombres réels, essayez avec 144.

(VAR)
144 BDISP



Appuyez sur (CANCEL) pour revenir à l'affichage de la pile.

Médiane de données statistiques

Cette section comporte deux programmes :

- *%TILE* renvoie la valeur du centile spécifié d'une liste.
- *MEDIAN* emploie *%TILE* pour calculer la valeur médiane des données statistiques en cours.

(*%TILE* et *MEDIAN* sont inclus dans le répertoire EXAMPLES de la fonction TEACH. Reportez-vous à la rubrique TEACH au chapitre 3)

%TILE (centile d'une liste)

%TILE trie une liste, puis renvoie la valeur d'un centile spécifié de la liste. Par exemple, en tapant { *liste* } 50 et en appuyant sur %TILE, vous obtenez la valeur médiane (50ème centile) de la liste.

Niveau 2	Niveau 1	→	Niveau 1
{ *liste* }	*n*	→	n^{ime} *centile de la liste triée*

Techniques employées dans %TILE

- **FLOOR et CEIL.** Pour un entier, FLOOR et CEIL renvoient tous deux cet entier ; pour un non entier, FLOOR et CEIL renvoient des entiers successifs qui encadrent ce non entier.
- **SORT.** La commande SORT trie les éléments de la liste par ordre croissant.

Listage du programme %TILE

Programme :	Commentaires :
«	
SWAP SORT	Place la liste au niveau 1 et la trie.
DUP SIZE	Copie la liste, puis en calcule la taille.
1 + ROT 100 / *	Calcule la position du centile spécifié.
→ p	Stocke la position centrale dans la locale variable *p*.
«	Commence la procédure de définition.

Programme :	**Commentaires :**
DUP	Crée une copie de la liste.
P FLOOR GET	Extrait le nombre situé au niveau ou au-dessous de la position centrale.
SWAP	Fait passer la liste au niveau 1.
P CEIL GET	Extrait le nombre situé au niveau ou en dessous de la position centrale.
+ 2 /	Calcule la moyenne des deux nombres.
»	Termine la procédure de définition.

»

ENTER ' %TILE STO — Stocke le programme dans *%TILE*.

Somme de contrôle : # 42718d
Octets : 99

Exemple : Calculez la valeur médiane de la liste { 8 3 1 5 2 }.

← { } 8 3 1 5 2 ENTER
VAR 50 %TILE

MEDIAN (médianes de données statistiques)

MEDIAN renvoie un vecteur contenant les valeurs médianes des colonnes de données statistiques. Notez que pour une liste triée possédant un nombre d'éléments impair, la médiane est la valeur de l'élément central ; pour une liste possédant un nombre d'éléments pair, la médiane est la valeur moyenne des éléments situés juste au-dessus et au-dessous du centre.

Niveau 1	→	**Niveau 1**
	→	[x_1 x_2 ... x_m]

Techniques employées dans MEDIAN

- **Tableaux, listes et éléments de la pile.** *MEDIAN* extrait une colonne de données de *ΣDAT* sous forme de vecteur. Pour convertir le vecteur en liste, *MEDIAN* place les éléments du vecteur dans la pile et les combine de manière à en faire une liste. A partir de cette dernière, la valeur médiane est calculée à l'aide de *%TILE*.

  La valeur médiane de la *m*ième colonne est calculée en premier et la valeur médiane de la première colonne est calculée en dernier. Tandis que chaque valeur médiane est calculée, ROLLD est employé pour la déplacer vers le haut de la pile.

  Une fois toutes les valeurs médianes calculées et positionnées dans la pile, elles sont combinées en un vecteur.

- **FOR ... NEXT (boucle finie avec compteur).** *MEDIAN* utilise une boucle pour calculer la valeur médiane de chaque colonne. Etant donné que les valeurs médianes sont calculées dans un ordre inversé (dernière colonne en premier), le compteur est employé pour inverser leur ordre.

Programme requis

- *%TILE* (page 2-16) trie une liste et renvoie la valeur d'un centile spécifié.

Listage du programme MEDIAN

Programme :	Commentaires :
«	
RCLΣ	Place une copie de la matrice de statistiques en cours ΣDAT dans la pile.
DUP SIZE	Place la liste { *n m* } dans la pile, où *n* est le nombre de lignes de ΣDAT et *m* le nombre de colonnes.
OBJ→ DROP	Place *n* et *m* dans la pile et élimine la taille de la liste.
→ s n m	Crée des variables locales pour *s*, *n* et *m*.
«	Commence la procédure de définition.
'ΣDAT' TRN	Rappelle et transpose ΣDAT. Désormais, *n* est le nombre de colonnes présentes dans ΣDAT et *m* est le nombre de lignes (pour taper le caractère Σ, appuyez sur ▶ Σ, puis supprimez les parenthèses).
1 m	Spécifie la première et la dernière lignes.
FOR j	Pour chaque ligne, effectue ce qui suit :
Σ-	Extrait la dernière ligne située dans ΣDAT. En premier, il s'agit de la *m*ième ligne, ce qui correspond à la *m*ième colonne dans la variable ΣDAT d'origine (pour taper la commande Σ−, appuyez sur ◀ STAT DATA Σ- .)

Programme :	**Commentaires :**
`OBJ→ DROP`	Place les éléments de ligne dans la pile. Elimine la liste d'index { *n* }.
`n →LIST`	Constitue une liste de *n* éléments.
`50 %TILE`	Trie la liste et calcule sa valeur médiane.
`j ROLLD`	Place la valeur médiane dans le niveau correct de la pile.
`NEXT`	Incrémente *j* et répète la boucle.
`m →ARRY`	Combine toutes les valeurs médianes en un vecteur à *m* éléments.
`s STOΣ`	Restaure ΣDAT à sa valeur antérieure.
`»`	Termine la procédure de définition.

`»`

ENTER ' MEDIAN STO — Stocke le programme dans *MEDIAN*.

Somme de contrôle : # 57504d
Octets : 140

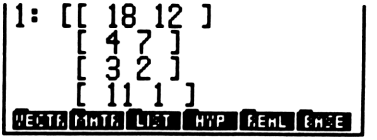
Exemple : Calculez la valeur médiane des données suivantes.

$$\begin{bmatrix} 18 & 12 \\ 4 & 7 \\ 3 & 2 \\ 11 & 1 \\ 31 & 48 \\ 20 & 17 \end{bmatrix}$$

Etant donné qu'il y a deux colonnes de données, *MEDIAN* renvoie un vecteur à deux éléments.

Saisissez la matrice.

(→) (MATRIX)
18 (ENTER) 12 (ENTER) (▼)
4 (ENTER) 7 (ENTER)
3 (ENTER) 2 (ENTER)
11 (ENTER) 1 (ENTER)
31 (ENTER) 48 (ENTER)
20 (ENTER) 17 (ENTER)
(ENTER)



Stockez la matrice dans ΣDAT, et calculez la valeur médiane.

(←) (STAT) DATA
CLΣ Σ+
(VAR) MEDIA

Développement et regroupement complets

Cette section comporte deux programmes :

- *MULTI* répète un programme jusqu'à ce qu'il n'ait plus d'effet sur son argument.
- *EXCO* appelle *MULTI* pour développer et regrouper complètement une expression algébrique.

MULTI (Exécution multiple)

Etant donné un objet et un programme agissant sur cet objet, *MULTI* applique le programme à l'objet jusqu'à ce qu'il ne le modifie plus.

Niveau 2	Niveau 1	→	Niveau 1
objet	« *programme* »	→	$objet_{résultat}$

Techniques employées dans MULTI

- **DO ... UNTIL ... END (boucle infinie).** La clause DO contient les étapes à répéter. La clause UNTIL contient le test qui répète les deux clauses (si faux) ou met fin à l'exécution (si vrai).
- **Programmes en tant qu'arguments.** Les programmes sont fréquemment nommés puis exécutés en appelant leurs noms, mais ils peuvent être placés dans la pile et employés en tant qu'arguments pour d'autres programmes.
- **Evaluation de variables locales.** L'argument programme devant être exécuté plusieurs fois est stocké dans une variable locale.

  Il est utile de stocker un objet dans une variable locale lorsque vous ne savez pas d'avance combien de copies vous seront nécessaires. Un objet stocké dans une variable locale est simplement placé dans la pile lorsque la variable locale est évaluée. *MULTI* utilise le nom de la variable locale pour placer l'argument programme dans la pile, puis exécute EVAL pour lancer le programme.

Listage du programme MULTI

Programme :	Commentaires :
«	
→ P	Crée une variable locale *p* qui contient le programme du niveau 1.
«	Commence la procédure de définition.
DO	Commence la clause de boucle DO.
DUP	Crée une copie de l'objet, à présent au niveau 1.
P EVAL	Applique le programme à l'objet, renvoyant sa nouvelle version.
DUP	Crée une copie du nouvel objet.
ROT	Déplace l'ancienne version vers le niveau 1.
UNTIL	Commence la clause de test DO.
SAME	Vérifie si l'ancienne version et la nouvelle version sont identiques.
END	Termine la structure DO.
»	Termine la procédure de définition.
»	
ENTER ' MULTI STO	Stocke le programme dans *MULTI*.

Somme de contrôle : # 34314d
Octets : 56

MULTI intervient dans l'exemple de programmation suivant.

EXCO (développer et regrouper complètement)

EXCO exécute plusieurs fois EXPAN sur une expression algébrique, jusqu'à ce que cette dernière ne soit plus modifiée, puis il exécute plusieurs fois COLCT, jusqu'à ce que l'expression algébrique ne soit plus modifiée. Dans certains cas, le résultat correspond à un nombre.

Les expressions comportant plusieurs produits de sommes ou des puissances peuvent nécessiter plusieurs itérations de EXPAN pour un

développement complet, prolongeant d'autant le temps d'exécution de *EXCO*.

Niveau 1	→	Niveau 1
'expr.alg.'	→	*'expr.alg.'*
'expr.alg.'	→	*z*

Techniques employées dans EXCO

- **Sous-programmes.** *EXCO* appelle deux fois le programme *MULTI*. Il est plus efficace de créer un programme *MULTI* et d'appeler simplement son nom deux fois que d'écrire deux fois chaque étape de *MULTI*.

Programmes requis

- *MULTI* (page 2-21) exécute plusieurs fois les programmes fournis par *EXCO* comme arguments.

Listage du programme EXCO

Programme :	**Commentaires :**
«	
« EXPAN »	Place un programme dans la pile en tant qu'argument de niveau 1 de *MULTI*. Le programme exécute la commande EXPAN.
MULTI	Exécute EXPAN jusqu'à ce que l'objet algébrique ne soit plus modifié.
« COLCT »	Place un autre programme dans la pile pour *MULTI*. Le programme exécute la commande COLCT.
MULTI	Exécute COLCT jusqu'à ce que l'objet algébrique ne soit plus modifié.
»	
ENTER ' EXCO STO	Stocke le programme dans *EXCO*.

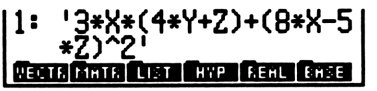
Somme de contrôle : # 48008d
Octets : 65.5

Exemple : Développez et regroupez complètement l'expression :

$$3x(4y + z) + (8x - 5z)^2$$

Saisissez l'expression.

' 3 × X ×
( ) 4 × Y + Z ▶ +
( ) 8 × X − 5 × Z
▶ y^x 2
ENTER



Sélectionnez le menu VAR et lancez le programme.

VAR EXCO

Eléments minimal et maximal d'un tableau

Cette section comporte deux programmes permettant d'extraire l'élément minimal ou maximal d'un tableau :

- *MNX* utilise une boucle (infinie) DO ... UNTIL ... END.
- *MNX2* utilise une boucle (finie) FOR ... NEXT.

MNX (élément minimal ou maximal—version 1)

MNX extrait l'élément minimal ou maximal d'un tableau situé dans la pile.

Niveau 1	→	Niveau 2	Niveau 1
[[*tableau*]]	→	[[*tableau*]]	z_{min} *ou* z_{max}

Techniques utilisées dans MNX

- **DO ... UNTIL ... END (boucle infinie).** La clause DO contient les instructions de tri. La clause UNTIL contient le test d'indicateur système qui détermine s'il faut répéter les instructions de tri.
- **Indicateurs utilisateur et système pour le contrôle de la logique :**
  - L'indicateur *utilisateur* 10 définit le tri : lorsqu'il est armé, *MNX* extrait l'élément maximal ; lorsqu'il est désarmé, *MNX* extrait l'élément minimal. Vous devez déterminer l'état de l'indicateur 10 au début du programme.
  - L'indicateur *système* −64, indicateur de boucle d'index, conditionne la fin du tri. Tant qu'il est désarmé, la boucle de tri continue. Lorsque l'index appelé par GETI boucle sur le premier élément de tableau, l'indicateur −64 est *automatiquement* armé et la boucle de tri prend fin.
- **Structure conditionnelle imbriquée.** Une structure conditionnelle IF ... THEN ... END est imbriquée dans la boucle DO ... UNTIL ... END ; elle détermine :
  - S'il faut maintenir l'élément minimal ou maximal en cours, ou faire du nouveau minimum ou maximum l'élément en cours.

- Le sens de la comparaison des éléments (soit <, soit >) en fonction de l'état de l'indicateur 10.

- **Menu personnalisé.** *MNX* crée un menu personnalisé qui permet de choisir d'effectuer le tri sur l'élément minimal ou maximal. La touche 1, libellée `MAX`, arme l'indicateur 10. La touche 2, libellée `MIN`, désarme l'indicateur 10.

- **Fonction logique.** *MNX* exécute XOR (*OU exclusif*) afin de tester l'état combiné de la valeur relative des deux éléments et l'état de l'indicateur 10.

Listage du programme MNX

Programme :	**Commentaires :**
`«`	

`{{ "MAX"`
`« 10 SF CONT »`
`{ "MIN"`
`« 10 CF CONT »}}`
	Définit le menu d'options. `MAX` arme l'indicateur 10 et poursuit l'exécution. `MIN` désarme l'indicateur 10 et poursuit l'exécution.

`TMENU`
`"Sort for MAX or MIN?"`
`PROMPT`
	Affiche le menu temporaire ainsi qu'un message d'invite.

`1 GETI`	Extrait le premier élément du tableau.
`DO`	Commence la boucle DO.
`ROT ROT GETI`	Place l'index et le tableau aux niveaux 1 et 2, puis extrait le nouvel élément de tableau.
`4 ROLL DUP2`	Fait passer l'élément de tableau minimal ou maximal en cours du niveau 4 au niveau 1, puis copie les deux.

Programme :	**Commentaires :**
`IF`	Teste l'état combiné de la valeur
`  > 10 FS? XOR`	relative des deux éléments et l'état de l'indicateur 10.
`THEN`	Si le nouvel élément est inférieur
`  SWAP`	au maximum en cours ou
`END`	supérieur au minimum en cours, fait passer le nouvel élément au niveau 1.
`DROP`	Elimine l'autre élément de la pile.
`UNTIL`	Commence la clause de test DO.
`  -64 FS?`	Vérifie si l'indicateur −64 est armé (si l'index a atteint la fin du tableau).
`END`	Termine la boucle DO.
`SWAP DROP 0 MENU`	Fait passer l'index au niveau 1 et l'élimine. Restaure le dernier menu.

»

ENTER ' MNX STO — Stocke le programme dans *MNX*.

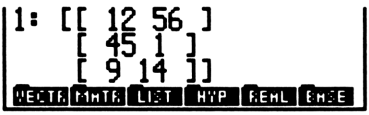
Somme de contrôle : # 57179d
Octets : 210.5

Exemple : Extrayez l'élément maximal de la matrice suivante :

$$\begin{bmatrix} 12 & 56 \\ 45 & 1 \\ 9 & 14 \end{bmatrix}$$

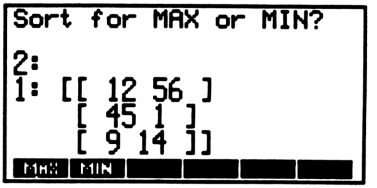
Entrez la matrice.

→ MATRIX
12 ENTER 56 ENTER ▼
45 ENTER 1 ENTER
9 ENTER 14 ENTER
ENTER



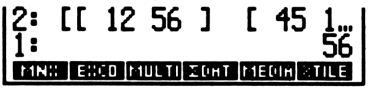
Sélectionnez le menu VAR et exécutez *MNX*.

VAR MNX



Extrayez l'élément maximal.

MAX



MNX2 (élément minimal ou maximal—version 2)

MNX2 extrait l'élément minimal ou maximal d'un tableau placé dans la pile. A cet effet, *MNX2* emploie une approche différente de celle de *MNX* : il exécute OBJ→ pour dissocier le tableau en éléments individuels dans la pile pour le test, au lieu d'exécuter GETI pour incrémenter un index sur la totalité du tableau.

Niveau 1	→	Niveau 2	Niveau 1
[[*tableau*]]	→	[[*tableau*]]	z_{max} *ou* z_{min}

Techniques employées dans MNX2

- **FOR ... NEXT (boucle finie).** La valeur de compteur initiale est 1. La valeur de compteur finale est *nm* − 1, où *nm* est le nombre d'éléments du tableau. La clause de la boucle contient les instructions de tri.
- **Indicateur utilisateur pour le contrôle de la logique.** L'indicateur *utilisateur* 10 définit le tri : lorsqu'il est armé, *MNX2* extrait l'élément maximal et lorsqu'il est désarmé, *MNX2* extrait l'élément minimal. Vous devez déterminer l'état de l'indicateur 10 au début du programme.
- **Structure conditionnelle imbriquée.** Une structure conditionnelle IF ... THEN ... END est imbriquée dans la boucle FOR ... NEXT ; elle détermine :

- S'il faut maintenir l'élément minimal ou maximal en cours, ou faire de l'élément en cours le nouveau minimum ou le nouveau maximum.
- Le sens de la comparaison des éléments (soit <, soit >) en fonction de l'état de l'indicateur 10.

- **Fonction logique.** *MNX2* exécute XOR (*OU exclusif*) pour tester l'état combiné de la valeur relative des deux éléments et l'état de l'indicateur 10.
- **Menu personnalisé.** *MNX2* crée un menu personnalisé qui permet de choisir d'effectuer le trier sur l'élément minimal ou maximal. La touche 1, libellée `MAX`, arme l'indicateur 10. La touche 2, libellée `MIN`, désarme l'indicateur 10.

Listage du programme MNX2

Programme :	Commentaires :
«	
<< "MAX" « 10 SF CONT » < "MIN" « 10 CF CONT »>>	Définit le menu d'options temporaire. `MAX` arme l'indicateur 10 et poursuit l'exécution. `MIN` désarme l'indicateur 10 et poursuit l'exécution.
TMENU "Sort for MAX or MIN?" PROMPT	Affiche le menu temporaire, ainsi qu'un message d'invite.
DUP OBJ→	Copie le tableau. Renvoie les différents éléments de tableau aux niveaux 2 à $nm+1$, et renvoie la liste contenant n et m au niveau 1.
1	Définit la valeur de compteur initiale.
SWAP OBJ→	Convertit la liste en éléments individuels dans la pile.
DROP * 1 -	Elimine la taille de la liste, puis calcule la valeur de compteur finale ($nm-1$).
FOR n	Commence la boucle FOR ... NEXT.
DUP2	Enregistre les éléments de tableau à tester (en premier, les deux derniers éléments). Utilise le dernier élément de tableau comme le minimum ou le maximum en cours.

Programme :	**Commentaires :**
`IF`	Teste l'état combiné de la
`> 10 FS? XOR`	valeur relative des deux éléments et l'état de l'indicateur 10.
`THEN`	Si le nouvel élément est
`SWAP`	inférieur au maximum en
`END`	cours ou supérieur au minimum en cours, fait passer ce nouvel élément au niveau 1.
`DROP`	Elimine l'autre élément de la pile.
`NEXT`	Termine la boucle FOR ... NEXT.
`0 MENU`	Restaure le dernier menu.
»	
ENTER ' MNX2 STO	Stocke le programme dans *MNX2*.

Somme de contrôle : # 12277d
Octets : 200.5

Exemple : Utilisez *MNX2* pour extraire l'élément minimal de la matrice provenant de l'exemple précédent :

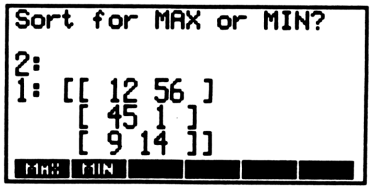
$$\begin{bmatrix} 12 & 56 \\ 45 & 1 \\ 9 & 14 \end{bmatrix}$$

Saisissez la matrice (ou récupérez-la de l'exemple précédent).

→ MATRIX
12 ENTER 56 ENTER ▼
45 ENTER 1 ENTER
9 ENTER 14 ENTER
ENTER

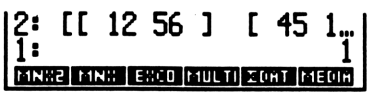
Sélectionnez le menu VAR et exécutez *MNX2*.

(VAR) MNX2



Extrayez l'élément minimal.

MIN



Application d'un programme à un tableau

APLY se sert du traitement des listes pour transformer chaque élément d'un tableau selon la procédure voulue. Le tableau d'entrée doit être numérique, mais le "tableau" de résultat peut être symbolique. En réalité, les tableaux ne peuvent pas contenir d'objets symboliques, aussi utilise-t-on une convention pour les représenter sous forme de "pseudo-tableaux" symboliques. Chaque ligne d'éléments est regroupée en une liste unique, et l'ensemble des lignes est regroupé en une autre liste. Par exemple, un pseudo-tableau 2 x 2 se présente ainsi :

{ { *élément*$_{11}$ *élément*$_{12}$ }
{ *élément*$_{21}$ *élément*$_{22}$ } }

La procédure appliquée à chaque élément doit être un programme qui prend exactement un argument (l'élément) et renvoie exactement un résultat (l'élément transformé).

Niveau 2	Niveau 1	→	Niveau 1
[*tableau*]	« *programme* »	→	[[*tableau*]] *ou* {{ *tableau* }}

Techniques employées dans APLY

- **Manipulation de méta-objets.** Les *méta-objets* sont des objets composites, tels que des tableaux et des listes, qui ont été dissociés dans la pile. APLY illustre plusieurs approches de manipulation des éléments et des dimensions de ces objets.
- **Traitement des listes.** APLY se sert de DOSUBS (même si DOLIST peut également être utilisé) pour effectuer la transformation réelle d'éléments de tableau.
- **Utilisation d'une structure IFERR ... THEN ... ELSE ... END.** Le cas du pseudo-tableau symbolique est entièrement traité au sein d'une structure d'interception d'erreurs, déclenchée lorsque la commande →ARRY génère une erreur en présence d'éléments symboliques.
- **Utilisation d'indicateurs.** L'indicateur utilisateur 1 est employé pour effectuer un suivi dans le cas où le tableau en entrée est un vecteur.

Listage du programme APLY

Programme :	**Commentaires :**
`«`	
`→ a p`	Stocke le tableau et le programme dans des variables locales.
`«`	Commence la structure de variables locales principale.
`1 CF`	Vérifie si l'indicateur 1 est désarmé afin de commencer la procédure.
`a DUP SIZE`	Extrait les dimensions du tableau.
`DUP SIZE`	Détermine si le tableau est un
`IF 1 ==`	vecteur.
`THEN 1 SF 1 +`	Si le tableau est un vecteur, arme l'indicateur 1 et ajoute une seconde dimension en traitant le vecteur comme une matrice $n \times 1$.

Programme :	**Commentaires :**
`SWAP OBJ→ OBJ→ DROP`	Dissocie le vecteur d'origine, en laissant le total d'éléments, *n*, au niveau 1.
`1 + ROLL`	Fait remonter les éléments dans la pile et passer les dimensions "matricielles" du vecteur au niveau 1.
`ELSE DROP2 a OBJ→`	Si le tableau est une matrice, efface la pile et décompose la matrice en ses différents éléments, en laissant la liste de dimensions au niveau 1.
`END DUP OBJ→ DROP *`	Duplique la liste de dimensions et calcule le nombre total d'éléments.
`SWAP OVER 2 +` `ROLLD →LIST`	Fait remonter le total d'éléments et combine tous les éléments en une liste. Notez que les éléments de la liste sont classés par ordre croissant de lignes.
`1 p DOSUBS`	Rappelle le programme et l'utilise comme argument pour DOSUBS (DOLIST fonctionne dans ce cas également). Le résultat est une liste d'éléments transformés.
`OBJ→ 1 + ROLL`	Dissocie la liste de résultats et fait passer les dimensions de tableau au niveau 1.

Programme :	Commentaires :
IFERR	Commence la structure d'interception d'erreurs. Son objectif est de déterminer et de traiter les cas où la liste de résultats contient des éléments symboliques.
IF 1 FS?	Le tableau d'origine était-il un vecteur ?
THEN OBJ→ DROP →LIST	Si oui, élimine la deuxième dimension (1) de la liste de dimensions.
END →ARRY	Convertit les éléments en un tableau aux dimensions données. Si des éléments symboliques sont présents, une erreur est générée et la clause d'erreur qui suit est exécutée.
THEN	Commence la clause d'erreur.
OBJ→	Place les dimensions de tableau aux niveaux 2 et 1. Si le tableau est un vecteur, le niveau 1 contient un 1.
IF 1 FC?C	Le tableau d'origine est-il une matrice ? Désarme l'indicateur 1 après avoir effectué le test.
THEN DROP	Elimine le nombre d'éléments de la matrice.
END → n m	Stocke les dimensions de tableau dans des variables locales.
« 1 n	Commence une structure de
FOR i	variables locales et lance une boucle FOR..NEXT pour chaque ligne.
m →LIST	Rassemble un groupe d'éléments pour en faire une ligne (une liste).

Programme :	**Commentaires :**
`'m*(n-i)+i' EVAL`	Calcule le nombre d'éléments à
`ROLLD`	faire remonter de manière à regrouper la ligne suivante.
`NEXT`	Répète la boucle pour la ligne suivante.
`n →LIST`	Rassemble les lignes en une liste, formant une liste de listes (pseudo-tableau symbolique).
`»`	Ferme la structure de variables
`END 1 CF`	locales et termine la structure IFERR..THEN..END. Désarme l'indicateur 1 avant de quitter le programme.
`»`	
`»`	
ENTER ' APLY STO	Stocke le programme dans *APLY*.

Somme de contrôle : # 49768d
Octets : 319

Exemple : Appliquez la fonction , $f(x) = Ax^3 - 7$ à chaque élément x du vecteur [3 -2 4 -8 1].

◀ [ ] 3 SPC 2 +/- 4 SPC 8 +/- 1 ENTER
◀ « » 3 y^x A × 7 − ENTER
VAR APLY

Conversion entre des bases

nBASE convertit un nombre décimal positif (x) en une chaîne identifiée représentant la valeur équivalente exprimée en une base numérique différente (b). x et b doivent être des nombres réels. *nBASE* arrondit automatiquement les deux arguments à l'entier le plus proche.

Niveau 2	Niveau 1	→	Niveau 1
x	*b*	→	*x* base*b*: "*chaîne*"

Techniques employées dans nBASE

- **Concaténation de chaînes et manipulation de caractères.** nBASE s'appuie sur plusieurs techniques de manipulation de chaînes et de caractères pour créer la chaîne résultante.
- **Résultat identifié.** nBASE identifie la chaîne résultante avec ses propres arguments de sorte qu'elle constitue un enregistrement complet de la commande.
- **Boucles infinies.** nBASE fonctionne en majeure partie avec des boucles infinies : les boucles DO..UNTIL..END et WHILE..REPEAT..END.

Listage du programme nBASE

Programme :	**Commentaires :**
«	
1 CF 0 RND SWAP 0 RND	Désarme l'indicateur 1 et arrondit les deux arguments en entiers.
→ b n	Stocke la base et le nombre dans des variables locales.
«	Commence la structure de variables locales externe.
n LOG b LOG /	Calcule le rapport des logarithmes communs du nombre et de la base.
10 RND	Arrondit le résultat afin de supprimer l'incertitude sur la dernière décimale.
IP n 0	Calcule la partie entière du rapport logarithmique, rappelle le nombre d'origine et initialise la variable de compteur *k* devant être utilisée dans la boucle DO..UNTIL.
→ i m k	Stocke les valeurs dans des variables locales.

Programme :	**Commentaires :**
`«`	Commence la structure de
`""`	variables locales interne, entre
`DO`	une chaîne vide et commence la boucle DO..UNTIL..END.
`'m' EVAL b i`	Calcule la valeur décimale de la
`'k' EVAL - ^`	$(i-k)$ième position de la chaîne.
`DUP2 MOD`	Effectue une copie des arguments et calcule la valeur décimale restante qui doit être prise en compte par les autres positions.
`IF DUP 0 ==`	Le reste est-il égal à zéro *et*
`'m' EVAL b ≥`	$m \geq b$?
`AND`	
`THEN 1 SF`	Si le test est vrai, arme l'indicateur 1.
`END 'm' STO`	Stocke le reste dans m.
`/ IP`	Calcule le nombre de fois où la valeur de la position en cours passe dans la valeur décimale restante. Il s'agit du "chiffre" qui se trouve dans la position en cours.
`IF DUP 10 ≥`	Le "chiffre" est-il ≥ 10 ?
`THEN 55 + CHR`	Convertit alors le chiffre en caractère alphabétique (tel que A, B, C ...)
`END +`	Ajoute le chiffre à la chaîne de
`'k' 1 STO+`	résultat en cours et incrémente la variable de compteur k.

Programme :	**Commentaires :**
`UNTIL 'm' EVAL 0 ==`	Répète la boucle DO..UNTIL jusqu'à ce que $m = 0$ (c'est-à-dire que la totalité de la valeur décimale ait été prise en compte).
`END`	L'indicateur 1 est-il armé ?
`IF 1 FS?C`	Désarme l'indicateur après le test.
`THEN 0 +`	Ajoute alors un zéro de garde à la chaîne résultante.
`WHILE i 'k' EVAL - 0 ≠`	Commence la boucle WHILE..REPEAT pour déterminer si des zéros de garde supplémentaires sont nécessaires. La boucle se répète tant que $i \neq k$.
`REPEAT 0 + 1 'k' STO`	Ajoute un zéro de garde supplémentaire et incrémente k avant de répéter la clause de test.
`END`	Termine la boucle
`END`	WHILE..REPEAT..END, la
`»`	structure IF..THEN..END, et la structure de variables locales interne.
`" base" b +`	Termine la structure
`n SWAP + →TAG`	IF..THEN..ELSE..END la plus externe, crée la chaîne d'identification et identifie la chaîne résultante avec les arguments d'origine.
`»`	
`»`	
ENTER ' nBASE STO	Stocke le programme dans *nBASE*.

Somme de contrôle : # 19700d
Octets : 417.5

Exemple : Convertissez 1000_{10} en base 23.

1000 ENTER 23 VAR NBASE

Vérification des arguments d'un programme

Les deux utilitaires de cette section vérifient si l'argument d'un programme correspond à un type d'objet correct.

- *NAMES* vérifie si un argument liste contient exactement deux noms.
- *VFY* vérifie si l'argument est un nom ou une liste contenant exactement deux noms. Il appelle *NAMES* si l'argument est une liste.

Vous pouvez modifier ces utilitaires pour vérifier d'autres types d'objets et leur contenu.

NAMES (Vérification du nombre de noms dans la liste)

Si l'argument d'un programme est une liste (comme déterminé par *VFY*), *NAMES* vérifie si elle contient exactement deux noms. Si ce n'est pas le cas, un message d'erreur s'affiche dans la zone d'état et l'exécution du programme est arrêtée prématurément.

Niveau 1	→	Niveau 1
{ *liste correcte* }	→	
{ *liste incorrecte* }	→	(*message d'erreur dans la zone d'état*)

Techniques employées dans NAMES

- **Structures conditionnelles imbriquées.** La structure conditionnelle externe vérifie s'il y a deux objets dans la liste. Si oui, la structure conditionnelle interne vérifie si ces deux objets sont des noms.
- **Fonctions logiques.** *NAMES* utilise la commande AND dans la structure conditionnelle interne pour déterminer si *les deux* objets

sont des noms, et la commande NOT pour afficher le message d'erreur si ce n'est pas le cas.

Listage du programme NAMES

Programme :	Commentaires :
«	
IF	Commence la structure conditionnelle externe.
OBJ→	Renvoie les *n* objets de la liste aux niveaux 2 à (*n* +1), et la taille de la liste *n* au niveau 1.
DUP 2 SAME	Copie la taille de la liste et vérifie si elle est égale à 2.
THEN	Si oui, déplace les objet vers les
DROP	niveaux 1 et 2, et commence la
IF	structure conditionnelle interne.
TYPE 6 SAME	Vérifie si le premier objet est un nom : renvoie 1 si tel est le cas, autrement, 0.
SWAP TYPE 6 SAME	Déplace le deuxième objet vers le niveau 1, puis vérifie s'il s'agit d'un nom (renvoie 1 ou 0).
AND	Combine les résultats de test : renvoie 1 si les deux tests étaient vrais, autrement, renvoie 0.
NOT	Inverse le résultat du test final.
THEN	Si les objets ne sont pas tous les
"List needs two names"	deux des noms, affiche un
DOERR	message d'erreur et arrête prématurément l'exécution.
END	Termine la structure conditionnelle interne.

Programme :	**Commentaires :**
`ELSE`	Si la taille de liste n'est pas égale à 2, l'élimine, affiche un message d'erreur et arrête prématurément l'exécution.
`DROPN`	
`"Illegal list size"`	
`DOERR`	
`END`	Termine la structure conditionnelle externe.
»	
ENTER ' NAMES STO	Stocke le programme dans *NAMES*.

Somme de contrôle : # 40666d
Octets : 141.5

NAMES intervient dans le programme *VFY*.

VFY (vérification de l'argument d'un programme)

VFY vérifie si l'argument placé dans la pile est un nom ou une liste qui contient exactement deux noms.

Niveau 1	→	Niveau 1
'nom'	→	*'nom'*
{ liste corr }	→	*{ liste corr }*
{ liste incorr }	→	*{ liste incorr } (et msge err en zone état)*
objet incorr	→	*objet incorr (et msge err en zone état)*

Techniques employées dans VFY

- **Utilitaires.** *VFY* présente peu d'utilité en soi. En revanche, il peut être utilisé avec des modifications mineures par d'autres programmes, afin de vérifier si des types d'objets donnés constituent des arguments corrects.
- **Structure conditionnelle CASE ... END.** *VFY* utilise une structure de ce type pour déterminer si l'argument est une liste ou un nom.
- **Programmation structurée.** Si l'argument est une liste, *VFY* appelle *NAMES* pour vérifier si elle contient exactement deux noms.

- **Structure de variables locales.** *VFY* stocke son argument dans une variable locale de manière à ce qu'il puisse être transféré à *NAMES* si nécessaire.
- **Fonction logique.** *VFY* utilise NOT pour afficher un message d'erreur.

Programmes requis

- *NAMES* (page 2-42) vérifie si un argument liste contient exactement deux noms.

Listage du programme VFY

Programme :	Commentaires :
«	
DUP	Copie l'argument d'origine à laisser dans la pile.
DTAG	Supprime tous les libellés éventuels de l'argument devant être testé par la suite.

Programme :	**Commentaires :**
`→ argm`	Stocke l'argument dans la variable locale *argm*.
`«`	Commence la procédure de définition.
`CASE`	Commence la structure CASE.
`argm TYPE 5 SAME`	Vérifie si l'argument est une
`THEN`	liste. Si tel est le cas, replace
`argm NAMES`	l'argument dans la pile et
`END`	appelle *NAMES* afin de vérifier si la liste est correcte, puis quitte la structure CASE.
`argm TYPE 6 SAME NOT`	Vérifie si l'argument n'est *pas*
`THEN`	un nom. Si tel est le cas,
`"Not name or list"`	affiche un message d'erreur et
`DOERR`	arrête prématurément
`END`	l'exécution.
`END`	Termine la structure CASE.
`»`	Termine la procédure de définition.

`»`

ENTER ' VFY STO — Stocke dans *VFY*.

Somme de contrôle : # 36796d
Octets : 139.5

Exemple : Exécutez *VFY* afin de vérifier la validité de l'argument nom *BEN* (l'argument est correct et il est simplement renvoyé dans la pile).

' BEN ENTER
VAR VFY

Exemple : Exécutez *VFY* pour vérifier la validité de l'argument liste { *BEN JEFF SARAH* }. Utilisez le nom de l'exemple précédent, saisissez les noms *JEFF* et *SARAH*, puis convertissez ces trois noms en une liste.

' JEFF ENTER
' SARAH ENTER
3 PRG LIST →LIST

Executez *VFY*. Etant donné que la liste contient trop de noms, le message d'erreur est affiché et l'exécution est arrêtée prématurément.

VAR VFY

Procédures de conversion d'une expression algébrique en RPN

Cette section contient un programme, →*RPN*, qui convertit une expression algébrique en une série (liste) d'objets classés dans un ordre RPN équivalent. Notez que →RPN est un programme fourni avec la commande TEACH. Vous pouvez le trouver dans le répertoire EXAMPLES en appuyant sur EXAM PRGS →RPN.

Niveau 1	→	Niveau 1
'symb'	→	{ *objets* }

Techniques employées dans →RPN

- **Récursion.** Le programme →RPN s'appelle lui-même en tant que sous-programme. Cette puissante technique revient à appeler un autre sous-programme tant que la pile contient les arguments corrects avant que le programme ne s'appelle lui-même. Dans ce

cas, l'argument de niveau 1 est testé en premier lieu pour vérifier s'il s'agit bien d'une expression algébrique avant que →RPN ne soit rappelé.

- **Vérification du type d'objet.** →RPN utilise une structure de branchement conditionnel qui dépend du type de l'objet de niveau 1.
- **Structures de programme imbriquées.** →RPN imbrique des structures IF ... THEN ... END dans des boucles FOR ... NEXT à l'intérieur d'une structure IF ... THEN ... ELSE ... END.
- **Concaténation de liste.** La liste résultante d'objets, en ordre RPN, est créée avec la commande +, qui permet d'ajouter séquentiellement des éléments supplémentaires à une liste. Il s'agit d'une technique pratique pour le regroupement de résultats issus d'une procédure en boucle.

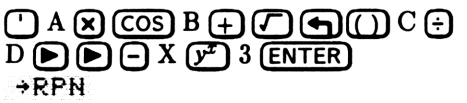
→Listage du programme RPN

Programme :	**Commentaires :**
«	
OBJ→	Isole l'expression.
IF OVER	Si le total d'arguments est non
THEN → n f	nul, stocke le total et la fonction.
«	Commence la procédure de définition de variables locales.
1 n	Commence la boucle FOR ...
FOR i	NEXT, qui convertit tous les arguments algébriques éventuels en listes.
IF DUP TYPE 9 SAME	Vérifie si l'argument est une expression algébrique.
THEN →RPN	Si c'est le cas, le convertit d'abord en liste.
END n ROLLD	Descend d'un niveau dans la pile pour la préparer pour le nouvel argument.
NEXT	Répète la boucle pour l'argument suivant.
IF DUP TYPE 5 ≠	Vérifie si l'objet de niveau 1 est une liste.
THEN 1 →LIST	Si ce n'est pas le cas, le convertit en liste.
END	Termine la structure IF ... THEN ... END.

Programme :	Commentaires :
`IF n 1 >`	Vérifie s'il y a plusieurs arguments.
`THEN 2 n`	Combine tous les arguments en
`START +`	une liste.
`NEXT`	
`END f +`	Ajoute la fonction à la fin de la liste.
`»`	Termine la procédure de définition de variables locales.
`ELSE 1 →LIST SWAP DROP`	Pour les fonctions dépourvues d'arguments, convertit en une liste simple.
`END`	Termine la structure IF ... THEN ... ELSE ... END.
`»`	

Somme de contrôle : # 28598d
Octets : 189.5

Exemple : Convertissez l'expression algébrique suivante en une série d'objets en syntaxe RPN : `'A*COS(B+√(C/D))-X^3'`.



Fonctions de Bessel

Cette section contient un programme, *BER*, qui calcule la partie réelle $\mathrm{Ber_n(x)}$ de la fonction de Bessel $J_n(xe^{3\pi i/4})$. Lorsque $n = 0$,

$$\mathrm{Ber}(x) = 1 - \frac{(x/2)^4}{2!^2} + \frac{(x/2)^8}{4!^2} - \cdots$$

Niveau 1	→	Niveau 1
z	→	*Ber(z)*

Techniques employées dans BER

- **Structure de variables locales.** Au niveau externe, *BER* comporte uniquement une structure de variables locales et possède donc deux propriétés d'une fonction-utilisateur : il peut prendre des arguments numériques ou symboliques dans la pile, ou prendre des arguments en syntaxe algébrique. Toutefois, étant donné que *BER* emploie une boucle DO ... UNTIL ... END, sa procédure de définition est un *programme* (les structures en boucle ne sont pas autorisées dans les expressions algébriques). Par conséquent, contrairement aux fonctions-utilisateur, *BER* n'est pas différenciable.
- **Boucle DO ... UNTIL ... END (boucle infinie avec compteur).** *BER* calcule des termes successifs de la série à l'aide d'une variable de compteur. Lorsque le nouveau terme ne diffère pas du terme précédent dans la mesure de la précision à 12 chiffres du calculateur, la boucle se termine.
- **Structures de variables locales imbriquées.** La structure externe est cohérente avec les besoins d'une fonction-utilisateur. La structure interne permet le stockage et le rappel de paramètres clés.

Listage du programme BER

Programme :	Commentaires :
«	
→ x	Crée une variable locale x.
«	Commence la procédure de définition extérieure.
'x/2' →NUM 2 1	Entre $x/2$, la première valeur de
→ xover2 j sum	compteur, et le premier terme de la série, puis crée des variables locales.
«	Commence la procédure de définition intérieure.
DO	Commence la boucle.
sum	Rappelle l'ancienne somme et
'sum+(-1)^(j/2)*	calcule la nouvelle somme.
xover2^(2*j)/SQ(j!)'	
EVAL	
2 'j' STO+	Incrémente le compteur.
DUP 'sum' STO	Stocke la nouvelle somme.
UNTIL	Termine la clause de boucle.
==	Teste les ancienne et nouvelle sommes.
END	Termine la boucle.
sum	Rappelle la somme.
»	Termine la procédure de définition interne.
»	Termine la procédure de définition externe.
»	
ENTER ' BER STO	Stocke le programme dans *BER*.

Somme de contrôle : # 36388d
Octets : 200.5

Exemple : Calculez Ber(3).

VAR
3 BER

Calculez Ber(2) en syntaxe algébrique.

Animation de polynômes de Taylor successifs

Cette section contient trois programmes qui manipulent des objets graphiques afin d'afficher une séquence de polynômes de Taylor pour la fonction sinus.

- *SINTP* trace une courbe sinusoïdale, et enregistre le tracé dans une variable.
- *SETTS* superpose des tracés de polynômes de Taylor successifs sur le tracé de la courbe sinusoïdale provenant de *SINTP*, et enregistre les objets graphiques résultants dans une liste.
- *TSA* fait appel à la commande ANIMATE pour afficher en séquence chaque objet graphique issu de la liste créée dans *SETTS*.

SINTP (conversion d'un tracé en objet graphique)

SINTP trace une courbe sinusoïdale, renvoie le tracé dans la pile sous forme d'objet graphique et stocke ce dernier dans une variable. Assurez-vous que votre calculateur est en mode radians.

Techniques employées dans SINTP

- **Utilisation par programme des commandes PLOT.** *SINTP* fait usage de commandes PLOT pour créer et afficher un objet graphique.

Listage du programme SINTP

Programme :	Commentaires :
«	
'SIN(X)' STEQ	Stocke l'expression pour $\sin x$ dans *EQ*.
FUNCTION '-2*π' →NUM	Définit le type de tracé et les
DUP NEG XRNG	plages d'affichage des axes x et y.
-2 2 YRNG	
ERASE DRAW	Efface *PICT*, puis trace l'expression.
PICT RCL 'SINT' STO	Rappelle l'objet graphique résultant et le stocke dans *SINT*.
»	
ENTER ' SINTP STO	Stocke le programme dans *SINTP*.

Somme de contrôle : # 1971d
Octets : 91.5

SINTP intervient dans le programme *TSA*.

SETTS (superposition de polynômes de Taylor)

SETTS superpose des polynômes de Taylor successifs sur une courbe sinusoïdale et stocke chaque objet graphique dans une liste.

Techniques employées dans SETTS

- **Programmation structurée.** *SETTS* appelle *SINTP* afin de créer une courbe sinusoïdale et la convertir en un objet graphique.
- **FOR ... STEP (boucle finie).** *SETTS* calcule des polynômes de Taylor successifs pour la fonction sinus dans une boucle finie. Le compteur de boucle est employé pour la valeur de l'ordre de chaque polynôme.
- **Utilisation par programme des commandes PLOT.** *SETTS* dessine un tracé de chaque polynôme de Taylor.
- **Manipulation d'objets graphique.** *SETTS* convertit chaque tracé de polynôme de Taylor en un objet graphique. Ensuite, il exécute

+ pour combiner chaque objet graphique avec la courbe sinusoïdale stockée dans *SINT*, créant neuf nouveaux objets graphiques, constituant chacun la superposition d'un polynôme de Taylor sur une courbe sinusoïdale. *SETTS* place ensuite ces neuf objets graphiques, ainsi que l'objet graphique de la courbe sinusoïdale lui-même, dans une liste.

Listage du programme SETTS

Programme :	**Commentaires :**
«	
`SINTP`	Trace une courbe sinusoïdale et stocke l'objet graphique dans *SINT*.
`1 17 FOR n`	Définit la plage de la boucle FOR à l'aide de la variable *n*.
`'SIN(X)' 'X' n TAYLR`	Trace le polynôme de Taylor
`STEQ ERASE DRAW`	d'ordre *n*.
`PICT RCL SINT +`	Renvoie le tracé dans la pile en tant qu'objet graphique et exécute + pour superposer la courbe sinusoïdale issue de *SINT*.
`2 STEP`	Incrémente le compteur de boucle *n* de 2 et répète la boucle.
`SINT`	Place l'objet graphique de la
`10 →LIST`	courbe sinusoïdale dans la pile,
`'TSL' STO`	puis crée une liste contenant cet objet, ainsi que les neufs objets créés dans la boucle. Stocke la liste dans *TSL*.
»	
ENTER ' SETTS STO	Stocke le programme dans *SETTS*.

Somme de contrôle : # 28102d
Octets : 138.5

SETTS intervient dans le programme *TSA*.

TSA (animation des polynômes de Taylor)

TSA affiche successivement chaque objet graphique créé dans *SETTS*.

Techniques employées dans TSA

- **Passage d'une variable globale.** L'exécution de *SETTS* dure plusieurs minutes, aussi *TSA* ne l'appelle pas. Vous devez d'abord exécuter *SETTS* pour créer la variable globale *TSL* contenant la liste des objets graphiques. *TSA* exécute ensuite cette variable globale pour placer la liste dans la pile.
- **FOR ... NEXT (boucle finie).** *TSA* exécute une boucle finie pour afficher successivement les objets graphiques de la liste.

Listage du programme TSA

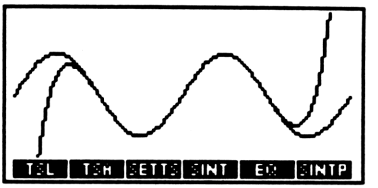
Programme :	**Commentaires :**
«	
TSL OBJ→	Place la liste *TSL* dans la pile et la convertit en 10 objets graphiques avec le total de liste.
{ { #0 #0 } .5 0 } +	Configure les paramètres pour ANIMATE.
ANIMATE	Affiche les graphiques en séquence.
11 DROPN	Supprime les objets graphiques et le total de liste de la pile.
»	
ENTER ' TSA STO	Stocke le programme dans *TSA*.

Somme de contrôle : # 59350d
Octets : 96.5

Exemple : Exécutez *SETTS* et *TSA* pour créer et afficher en séquence une série d'approximations polynômiales de Taylor de la fonction sinus.

Activez le mode radians et exécutez *SETTS* afin de créer la liste d'objets graphiques (l'exécution de *SETTS* dure plusieurs minutes). Ensuite, exécutez *TSA* pour afficher chaque tracé en séquence. L'affichage montre *TSA* en cours d'exécution.

(↩) (RAD) (si nécessaire)
(VAR) SETTS
TSA



Appuyez sur (CANCEL) pour arrêter l'animation. Appuyez sur (↩) (RAD) pour restaurer le mode degrés.

Utilisation par programme des commandes statistiques et de traçage

Cette section décrit un programme *PIE* que vous pouvez employer pour tracer des graphiques circulaires. *PIE* vous invite à fournir des données à une seule variable, les stocke dans la matrice statistique ΣDAT, puis dessine un graphique circulaire libellé présentant chaque point de données en pourcentage du total.

Techniques employées dans PIE

- **Utilisation par programme des commandes PLOT.** *PIE* exécute XRNG et YRNG pour définir les plages d'affichage des axes x et y en unités-utilisateur, et exécute ARC et LINE pour dessiner le cercle et les sections.
- **Utilisation par programme de matrices et de commandes statistiques.**
- **Manipulation d'objets graphiques.** *PIE* rappelle *PICT* dans la pile et exécute GOR pour fusionner le libellé de chaque section avec le tracé.
- **FOR ... NEXT (boucle finie).** Chaque section est calculée, dessinée et libellée dans une boucle finie.

- **Structure CASE ... END.** Pour éviter de les superposer au cercle, les libellés sont décalés à partir du point médian de l'arc de la section. Ce décalage dépend de la position de la section dans le cercle. La structure CASE ... END le détermine en fonction de cette position.
- **Sauvegarde de l'état de l'indicateur du calculateur.** Avant de spécifier le mode radians, *PIE* enregistre l'état de l'indicateur en cours dans une variable locale, puis le restaure à la fin du programme.
- **Structures de variables locales imbriquées** A différents stades du processus, des résultats intermédiaires sont enregistrés dans des variables locales pour pouvoir être facilement rappelés le cas échéant.
- **Menu temporaire pour la saisie des données.**

Listage du programme PIE

Programme :	Commentaires :
`«`	
`RCLF → flags`	Rappelle l'état d'indicateur en cours et le stocke dans des *indicateurs* de variable.
`«`	
`RAD`	Active le mode radians.
`{{ "SLICE" Σ+ }`	Définit le menu d'entrée : La touche 1 exécute Σ+ pour stocker chaque point de données dans ΣDAT, la touche 3 efface ΣDAT, et la touche 6 continue l'exécution du programme après l'entrée des données.
`{ }`	
`{ "CLEAR" CLΣ }`	
`{ } { }`	
`{ "DRAW" CONT }}`	
`TMENU`	Affiche le menu temporaire.
`"Key values into`	Demande de fournir des entrées.
`SLICE,▪DRAW`	▪ représente le caractère de saut de ligne (→ ↵) une fois que vous avez entré le programme dans la pile.
`restarts program."`	
`PROMPT`	
`ERASE 1 131 XRNG`	Efface le *PICT* en cours et définit les paramètres de tracé.
`1 64 YRNG CLLCD`	
`"Please wait...▪`	Affiche le message indiquant que le tracé est en cours.
`Drawing Pie Chart"`	
`1 DISP`	
`(66,32) 20 0 6.28`	Dessine le cercle.
`ARC`	
`PICT RCL →LCD`	Affiche le cercle vide.
`RCLΣ TOT /`	Rappelle la matrice de données statistiques, calcule des totaux et les proportions.
`DUP 100. *`	Convertit les proportions en pourcentages.
`→ prcnts`	Stocke la matrice de pourcentage dans *prcnts*.
`«`	
`2 π →NUM * *`	Multiplie la matrice de proportion par 2π, et entre l'angle initial (0).
`0`	

Programme :	**Commentaires :**
`→ prop angle`	Stocke la matrice d'angle dans *prop* et l'angle dans *angle*.
`«`	
`prop SIZE OBJ→`	Définit 1 à *m* comme plage de
`DROP SWAP`	compteur de boucle.
`FOR n`	Commence la clause de la boucle.
`(66,32) prop n GET`	Place le centre du cercle dans la
`'angle' STO+`	pile, puis extrait la *n*ième valeur de la matrice de proportion et l'ajoute à *angle*.
`angle COS angle SIN`	Calcule le point d'extrémité et
`R→C 20 * OVER +`	trace la ligne de la *n*ième
`LINE`	section.
`PICT RCL`	Rappelle *PICT* dans la pile.
`angle prop n GET`	Pour libeller la section, calcule
`2 / - DUP DUP`	le point médian de l'arc de cette
`COS SWAP SIN R→C`	dernière.
`26 * (66,32) +`	
`SWAP`	
`CASE`	Commence la structure CASE pour vérifier *angle* et déterminer la valeur de décalage du libellé.
`DUP 1.5 ≤`	De 0 à 1.5 radians : ne décale
`THEN`	pas le libellé.
`DROP`	
`END`	
`DUP 4.4 ≤`	De 1.5 à 4.4 radians : décale le
`THEN`	libellé de 15 unités-utilisateur
`DROP 15 -`	vers la gauche.
`END`	
`5 <`	De 4.4 à 5 radians : décale le
`THEN`	libellé de 3 unités vers la droite
`(3,2) +`	et de 2 unités vers le haut.
`END`	
`END`	Termine la structure CASE.

Programme :	**Commentaires :**
`prcnts n GET`	Extrait la *n*ième valeur de la matrice de pourcentage,
`1 RND`	
`→STR "%" +`	l'arrondit à une décimale et la convertit en une chaîne à laquelle "%" est ajouté.
`1 →GROB`	Convertit la chaîne en objet graphique.
`GOR DUP PICT STO`	Ajoute le libellé au tracé et stocke le nouveau tracé.
`→LCD`	Affiche le tracé modifié.
`NEXT`	Termine la structure en boucle.
`( ) PVIEW`	Affiche le tracé final.
`»`	
`»`	
`flags STOF`	Restaure l'état de l'indicateur d'origine.
`» 0 MENU`	Restaure le menu précédent (vous devez d'abord appuyer sur CANCEL pour effacer le tracé).
`»`	
ENTER ' PIE STO	Stocke le programme dans *PIE*.

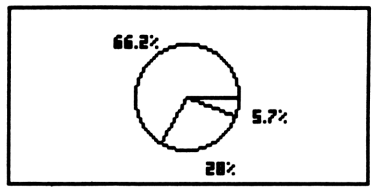
Somme de contrôle : # 1177d
Octets : 765

Exemple : Le stock d'un marchand de primeurs est composé de 983 oranges, 416 pommes, 85 bananes. Tracez un graphique circulaire, représentant le pourcentage de chaque sorte de fruit par rapport à l'ensemble du stock.

VAR PIE

Effacez les données statistiques en cours (le message est supprimé de l'affichage). Tapez les nouvelles données et tracez le graphique circulaire.

```
CLEAR
983 SLICE
416 SLICE
85 SLICE
DRAW
```



Appuyez sur CANCEL pour revenir à l'affichage de la pile.

Mode TRACE

Cette section comporte deux programmes, *αENTER* et *βENTER*, qui fournissent ensemble le "mode TRACE" du HP 48 en employant une imprimante externe. Pour activer le mode TRACE, armez l'indicateur −63 et activez le mode utilisateur (USER). Pour désactiver le mode TRACE, désarmez l'indicateur −63 ou désactivez le mode utilisateur.

Techniques employées dans αENTER et βENTER

- **Rôle de ENTER.** L'armement de l'indicateur −63 et l'activation du mode utilisateur, provoque l'activation de ENTER. Dans ce cas, et s'il existe une variable *αENTER*, le texte de la ligne de commande est placé dans la pile en tant que chaîne et *αENTER* est évalué. Ensuite, si une variable *βENTER* existe, la commande qui a déclenché le traitement de la ligne de commande est placée dans la pile sous forme de chaîne et *βENTER* est évaluée.

Listage du programme αENTER

Programme :	**Commentaires :**
≪	
PR1	Imprime le texte de la ligne de commande, puis convertit la chaîne en objet et l'évalue.
OBJ→	
≫	
ENTER ' αENTER STO	Stocke le programme dans *αENTER* (appuyez sur α A pour taper α. Vous devez absolument utiliser ce nom).

Somme de contrôle : # 51789d
Octets : 25.5

Listage du programme βENTER

Programme :	**Commentaires :**
≪	
PR1 DROP	Imprime la commande qui a provoqué le traitement, puis l'élimine et imprime la pile sous forme compacte.
PRSTC	
≫	
ENTER ' βENTER STO	Stocke le programme dans *βENTER* (appuyez sur α B pour taper β. Vous devez absolument utiliser ce nom).

Somme de contrôle : # 37631d
Octets : 28

Solver de fonctions inverses

Cette section décrit le programme *ROOTR*, qui détermine la valeur de x pour laquelle $f(x) = y$. Vous devez fournir le nom de la variable pour le programme qui calcule $f(x)$, la valeur de y, et une estimation pour x (s'il y a plusieurs solutions).

Level 3	Level 2	Niveau 1	→	Niveau 1
'nom fonction'	y	x_{estim}	→	x

Techniques utilisées dans ROOTR

- **Utilisation par programme de l'extracteur de racines.** *ROOTR* exécute ROOT pour calculer la valeur x voulue.
- **Programmes utilisés en tant qu'arguments.** Si les programmes sont en général nommés, puis exécutés en appelant leurs noms, ils peuvent également être placés dans la pile et employés comme arguments pour d'autres programmes.

Listage du programme ROOTR

Programme :	**Commentaires :**
«	
→ fname yvalue xguess	Crée des variables locales.
«	Commence la procédure de défintion.
xguess 'XTEMP' STO	Crée la variable *XTEMP* (devant être résolue).
« XTEMP fname yvalue - »	Entre le programme qui évalue $f(x) - y$.
'XTEMP'	Entre le nom de la variable inconnue.
xguess	Entre une estimation pour *XTEMP*.
ROOT	Résout le programme pour *XTEMP*.
»	Termine la procédure de définition.
'XTEMP' PURGE	Supprime la variable temporaire.
»	
ENTER ' ROOTR STO	Stocke le programme dans *ROOTR*.

Somme de contrôle : # 13007d
Octets : 163

Exemple : Supposons que vous travailliez souvent avec l'expression $3.7x^3 + 4.5x^2 + 3.9x + 5$ et que vous ayez créé le programme $X \rightarrow FX$ pour calculer la valeur :

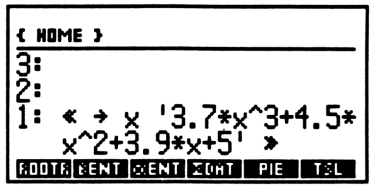
```
« → x '3.7*x^3+4.5*x^2+3.9*x+5' »
```

Vous pouvez utiliser *ROOTR* pour calculer la fonction *inverse*.

Exemple : Calculez la valeur de x pour laquelle $X \rightarrow FX$ égale 599.5. Choisissez une estimation voisine de 1.

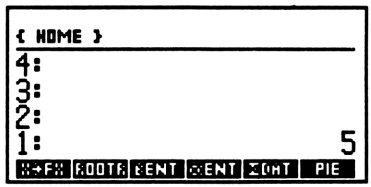
Tapez d'abord $X \rightarrow FX$:

◀ « » ▶ → x SPC ' 3.7
× x y^x 3 + 4.5 × x y^x 2
+ 3.9 × x + 5 ENTER



Stockez le programme dans $X \rightarrow FX$, saisissez le nom de programme, la valeur y 599.5 et l'estimation 1, puis exécutez *ROOTR* :

' X→FX STO
' VAR X→FX ENTER
599.5 ENTER 1 ROOTR



Animation d'une image

Le programme *WALK* permet à un petit personnage de traverser l'écran en marchant. Il anime cette image personnalisée en incrémentant sa position dans une structure en boucle.

Techniques employées dans WALK

- **Image personnalisée.** Notez qu'avant d'écrire le programme, le programmeur compile toutes les informations de l'image en la créant de manière *interactive* dans l'environnement graphique, puis en la renvoyant dans la ligne de commande.
- **FOR ... STEP (boucle finie).** *WALK* se sert de cette boucle pour animer l'image. La valeur finale de la boucle est MAXR. Etant donné que la valeur de compteur ne peut pas dépasser MAXR, la boucle s'exécute indéfiniment.

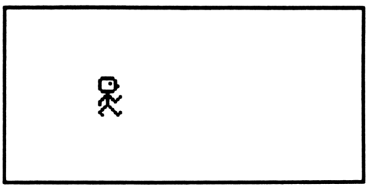
Listage du programme WALK

Programme :	Commentaires :
«	
GROB 9 15 E300 14001500 1C001400 E300 8000C110 AA009400 9000 4100220014102800	Place l'image dans la ligne de commande (notez que la partie hexadécimale de l'objet graphique est un entier continu E300 ... 2800. Les ruptures de ligne ne représentent *pas* des espaces).
→ walk	Crée la variable locale *walk* contenant l'objet graphique.
«	
ERASE (# 0d # 0d) PVIEW	Efface *PICT*, puis l'affiche.
(# 0d # 25d) PICT OVER walk GXOR	Place la première position dans la pile et active la première image. La pile et *PICT* sont ainsi préparés pour la boucle.
5 MAXR FOR i	Commence la boucle pour générer indéfiniment des coordonnées horizontales.
i 131 MOD R→B	Calcule la coordonnée horizontale de l'image suivante.
# 25d 2 →LIST	Spécifie une coordonnée verticale fixe. Place les deux coordonnées dans une liste.
PICT OVER walk GXOR	Affiche la nouvelle image, en laissant ses coordonnées dans la pile.
PICT ROT walk GXOR	Désactive l'ancienne image, en supprimant ses coordonnées de la pile.
5 STEP	Incrémente la coordonnée horizontale de 5.
»	
»	
ENTER ' WALK STO	Stocke le programme dans *WALK*.

Somme de contrôle : # 18146d
Octets : 240.5

Exemple : Lancez l'animation.

VAR WALK



Appuyez sur CANCEL pour mettre fin à l'animation.

3

Commandes

Ce chapitre présente dans l'ordre alphabétique toutes les commandes et fonctions programmables disponibles dans le HP 48. Cette liste inclut notamment les informations suivantes :

- une courte définition du rôle des commandes ou des fonctions ;
- un diagramme de la pile montrant les arguments nécessaires, le cas échéant ;
- les touches à actionner pour y accéder ;
- les indicateurs susceptibles d'en affecter le fonctionnement ;
- des informations complémentaires sur leur mode de fonctionnement et leur utilisation ;
- un exemple d'utilisation ;
- les commandes ou fonctions associées.

Les quelques pages ci-dessous expliquent comment lire les diagrammes de la pile, la façon dont les commandes sont classées, et la signification des différentes classifications mentionnées en haut à droite de chaque diagramme.

Commer lire les diagrammes de la pile

Chaque entrée de ce chapitre présente un *diagramme de la pile*. Il s'agit d'un tableau indiquant les *arguments* pris dans la pile par la commande, la fonction ou la fonction analytique et les *résultats* qui y sont renvoyés. Dans ce tableau, le caractère "→" sépare les arguments des résultats. Le diagramme de la pile peut contenir plusieurs lignes "argument → résultat", exprimant toutes les combinaisons possibles d'arguments et de résultats pour une commande donnée. Prenons l'exemple suivant :

ACOS

Fonction analytique arc-cosinus Renvoie la valeur de l'angle dont le cosinus est donné.

{ }

Niveau 1	→	Niveau 1
z	→	arc cos *z*
'*symb*'	→	'ACOS(*symb*)'

Ce diagramme indique que la *fonction analytique* ACOS (*Arc-cosinus*) prend un seul argument au niveau 1 et renvoie un résultat (dans le niveau 1). ACOS accepte comme argument un nombre réel ou complexe, ou bien un objet algébrique. Dans le premier cas, elle renvoie l'arc-cosinus numérique ; dans le second cas, elle renvoie l'expression symbolique de l'arc-cosinus de l'argument.

Certaines commandes affectent un état du calculateur—un mode, une variable réservée, un indicateur ou un affichage— sans prendre d'arguments dans la pile ni y renvoyer de résultats. Dans ce cas, aucun diagramme de pile n'apparaît.

Traitement en parallèle de listes

Les commandes qui peuvent faire appel au traitement en parallèle de listes sont signalées par le symbole "{ }" placé au-dessus du

diagramme de la pile. Cette fonction est décrite en détail à l'Annexe G.

En général, une commande peut utiliser le traitement en parallèle si les conditions suivantes sont remplies :

- La commande vérifie l'existence de types d'arguments corrects. Les commandes applicables à tous les types d'objets, comme DUP, SWAP, ROT, etc., n'ont pas recours au traitement en parallèle.
- La commande prend exactement un, deux, trois, quatre ou cinq arguments dont aucun n'est susceptible d'être une liste. Les commandes comme →LIST, qui ont un nombre indéfini d'arguments, n'utilisent pas le traitement en parallèle.
- La commande n'est pas une commande de branchement dans un programme (comme IF, FOR, CASE, NEXT, etc).

Le HP 48 dispose aussi de quelques commandes (PURGE et DELKEYS par exemple) qui ont, intégrée dans leur définition, la capacité de traiter des listes, de sorte qu'elles n'utilisent pas non plus la fonction de traitement en parallèle.

Classement des commandes

Les commandes sont classées dans l'ordre alphabérique. Celles dont le nom contient un caractère spécial (non alphabétique) sont classées comme suit :

- En ce qui concerne les commandes contenant *à la fois* des caractères spéciaux et alphabétiques :
  - Un caractère spécial figurant au *début* du nom est *ignoré*. En conséquence, la commande *H suit la commande GXOR et précède HALT.
  - Un caractère spécial *dans* ou à la *fin* du nom est considéré comme suivant le "Z" de l'alphabet. Ainsi, la commande R→B suit RSD et précède R→C.
- Les commandes contenant *uniquement* des caractères spéciaux sont placées à la fin.

Classification des opérations

La liste des commandes englobe des *commandes*, des *fonctions* et des *fonctions analytiques* du HP 48. Les commandes sont des opérations exécutables à partir d'un programme. Les fonctions sont des commandes qu'il est possible d'inclure dans des objets algébriques.

Les fonctions analytiques sont des fonctions pour lesquelles le HP 48 fournit un inverse et une dérivée. Il existe aussi quatre *opérations* non programmables (DBUG, NEXT, SST et SST↓) qui sont mentionnées avec les commandes programmables pour des raisons de commodité, car elles sont utilisées de manière interactive pendant la programmation.

Les définitions des abréviations utilisées pour les arguments et les résultats sont fournies dans le tableau suivant, "Termes employés dans les diagrammes de la pile". Souvent, un indice descriptif est fourni pour plus de clarté.

Termes employés dans les diagrammes de la pile

Terme	Description
arg	Argument.
[*tableau*]	Vecteur ou matrice de type réel ou complexe.
[*C-tableau*]	Vecteur ou matrice de type complexe
date	Date au format MM.JJAAAA ou JJ.MMAAAA.
{ *dim* }	Liste d'une ou deux dimensions de tableau (nombres réels).
' *global* '	Nom global.
grob	Objet graphique.
HMS	Heure en nombre réel ou angle au format heures-minutes-secondes.
{ *liste* }	Liste d'objets.
local	Nom local.
[[*matrice*]]	Matrice réelle ou complexe.
n ou *m*	Nombre réel entier positif (arrondi si non entier).
$:n_{port}:nom_{sauvegarde}$	Identificateur de sauvegarde.
$:n_{port}:n_{bibliothèque}$	Identificateur de bibliothèque.
#*n*	Entier binaire.
{ #*n* #*m* }	Coordonnées en pixels. (Utilise des entiers binaires.)
' *nom* '	Nom global ou local.
obj	Objet quelconque.
PICT	Objet graphique en cours.
≪ *programme* ≫	Programme.
[*R-tableau*]	Vecteur réel ou matrice réelle.
"*chaîne*"	Chaîne de caractères.
' *symb* '	Expression, équation ou nom traité comme objet algébrique.
V/F	Résultat de test utilisé comme *argument*: nombre réel égal à zéro (faux) ou différent de zéro (vrai).
0/1	Résultat de test *renvoyé* par une commande: zéro (faux) ou un (vrai).
heure	Heure au format HH.MMSSs.
[*vecteur*]	Vecteur réel ou complexe.
x ou *y*	Nombre réel.
x_unité	Objet-unité, ou nombre réel traité comme objet sans dimensions.
(*x*,*y*)	Nombre complexe sous forme rectangulaire, ou coordonnées en unités-utilisateur.
z	Nombre réel ou complexe.

ABS

Fonction de valeur absolue : Renvoie la valeur absolue de son argument.

{ }

Niveau 1	→	Niveau 1
x	→	$\|x\|$
(*x*,*y*)	→	$\sqrt{x^2+y^2}$
x_unité	→	$\|x\|$_*unité*
[*tableau*]	→	|| *tableau* ||
'*symb*'	→	'ABS(*symb*)'

Accès clavier :

(MTH) REAL (NXT) ABS

(MTH) MATR NORM ABS

(MTH) (NXT) CMPL ABS

(MTH) VECTR ABS

Indicateurs : Résultats numériques (−3)

Remarques : ABS a une dérivée (SIGN) mais pas d'inverse.

Dans le cas d'un tableau, ABS renvoie la norme de Frobenius (euclidienne) du tableau, définie comme la racine carrée de la somme des carrés des valeurs absolues des n éléments. A savoir,

$$\sqrt{\sum_{i=1}^{n} |z_i|^2}$$

Commandes associées : NEG, SIGN

ACK

Commande d'accusé de réception d'une alarme : Accuse réception de la plus ancienne alarme échue.

Accès clavier : ↩ TIME ALRM ACK

Indicateurs : Alarmes répétitives non reprogrammées (−43), Alarmes ayant reçu un accusé de réception sauvegardées (−44)

Remarques : ACK supprime le témoin d'alerte s'il n'existe pas d'autres alarmes échues ni d'autres sources d'alerte actives (batterie faible par exemple).

ACK n'a aucun effet sur les alarmes de commande. Celles-ci, lorsqu'elles arrivent à échéance, reçoivent automatiquement un accusé de réception *et* sont sauvegardées dans la liste des alarmes du système.

Commande associée : ACKALL

ACKALL

Commande d'accusé de réception de toutes les alarmes : Accuse réception de toutes les alarmes échues.

Accès clavier : ↩ TIME ALRM ACKA

Indicateurs : Alarmes répétitives non reprogrammées (−43), Alarmes ayant reçu un accusé de réception sauvegardées (−44)

Remarques : ACKALL supprime le témoin d'alerte s'il n'existe pas d'autres sources d'alerte actives (par exemple une batterie faible).

ACKALL n'a aucun effet sur les alarmes de commande. Celles-ci, lorsqu'elles arrivent à échéance, reçoivent automatiquement un accusé de réception *et* sont sauvegardées dans la liste des alarmes système.

Commande associée : ACK

ACOS

Fonction analytique arc-cosinus : Renvoie la valeur de l'angle dont le cosinus est donné.

{ }

Niveau 1	→	Niveau 1
z	→	arc cos *z*
'*symb*'	→	'ACOS(*symb*)'

Accès clavier : ◀ ACOS

Indicateurs : Solution principale (−1), Résultats numériques (−3), Mode d'angle (−17, −18)

Remarques : Pour un argument réel x compris dans le domaine $-1 \leq x \leq 1$, le résultat est compris entre 0 et 180 degrés (0 à π radians; 0 à 200 grades).

Un argument réel hors de ce domaine est converti en argument complexe $z = x + 0i$, et le résultat est complexe.

L'inverse de COS est une *relation*, non une fonction, puisque COS envoie plusieurs arguments au même résultat. La relation inverse pour COS est exprimée par ISOL comme *solution générale*:

```
'≤1*ACOS(Z)+2*π*n1'
```

La fonction ACOS est l'inverse d'une *partie* de COS, partie définie par restriction du domaine de COS de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points de ce domaine restreint de COS constituent les *valeurs principales* de la relation inverse. ACOS dans sa totalité est appelée *détermination principale* de la relation inverse, et les points envoyés par ACOS à la frontière du domaine restreint de COS constituent les *coupures* de ACOS.

La détermination principale utilisée par le HP 48 pour ACOS a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse à *valeurs réelles* sont définis. La coupure pour la fonction arc-cosinus à valeur complexe apparaît dans une zone

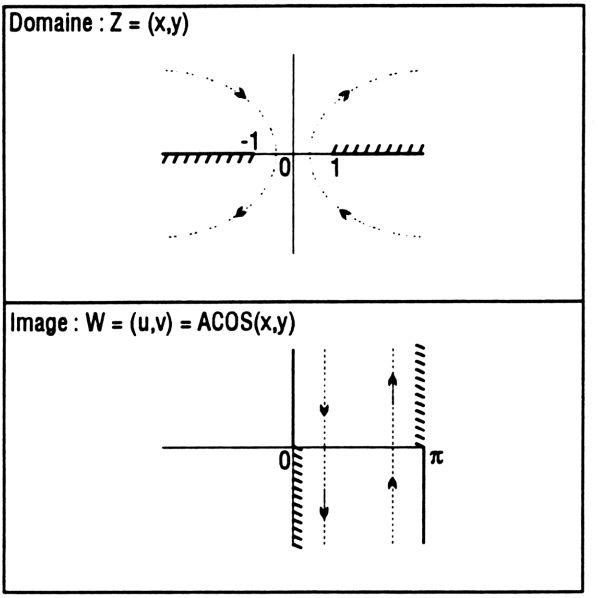
où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Les graphiques ci-dessous montrent le domaine et l'image de ACOS. Le graphique représentant le domaine montre l'emplacement de la coupure : le trait plein marque un bord de la coupure, les hachures marquent l'autre bord. Le graphique représentant l'image indique l'endroit où chaque bord des coupures est projeté par la fonction .

Ces graphiques montrent la relation inverse '⍵1*ACOS⍳Z⍴+2*⍵*⍳n1' dans le cas où *s1*=1 et *n1*=0. Pour d'autres valeurs de *s1* et *n1*, la bande verticale du graphique inférieur est translatée vers la droite ou vers la gauche. La réunion de ces bandes couvre la totalité du plan complexe, qui correspond au domaine de COS.

Il suffit de regarder ces graphiques en inversant le domaine et l'image pour voir comment le domaine de COS est restreint de façon à ce qu'une *fonction* inverse soit possible. Considérez la bande verticale du graphique inférieur comme étant le domaine restreint Z = (*x*, *y*). COS projette ce domaine sur la totalité du plan complexe dans l'image W = (*u*, *v*) = COS(*x*, *y*) du graphique supérieur.

Commandes associées : ASIN, ATAN, COS, ISOL



Coupures pour ACOS(Z)

ACOSH

Fonction analytique cosinus hyperbolique inverse : Renvoie le cosinus hyperbolique inverse de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	acosh *z*
'*symb*'	→	'ACOSH(*symb*)'

Accès clavier : (MTH) HYP ACOSH

Indicateurs : Solution principale (−1), Résultats numériques (−3)

Remarques : Pour des arguments réels $|x| < 1$, ACOSH renvoie le résultat complexe obtenu pour l'argument $(x, 0)$.

L'inverse de ACOSH est une *relation*, non une fonction, puisque COSH envoie plusieurs arguments au même résultat. La relation inverse pour COSH est exprimée par ISOL comme la *solution générale*:

```
'≤1*ACOSH(Z)+2*π*i*n1'
```

La fonction ACOSH est l'inverse d'une *partie* de COSH, partie définie par restriction du domaine de COSH de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points de ce domaine restreint de COSH constituent les *valeurs principales* de la relation inverse. ACOSH dans sa totalité est appelée *détermination principale* de la relation inverse, et les points envoyés par ACOSH à la frontière du domaine restreint de COSH constituent les *coupures* de ACOSH.

La détermination principale utilisée par le HP 48 pour ACOSH a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse à *valeurs réelles* sont définis. La coupure pour la fonction arc-cosinus hyperbolique à valeur complexe apparaît dans une zone où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Les graphiques ci-dessous montrent le domaine et l'image de ACOSH. Le graphique représentant le domaine montre l'emplacement de la coupure : le trait plein marque un bord de la coupure, les hachures marquent l'autre bord. Le graphique représentant l'image indique l'endroit où chaque bord des coupures est projeté par la fonction .

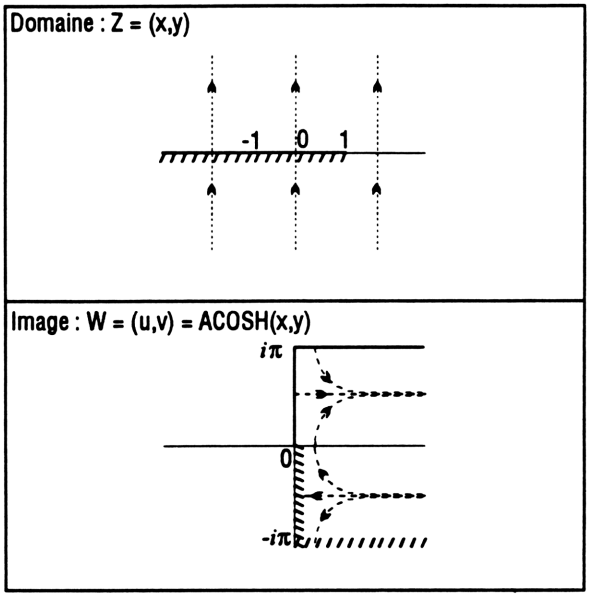
Ces graphiques montrent la relation inverse `'≤1*ACOSH(Z)+2*π*i*n1'` dans le cas où $s1=1$ et $n1=0$. Pour d'autres valeurs de *s1* et *n1*, la demi-bande horizontale du graphique inférieur pivote vers la gauche et est translatée de bas en haut. La réunion de ces bandes couvre l'ensemble du plan complexe, qui est le domaine de COSH.

Il suffit de regarder ces graphiques en inversant le domaine et l'image pour voir comment le domaine de COSH est restreint de telle sorte qu'une *fonction* inverse soit possible. Considérez la demi-bande horizontale du graphique inférieur comme étant le domaine restreint `Z =·(x, y)`. COSH projette ce domaine sur la totalité du plan

ACOSH

complexe dans l'image W = ⟨*u*, *v*⟩ = COSH⟨*x*, *y*⟩ du graphique supérieur.

Commandes associées : ASINH, ATANH, COSH, ISOL



Coupures pour ACOSH(Z)

ADD

Commande d'addition de listes : Additionne les éléments correspondants de deux listes ou ajoute un nombre à chacun des éléments d'une liste.

{}

Niveau 2	Niveau 1	→	Niveau 1
{ *liste*$_1$ }	{ *liste*$_2$ }	→	{ *liste*$_{résultat}$ }
{ *liste* }	*obj*$_{non-liste}$	→	{ *liste*$_{résultat}$ }
obj$_{non-liste}$	{ *liste* }	→	{ *liste*$_{résultat}$ }

Accès clavier : (MTH) LIST ADD

Indicateurs : Aucun

Remarques : ADD exécute la commande + une fois pour chacun des éléments de la liste. S'il y a deux listes comme arguments, elles doivent contenir le même nombre d'éléments puisque ADD exécute la commande + une fois pour chaque paire d'éléments correspondante. Si un argument n'est pas une liste, ADD tente d'exécuter la commande en utilisant cet objet non-liste et chaque élément de l'argument liste, en renvoyant le résultat à la position correspondante. (Reportez-vous à la commande + pour voir les combinaisons d'objets qui sont définies.) En cas d'addition non définie, le message d'erreur Bad Argument Type s'affiche.

Commandes associées : ΔLIST, ΠLIST, ΣLIST

ALOG

Fonction analytique antilogarithme commun : Renvoie l'antilogarithme commun, à savoir 10 élevé à la puissance donnée.

{ }

Niveau 1	→	Niveau 1
z	→	10^z
'*symb*'	→	'ALOG(*symb*)'

Accès clavier : ◀ 10^x

Indicateurs : Résultats numériques (−3)

Remarque : Pour des arguments complexes :

$10^{(x,y)} = e^{cx} \cos cy + i\, e^{cx} \sin cy$

où $c = \ln 10$.

Commandes associées : EXP, LN, LOG

AMORT

Commande d'amortissement : Amortit un prêt ou un investissement en fonction des valeurs d'amortissement en vigueur.

{ }

Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
n	→	*capital*	*intérêts*	*solde*

Accès clavier : ◀ SOLVE TVM AMOR

Indicateurs : Indicateur de mode Paiement financier (−14)

Remarque : Les valeurs doivent être stockées dans les variables TVM (*I%YR*, *PV*, *PMT* et *PYR*). Le nombre de versements *n* est fourni par le niveau 1 et par l'indicateur −14.

Commandes associées : TVM, TVMBEG, TVMEND, TVMROOT

AND

Fonction de somme (And) : Renvoie la somme (AND) logique de deux arguments.

{ }

Niveau 2	Niveau 1	→	Niveau 1
#n_1	#n_2	→	#n_3
"*chaîne*$_1$"	"*chaîne*$_2$"	→	"*chaîne*$_3$"
V/F$_1$	*V/F*$_2$	→	0/1
V/F	'*symb*'	→	'*V/F* AND *symb*'
'*symb*'	*V/F*	→	'*symb* AND *V/F*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$ AND *symb*$_2$'

Accès clavier :

(MTH) BASE (NXT) LOGIC AND

(PRG) TEST (NXT) AND

Indicateurs : Résultats numériques (−3), Taille de mot entier binaire (−5 à −10)

Remarques : Lorsque les arguments sont des entiers binaires ou des chaînes, AND procède à une comparaison logique bit par bit (base 2).

- Un argument qui est un entier binaire est traité comme une séquence de bits sur toute la taille du mot. Chaque bit du résultat est déterminé par la comparaison des bits correspondants (bit_1 et bit_2) des deux arguments, comme le montre le tableau ci-dessous :

bit_1	bit_2	bit_1 **AND** bit_2
0	0	0
0	1	0
1	0	0
1	1	1

- Un argument qui est une chaîne est traité comme une séquence de bits, à raison de 8 bits par caractère (soit l'utilisation de la version binaire du code de caractères). Les deux arguments chaînes doivent comporter le même nombre de caractères.

Lorsque les arguments sont des nombres réels ou symboliques, AND exécute simplement un test vrai/faux. Le résultat est 1 (vrai) si les deux arguments sont différents de zéro, et il est égal à 0 (faux) si l'un des arguments ou les deux égalent zéro. Ce test sert généralement à comparer deux résultats.

Si l'un des arguments ou les deux sont des expressions algébriques, le résultat est une expression algébrique de la forme '$symb_1$ AND $symb_2$'. Il suffit d'exécuter →NUM (ou de définir l'indicateur −3 avant d'exécuter AND) afin d'obtenir un résultat numérique à partir du résultat algébrique.

Commandes associées : NOT, OR, XOR

ANIMATE

Commande d'animation : Affiche des objets graphiques en séquence.

Niveau n+1...Niveau 2	Niveau 1	→ Niveau 1
$grob_n ... grob_1$	n_{grobs}	→ *même pile*
$grob_n ... grob_1$	{ *n* { #*X* #*Y* } *durée rép* }	→ *même pile*

Accès clavier :

PRG GROB NXT ANIM

Indicateurs : Aucun

Remarques : ANIMATE affiche une séquence d'objets graphiques (ou des variables qui en contiennent) les uns après les autres. Vous pouvez utiliser une liste pour spécifier la zone de l'écran à animer (coordonnées en pixels #*X* et #*Y*), la durée, en secondes, entre chaque plan de l'animation (*durée*) et le nombre de répétitions de la séquence (*rép*). Si *rép* est égal à 0, la séquence se répète un million de fois ou jusqu'à ce que vous appuyiez sur CANCEL.

Si vous utilisez une liste au niveau 1, tous les paramètres doivent être présents.

L'animation affiche PICT pendant l'affichage des objets graphiques. Ceux-ci, ainsi que les paramètres d'animation, sont laissés dans la pile.

Exemple : Le programme suivant dessine la moitié d'un cylindre et le fait pivoter :

```
« PARSURFACE ( 'COS(X)' 'SIN(X)' Y )
  STEQ
    « I 180 I + XXRNG ERASE DRAW PICT RCL
    »
  I 0 359 8 SEQ OBJ→ ANIMATE DROPN
»
```

Ce programme illustre également l'emploi de SEQ et de PARSURFACE. Vous pouvez changer la valeur adoptée pour SEQ (ici, elle est égale à 8) afin de modifier le nombre d'images dessinées par le programme ou d'utiliser moins de mémoire.

APPLY

Fonction d'application à des arguments : Crée une expression à partir du nom de fonction spécifié et des arguments.

Niveau 2	Niveau 1	→	Niveau 1
{ $symb_1$ ... $symb_n$ }	'*nom*'	→	'*nom*($symb_1$ ... $symb_n$)'

Accès clavier : ◀ SYMBOLIC NXT APPLY

Indicateurs : Aucun

Remarques : Une fonction-utilisateur *f* qui recherche dans ses arguments des cas spéciaux est souvent incapable de déterminer si un argument symbolique *x* correspond à un de ces cas spéciaux. La fonction *f* peut alors utiliser APPLY afin de créer une nouvelle expression `'f(x)'`. Si l'utilisateur évalue `'f(x)'`, *x* est évaluée avant *f*, de sorte que l'argument pour *f* sera le résultat obtenu par l'évalution de *x*.

La syntaxe algébrique pour APPLY est la suivante :

'APPLY(*nom*, $symb_1$, ... , $symb_n$)'

Dans une expression algébrique, APPLY évalue les arguments (pour résoudre des noms locaux dans les fonctions-utilisateur) avant de créer le nouvel objet.

Exemple : La fonction-utilisateur *Asin* suivante est une variante de la fonction intégrée ASIN. *Asin* recherche des arguments numériques spéciaux. Si l'argument figurant dans la pile est symbolique (deuxième cas dans la structure proposée), *Asin* utilise APPLY pour renvoyer l'expression `'Asin(`*argument*`)'`.

APPLY

```
«
 → argument
 «
  CASE
   -3 FS? THEN argument ASIN END
   { 6 7 9 } argument TYPE POS
    THEN 'APPLY(Asin,argument)' EVAL END
   'argument==1' THEN 'π/2' END
   'argument==-1' THEN '-π/2' END
   argument ASIN
  END
 »
»
```

ENTER ' Asin STO

Commandes associées : QUOTE, |

ARC

Commande de dessin d'un arc : Dessine un arc dans *PICT* dans le sens antihoraire, de x_{θ_1} à x_{θ_2}, centré sur les coordonnées du niveau 4 et avec un rayon donné au niveau 3.

Niveau 4	Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
(x, y)	x_{rayon}	x_{θ_1}	x_{θ_2}	→	
$\{\#n\ \#m\}$	$\#n_{\text{rayon}}$	x_{θ_1}	x_{θ_2}	→	

Accès clavier : PRG PICT ARC

Indicateurs : Mode d'angle (−17 et −18)

Le paramétrage des indicateurs −17 et −18 determine l'interprétation de x_{θ_1} et x_{θ_2} (degrées, radians ou grades).

Remarques : ARC dessine toujours un arc d'un rayon constant exprimé en pixels, même si le rayon et le centre sont spécifiés en unités-utilisateur, indépendamment des échelles relatives des axes x et y spécifiées en unités-utilisateur. Avec des arguments exprimés en

unités-utilisateur, l'arc commence au pixel désigné par $(x, y) + (a, b)$, où (a, b) est la conversion rectangulaire de la coordonnée polaire $(x_{\text{rayon}}, x_{\theta_1})$. La distance en pixels obtenue entre le point de départ et le pixel central est utilisée comme rayon réel r'. L'arc se termine au niveau du pixel spécifié par (r', x_{θ_2}).

Si $x_{\theta_1} = x_{\theta_2}$, ARC trace un point. Si $|x_{\theta_1} - x_{\theta_2}| > 360$ degrés, 2π radians, ou 400 grades, ARC dessine un cercle.

Exemple : En mode Degrés, avec la plage d'affichage de l'axe des x (XRNG) allant de −6.5 à 6.5, la séquence de commande `‹0,0› 1 0 90 ARC` dessine un arc dans le sens antihoraire, de 0 à 90 degrés, avec un rayon constant de 10 pixels.

Commandes associées : BOX, LINE, TLINE

ARCHIVE

Commande d'archivage du répertoire HOME : Crée une copie de sauvegarde du répertoire *HOME* (c'est-à-dire de toutes les variables), des définitions de touches utilisateur et du catalogue des alarmes, qu'elle place dans l'objet-sauvegarde spécifié (**:** n_{port} **:** *nom*) en RAM indépendante.

Niveau 1	→	Niveau 1
:n_{port} :*nom*	→	
:ES :*nom*	→	

Accès clavier : ↩ MEMORY NXT ARCHI

Indicateurs : Unité d'E-S (−33), Messages d'E-S (−39) *si* l'argument est **:**ES**:** *nom*

Remarques : Le numéro de port spécifié peut aller de 0 à 33. Le port utilisé (à l'exception du numéro 0) doit être configuré comme RAM indépendante (voir FREE). Une erreur se produira si la taille de la RAM indépendante n'est pas suffisante pour recevoir la copie du répertoire HOME.

ARCHIVE

Si l'objet-sauvegarde est :ES:*nom*, le répertoire copié est transmis en binaire via le protocole Kermit, et enregistré sous le nom de fichier spécifié en utilisant le port d'E-S en cours.

Pour sauvegarder le paramétrage des indicateurs, exécutez RCLF et stockez la liste obtenue dans une variable.

Commande associée : RESTORE

ARG

Fonction Argument : Renvoie l'angle polaire (réel) θ d'un nombre complexe (x, y).

{ }

Niveau 1	→	Niveau 1
(*x*,*y*)	→	θ
'*symb*'	→	'ARG(*symb*)'

Accès clavier : MTH NXT CMPL ARG

Indicateurs : Mode d'angle (−17, −18)

Remarques : L'angle polaire θ est égal à :

- arc-tangente y/x pour $x \geq 0$
- arc-tangente $y/x + \pi$ signe y pour $x < 0$, mode radians
- arc-tangente $y/x + 180$ signe y pour $x < 0$, mode degrés
- arc-tangente $y/x + 200$ signe y pour $x < 0$, mode grades

Un argument réel x est traité comme l'argument complexe $(x, 0)$.

ARRY→

Commande d'envoi de tableau dans la pile : Renvoie les éléments d'un tableau dans la pile, sous la forme de nombres réels ou complexes distincts, ainsi qu'une liste des dimensions du tableau.

Niveau 1	→	Niveau nm+1 ... Niveau 2	Niveau 1
[*vecteur*]	→	$z_1 \ldots z_n$	{ $n_{élément}$ }
[[*matrice*]]	→	$z_{11} \ldots z_{nm}$	{ n_{lig} m_{col} }

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : La commande OBJ→ inclut cette fonctionnalité. ARRY→ est fournie à des fins de compatibilité avec le HP 28S. ARRY→ ne figure pas dans un menu.

Si l'argument est un vecteur de n éléments, le premier élément est renvoyé au niveau $n + 1$ (et non au niveau $nm + 1$), et le $n$ième élément au niveau 2.

Commandes associées : →ARRY, DTAG, EQ→, LIST→, OBJ→, STR→

→ARRY

Commande de création d'un tableau à partir du contenu de la pile : Renvoie un vecteur de n éléments, réels ou complexes, ou une matrice de $n \times m$ éléments réels ou complexes.

Niveau nm+1 ... Niveau 2	Niveau 1	→	Niveau 1
$z_1 \ldots z_n$	$n_{élément}$	→	[*vecteur*]
$z_{11} \ldots z_{nm}$	{ n_{lig} m_{col} }	→	[[*matrice*]]

→ARRY

Accès clavier : PRG TYPE →ARR

Indicateurs : Aucun

Remarque : Les éléments du tableau obtenu doivent être saisis dans la pile dans l'ordre des lignes, avec z_{11} (ou z_1) au niveau $nm + 1$ (ou $n + 1$), et z_{nm} (ou z_n) au niveau 2. Si un ou plusieurs des éléments est un nombre complexe, le tableau obtenu sera complexe.

Commandes associées : ARRY→, LIST→, →LIST, OBJ→, STR→, →TAG, →UNIT

ASIN

Fonction analytique arc-sinus : Renvoie la valeur d'un angle dont le sinus est donné.

{ }

Niveau 1	→	Niveau 1
z	→	arc sin *z*
'*symb*'	→	'ASIN(*symb*)'

Accès clavier : ← ASIN

Indicateurs : Solution principale (−1), Résultats numériques (−3), Mode d'angle (−17, −18)

Remarques : Pour un argument réel x situé dans le domaine $-1 \leq x \leq 1$, le résultat est compris entre −90 et +90 degrés ($-\pi/2$ à $+\pi/2$ radians ; −100 à +100 grades).

Un argument réel hors de ce domaine est converti en argument complexe $z = x + 0i$, et le résultat est complexe.

L'inverse de SIN est une *relation*, non une fonction, puisque SIN envoie plusieurs arguments au même résultat. La relation inverse pour SIN est exprimée par ISOL comme la *solution générale* :

```
'ASIN(Z)*(-1)^n1+π*n1'
```

La fonction ASIN est l'inverse d'une *partie* de SIN, partie définie par restriction du domaine de SIN de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points situés dans ce domaine restreint de SIN constituent les *valeurs principales* de la relation inverse. ASIN dans sa totalité est appelée la *détermination principale* de la relation inverse, et les points envoyés par ASIN à la frontière du domaine restreint de SIN forment les *coupures* de ASIN.

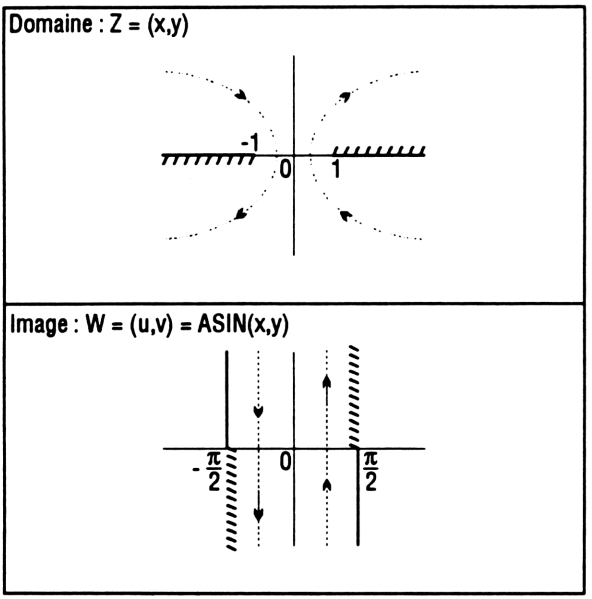
La détermination principale utilisée par le HP 48 pour ASIN a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse à *valeurs réelles* sont définis. La coupure pour la fonction arc-sinus à valeur complexe apparaît dans la région où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Les graphiques ci-dessous montrent le domaine et l'image de ASIN. Le graphique représentant le domaine indique l'emplacement de la coupure : le trait plein marque un bord de la coupure, les hachures marquent l'autre bord. Le graphique représentant l'image montre l'emplacement où chaque bord de la coupure est projeté par la fonction.

Ces graphiques montrent la relation inverse `'ASIN(Z)*(-1)^n1+π*n1'` dans le cas où *n1*=0. Pour d'autres valeurs de *n1*, la bande verticale du graphique inférieur est translatée vers la droite (pour *n1* positif) ou vers la gauche (pour *n1* négatif). La réunion de ces bandes couvre la totalité du plan complexe, qui est le domaine de SIN.

Il suffit de regarder ces graphiques en inversant le domaine et l'image pour voir comment le domaine de SIN est restreint de façon à ce qu'une *fonction* inverse soit possible. Considérez la bande verticale du graphique inférieur comme le domaine restreint `Z = (x, y)`. SIN projette ce domaine sur la totalité du plan complexe dans l'image `W = (u, v) = SIN(x, y)` du graphique supérieur.

Commandes associées : ACOS, ATAN, ISOL, SIN



Coupures pour ASIN(Z)

ASINH

Fonction analytique arc-sinus hyperbolique : Renvoie le sinus hyperbolique inverse de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	asinh *z*
'*symb*'	→	'ASINH(*symb*)'

Accès clavier : (MTH) HYP ASINH

Indicateurs : Solution principale (−1), Résultats numériques (−3)

Remarques : L'inverse de SINH est une *relation*, non une fonction, puisque SINH envoie plusieurs arguments au même résultat. La relation inverse pour SINH est exprimée par ISOL comme la *solution générale* :

```
'ASINH(Z)*(-1)^n1+π*i*n1'
```

La fonction ASINH est l'inverse d'une *partie* de SINH, partie définie par restriction du domaine de SINH de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points situés dans ce domaine restreint de SINH constituent les *valeurs principales* de la relation inverse. ASINH dans sa totalité est appelée la *détermination principale* de la relation inverse, et les points envoyés par ASINH à la frontière du domaine restreint de SINH forment les *coupures* de ASINH.

La détermination principale utilisée par le HP 48 pour ASINH a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse à *valeurs réelles* sont définis. La coupure pour la fonction ASINH à valeur complexe apparaît là où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Le graphique pour ASINH est dérivé de celui de ASIN (voir ASIN) et la relation asinh $z = -i$ asin iz.

Commandes associées : ACOSH, ATANH, ISOL, SINH

ASN

Commande Affecter : Définit une touche du clavier utilisateur en affectant l'objet donné à la touche x_{touche}, spécifiée par *lc.s*.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	x_{touche}	→	
'SKEY'	x_{touche}	→	

Accès clavier : ◀ MODES KEYS ASN

ASN

Indicateurs : Verrouillage mode utilisateur (−61) et Mode utilisateur (−62).

Remarques : L'argument x_{touche} est un nombre réel *lc.s* désignant la touche par son numéro de ligne *l*, son numéro de colonne *c*, et le code shift *s*. Les valeurs autorisées pour *s* sont les suivantes :

Code *s*	Shift
0 ou 1	non shifté
2	◀ shifté-gauche
3	▶ shifté-droite
4	α alpha
5	α ◀ alpha shifté-gauche
6	α ▶ alpha shifté-droite

Une fois que ASN a été exécutée, la pression d'une touche en mode utilisateur (USER ou 1USR) déclenche l'exécution de l'objet affecté par l'utilisateur. Les définitions des touches utilisateur restent en vigueur jusqu'à leur modification au moyen de ASN, STOKEYS ou DELKEYS. Les touches sans affectation conservent leur définition standard.

Si l'argument *obj* est `'SKEY'`, la touche spécifiée reprend sa définition *standard* sur le clavier utilisateur. Ceci n'a de sens que lorsque toutes les définitions de touches standard ont été supprimées (pour le clavier utilisateur) par la commande `'S' DELKEYS` (voir DELKEYS).

Pour définir plusieurs touches simultanément, utilisez STOKEYS. Pour supprimer des définitions, utilisez DELKEYS.

Attention de ne pas réaffecter ou supprimer les touches servant à annuler le mode utilisateur. Si toutefois cela se produisait, quittez le mode utilisateur en interrompant le système ("démarrage à chaud") : appuyez simultanément sur ON et sur C, puis relâchez la touche C en premier. Vous annulez ainsi le mode utilisateur.

Exemple : L'exécution de ASN avec `GETI` au niveau 2 et `85.3` au niveau 1 affecte GETI à ▶ " " sur le clavier utilisateur. La combinaison (▶ " " est désignée par 85.3 car elle est à la huitième

ligne, cinquième colonne et shiftée-droite.) Lorsque le mode utilisateur est activé, une pression sur ▶ " " exécute GETI (au lieu de " ").

Commandes associées : DELKEYS, RCLKEYS, STOKEYS

ASR

Commande de décalage arithmétique à droite : Décale un entier binaire d'un bit à droite, sauf pour le bit de poids fort qui est inchangé.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : MTH BASE NXT BIT ASR

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : Le bit de poids fort est maintenu tel quel alors que les autres bits (*tailledemot*−1) sont décalés d'un bit vers la droite. Le bit qui suit le bit de poids fort est remplacé par zéro. Le bit de poids faible est décalé à droite et donc perdu.

Ce décalage arithmétique permet de conserver le bit de signe d'un entier binaire qui sera décalé. Bien que le HP 48 ne prévoit pas spécialement des entiers binaires signés, vous pouvez toujours *interpréter* un nombre comme une quantité signée.

Commandes associées : SL, SLB, SR, SRB

ATAN

Fonction analytique arc-tangente : Renvoie la valeur de l'angle dont la tangente est donnée.

{ }

Niveau 1	→	Niveau 1
z	→	arc tan *z*
'*symb*'	→	'ATAN(*symb*)'

Accès clavier : ◀ ATAN

Indicateurs : Solution principale (−1), Résultats numériques (−3), Mode d'angle (−17, −18)

Remarques : Pour un argument réel, le résultat est compris entre −90 et +90 degrés ($-\pi/2$ à $+\pi/2$ radians ; −100 à +100 grades).

L'inverse de TAN est une *relation*, non une fonction, puisque TAN envoie plusieurs arguments au même résultat. La relation inverse pour TAN est exprimée par ISOL comme la *solution générale* :

```
'ATAN(Z)+π*n1'
```

La fonction ATAN est l'inverse d'une *partie* de TAN, partie définie par restriction du domaine de TAN de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points situés dans ce domaine restreint de TAN constituent les *valeurs principales* de la relation inverse. ATAN dans sa totalité est appelée la *détermination principale* de la relation inverse, et les points envoyés par ATAN à la frontière du domaine restreint de TAN forment les *coupures* de ATAN.

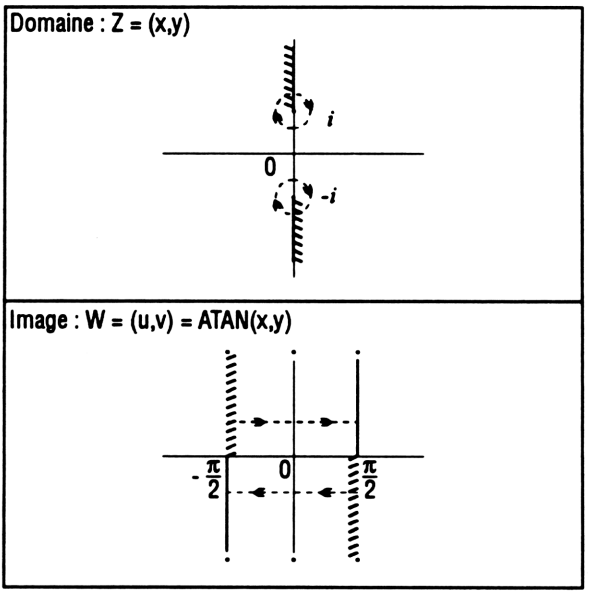
La détermination principale utilisée par le HP 48 pour ATAN a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse à *valeurs réelles* sont définis. Les coupures pour la fonction arc-tangente à valeur complexe apparaissent dans la région où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Les graphiques ci-dessous montrent le domaine et l'image de ATAN. Le graphique représentant le domaine indique l'emplacement de la coupure : le trait plein marque un bord de la coupure, les hachures marquent l'autre bord. Le graphique représentant l'image montre l'emplacement où chaque bord de la coupure est projeté par la fonction.

Ces graphiques montrent la relation inverse 'ATAN(Z)+π*n1' dans le cas où *n1*=0. Pour les autres valeurs de *n1*, la bande verticale du graphique inférieur est translatée vers la droite (pour *n1* positif) ou vers la gauche (pour *n1* négatif). La réunion de ces bandes couvre la totalité du plan complexe, qui est le domaine de TAN.

Il suffit de regarder ces graphiques en inversant le domaine et l'image pour voir comment le domaine de TAN est restreint de façon à ce qu'une *fonction* inverse soit possible. Considérez la bande verticale du graphique inférieur comme le domaine restreint Z = (*x*, *y*). TAN projette ce domaine sur la totalité du plan complexe dans l'image W = (*u*, *v*) = TAN(*x*, *y*) du graphique supérieur.

Commandes associées : ACOS, ASIN, ISOL, TAN



Coupures pour ATAN(Z)

ATANH

Fonction analytique arc-tangente hyperbolique : Renvoie la tangente hyperbolique inverse de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	atanh *z*
'*symb*'	→	'ATANH(*symb*)'

Accès clavier : (MTH) HYP ATAN

Indicateurs : Solution principale (−1), Résultats numériques (−3), Exception de résultat infini (−22)

Remarques : Pour des arguments réels $|x| > 1$, ATANH renvoie le résultat complexe obtenu pour l'argument $(x, 0)$. Pour un argument réel $x=\pm 1$, une exception `Infinite Result` est signalée. Si l'indicateur −22 est armé (pas de message d'erreur), le signe du résultat (MAXR) correspond à celui de l'argument.

L'inverse de TANH est une *relation*, non une fonction, puisque TANH envoie plusieurs arguments au même résultat. La relation inverse pour TANH est exprimée par ISOL comme la *solution générale* :

```
'ATANH(Z)+π*i*n1'
```

La fonction ATANH est l'inverse d'une *partie* de TANH, partie définie par restriction du domaine de TANH de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points situés dans ce domaine restreint de TANH constituent les *valeurs principales* de la relation inverse. ATANH dans sa totalité est appelée la *détermination principale* de la relation inverse, et les points envoyés par ATANH à la frontière du domaine restreint de TANH forment les *coupures* de ATANH.

La détermination principale utilisée par le HP 48 pour ATANH a été choisie car elle est analytique dans les régions où les arguments de la fonction à *valeurs réelles* sont définis. Les coupures pour la fonction ATANH à valeur complexe apparaissent dans la région où la fonction

à valeurs réelles correspondante n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Le graphique pour ATANH peut être dérivé du graphique de ATAN (voir ATAN) et de la relation atanh $z = -i$ atan iz.

Commandes associées : ACOSH, ASINH, ISOL, TANH

ATICK

Commande de graduation des axes : Définit les marques de graduation des axes dans la variable réservée *PPAR*.

Niveau 1	→	Niveau 1
x	→	
#n	→	
{ *x,y* }	→	
{ *#n #m* }	→	

Accès clavier : ◀ PLOT PPAR NXT ATICK

Indicateurs : Aucun

Remarques : Etant donné x, ATICK définit la graduation à x unités sur les axes x et y. Par exemple, avec 2 les deux axes sont gradués toutes les deux unités.

Etant donné $\#n$, ATICK définit la graduation à $\#n$ pixels sur les axes x et y. Par exemple, avec #5 les deux axes sont gradués tous les 5 pixels.

Etant donné { x y }, ATICK définit la graduation pour chaque axe séparément. Par exemple, avec { 10 3 } l'axe x est gradué toutes les 10 unités et l'axe y toutes les 3 unités.

Etant donné { $\#n$ $\#m$ } ATICK définit la graduation en pixel sur chaque axe séparément. Par exemple, avec { 6 2 } l'axe x est gradué tous les 6 pixels et l'axe y tous les 2 pixels.

ATICK

Commandes associées : AXES, DRAX

ATTACH

Commande d'association d'une bibliothèque : Associe la bibliothèque portant le numéro spécifié au répertoire en cours. Chaque bibliothèque possède un numéro unique. Si le numéro de port est indiqué, le système l'ignore.

{ }

Niveau 1	→	Niveau 1
$n_{bibliothèque}$	→	
$:n_{port}\ :n_{bibliothèque}$	→	

Accès clavier : ◀ LIBRARY NXT ATTAC

Indicateurs : Aucun

Remarques : Pour utiliser un objet-bibliothèque, celui-ci doit être installé dans un port et lui être associé. Un objet-bibliothèque résidant sur une carte d'application (ROM) est automatiquement associé à un port (1 à 33), mais s'il s'agit d'un objet copié en RAM (par la liaison PC par exemple), il doit être stocké dans un port à l'aide de STO.

Bon nombre de bibliothèques sont associées automatiquement au moment de l'installation d'une carte d'application. Vous devez en associer d'autres avec ATTACH, comme les objets-bibliothèques copiés en RAM. (Le manuel de la carte d'application ou de la bibliothèque indique ceux qui doivent être associés manuellement). Vous pouvez aussi vérifier si une bibliothèque est associée ou non au répertoire en cours en exécutant LIBS.

Une bibliothèque copiée en RAM puis stockée (à l'aide de STO) dans un port ne peut être associée *que si vous éteignez puis rallumez le calculateur* à la suite de la commande STO. Cette action, en créant une *interruption système*, rend l'objet-bibliothèque "associable". Elle a également pour effet d'effacer le contenu de la pile, les

variables locales et la pile LAST, et d'afficher le menu MATH. (Pour sauvegarder la pile, exécutez DEPTH →LIST '*nom*' STO .)

Le nombre de bibliothèques qu'il est possible d'associer au répertoire HOME n'est limité que par la taille de la mémoire. Cependant, vous ne pouvez en associer qu'une à la fois à un autre répertoire. Si vous tentez d'associer une deuxième bibliothèque à un répertoire autre que *HOME*, elle remplacera la précédente.

Commandes associées : DETACH, LIBS

AUTO

Commande d'échelle automatique : Calcule une plage d'affichage pour l'axe y , ou pour les axes x et y.

Accès clavier : ◀ PLOT NXT AUTO

Indicateurs : Aucun

Remarques : L'action de la commande AUTO dépend du type de tracé :

Type de tracé	Action de la commande.
FUNCTION	Echantillonne l'équation figurant dans *EQ* à 40 valeurs de la variable indépendante, également espacées dans le domaine de traçage de l'axe x, supprime les points qui renvoient $\pm\infty$, puis définit la plage d'affichage de l'axe y de façon à inclure le maximum, le minimum, et l'origine.
CONIC	Définit l'échelle de l'axe y de sorte qu'elle soit égale à celle de l'axe x.
POLAR	Echantillonne l'équation figurant dans *EQ* à 40 valeurs de la variable indépendante, également espacées dans le domaine de traçage, supprime les points qui renvoient $\pm\infty$, puis définit les plages d'affichage des axes x et y de la même manière que pour le tracé FUNCTION.
PARAMETRIC	Procède comme pour le tracé POLAR.
TRUTH	N'a aucun effet.
BAR	Définit la plage d'affichage de l'axe x de 0 au nombre d'éléments contenus dans ΣDAT, plus 1. Définit la plage d'affichage de l'axe y en prenant les valeurs minimale et maximale de ces éléments. L'axe x est toujours inclus.

Type de tracé	Action de la commande
HISTOGRAM	Définit la plage d'affichage de l'axe x en fonction des valeurs minimale et maximale des éléments de ΣDAT. Définit la plage d'affichage de l'axe y de 0 au nombre de lignes contenues dans ΣDAT.
SCATTER	Définit la plage d'affichage de l'axe x en prenant les valeurs minimale et maximale de la colonne de variable indépendante (XCOL) contenue dans ΣDAT. Définit la plage d'affichage de l'axe y en prenant les valeurs minimale et maximale de la colonne de variable dépendante (YCOL).

AUTO n'a aucun effet sur les tracés en 3 dimensions.

AUTO calcule une plage d'affichage de l'axe y puis étend cette plage de telle sorte que les libellés de menus ne dissimulent pas le tracé obtenu.

AUTO ne permet pas de dessiner un tracé—pour cela, exécutez DRAW.

Exemple : Le programme « FUNCTION AUTO DRAW DRAX » définit le type de tracé FUNCTION, calcule l'échelle de l'axe y, trace l'équation stockée dans EQ et ajoute des axes au tracé.

Commandes associées : DRAW, *H, SCALE, SCLΣ, *W, XRNG, YRNG

AXES

Commande de contrôle des axes : Spécifie le point d'intersection des axes x et y, l'intervalle de graduation, et les libellés des axes. Ces informations sont enregistrées dans la variable réservée $PPAR$.

AXES

Niveau 1	→ Niveau 1
(x, y)	→
{ *(x, y) atick "libellé de l'axe x" "libellé de l'axe y"* }	→

Accès clavier : ◀ PLOT PPAR NXT AXES

Indicateurs : Aucun

Remarques : L'argument pour la commande AXES (nombre complexe ou liste) est stocké comme cinquième paramètre de la variable réservée *PPAR*. L'utilisation de cet argument dépend de son type :

- Si l'argument est un nombre complexe, il remplace la valeur en cours contenue dans *PPAR*.
- Si l'argument est une liste contenant une ou toutes les variables ci-dessus, seules celles qui sont spécifiées sont affectées.

atick a le même format que l'argument de la commande ATICK. Il s'agit de la variable affectée par la commande ATICK.

La valeur par défaut pour AXES est `(0,0)`.

Les libellés des axes ne sont pas affichés dans *PICT* tant que la commande LABEL n'est pas réexécutée.

Exemple : La séquence :

```
( (0,0) 2 "t" "y" ) AXES LABEL
```

spécifie une intersection des axes à `(0,0)`, un intervalle de graduation toutes les 2 unités, et place les libellés `t` et `y` dans *PICT*. Ces libellés sont positionnés de façon à identifier respectivement les axes horizontal et vertical.

Commandes associées : ATICK, DRAW, DRAX, LABEL

BAR

Commande de type de tracé Bar : Définit le type de tracé BAR.

Accès clavier : ◀ PLOT NXT STAT PTYPE BAR

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est BAR, la commande DRAW dessine un diagramme à barres en utilisant les données d'une colonne de la matrice statistique en cours (variable réservée ΣDAT). La colonne à tracer est spécifiée par la commande XCOL, et est stockée dans le premier paramètre de la variable réservée ΣPAR. Les paramètres de traçage sont indiqués dans la variable réservée *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé BAR, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant les coordonnées de l'angle inférieur gauche de *PICT* (la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant les coordonnées de l'angle supérieur droit de *PICT* (la plage d'affichage). La valeur par défaut est (6.5,3.2).
- *indep* est soit le nom d'un libellé de l'axe horizontal, soit une liste contenant un nom et deux nombres dont le plus petit spécifie l'emplacement horizontal de la première barre. La valeur par défaut de *indep* est *X*.
- *res* est un nombre réel indiquant la largeur de la barre en unités-utilisateur, ou un entier binaire spécifiant la largeur de la barre en pixels. La valeur par défaut est #0, pour une largeur de 1 en unités-utilisateur.
- *axes* est une liste contenant un ou plusieurs des éléments ci-après, dans l'ordre indiqué : un nombre complexe représentant les coordonnées en unités-utilisateur du point de départ du tracé, une liste des intervalles de graduation, et deux chaînes désignant les libellés des deux axes, horizontal et vertical. La valeur par défaut est (0,0).

BAR

- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de la commande BAR place le nom BAR dans *PPAR*.
- *depend* est un nom correspondant au libellé de l'axe vertical. La valeur par défaut est *Y*.

Une barre est tracée pour chaque élément de la colonne stockée dans *ΣDAT*. Sa largeur est indiquée par *res* et sa hauteur est liée à la valeur de l'élément. L'emplacement de la première barre peut être spécifié par *indep*; sinon, la valeur adoptée est $\langle x_{min}, y_{min} \rangle$.

Commandes associées : CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

BARPLOT

Commande de dessin d'un tracé Bar : Trace un diagramme à barres à partir de la colonne spécifiée de la matrice statistique en cours (variable réservée *ΣDAT*).

Accès clavier : ↩ STAT PLOT BARPLOT

Indicateurs : Aucun

Remarques : La colonne de données à représenter est spécifiée par la commande XCOL et constitue le premier paramètre de la variable réservée *ΣPAR*. Les données peuvent être positives ou négatives, donnant lieu à un diagramme dont les barres apparaissent au-dessous ou au-dessous de l'axe, selon le cas. L'échelle de l'axe *y* est automatiquement calculée et le type de tracé est défini à BAR.

Lorsque la commande BARPLOT est exécutée depuis un programme, le tracé obtenu ne sera pas conservé si vous n'exécutez pas immédiatement après la commande PICTURE, PVIEW (avec un argument de liste vide) ou FREEZE.

Commandes associées : FREEZE, HISTPLOT, PICTURE, PVIEW, SCATRPLOT, XCOL

BAUD

Commande de vitesse de transmission : Spécifie une vitesse de transmission des bits de données.

{ }

Niveau 1	→	Niveau 1
$n_{\text{vitessetransmission}}$	→	

Accès clavier : ↩ I/O IOPAR BAUD

Indicateurs : Aucun

Remarques : Les vitesses de transmission standard sont 1200, 2400, 4800 et 9600 (par défaut).

Pour plus d'informations, reportez-vous aussi à la variable réservée *IOPAR* (*paramètres d'E-S*) en Annexe D, "Variables réservées".

Commandes associées : CKSM, PARITY, TRANSIO

BEEP

Commande d'avertissement sonore : Emet un son pendant x secondes, à la fréquence n hertz.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$n_{\text{fréquence}}$	$x_{\text{durée}}$	→	

Accès clavier : PRG NXT OUT NXT BEEP

Indicateurs : Signal sonore d'erreur (−56)

Remarque : La fréquence du signal est fonction du générateur de sons intégré. Elle est au maximum d'environ 4400 Hz ; la durée

maximale est de 1048,575 secondes. les arguments dépassant ces maxima y sont automatiquement ramenés.

Commandes associées : HALT, INPUT, PROMPT, WAIT

BESTFIT

Commande de sélection du modèle le mieux adapté : Exécute LR avec chacun des quatre modèles d'ajustement de courbe, et sélectionne le modèle présentant le coefficient de corrélation le plus élevé.

Accès clavier : ◀ STAT ΣPAR MODL BESTF

Indicateurs : Aucun

Remarque : Le modèle sélectionné est stocké dans le cinquième paramètre de la variable réservée ΣPAR, et les coefficients de régression associés, ainsi que l'intersection et la pente, constituent respectivement les troisième et quatrième paramètres. Pour une description de ΣPAR, voir l'Annexe D, "Variables réservées".

Commandes associées : EXPFIT, LINFIT, LOGFIT, LR, PWRFIT

BIN

Commande de mode binaire : Sélectionne la base binaire pour les opérations sur des entiers binaires. La base par défaut est la base 10.

Accès clavier : MTH BASE BIN

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : Les entiers binaires doivent être précédés du préfixe #. Les entiers binaires saisis et renvoyés en base binaire contiennent automatiquement le suffixe b. Si la base en vigueur n'est pas binaire, il reste possible de taper des nombres binaires en utilisant le suffixe b (cependant, les nombres s'affichent dans la base en vigueur).

La base en cours n'affecte pas la représentation interne des entiers binaires comme nombres binaires non signés.

Commandes associées : DEC, HEX, OCT, STWS, RCWS

BINS

Commande de tri par bloc : Trie les éléments de la colonne indépendante (XCOL) de la matrice statistique en cours (variable réservée ΣDAT) en (n_{blocs} + 2) blocs, le bord gauche du bloc 1 commençant à la valeur x_{min} et chaque bloc étant large de $x_{largeur}$.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
x_{min}	x_{larg}	n_{bloc}	→	[[n_{bloc1} ··· n_{blocn}]]	[n_{blocL} n_{blocR}]

Accès clavier : ◀ STAT 1VAR BINS

Indicateurs : Aucun

Remarques : BINS renvoie une matrice contenant la fréquence des occurrences dans chaque bloc, et un tableau à deux éléments contenant la fréquence des occurrences se situant au-dessous ou au-dessus de la fourchette définie des valeurs x. Le tableau peut être stocké dans la variable réservée ΣDAT et utilisé pour tracer un histogramme représentant les données du bloc (par exemple au moyen de BARPLOT).

Pour chaque élément x de ΣDAT, le $n$ième numéro de bloc $n_{fréq\ bloc\ n}$ est incrémenté, où :

$$n_{frq\ bloc\ n} = IP\left[\frac{x - x_{min}}{x_{largeur}}\right]$$

pour $x_{min} \leq x \leq x_{max}$, où $x_{max} = x_{min} + (n_{blocs})(x_{largeur})$.

Exemple : Si la colonne indépendante de ΣDAT contient les données suivantes :

7 2 3 1 4 6 9 0 1 1 3 5 13 2 6 9 5 8 5

BINS

1 2 5 BINS renvoie [[5] [3] [5] [2] [2]] et [1 1].

Les données sont triées en 5 blocs d'une largeur de 2, en commençant par la valeur x 1 et en terminant par la valeur x 11. Le premier élément de la matrice montre que 5 valeurs x (2 1 1 1 2) vont dans le bloc 1, le bloc 1 allant de la valeur x 1 à 2.99999999999. Le vecteur montre qu'une valeur x est inférieure à x_{min} (0), et qu'une est supérieure à x_{max} (13).

Commandes associées : BARPLOT, XCOL

BLANK

Commande d'objet graphique vierge : Crée un objet graphique vierge de la largeur et de la hauteur spécifiées.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$\#n_{largeur}$	$\#m_{hauteur}$	→	$grob_{vierge}$

Accès clavier : PRG GROB BLAN

Indicateurs : Aucun

Commandes associées : →GROB, LCD→

BOX

Commande de fenêtre : Dessine dans *PICT* une fenêtre dont les angles opposés sont définis par les coordonnées spécifiées en unités-utilisateur ou en pixels.

{ }

Niveau 2	Niveau 1	→	Niveau 1
{ #n_1 #m_1 }	{ #n_2 #m_2 }	→	
(x_1 , y_1)	(x_2 , y_2)	→	

Accès clavier : PRG PICT BOX

Indicateurs : Aucun

Commandes associées : ARC, LINE, TLINE

BUFLEN

Commande de longueur de tampon : Renvoie le nombre de caractères contenus dans la mémoire tampon série du HP 48 et un chiffre indiquant si une erreur s'est produite durant la réception des données.

Niveau 1	→	Niveau 2	Niveau 1
	→	$n_{caractères}$	0/1

Accès clavier : ← I/O NXT SERIA BUFLE

Indicateurs : Aucun

Remarques : Le chiffre renvoyé au niveau 1 est 1 s'il ne s'est produit aucune erreur d'encadrement des bits, aucune surcharge de l'UART ni de dépassement de la capacité du tampon durant la réception. En revanche, si l'une de ces erreurs s'est produite, le chiffre renvoyé

est égal à 0. Le tampon peut contenir jusqu'à 255 octets. En cas d'erreur d'encadrement ou de surcharge, la réception des données est interrompue jusqu'à effacement de l'erreur (opération exécutée par BUFLEN) ; ainsi, *n* représente les données reçues *avant* l'erreur.

Lorsque BUFLEN renvoie la valeur 0 au niveau 1, utilisez ERRM pour déterminer la nature de l'erreur.

Commandes associées : CLOSEIO, OPENIO, SBRK, SRECV, STIME, XMIT

BYTES

Commande d'indication de taille des objets en octets : Renvoie le nombre d'octets et le total de contrôle d'un objet donné.

{}

Niveau 1	→	Niveau 2	Niveau 1
obj	→	$\#n_{totaldecontrôle}$	x_{taille}

Accès clavier : ◀ MEMORY BYTES

Indicateurs : Aucun

Remarques : Si l'argument est un objet intégré, sa taille est de 2,5 octets et le total de contrôle est # 0.

Si l'argument est un nom global, la taille représente le nom *et* le contenu, tandis que le total de contrôle ne représente que le contenu. La taille du nom seul est de $(3{,}5 + 2 \times n)$, où n est le nombre de caractères composant le nom.

Exemple : Les objets qui se décompilent de façon identique peuvent avoir une taille en octets et des totaux de contrôle différents. Par exemple,

```
« 1 »
```

et

```
1 'A' STO A « » +
```

génèrent tous les deux une liste contenant le nombre 1. Cependant, la première liste contient l'objet intégré 1 (d'une taille de 7,5 octets), la seconde contient une copie RAM de 1 (pour une taille de 15,5 octets).

Commande associée : MEM

B→R

Commande de conversion d'un entier binaire en nombre réel : Convertit un entier binaire en son équivalent à virgule flottante.

{ }

Niveau 1	→	Niveau 1
#*n*	→	*n*

Accès clavier : (MTH) BASE B→R

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarque : Si # *n* ≥ # 1000000000000 (base 10), seuls les 12 chiffres décimaux les plus significatifs sont conservés dans la mantisse.

Commande associée : R→B

CASE

Commande de structure conditionnnelle CASE : Commence une structure conditionnelle CASE ... END.

CASE

	Niveau 1	→	Niveau 1
CASE		→	
THEN	*V/F*	→	
END		→	
END		→	

Accès clavier : PRG BRCH CASE CASE

Indicateurs : Aucun

Remarques : La structure CASE ... END exécute une série de *tests*. Le premier test qui renvoie un résultat vrai déclenche l'exécution de la clause vraie correspondante, mettant fin à la structure CASE ... END. Il est possible d'inclure une clause par défaut, qui sera exécutée si tous les tests sont évalués comme étant faux.

La structure CASE ... END présente la syntaxe suivante :

```
CASE
        clause-de-test1 THEN clause-vraie1 END
        clause-de-test2 THEN clause-vraie2 END
        :
        clause-de-testn THEN clause-vraien END
        clause-par-défaut (facultative)
END
```

Lorsque CASE est exécutée, la *clause-de-test*$_1$ est évaluée. Si le test est vrai, la *clause-vraie*$_1$ est exécutée et le programme passe directement à END. Si la *clause-de-test*$_1$ est fausse, la *clause-de-test*$_2$ est exécutée. L'exécution au sein de la structure CASE continue jusqu'à ce qu'une clause vraie soit exécutée ou que toutes les clauses de test soient évaluées comme étant fausses. Si une clause par défaut est incluse, elle est exécutée si toutes les clauses de test ont été évaluées comme étant fausses.

Exemple : Le programme suivant prend un argument numérique dans la pile :

- si l'argument est négatif, il est ajouté à lui-même,

- si l'argument est positif, il prend le signe négatif,
- si l'argument est égal à zéro, le programme s'interrompt.

```
« → X
  « CASE
    'X>0'
    THEN X NEG END
    'X<0'
    THEN X DUP + END
    'X==0'
    THEN 0 DOERR END
  END
  »
»
```

Commandes associées : END, IF, IFERR, THEN

CEIL

Fonction plafond : Renvoie le plus petit entier supérieur ou égal à l'argument.

Niveau 1	→	Niveau 1
x	→	*n*
x_unité	→	*n_unité*
'*symb*'	→	'CEIL(*symb*)'

Accès clavier : (MTH) REAL (NXT) (NXT) CEIL

Indicateurs : Résultats numériques (−3)

Exemples : `3.2 CEIL` renvoie 4; `-3.2 CEIL` renvoie −3.

CEIL

Commandes associées : FLOOR, IP, RND, TRNC

CENTR

Commande de centrage : Ajuste les deux premiers paramètres figurant dans la variable réservée *PPAR*, ⟨x_{min}, y_{min}⟩ et ⟨x_{max}, y_{max}⟩, de telle sorte que le point correspondant à l'argument ⟨x, y⟩ soit le centre du tracé.

{}

Niveau 1	→	Niveau 1
(*x*, *y*)	→	
x	→	

Accès clavier : ◀ PLOT PPAR NXT CENT

Indicateurs : Aucun

Remarques : Le pixel central est situé en ligne 32, colonne 65 lorsque *PICT* possède sa taille par défaut (131 × 64).

Si l'argument est un nombre réel x, CENTR positionne le point central aux coordonnées ⟨x, 0⟩.

Commande associée : SCALE

CF

Commande de désarmement d'un indicateur : Désarme l'indicateur système ou utilisateur spécifié.

{ }

Niveau 1	→	Niveau 1
$n_{\text{numéroindicateur}}$	→	

Accès clavier :

◀ MODES FLAG CF

PRG TEST NXT NXT CF

Indicateurs : Aucun

Remarque : Les indicateurs définis par l'utilisateur sont numérotés de 1 à 64. Les indicateurs système sont numérotés de −1 à −64. Voir l'Annexe C qui fournit une liste des indicateurs système du HP 48 avec leurs numéros.

Commandes associées : FC?, FC?C, FS?, FS?C, SF

CHOOSE

Commande de création d'une liste déroulante personnalisée : Crée une liste déroulante définie par l'utilisateur.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
"*message*"	{ c_1 ... c_n }	n_{pos}	→	*obj* ou *résultat*	"*1*"
"*message*"	{ c_1 ... c_n }	n_{pos}	→		"*0*"

Accès clavier : PRG NXT IN CHOOS

Indicateurs : Aucun

CHOOSE

Remarques : CHOOSE crée une liste de sélection standard, avec les spécifications suivantes :

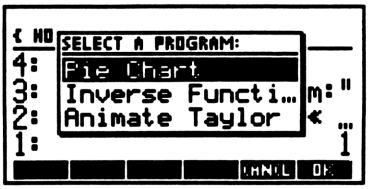
Variable	Fonction
"*message*"	Message qui apparaît en haut de la liste déroulante. Si "*message*" est une chaîne vide (""), aucun message ne s'affiche.
$\{c_1 \ \ldots \ c_n\}$	Définitions qui apparaissent dans la liste. Une définition (c_x) peut avoir deux formats : ■ *obj*, tout objet. ■ { $obj_{affichage}$ $obj_{résultat}$ }, l'objet à afficher suivi du résultat renvoyé dans la pile si cet objet est sélectionné.
n_{pos}	Numéro de position de la définition de l'élément. Ce dernier est mis en surbrillance lorsque la liste déroulante apparaît. Si n_{pos}=0, aucun n'est mis en surbrillance et la liste déroulante peut simplement servir à visualiser des éléments.

Si vous choisissez un élément de la liste déroulante et si vous appuyez sur OK , CHOOSE renvoie le *résultat* (ou l'objet lui-même si aucun résultat n'est spécifié) au niveau 2, et 1 au niveau 1. Si vous appuyez sur CANCL, CHOOSE renvoie 0. De même, si n_{pos}=0, CHOOSE renvoie 0.

Exemple : Avec les trois lignes suivantes :

```
"Select a Program:"
{ {"Pie Chart" «PIE»} {"Inverse Function" «ROOTR»}
{"Animate Taylor" «TSA»} }
1
```

CHOOSE donnerait lieu à l'affichage suivant :



Commandes associées : INFORM, NOVAL

%CH

Fonction de variation en pourcentage : Renvoie la variation entre x (niveau 2) et y (niveau 1) en pourcentage de x.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	*100(y−x)/x*
x	*'symb'*	→	*'%CH(x,symb)'*
'symb'	*x*	→	*'%CH(symb,x)'*
'$symb_1$'	*'$symb_2$'*	→	*'%CH($symb_1$,$symb_2$)'*
x_unité	*y_unité*	→	*100(y_unité−x_unité)/x_unité*
x_unité	*'symb'*	→	*'%CH(x_unité,symb)'*
'symb'	*x_unité*	→	*'%CH(symb,x_unité)'*

Accès clavier : (MTH) REAL %CH

Indicateurs : Résultats numériques (−3)

Remarques : Si les deux arguments sont des objets-unités, les unités doivent être cohérentes entre elles. Les dimensions d'un objet-unité n'apparaissent pas dans le résultat, *mais les unités font partie du calcul.*

%CH

Pour de plus amples informations sur l'utilisation des unités de température avec les fonctions arithmétiques, reportez-vous à la rubrique concernant l'opération +.

Exemples : `1_m 500_cm %CH` renvoie `400`, car 500 cm représente une augmentation de 400% par rapport à 1 m.

`100_K 150_K %CH` renvoie `50`.

Commandes associées : %, %T

CHR

Commande de code de caractère : Renvoie une chaîne représentant le caractère HP 48 correspondant au code *n*.

{ }

Niveau 1	→	Niveau 1
n	→	*" chaîne "*

Accès clavier :

(↩) (CHARS) CHR

(PRG) TYPE (NXT) CHR

Indicateurs : Aucun

Remarques : Les codes de caractères sont une extension de l'ISO 8859/1. Les codes 128 à 159 sont uniques au HP 48. Voir la rubrique NUM dans le présent manuel pour avoir une liste complète des caractères et des codes afférents.

Le caractère par défaut ▪ est fourni pour tous les codes de caractères qui ne font *pas* partie du jeu de caractères standard du HP 48.

Le code 0 a une fonction spécifique : marquer la fin de la ligne de commande. Si vous tentez de modifier une chaîne contenant ce caractère, vous obtenez le message d'erreur suivant : `Can't Edit CHR(0)`.

L'application CHARS permet de retrouver le code de n'importe quel caractère utilisé par le HP 48. Voir la section "Saisie de caractères spéciaux" dans le chapitre 2 du *Manuel d'utilisation du HP 48.*

Commandes associées : NUM, POS, REPL, SIZE, SUB

CKSM

Commande de total de contrôle : Spécifie le type de détection d'erreur utilisé.

{ }

Niveau 1	→	Niveau 1
$n_{totaldecontrôle}$	→	

Accès clavier : ◀ I/O IOPAR CKSM

Indicateurs : Aucun

Remarques : Les valeurs possibles pour $n_{totaldecontrôle}$ sont les suivantes :

- 1 : total de contrôle arithmétique à 1 chiffre.
- 2 : total de contrôle arithmétique à 2 chiffres.
- 3 : contrôle par redondance cyclique à 3 chiffres (par défaut).

La commande CKSM spécifiée correspond au type de détection d'erreur qui sera demandé par KGET, PKT ou SEND. S'il y a désaccord sur la demande entre l'émetteur et le récepteur, la détection par total de contrôle arithmétique à 1 chiffre est appliqué.

Dans le cas des transmissions infrarouges, il faut utiliser le type 3.

Commandes associées : BAUD, PARITY, TRANSIO

CLEAR

Commande d'effacement : Supprime tous les objets de la pile.

Niveau n ... Niveau 1	→	Niveau n ... Niveau 1
obj_n ... obj_1	→	

Accès clavier : ◀ CLEAR

Indicateurs : Aucun

Remarque : Pour récupérer le contenu de la pile après son effacement, appuyez sur ▶ UNDO avant d'exécuter toute autre opération. Il n'existe pas de commande programmable pour récupérer les données de la pile.

Commandes associées : CLVAR, PURGE

CLKADJ

Commande de réglage de l'horloge système: Règle l'heure système de x tops d'horloge, sachant que 8192 tops d'horloge correspondent à 1 seconde.

{ }

Niveau 1	→	Niveau 1
x	→	

Accès clavier : ◀ TIME NXT NXT CLKA

Indicateurs : Aucun

Remarque : Si x est positif, x tops d'horloge sont ajoutés à l'heure système. Si x est négatif, x tops d'horloge sont soustraits.

Exemple : -20480 CLKADJ diminue l'heure système de 2,5 secondes.

Commande associée : →TIME

CLLCD

Commande d'effacement de l'affichage : Vide l'affichage de la pile.

Accès clavier : PRG NXT OUT CLLCD

Indicateurs : Aucun

Remarques : Les libellés de menus restent affichés après exécution de CLLCD.

Lorsque cette commande est exécutée depuis un programme, l'affichage ainsi vidé dure jusqu'à ce que le contrôle soit rendu au clavier pour la saisie. Si vous souhaitez qu'il se prolonge jusqu'à activation d'une touche, exécutez FREEZE à la suite de CLLCD. (Exécutée depuis le clavier, CLLCD gèle *automatiquement* l'affichage.)

Exemple : L'évalution de « CLLCD 7 FREEZE » vide l'affichage (sauf les libellés de menus), puis le gèle en totalité.

Commandes associées : DISP, FREEZE

CLOSEIO

Commande de fermeture d'un port d'E-S : Ferme le port série et le port IR, et efface le contenu du tampon d'entrée ainsi que les messages d'erreur de KERRM.

Accès clavier : ← I/O NXT CLOSE

Indicateurs : Aucun

Remarques : Lorsque vous éteignez le HP 48, il ferme automatiquement les ports série et IR, mais n'efface pas KERRM. Ainsi, il n'est pas obligatoire d'exécuter CLOSEIO pour fermer les

CLOSEIO

ports, mais son utilisation permet d'économiser l'énergie sans mettre le calculateur hors tension.

L'exécution des commandes Kermit du HP 48 efface automatiquement le contenu du tampon d'entrée ; ce qui n'est pas le cas des commandes non Kermit (comme SRECV et XMIT).

CLOSEIO efface également les messages d'erreur de KERRM, ce qui peut s'avérer utile en cas de débogage.

Commandes associées : BUFLEN, OPENIO

CLΣ

Commande d'effacement de données statistiques : Supprime les données statistiques de la variable réservée ΣDAT.

Accès clavier : ◀ STAT DATA CLΣ

Indicateurs : Aucun

Commandes associées : RCLΣ, STOΣ, Σ+, Σ–

CLTEACH

Commande de suppression des exemples du didacticiel : Supprime le sous-répertoire EXAMPLES et son contenu du répertoire HOME.

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Commande associée : TEACH

CLUSR

Commande d'effacement de variables : Cette commande est fournie à des fins de compatibilité avec le HP 28. CLUSR est l'équivalent de CLVAR (voir cette dernière).

CLVAR

Commande d'effacement de variables : Supprime toutes les variables et les sous-répertoires vides du répertoire en cours.

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Commandes associées : CLUSR, PGDIR, PURGE

CNRM

Commande de norme de colonne : Renvoie la norme de colonne (norme 1) du tableau.

{ }

Niveau 1	→	Niveau 1
[*tableau*]	→	$x_{normedecolonne}$

Accès clavier : (MTH) MATR NORM CNRM

Indicateurs : Aucun

Remarques : La norme de colonne d'une matrice est la valeur maximale (pour toutes les colonnes) des sommes des valeurs absolues de tous les éléments d'une colonne. Pour un vecteur, elle est la somme des valeurs absolues de ses éléments. Pour les tableaux complexes, la valeur absolue d'un élément donné (x, y) est $\sqrt{x^2 + y^2}$.

CNRM

Commandes associées : CROSS, DET, DOT, RNRM

→COL

Commande de conversion d'une matrice en colonnes :
Transforme une matrice en une série de vecteurs-colonnes et renvoie les vecteurs ainsi qu'un nombre de colonnes, ou transforme un vecteur en ses éléments et renvoie les éléments ainsi qu'un nombre d'éléments.

{ }

Niveau 1	→	Niveau n+1 ...	Niveau 2	Niveau 1
[[*matrice*]]	→	[*vecteur*]$_{col1}$	[*vecteur*]$_{coln}$	$n_{nbrecol}$
[*vecteur*]	→	*élément*$_1$	*élément*$_n$	$n_{nbreéléments}$

Accès clavier : (MTH) MATR COL →COL

Indicateurs : Aucun

Remarque : →COL n'introduit aucune erreur d'arrondi.

Commandes associées : COL→, →ROW, ROW→

COL+

Commande d'insertion de colonnes : Insère un tableau (vecteur ou matrice) dans une matrice (ou un ou plusieurs éléments dans un vecteur) à la position indiquée par n_{index}, et renvoie le tableau modifié.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	[*matrice*]$_2$	n_{index}	→	[[*matrice*]]$_3$
[[*matrice*]]$_1$	[*vecteur*]$_{colonne}$	n_{index}	→	[[*matrice*]]$_2$
[*vecteur*]$_1$	$n_{élément}$	n_{index}	→	[*vecteur*]$_2$

Accès clavier : (MTH) MATR COL COL+

Indicateurs : Aucun

Remarques : Le tableau inséré doit avoir le même nombre de lignes que le tableau cible.

n_{index} est arrondi à l'entier le plus proche. Le tableau initial est redimensionné afin de contenir les nouvelles colonnes ou éléments. Les éléments qui se trouvaient au point d'insertion, ainsi que les colonnes de droite sont décalés vers la droite.

Commandes associées : COL–, CSWP, ROW+, ROW–

COL–

Commande de suppression de colonnes : Supprime la colonne n d'une matrice (ou l'élément n d'un vecteur) et renvoie la matrice modifiée (ou le vecteur), ainsi que la colonne supprimée (ou l'élément).

{ }

Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
[[*matrice*]]$_1$	$n_{colonne}$	→	[[*matrice*]]$_2$	[*vecteur*]$_{colonne}$
[*vecteur*]$_1$	$n_{élément}$	→	[*vecteur*]$_2$	*élément*$_n$

Accès clavier : (MTH) MATR COL COL–

Indicateurs : Aucun

Remarque : n est arrondi à l'entier le plus proche.

COL–

Commandes associées : COL+, CSWP, ROW+, ROW–

COL→

Commande de conversion de colonnes en matrice : Transforme une série de vecteurs-colonnes et un nombre de colonnes en une matrice contenant ces colonnes, ou transforme une série de nombres et un nombre d'éléments en un vecteur dont les éléments seront ces nombres.

Niveau n+1 ...	Niveau 2	Niveau 1	→	Niveau 1
[*vecteur*]$_{colonne1}$	[*vecteur*]$_{colonnen}$	$n_{nbrecolonnes}$	→	[[*matrice*]]
$élément_1$	$élément_n$	$n_{nbreéléments}$	→	[*vecteur*]

Accès clavier : (MTH) MATR COL COL→

Indicateurs : Aucun

Remarque : Tous les vecteurs doivent avoir la même longueur. Le nombre de colonnes ou d'éléments est arrondi à l'entier le plus proche.

Commandes associées : →COL, →ROW, ROW→

COLCT

Commande de regroupement des termes identiques : Simplifie une expression algébrique ou une équation en "regroupant" les termes identiques.

{ }

Niveau 1	→	Niveau 1
'$symb_1$'	→	'$symb_2$'
x	→	x
(x, y)	→	(x, y)

Accès clavier : ◀ SYMBOLIC COLCT

Indicateurs : Aucun

Remarque : COLCT fonctionne séparément sur les deux membres d'une équation, de sorte que des termes identiques de part et d'autre de l'équation ne sont pas regroupés.

Exemples : `'6+EXP(10)' COLCT` renvoie `8.71828182846`.

`'5+X+9' COLCT` renvoie `'14+X'`.

`'X*1_m+X*9_cm' COLCT` renvoie `'(109_cm)*X'`.

`'X^Z*Y*X^T*Y' COLCT` renvoie `'X^(T+Z)*Y^2'`.

`'X+3*X+Y+Y' COLCT` renvoie `'4*X+2*Y'`.

Commandes associées : EXPAN, ISOL, QUAD, SHOW

COLΣ

Commande de colonnes dépendantes et indépendantes : Spécifie les colonnes de variables indépendantes et dépendantes de la matrice statistique en cours (variable réservée ΣDAT).

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_{xcol}	x_{ycol}	→	

Accès clavier : Aucun. A saisir.

COLΣ

Indicateurs : Aucun

Remarques : COLΣ combine les fonctionnalités de XCOL et YCOL. Elle est fournie dans le HP 48 à des fins de compatibilité avec le HP 28S.

Le numéro de colonne indépendante x_{xcol} est stocké comme premier paramètre de la variable réservée ΣPAR (le numéro par défaut est 1). Le numéro de colonne dépendante x_{ycol} constitue le second paramètre de ΣPAR (le numéro par défaut est 2).

COLΣ accepte et stocke les nombres réels non entiers, mais les commandes qui, après elle, utilisent ces deux paramètres dans ΣPAR provoquent des erreurs.

Exemple : 2 5 COLΣ définit la colonne 2 dans ΣDAT comme la colonne de variable indépendante et la colonne 5 comme colonne de variable dépendante, puis elle stocke 2 et 5 comme les premier et second éléments de ΣPAR.

Commandes associées : BARPLOT, BESTFIT, CORR, COV, EXPFIT, HISTPLOT, LINFIT, LOGFIT, LR, PREDX, PREDY, PWRFIT, SCATRPLOT, XCOL, YCOL

COMB

Fonction de combinaison : Renvoie le nombre de combinaisons possibles de n éléments prélevés par quantités de m à la fois.

{ }

Niveau 2	Niveau 1	→	Niveau 1
n	m	→	$C_{n;m}$
'$symb_n$'	m	→	'COMB($symb_n$, m)'
n	'$symb_m$'	→	'COMB(n, $symb_m$)'
'$symb_n$'	'$symb_m$'	→	'COMB($symb_n$, $symb_m$)'

Accès clavier : MTH NXT PROB COMB

Indicateurs : Résultats numériques (−3)

Remarque : La formule suivante sert à calculer $C_{n,m}$.

$$C_{n,m} = \frac{n!}{m!(n-m)!}$$

Les arguments n et m doivent être inférieurs à 10^{12}.

Commandes associées : PERM, !

CON

Commande de tableau de constantes : Renvoie un tableau de constantes, dont les éléments ont tous la même valeur.

Niveau 2	Niveau 1	→	Niveau 1
{ $n_{colonnes}$ }	$z_{constante}$	→	[$vecteur_{constante}$]
{ n_{lignes} $m_{colonnes}$ }	$z_{constante}$	→	[[$matrice_{constante}$]]
[*R-tableau*]	$x_{constante}$	→	[*R-tableau*$_{constante}$]
[*C-tableau*]	$z_{constante}$	→	[*C-tableau*$_{constante}$]
'nom'	$z_{constante}$	→	

Accès clavier : (MTH) MATR MAKE CON

Indicateurs : Aucun

Remarques : La valeur de la constante est un nombre réel ou complexe pris au niveau 1. Il en résulte un tableau entièrement nouveau, ou un tableau qui existait déjà mais dont les éléments sont remplacés par la constante, selon l'objet figurant au niveau 2.

- **Création d'un nouveau tableau :** Si le niveau 2 contient une liste d'un ou deux entiers, CON renvoie un nouveau tableau. Si la liste contient un seul entier $n_{colonnes}$, CON renvoie un vecteur de constantes de n éléments. Si la liste contient deux entiers n_{lignes} et $m_{colonnes}$, CON renvoie une matrice de constantes avec n lignes et m colonnes.

CON

- **Remplacement des éléments d'un tableau existant :** Si le niveau 2 contient un tableau, CON renvoie un tableau de mêmes dimensions, dont chaque élément est égal à la constante. Si la constante est un nombre complexe, le tableau d'origine doit aussi être complexe.

  Si le niveau 2 contient un nom, celui-ci doit identifier une variable contenant un tableau. Dans ce cas, les éléments du tableau sont remplacés par la constante. Si la constante est un nombre complexe, le tableau d'origine doit aussi être complexe.

Exemples : `< 2 2 > 6 CON` renvoie la matrice `[[ 6 6 ][ 6 6 ]]`.

`[ (2,4) (7,9) ] 3 CON` renvoie le vecteur complexe `[ (3,0) (3,0) ]`.

Commande associée : IDN

COND

Commande du nombre de condition : Renvoie le nombre de condition de la norme 1 (norme de colonne) d'une matrice carrée.

{}

Niveau 1	→	Niveau 1
[[*matrice*]]$_{n \times n}$	→	$x_{nombredecondition}$

Accès clavier : (MTH) MATR NORM COND

Indicateurs : Aucun

Remarques : Le nombre de condition d'une matrice est le produit de la norme de la matrice par la norme de colonne de son inverse. COND utilise la norme de colonne et calcule le nombre de condition de la matrice sans calculer son inverse.

Le nombre de condition exprime la sensibilité du problème de la résolution d'un système d'équations linéaires dont les coefficients sont représentés par les éléments de la matrice (ceci inclut l'inversion de la matrice). Il indique donc jusqu'à quel point une erreur en entrée peut être amplifiée dans le résultat des calculs utilisant la matrice.

Dans plusieurs calculs algébriques linéaires, le logarithme en base 10 du nombre de condition de la matrice est une estimation du nombre de chiffres de précision qui risquent d'être perdus dans les calculs utilisant cette matrice. Il paraît raisonnable d'estimer le nombre des chiffres de précision à environ MIN(12,15−$\log_{10}$(COND)).

Exemple : Selon la règle approximative ci-dessus, le programme suivant calcule ainsi le nombre de chiffres de précision :

```
«
 DUP SIZE 1 GET LOG SWAP COND LOG + 11 SWAP -
»
```

Commandes associées : SNRM, SRAD, TRACE

CONIC

Commande de type de tracé Conic : Définit le type de tracé CONIC.

Accès clavier : ◀ PLOT PTYPE CONIC

Indicateurs : Aucun

Remarques : Lorsque le tracé est de type CONIC, la commande DRAW représente l'équation en cours comme le polynôme de second degré de deux variables réelles. L'équation en cours est spécifiée dans la variable réservée *EQ*. Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour un tracé de type CONIC, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant les coordonnées de l'angle inférieur gauche de la place d'affichage *PICT*. La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant les coordonnées de l'angle supérieur droit de la plage d'affichage *PICT*. La valeur par défaut est (6.5,3.2).

- *indep* est le nom de la variable indépendante ou d'une liste contenant un nom et deux nombres spécifiant les valeurs minimum et maximum de la variable indépendante (domaine de traçage). La valeur par défaut est *X*.
- *res* est un nombre réel spécifiant l'intervalle (en unités-utilisateur) séparant deux valeurs tracées de la variable indépendante, ou un entier binaire précisant cet intervalle en pixels. La valeur par défaut est 0, soit un intervalle de 1 pixel.
- *axes* est un nombre complexe spécifiant les coordonnées, en unités-utilisateur, de l'intersection des axes, ou une liste contenant un nombre et les libellés (chaînes) des deux axes. La valeur par défaut est (0,0).
- *ptype* est un nom de commande spécifiant le type de tracé. Lorsque CONIC est exécutée, le nom de commande CONIC est placé dans *PPAR*.
- *depend* est le nom de la variable dépendante. La valeur par défaut est *Y*.

L'équation en cours sert à définir une paire de fonctions de la variable indépendante. Ces fonctions sont dérivées de l'estimation de Taylor au second degré pour l'équation en cours. Les valeurs minimum et maximum de la variable indépendante (domaine de traçage) peuvent être spécifiées dans *indep*; sinon, les valeurs de (x_{min}, y_{min}) et (x_{max}, y_{max}) (plage d'affichage) sont utilisées. Des lignes relient les points tracés, à moins que l'indicateur −31 soit armé.

Commandes associées : BAR, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

CONJ

Fonction analytique de conjugaison : Renvoie le conjugué d'un nombre ou d'un tableau complexe.

{ }

Niveau 1	→	Niveau 1
x	→	*x*
(*x*, *y*)	→	(*x*, −*y*)
[*R-tableau*]	→	[*R-tableau*]
[*C-tableau*]$_1$	→	[*C-tableau*]$_2$
'*symb*'	→	'CONJ(*symb*)'

Accès clavier : MTH NXT CMPL NXT CONJ

Indicateurs : Résultats numériques (−3)

Remarques : Le conjugué est l'opposé (inversion de signe) de la partie imaginaire d'un nombre complexe. Pour les nombres et les tableaux réels, le conjugué est identique à l'argument d'origine.

Exemple : `[ (3,4) (7,2) ] CONJ` renvoie `[ (3,-4) (7,-2) ]`.

Une matrice carrée A contenant des éléments complexes est dite *hermitienne* si $A^H = A$, où A^H équivaut à une transposition normale, si ce n'est que le conjugué complexe de chaque élément est utilisé. Le programme suivant renvoie 1 si la matrice en entrée est hermitienne, et 0 dans le cas contraire.

```
« DUP TRN CONJ SAME »
```

Commandes associées : ABS, IM, RE, SCONJ, SIGN

CONLIB

Commande d'ouverture d'une bibliothèque de constantes : Ouvre le catalogue de la bibliothèque des constantes.

Accès clavier : ◀ EQ LIB COLIB CONLI

Indicateurs : Aucun

Commande associée : CONST

CONST

Commande d'évaluation d'une constante : Renvoie la valeur d'une constante.

{ }

Niveau 1	→	Niveau 1
'nom'	→	*x*

Accès clavier : ◀ EQ LIB COLIB CONS

Indicateurs : Système d'unités (60), Emploi des unités (61)

Remarques : CONST renvoie la valeur de la constante spécifiée. Elle choisit le type d'unité en fonction de l'indicateur 60 (désarmé : unités SI, armé : unités anglo-saxonnes) et utilise les unités conformément à l'indicateur 61 (désarmé : emploi des unités, armé : pas d'unités).

Reportez-vous à la section "Bibliothèque de constantes" au chapitre 25 du *Manuel d'utilisation du HP 48*, qui fournit une liste des constantes disponibles.

Commande associée : CONLIB

CONT

Commande de reprise de l'exécution d'un programme : Relance un programme interrompu.

Accès clavier : ◀ CONT

Indicateurs : Aucun

Remarque : CONT étant une commande, il est possible de l'affecter à une touche ou à un menu personnalisé.

Exemple : Le programme :

```
"Tapez A, appuyez sur ‹ CONT ›" ‹ CONT › MENU PROMPT »
```

affiche un message, crée un menu dont la première touche est associée à la commande CONT, et interrompt le programme pour permettre la saisie des données. Une fois la saisie terminée, il suffit d'appuyer sur CONT pour reprendre l'exécution du programme. (Le fait d'appuyer sur ◀ CONT équivaut à une pression sur CONT .)

Commandes associées : HALT, KILL, PROMPT

CONVERT

Commande de conversion d'unités : Convertit un objet-unité source en dimensions exprimées en une unité cible.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_1 _unités$_{source}$	x_2 _unités$_{cible}$	→	x_3 _unités$_{cible}$

Accès clavier : ◀ UNITS CONV

Indicateurs : Aucun

Remarque : Les unités source et cible doivent être compatibles. La partie numérique x_2 de l'objet-unité cible est ignorée.

CONVERT

Commandes associées : UBASE, UFACT, →UNIT, UVAL

CORR

Commande de corrélation : Renvoie le coefficient de corrélation des colonnes de données dépendantes et indépendantes dans la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{\text{corrélation}}$

Accès clavier : ◀ STAT FIT CORR

Indicateurs : Aucun

Remarques : Les colonnes sont spécifiées par les deux premiers éléments de la variable réservées ΣPAR, définis respectivement par XCOL et YCOL. Si ΣPAR n'existe pas, CORR la crée et définit les éléments avec leurs valeurs par défaut (1 et 2).

La corrélation est calculée à l'aide de la formule suivante :

$$\frac{\sum_{i=1}^{n}(x_{in_1}-\overline{x}_{n_1})(x_{in_2}-\overline{x}_{n_2})}{\sqrt{\sum_{i=1}^{n}(x_{in_1}-\overline{x}_{n_1})^2\sum_{i=1}^{n}(x_{in_2}-\overline{x}_{n_2})^2}}$$

où x_{in_1} est la $i$ième valeur de coordonnées dans la colonne n_1, x_{in_2} est la $i$ème valeur de cordonnées dans la colonne n_2, $\overline{x}_{n_1}$ est la moyenne des données de la colonne n_1, $\overline{x}_{n_2}$ est la moyenne des données de la colonne n_2, et n est le nombre de points de données.

Commandes associées : COLΣ, COV, PREDX, PREDY, XCOL, YCOL

COS

Fonction analytique cosinus : Renvoie le cosinus de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	cos *z*
'*symb*'	→	'COS(*symb*)'
x_unité$_{\text{angulaire}}$	→	cos (*x_unité*$_{\text{angulaire}}$)

Accès clavier : (COS)

Indicateurs : Résultats numériques (−3), Mode d'angle (−17, −18)

Remarques : Pour des arguments réels, le mode d'angle en cours détermine l'interprétation du nombre comme angle, à moins que ne soient spécifiées les unités angulaires.

Pour des arguments complexes, $\cos(x + iy) = \cos x \cosh y - i \sin x \sinh y$.

Si l'argument de COS est un objet-unité, l'unité angulaire spécifiée remplace le mode d'angle pour déterminer le résultat. En revanche, l'intégration et la différenciation respectent toujours le mode d'angle. Ainsi, pour une intégration ou une différenciation correcte d'exprressions contenant COS avec un objet-unité, le mode d'angle doit être Radians (puisque qu'il s'agit d'un mode "neutre").

Commandes associées : ACOS, SIN, TAN

COSH

Fonction analytique cosinus hyperbolique : Renvoie le cosinus hyperbolique de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	cosh *z*
'*symb*'	→	'COSH(*symb*)'

Accès clavier : (MTH) HYP COSH

Indicateurs : Résultats numériques (−3)

Remarque : Pour des arguments complexes, $\cosh(x + iy) = \cosh x \cos y + i \sinh x \sin y$.

Commandes associées : ACOSH, SINH, TANH

COV

Commande de covariance : Renvoie la covariance des colonnes de données dépendantes et indépendantes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{covariance}$

Accès clavier : (←)(STAT) FIT COV

Indicateurs : Aucun

Remarques : Les colonnes sont spécifiées par les deux premiers éléments de la variable réservée ΣPAR, définis respectivement par

XCOL et YCOL. Si *ΣPAR* n'existe pas, COV la crée et définit les éléments avec leurs valeurs par défaut (1 et 2).

La covariance est calculée à l'aide de la formule suivante :

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_{in_1}-\overline{x_{n_1}})(x_{in_2}-\overline{x_{n_2}})$$

où x_{in_1} est la $i$ème valeur de coordonnées figurant dans la colonne n_1, x_{in_2} est la $i$ème valeur de coordonnées dans la colonne n_2, $\overline{x_{n_1}}$ est la moyenne des données de la colonne n_1, $\overline{x_{n_2}}$ est la moyenne des données de la colonne n_2, et n est le nombre de points de données.

Commandes associées : COLΣ, CORR, PCOV, PREDX, PREDY, XCOL, YCOL

CR

Commande d'impression : Imprime le contenu du tampon d'impression.

Accès clavier : ↩ I/O PRINT CR

Indicateurs : Impression en double interligne (−37), Unité d'impression (−34), Unité d'E-S (−33)

Si l'indicateur −34 est armé (la sortie est directement envoyée sur le port série), l'indicateur −33 doit être désarmé.

Remarques : Si vous utilisez une imprimante infrarouge HP 82240B (indicateur −34 désarmé), CR laisse la tête d'impression à la fin de la ligne qui vient de s'imprimer.

En cas d'impression sur le port série (indicateur −34 armé), CR envoie à l'imprimante une chaîne destinée à coder le mode de fin de ligne. Par défaut, un retour chariot/un saut de ligne est introduit. La chaîne correspond au quatrième paramètre de la variable réservée *PRTPAR*.

Commandes associées : DELAY, OLDPRT, PRLCD, PRST, PRSTC, PRVAR, PR1

CRDIR

Commande création de répertoire : Crée dans le répertoire en cours un sous-répertoire vide, portant le nom spécifié.

{ }

Niveau 1	→	Niveau 1
'global'	→	

Accès clavier : ◀ MEMORY DIR CRDIR

Indicateurs : Aucun

Remarques : Lorsque vous exécutez CRDIR, le répertoire en cours ne change pas. Pour que le nouveau sous-répertoire soit activé, vous devez évaluer son nom.

Commandes associées : HOME, PATH, PGDIR, UPDIR

CROSS

Commande de produit vectoriel : CROSS renvoie le produit vectoriel $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ des vecteurs A et B.

{ }

Niveau 2	Niveau 1	→	Niveau 1
[*vecteur*]$_A$	[*vecteur*]$_B$	→	[*vecteur*]$_{A \times B}$

Accès clavier : MTH VECTR CROSS

Indicateurs : Aucun

Remarque : Les arguments doivent être des vecteurs à deux ou trois éléments, sans pour autant contenir le même nombre d'éléments. (Le

HP 48 convertit automatiquement un argument à deux éléments [d_1 d_2] en argument à trois éléments [d_1 d_2 0].)

Commandes associées : CNRM, DET, DOT, RNRM

CSWP

Commande de permutation de colonnes : Permute les colonnes i et j de la matrice et renvoie celle-ci après modification, ou permute les éléments i et j du vecteur et le renvoie une fois modifié.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	n_{colonnei}	n_{colonnej}	→	[[*matrice*]]$_2$
[*vecteur*]$_1$	$n_{\text{élémenti}}$	$n_{\text{élémentj}}$	→	[*vecteur*]$_2$

Accès clavier : (MTH) MATR COL CSWP

Indicateurs : Aucun

Remarque : Les numéros de colonnes sont arrondis à l'entier le plus proche. Les arguments vecteurs sont traités comme des vecteurs-lignes.

Commandes associées : COL+, COL−, RSWP

CYLIN

Commande de mode cylindrique : Définit le mode de coordonnées cylindriques.

Accès clavier :

◀ MODES ANGL CYLIN

MTH VECTR NXT CYLIN

Indicateurs : Aucun

Remarques : CYLIN désarme l'indicateur −15, arme l'indicateur −16, et affiche le témoin R∠Z.

En mode de coordonnées cylindriques, les vecteurs sont affichés comme composants polaires. Ainsi, un vecteur 3D apparaîtra comme [$R \angle \theta$ Z].

Commandes associées : RECT, SPHERE

C→PX

Commande de conversion de coordonnées en pixels : Convertit en pixels des coordonnées spécifiées en unités-utilisateur.

{ }

Niveau 1	→	Niveau 1
(*x*, *y*)	→	{ #*n* #*m* }

Accès clavier : PRG PICT NXT C→PX

Indicateurs : Aucun

Remarque : Les coordonnées en unités-utilisateur sont fournis par les paramètres (x_{min}, y_{min}) et (x_{max}, y_{max}) de la variable réservée *PPAR*.

Commandes associées : PX→C

C→R

Commande de séparation de nombres complexes en nombres réels : Sépare les parties réelles et imaginaires d'un nombre ou d'un tableau complexe.

{ }

Niveau 1	→	Niveau 2	Niveau 1
(*x*, *y*)	→	*x*	*y*
[*C-tableau*]	→	[*R-tableau*]$_1$	[*R-tableau*]$_2$

Accès clavier :

MTH NXT CMPL C→R

PRG TYPE NXT C→R

Indicateurs : Aucun

Remarque : Le résultat envoyé au niveau 2 représente la partie réelle de l'argument complexe. Le résultat au niveau 1 en représente la partie imaginaire.

Commandes associées : R→C, RE, IM

DARCY

Fonction facteur de friction Darcy : Calcule le facteur de friction Darcy d'un fluide en circulation.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$x_{e/D}$	y_{Re}	→	x_{Darcy}

Accès clavier : ↶ EQ LIB UTILS DARCY

DARCY

Indicateurs : Aucun

Remarques : DARCY calcule le facteur de friction Fanning et le multiplie par 4. $x_{e/D}$ est la rugosité relative—rapport entre la rugosité du conduit et son diamètre. y_{Re} est le nombre de Reynolds. La fonction utilise des routines de calcul différentes pour un écoulement laminaire ($Re \leq 2100$) et un écoulement turbulent ($Re > 2100$). $x_{e/D}$ et y_{Re} doivent être des nombres réels ou des objets-unités réductibles à des nombres sans dimension, et être supérieurs à 0.

Commande associée : FANNING

DATE

Commande relative à la date : Renvoie la date système au niveau 1.

Niveau 1	→	Niveau 1
	→	*date*

Accès clavier : ↩ TIME DATE

Indicateurs : Format de date (−42)

Exemple : Si la date en cours est le 12 mai 1990, si l'indicateur −42 est armé et si le mode d'affichage est standard, DATE renvoie `5.12199`. (Les zéros à droite sont ignorés.)

Commandes associées : DATE+, DDAYS, TIME, TSTR

→DATE

Commande de définition de la date : Définit la date système à *date*.

{ }

Niveau 1	→	Niveau 1
date	→	

Accès clavier : ◀ TIME →DAT

Indicateurs : Format de date (−42)

Remarques : *date* prend le format *MM.JJAAAA* ou *JJ.MMAAAA*, selon l'état de l'indicateur −42. *MM* est le mois, *JJ* le jour et *AAAA* l'année. Si *AAAA* n'est pas indiqué, la spécification en cours est utilisée. La fourchette des dates possibles va du 1er janvier 1991 au 31 décembre 2090.

Exemple : Si l'indicateur −42 est armé et si l'année système en cours est 1995, `28.07 →DATE` définit le 28 juillet 1995 comme date système.

Commande associée : →TIME

DATE+

Commande d'addition de date : Renvoie une date passée ou future, en fonction de la date spécifiée au niveau 2 et du nombre de jours mentionné au niveau 1.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$date_1$	x_{jours}	→	$date_{nouvelle}$

Accès clavier : ◀ TIME NXT DATE+

DATE+

Indicateurs : Format de date (−42)

Remarques : Si x_{jours} est négatif, DATE+ calcule une date dans le passé. La fourchette des dates possibles va du 15 octobre 1582 au 31 décembre 9999.

Commandes associées : DATE, DDAYS

DBUG

Opération de débogage : Démarre l'exécution d'un programme, puis l'interrompt comme si HALT était la première commande du programme.

Niveau 1	→	Niveau 1
≪ *programme* ≫ *ou* '*nom de programme*'	→	

Accès clavier : PRG NXT RUN DBUG

Indicateurs : Aucun

Remarques : DBUG n'est pas programmable.

Commandes associées : HALT, NEXT, SST, SST↓

DDAYS

Commande de calcul de l'écart entre deux dates : Renvoie le nombre de jours entre deux dates.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$date_1$	$date_2$	→	x_{jours}

Accès clavier : ◀ TIME NXT DDAYS

Indicateurs : Format de date (−42)

Remarques : Si la date au niveau 2 se situe chronologiquement après celle du niveau 1, le résultat est négatif. La fourchette des dates possibles va du 15 octobre 1582 au 31 décembre 9999.

Commandes associées : DATE, DATE+

DEC

Commande de mode décimal : Sélectionne la base décimale pour les opérations sur les entiers binaires. (La base par défaut est décimale.)

Accès clavier : MTH BASE DEC

Indicateurs : Taille de mot entier binaire (−5 through −10), Base entier binaire (−11, −12)

Remarques : Les entiers binaires doivent être précédés du préfixe #. Les entiers binaires saisis et renvoyés en base décimale affichent automatiquement le suffixe d. Si la base en cours n'est pas décimale, vous pouvez taper un nombre décimal en le faisant suivre de d. Il s'affichera dans la base en cours.

La base en cours n'affecte pas la représentation interne des entiers binaires en tant que nombres binaires non signés.

Commandes associées : BIN, HEX, OCT, RCWS, STWS

DECR

Commande de décrémentation : Prend une variable au niveau 1, soustrait 1, stocke la nouvelle valeur dans la variable d'origine et renvoie la nouvelle valeur au niveau 1.

{ }

Niveau 1	→	Niveau 1
'nom'	→	$x_{nouvelle}$

Accès clavier : ◀ MEMORY ARITH DECR

Indicateurs : Aucun

Remarque : Le contenu de *nom* doit être un nombre réel.

Exemple : Si 35.7 est stocké dans *A*, 'A' DECR renvoie 34.7.

Le programme suivant effectue un décompte de 100 à 0 et laisse les entiers 100 à 0 dans la pile :

```
«
 100 'A' STO
 WHILE A REPEAT 'A' DECR END
 'A' PURGE
»
```

Commande associée : INCR

DEFINE

Commande de définition de variable ou de fonction : Stocke l'expression située à droite du signe = dans la variable spécifiée à gauche, ou crée une fonction-utilisateur.

{ }

Niveau 1	→	Niveau 1
'nom=exp'	→	
'nom(nom$_1$ ... nom$_n$)=exp(nom$_1$ ... nom$_n$)'	→	

Accès clavier : ◀ DEF

Indicateurs : Résultats numériques (−3)

Pour les arguments de la forme *'nom=exp'*, si l'indicateur −3 est armé, *expression* est évaluée comme nombre avant d'être stockée dans *nom*. (Si *exp* contient une variable formelle et si l'indicateur −3 est armé, DEFINE provoque une erreur.)

Remarques : Si le membre gauche de l'équation est *nom* uniquement, DEFINE stocke *exp* dans la variable *nom*.

Si le membre gauche de l'équation est *nom* suivi d'arguments entre parenthèses *nom$_1$ ... nom$_n$*, DEFINE crée une fonction-utilisateur et la stocke dans la variable *nom*.

Exemples : `'A=2*X'` DEFINE stocke `'2*X'` dans la variable *A*.

`'A(X,Y)=2*X+3/Y'` DEFINE crée une fonction-utilisateur *A*. Le contenu de *A* est le programme `« → X Y '2*X+3/Y' »`.

Commande associée : STO

DEG

Commande de définition en degrés : Définit le mode d'angle degrés.

Accès clavier : ◀ MODES ANGL DEG

Indicateurs : Aucun

Remarque : DEG désarme les indicateurs −17 et −18, et efface les témoins RAD et GRAD.

En mode d'angle en degrés, les arguments en nombres réels qui représentent des angles sont interprétés comme des degrés, et les résultats en nombres réels qui représentent des angles sont exprimés en degrés.

Commandes associées : GRAD, RAD

DELALARM

Commande de suppression d'alarme : Supprime l'alarme spécifiée au niveau 1.

{ }

Niveau 1	→	Niveau 1
n_{index}	→	

Accès clavier : ◀ TIME ALRM DELAL

Indicateurs : Aucun

Remarque : Si n_{index} est égal à 0, toute la liste des alarmes système est supprimée.

Commandes associées : FINDALARM, RCLALARM, STOALARM

DELAY

Commande de définition de l'intervalle de transmission : Indique le délai (en secondes) s'écoulant entre chaque envoi de lignes à l'imprimante par le HP 48.

{ }

Niveau 1	→	Niveau 1
$x_{délai}$	→	

Accès clavier : ⬅ I/O PRINT PRTPA DELAY

Indicateurs : Unité d'impression (−34) et Unité d'E-S (−33)

Armer l'indicateur −34 a pour effet d'envoyer les données à imprimer vers le port série. L'indicateur −33 doit alors être désarmé.

Si l'indicateur −34 est armé et si l'intervalle de transmission est activé (valeur différente de zéro) dans la variable réservée *IOPAR*, le protocole XON/XOFF contrôle la transmission des données et le délai spécifié n'a aucun effet.

Remarques : $x_{délai}$ spécifie l'intervalle en secondes. Par défaut, il est de 1,8 secondes. Le délai maximal est de 6,9 secondes. (Le signe de $x_{délai}$ est ignoré, de sorte que −4 DELAY équivaut à 4 DELAY.)

Le délai défini est le premier paramètre de la variable réservée *PRTPAR*.

Il est possible de fixer un délai plus court lorsque le HP 48 envoie plusieurs lignes d'information à l'imprimante (par exemple lors de l'impression d'un programme). Pour optimiser le rendement de l'impression, définissez un délai légèrement supérieur au temps mis par la tête d'impression pour imprimer une ligne.

Si vous choisissez un délai *plus court* , vous risquez de perdre des informations. De même, lorsque les batteries de l'imprimante faiblissent, la tête d'impression ralentit et si vous avez raccourci le délai, vous devez dès lors le rallonger afin de ne perdre aucune donnée. La décharge de la batterie n'entraînera pas la réduction du temps d'impression au-dessous de 1,8 seconde, qui est le délai par défaut.

DELAY

Commandes associées : CR, OLDPRT, PRLCD, PRST, PRSTC, PRVAR, PR1

DELKEYS

Commande d'annulation des définitions de touches : Annule les touches utilisateur.

Niveau 1	→	Niveau 1
x_{touche}	→	
$\{ x_{touche1} \cdots x_{touchen} \}$	→	
0	→	
'S'	→	

Accès clavier : ◀ (MODES) KEYS DELK

Indicateurs : Verrouillage mode Utilisateur (−61) et Mode utilisateur (−62).

Remarques : L'argument x_{touche} est un nombre réel *lc.s* désignant la touche par son numéro de ligne *l*, son numéro de colonne *c* et le code shift *s*. Pour connaître les valeurs admissibles de *s*, reportez-vous à ASN.

Si vous donnez la valeur 0 à x_{touche}, *toutes* les définitions effectuées par l'utilisateur sont supprimées et les définitions standard sont rétablies.

La valeur S comme argument pour DELKEYS supprime toutes les définitions standard du clavier utilisateur. Ainsi, les touches non définies par l'utilisateur sont inactives (clavier utilisateur). Vous pouvez introduire des exceptions à l'aide de ASN ou rétablir les définitions avec STOKEYS. Si vous êtes bloqué en mode utilisateur, probablement avec un clavier "verrouillé", à la suite de la réaffectation ou de l'annulation des touches servant à quitter le mode utilisateur, redémarrez le système à chaud : appuyez simultanément sur la touche (ON) et sur C, puis relâchez la touche C en premier. Vous désactivez ainsi le mode utilisateur.

Les définitions annulées occupent toujours de 2,5 à 15 octets de mémoire chacune. Vous pouvez les condenser en exécutant RCLKEYS Ø DELKEYS STOKEYS afin de libérer cette mémoire.

Commandes associées : ASN, RCLKEYS, STOKEYS

DEPND

Commande de définition de variable dépendante : Spécifie la variable dépendante (et son domaine de traçage pour les tracés TRUTH).

Niveau 2	Niveau 1	→	Niveau 1
	'globale'	→	
	{ *global* }	→	
	{ *global* $y_{début}$ y_{fin} }	→	
	{ $y_{début}$ y_{fin} }	→	
$y_{début}$	y_{fin}	→	

Accès clavier : ↩ PLOT PPAR DEPN

Indicateurs : Aucun

Remarques : Le nom de la variable dépendante spécifié et son domaine de traçage sont stockés dans la variable réservée *PPAR* comme suit :

- Si l'argument est un nom de variable globale, ce nom remplace la variable dépendante dans *PPAR*.
- Si l'argument est une liste contenant un nom global, celui-ci remplace le nom de la variable dépendante mais le domaine de traçage existant ne change pas.
- Si l'argument est une liste contenant un nom global et deux nombres réels, ou une liste contenant un nom, un tableau et un nombre réel, cette liste remplace la variable dépendante.

DEPND

- Si l'argument est une liste contenant deux nombres réels, ou deux nombres réels des niveaux 1 et 2, ceux-ci spécifient un nouveau domaine de traçage, sans toucher au nom de la variable dépendante. (LASTARG renvoie une liste, même si les deux nombres ont été entrés séparément.)

L'entrée par défaut est Y.

Le domaine de traçage pour la variable dépendante n'a d'importance que pour les tracés TRUTH, où il restreint la région pour laquelle l'équation est testée, et pour les tracés DIFFEQ, où il spécifie la solution initiale et la tolérance d'erreur absolue.

Commande associée : INDEP

DEPTH

Commande indiquant le nombre d'objets dans la pile : Renvoie un nombre réel représentant le nombre d'objets figurant dans la pile (avant exécution de DEPTH).

Niveau 1	→	Niveau 1
	→	*n*

Accès clavier : ◀ STACK DEPTH

Indicateurs : Aucun

DET

Fonction de calcul du déterminant : Renvoie le déterminant d'une matrice carrée.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]	→	$x_{\text{déterminant}}$

Accès clavier : (MTH) MATR NORM (NXT) DET

Indicateurs : Eléments "très petits" (−54)

Remarques : La matrice-argument doit être carrée. DET calcule le déterminant des matrices 1×1 et 2×2 directement à partir de l'expression de définition du déterminant. Elle obtient le déterminant d'une matrice plus grande en calculant la décomposition LU de Crout de la matrice et en cumulant le produit des éléments diagonaux de cette décomposition.

Cette opération utilisant une division à virgule flottante, le déterminant calculé d'une matrice d'entiers n'est pas toujours un entier, même si le déterminant réel de ce type de matrice doit être un entier. DET corrige ce résultat en arrondissant le déterminant obtenu à une valeur d'entier. Cette technique est également appliquée aux matrices de non-entiers avec des déterminants ayant moins de 15 chiffres différents de zéro : le déterminant calculé est arrondi à la position la plus adéquate de façon à rétablir, en totalité ou partiellement, la précision perdue.

Cette technique peut aboutir à une certaine discontinuité du déterminant calculé. Pour remédier à cet inconvénient, vous pouvez désactiver l'arrondi en armant l'indicateur −54.

Exemple : Pour une matrice carrée A, le *mineur* de l'élément a_{ij} est le déterminant de la sous-matrice qui reste après suppression de la ligne i et de la colonne j de la matrice d'origine. Etant donné une matrice carrée au niveau 3, i au niveau 2 et j au niveau 1, le programme suivant *MINOR* détermine le mineur de la sous-matrice :

```
« → M row col
```

DET

```
« M row ROW- DROP col COL- DROP DET
»
»
```

Par exemple, la saisie de `[[ 1 2 3 ][ 4 5 6 ][ 7 8 9 ]] 2 3 MINOR` renvoie -6.

Commandes associées : CNRM, CROSS, DOT, RNRM

DETACH

Commande de dissociation de bibliothèque : Dissocie du répertoire en cours la bibliothèque dont le numéro est indiqué. Chaque bibliothèque possède un numéro unique. Si un numéro de port est spécifié, il est ignoré.

{ }

Niveau 1	→	Niveau 1
$n_{bibliothèque}$	→	
$:n_{port}\ :n_{bibliothèque}$	→	

Accès clavier : ◀ LIBRARY DETAC

Indicateurs : Aucun

Remarques : Un objet-bibliothèque en RAM associé au répertoire HOME doit être dissocié avant qu'il soit possible de le supprimer, ce qui n'est pas nécessaire si la bibliothèque est associée à un autre répertoire. De même, un objet-bibliothèque associé à un répertoire différent de *HOME* est ***automatiquement*** dissocié (sans l'aide de DETACH) chaque fois qu'un autre objet-bibliothèque est associé au répertoire.

Commandes associées : ATTACH, LIBS, PURGE

DIAG→

Commande de construction d'une matrice à partir d'éléments diagonaux : Prend un tableau et une dimension spécifiée et renvoie une matrice dont les principaux éléments diagonaux sont les éléments du tableau.

Niveau 2	Niveau 1	→	Niveau 1
[*tableau*]$_{\text{él:diagonaux}}$	{ *dim* }	→	[[*matrice*]]

Accès clavier : (MTH) MATR (NXT) DIAG→

Indicateurs : Aucun

Remarques : Les dimensions en nombres réels sont arrondies à des entiers. Dans le cas d'une seule dimension, le calculateur renvoie une matrice carrée. Dans le cas de deux dimensions, l'ordre correct est { *nombre de lignes, nombre de colonnes* }. Il est impossible de spécifier plus de deux dimensions.

Si la diagonale principale de la matrice obtenue contient plus d'éléments que le tableau, les éléments supplémentaires sont définis à zéro. Si elle en contient moins, les éléments en surnombre du tableau sont supprimés.

Commande associée : →DIAG

→DIAG

Commande de création d'un vecteur à partir des éléments diagonaux d'une matrice : Renvoie un vecteur contenant les principaux éléments diagonaux d'une matrice.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]	→	[*vecteur*]$_{\text{él:diagonaux}}$

→DIAG

Accès clavier : (MTH) MATR (NXT) →DIAG

Indicateurs : Aucun

Remarque : La matrice entrée ne doit pas nécessairement être carrée.

Commande associée : DIAG→

DIFFEQ

Commande de type de tracé d'une équation différentielle : Définit le type de tracé DIFFEQ.

Accès clavier : ◀ (PLOT) PTYPE DIFFE

Indicateurs : Aucun

Remarques : Lorsque le type du tracé est DIFFEQ et que la variable réservée *EQ* ne contient pas de liste, l'équation à valeur initiale est résolue et tracée sur un intervalle donné, selon la méthode Runge-Kutta-Fehlberg (4,5). Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour les tracés DIFFEQ, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant l'angle inférieur gauche de la plage d'affichage *PICT*. La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant l'angle supérieur droit de la plage d'affichage *PICT*. La valeur par défaut est (6.5,3.2).
- *indep* est une liste, { 'X' x_0 x_f }, contenant le nom de la variable indépendante, et deux nombres spécifiant les valeurs initiale et finale de cette variable. Les valeurs par défaut de ces éléments sont { 'X' 0 x_{max} }.
- *res* est un nombre réel indiquant l'intervalle maximal, en unités-utilisateur, entre les valeurs de la variable indépendante. La valeur par défaut est 0. Si *res* est différent de zéro, l'intervalle

maximal est *res*. Si *res* égale zéro, l'intervalle maximal n'a pas de limite.

- *axes* est une liste contenant un ou plusieurs des éléments suivants, dans l'ordre cité : un nombre complexe représentant les coordonnées, en unités-utilisateur, de l'origine du tracé, une liste spécifiant l'intervalle de graduation, et les libellés (chaînes) des deux axes. Si la solution est à valeurs réelles, ces chaînes peuvent spécifier la variable dépendante ou indépendante ; si la solution est un vecteur, les chaînes peuvent spécifier un composant de la solution :
  - "0" spécifie la variable indépendante (X)
  - "1" spécifie la variable dépendante (Y)
  - "*n*" spécifie un composant de la solution Y_n

  Si *axes* contient des chaînes autres que "0", "1", ou "*n*", le programme de traçage DIFFEQ utilise les chaînes par défaut "0" et "1", et trace la variable indépendante sur l'axe horizontal et la variable dépendante sur l'axe vertical.
- *ptype* est un nom de commande spécifiant le type du tracé. L'exécution de DIFFEQ place le nom de commande DIFFEQ dans *PPAR*.
- *depend* est une liste, { 'Y' y_0 x_{TolErr} }, contenant le nom de la variable dépendante (la solution) et deux nombres représentant la valeur initiale de Y ainsi que la tolérance d'erreur absolue globale dans la solution Y. Les valeurs par défaut de ces éléments sont { 'Y' 0 .0001 }.

EQ doit définir le membre droit de l'équation à valeur initiale Y'(X)=F(X,Y). Y peut renvoyer une valeur réelle ou un vecteur réel lorsqu'elle est évaluée.

Le programme de traçage DIFFEQ tente de maintenir entre les valeurs de la variable indépendante un intervalle aussi grand que possible, tout en conservant la solution calculée dans les limites de tolérance spécifiée x_{TolErr}. Cette tolérance peut ne s'appliquer qu'aux points calculés. Des lignes droites relient les points limites calculés à chaque pas, mais elles risquent de ne pas refléter précisément la forme réelle de la solution. *res* limite la taille de l'intervalle maximal de façon à assurer une meilleure résolution du tracé.

DIFFEQ

Lorsque vous quittez le tracé DIFFEQ, les premiers éléments de *indep* et *depnd* (identificateurs) contiennent les valeurs finales de X et Y, respectivement.

Si *EQ* contient une liste, le problème à valeur initiale est résolu et tracé en combinant les méthodes Rosenbrock (3,4) et Runge-Kutta-Fehlberg (4,5). Dans ce cas, DIFFEQ utilise RRKSTEP pour calculer y_f, et *EQ* doit comporter deux éléments supplémentaires :

- Le second élément de *EQ* doit s'évaluer à la dérivée partielle de Y ' par rapport à X, et, une fois évalué, peut renvoyer une valeur réelle ou un vecteur réel.
- Le troisième élément *EQ* doit s'évaluer à la dérivée partielle de Y ' par rapport à Y, et, une fois évalué, peut renvoyer une valeur ou une matrice réelle.

Commandes associées : AXES, CONIC, FUNCTION, PARAMETRIC, POLAR, RKFSTEP, RRKSTEP, TRUTH

DISP

Commande d'affichage : Affiche *obj* dans la *n*ème ligne d'affichage.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	*n*	→	

Accès clavier : PRG NXT OUT DISP

Indicateurs : Aucun

Remarques : $n \leq 1$ désigne la ligne supérieure de l'affichage ; $n \geq 7$ désigne la ligne inférieure.

Pour faciliter l'affichage des messages, les chaînes apparaissent sans les délimiteurs " ". Tous les autres objets sont affichés sous la même forme que s'ils étaient au niveau 1 de l'affichage multiligne. Si l'affichage d'un objet nécessite plusieurs lignes, il commence à la ligne

n et se poursuit sur les lignes suivantes jusqu'à la fin de l'objet ou de l'affichage.

L'objet affiché par DISP reste à l'écran jusqu'à ce que le contrôle soit rendu au clavier pour la saisie. Pour le maintenir à l'écran jusqu'à ce qu'une touche soit actionnée, utilisez la commande FREEZE.

Exemple : Le programme

```
« "ENTRER les données" 1 DISP 7 FREEZE HALT »
```

affiche le message ENTRER les données en haut de l'affichage, "gèle" la totalité de l'affichage, puis s'interrompt.

Commandes associées : FREEZE, HALT, INPUT, PROMPT

DO

Commande de structure en boucle infinie : Commence la structure en boucle infinie DO ... UNTIL ... END.

	Niveau 1	→	Niveau 1
DO		→	
UNTIL		→	
END	*V/F*	→	

Accès clavier : PRG BRCH DO DO

Indicateurs : Aucun

Remarques : DO ... UNTIL ... END répète l'exécution d'une boucle jusqu'à ce qu'un test renvoie un résultat vrai (différent de zéro). La clause de test étant exécutée après la clause de la boucle, celle-ci s'exécute au moins une fois. La syntaxe est la suivante :

DO *clause-de-la-boucle* UNTIL *clause-de-test* END

DO commence l'exécution de la clause de la bouche. UNTIL la termine et lance la clause de test. Celle-ci doit renvoyer un résulat de test dans la pile. END retire ce résultat. Si sa valeur est zéro, la

clause de la bouche est à nouveau exécutée ; sinon, l'exécution reprend après END.

Exemple : Le programme suivant effectue un décompte de 100 à 0 et laisse les entiers 100 à 0 dans la pile :

```
«
 100 'A' STO A
  DO 'A' DECR
  UNTIL 'A==0'
  END
 'A' PURGE
»
```

Commandes associées : END, UNTIL, WHILE

DOERR

Commande de provocation d'une erreur : Exécute une "erreur-utilisateur", afin qu'un programme se comporte exactement comme si une erreur normale s'était produite durant son déroulement.

{ }

Niveau 1	→	Niveau 1
n_{erreur}	→	
$\#n_{erreur}$	→	
"erreur"	→	
0	→	

Accès clavier : PRG NXT ERROR DOERR

Indicateurs : Aucun

Remarques : DOERR oblige un programme à se comporter exactement comme si une erreur se produisait durant son déroulement. Le message d'erreur dépend de l'argument fourni à DOERR :

- n_{erreur} ou #n_{erreur} affiche le message d'erreur intégré correspondant.
- "*erreur*" affiche le contenu de la chaîne. (L'exécution de ERRM immédiatement après renvoie "*erreur*". ERRN renvoie # 70000h.)
- 0 abandonne l'exécution du programme sans afficher de message—0 DOERR équivaut à appuyer sur CANCEL.

Exemple : Le programme suivant prend un nombre dans la pile et renvoie une erreur si celui-ci est supérieur à 10 :

```
« → X
  CASE
  'X>10'
  THEN "X TROP GRAND" DOERR END
  END
»
```

Commandes associées : ERRM, ERRN, ERR0

DOLIST

Commande d'application à une liste : Applique des commandes, des programmes ou des fonctions-utilisateurs à des listes.

DOLIST

Niveau n...Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
{ *liste* }$_1$ ... { *liste* }$_n$	*n*	≪ *programme* ≫	→	{ *résultats* }
{ *liste* }$_1$ ... { *liste* }$_n$	*n*	*commande*	→	{ *résultats* }
{ *liste* }$_1$ ... { *liste* }$_n$	*n*	*nom*	→	{ *résultats* }
{ *liste* }$_1$ ...	{ *liste* }$_n$	≪ *programme* ≫	→	{ *résultats* }
{ *liste* }$_1$ ...	{ *liste* }$_n$	*commande*	→	{ *résultats* }
{ *liste* }$_1$ ...	{ *liste* }$_n$	*nom*	→	{ *résultats* }

Accès clavier : PRG LIST PROC DOLIS

Indicateurs : Aucun

Remarques : Le nombre de listes *n* n'est pas obligatoire lorsque l'argument du niveau 1 est l'un des éléments suivants :

- Une commande.
- Un programme contenant exactement une commande (par ex. ≪ DUP ≫).
- Un programme conforme à la structure d'une fonction-utilisateur.

L'objet du niveau 1 peut être un nom global ou local faisant référence à un programme ou à une commande.

Toutes les listes doivent avoir la même longueur *l*. Le programme est exécuté *l* fois : à la *i*ème itération, *n* objets pris chacun à la *i*ème position dans chaque liste sont introduits dans la pile, dans le même ordre que dans les listes d'origine, et le programme est exécuté. Les résultats de chaque exécution sont laissés dans la pile. Après la dernière itération, les nouveaux résultats sont regroupés en une liste unique.

Exemple : { 1 2 3 } { 4 5 6 } { 7 8 9 } 3 « + * » DOLIST renvoie { 11 26 45 }.

Commandes associées : DOSUBS, ENDSUB, NSUB, STREAM

DOSUBS

Commande d'application aux éléments d'une liste : Applique un programme ou une commande à des groupes d'éléments d'une liste.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
{ *liste* }$_1$	*n*	≪ *programme* ≫	→	{ *liste* }$_2$
{ *liste* }$_1$	*n*	*commande*	→	{ *liste* }$_2$
{ *liste* }$_1$	*n*	*nom*	→	{ *liste* }$_2$
	{ *liste* }$_1$	≪ *programme* ≫	→	{ *liste* }$_2$
	{ *liste* }$_1$	*commande*	→	{ *liste* }$_2$
	{ *liste* }$_1$	*nom*	→	{ *liste* }$_2$

Accès clavier : (PRG) LIST PROC DOSUB

Indicateurs : Aucun

Remarques : Le nombre réel *n* n'est pas obligatoire lorsque l'argument du niveau 1 est un des éléments suivants :

- Une commande.
- Un programme-utilisateur contenant une seule commande.
- Une programme avec une structure de fonction-utilisateur.
- Un nom global ou local faisant référence à l'un des éléments ci-dessus.

La première itération utilise les éléments 1 à *n* pris dans la liste ; la seconde itération utilise les éléments 2 à *n*+1, etc. En général, la $m^{ième}$ itération utilise les éléments de la liste correspondant aux positions *m* à *m*+*n*−1.

Pendant une itération, la position du premier élément utilisé est accessible à l'utilisateur via la commande NSUB, et le nombre de groupes d'éléments l'est à l'aide de la commande ENDSUB. Ces deux commandes renvoient le message d'erreur "Undefined Local Name" si elles sont exécutées sans que DOSUBS ait été activée.

DOSUBS renvoie le message d'erreur "Invalid User Function" si l'argument du niveau 1 est un programme-utilisateur contenant plusieurs commandes et s'il n'a pas la structure d'une

DOSUBS

fonction-utilisateur. DOSUBS renvoie aussi le message "Wrong Argument Count" si l'argument du niveau 1 est une commande qui n'accepte pas jusqu'à 5 arguments d'un type spécifique (DUP, ROT, ou →LIST, par exemple).

Exemples : { A B C D E } « - » DOSUBS renvoie
{ 'A-B' 'B-C' 'C-D' 'D-E' }.

{ A B C } 2 « DUP * * » DOSUBS renvoie
{ 'A*(B*B)' 'B*(C*C)' }.

La saisie de :

```
{ 1 2 3 4 5 }
« → a b

 « CASE  'NSUB==1' THEN a END

  'NSUB==ENDSUB' THEN b END

  'a+b' EVAL

  END

 »

»

DOSUBS
```

renvoie
{ 1 5 7 5 }.

Commandes associées : DOLIST, ENDSUB, NSUB, STREAM

DOT

Commande de produit scalaire : Renvoie le produit scalaire **A**·**B** de deux tableaux **A** et **B**, calculé comme la somme des produits des éléments correspondants des deux tableaux.

{ }

Niveau 2	Niveau 1	→	Niveau 1
[*tableau A*]	[*tableau B*]	→	*x*

Accès clavier : (MTH) VECTR DOT

Indicateurs : Aucun

Remarques : Les deux tableaux doivent avoir les mêmes dimensions.

Certains spécialistes définissent le produit scalaire de deux tableaux complexes comme la somme des produits des éléments conjugués d'un tableau avec les éléments correspondants de l'autre tableau. Le HP 48 utilise les produits ordinaires, non conjugués. Si vous préférez l'autre définition, appliquez CONJ à l'un des tableaux avant d'exécuter DOT.

Exemple : [1 2 3] [4 5 6] DOT renvoie 32 (en calculant 1 × 4 + 2 × 5 + 3 × 6).

Commandes associées : CNRM, CROSS, DET, RNRM

DRAW

Commande de dessin : Trace les données mathématiques de la variable réservée *EQ* ou les données statistiques de la variable réservée ΣDAT en utilisant la plage d'affichage spécifiée des axes x et y.

Accès clavier : (←) (PLOT) DRAW

Indicateurs : Indicateur de traçage simultané d'équations multiples (−28), Remplissage de courbe (−31)

Remarques : Le type de tracé détermine si les données figurant dans la variable réservée *EQ* ou dans ΣDAT sont tracées. DRAW

DRAW

n'efface pas *PICT* avant le traçage ; pour cela, vous devez exécuter ERASE. DRAW ne dessine pas non plus les axes (exécutez DRAX si nécessaire).

Lorsque vous exécutez DRAW depuis un programme, le graphique s'affiche, mais il ne persiste pas si vous n'exécutez pas immédiatement après la commande PICTURE, PVIEW (avec un argument liste vide) ou FREEZE.

Commandes associées : AUTO, AXES, DRAX, ERASE, FREEZE, PICTURE, LABEL, PVIEW

DRAX

Commande de dessin des axes : Dessine des axes dans *PICT*.

Accès clavier : ◀ PLOT DRAX

Indicateurs : Aucun

Remarques : Les coordonnées de l'intersection des axes sont spécifiées par AXES. Les graduations sur les axes sont spécifiées dans PPAR au moyen de la commande ATICK ou AXES. DRAX ne libelle pas les axes (exécutez LABEL si nécessaire).

Commandes associées : AXES, DRAW, LABEL

DROP

Commande de suppression d'un objet : Supprime l'objet de niveau 1 de la pile.

Niveau 1	→	Niveau 1
obj	→	

Accès clavier : ◀ DROP

Indicateurs : Aucun

Commandes associées : CLEAR, DROPN, DROP2

DROPN

Commande de suppression de n objets : Supprime de la pile les $n + 1$ premiers objets (les n premiers objets excluent l'entier n lui-même).

Niveau n+1 ... Niveau 2	Niveau 1	→	Niveau 1
obj_1 ... obj_n	n	→	

Accès clavier : ◀ STACK NXT DRPN

Indicateurs : Aucun

Commandes associées : CLEAR, DROP, DROP2

DROP2

Commande de suppression de 2 objets : Supprime les deux premiers objets de la pile.

Niveau 2	Niveau 1	→	Niveau 1
obj_1	obj_2	→	

Accès clavier : ◀ STACK NXT DROP2

Indicateurs : Aucun

Commandes associées : CLEAR, DROP, DROPN

DTAG

Commande de suppression de libellés : DTAG supprime toutes les identifications (libellés) d'un objet.

{ }

Niveau 1	→	Niveau 1
:id:obj	→	*obj*

Accès clavier : PRG TYPE NXT DTAG

Indicateurs : Aucun

Remarques : Pour une meilleure lisibilité, le signe "deux points" placé en tête n'apparaît pas lorsque l'objet identifié est dans la pile.

DTAG n'a aucun effet sur un objet non identifié.

Commandes associées : LIST→, →TAG

DUP

Commande de duplication d'objets : DUP renvoie une copie au niveau 1 de l'objet de niveau 1.

Niveau 1	→	Niveau 2	Niveau 1
obj	→	*obj*	*obj*

Accès clavier :

Une pression sur ENTER copie l'élément au niveau 1.

◀ STACK NXT DUP

Indicateurs : Aucun

Commandes associées : DUPN, DUP2, PICK

DUPN

Commande de duplication de n objets : Prend un entier *n* au niveau 1 de la pile et renvoie des copies des objets des niveaux 2*n* à *n* + 1.

Niv n+1...Niv 2	Niv 1	→	Niv 2n...Niv n+1	Niv n...Niv 1
$obj_n \ldots obj_1$	*n*	→	$obj_n \ldots obj_1$	$obj_n \ldots obj_1$

Accès clavier : ◀ STACK NXT DUPN

Indicateurs : Aucun

Commandes associées : DUP, DUP2, PICK

DUP2

Commande de duplication de 2 objets : DUP2 renvoie des copies des objets des niveaux 1 et 2 de la pile.

Niveau 2	Niveau 1	→	Niveau 4	Niveau 3	Niveau 2	Niveau 1
obj_2	obj_1	→	obj_2	obj_1	obj_2	obj_1

Accès clavier : ◀ STACK NXT DUP2

Indicateurs : Aucun

Commandes associées : DUP, DUPN, PICK

D→R

Fonction de conversion de degrés en radians : Convertit en radians un nombre réel représentant un angle en degrés.

{ }

Niveau 1	→	Niveau 1
x	→	(π/180) *x*
'*symb*'	→	'D→R(*symb*)'

Accès clavier : MTH REAL NXT NXT D→R

Indicateurs : Résultats numériques (−3)

Remarques : Cette fonction agit indépendamment du mode d'angle.

Commande associée R→D

e

Fonction e : Renvoie la constante symbolique *e* ou sa représentation numérique 2.71828182846.

Niveau 1	→	Niveau 1
	→	'e'
	→	2.71828182846

Accès clavier :

α ◀ E ENTER

MTH NXT CONS E

Indicateurs : Constantes symboliques (−2), Résultats numériques (−3)

Lorsqu'elle est évaluée, *e* renvoie sa représentation numérique si l'indicateur −2 ou −3 est armé ; sinon, *e* renvoie sa représentation symbolique.

Remarques : Le nombre renvoyé pour e est la plus proche approximation de la constante *e* avec une précision de 12 chiffres. Pour un calcul d'exponentielle, utilisez l'expression 'EXP(X)' au lieu de 'e^X', car la fonction EXP met en oeuvre un algorithme spécial pour une plus grande précision.

Commandes associées : EXP, EXPM, i, LN, LNP1, MAXR, MINR, π

EGV

Commande relative aux valeurs et aux vecteurs propres : Calcule les valeurs propres et les vecteurs propres de droite pour une matrice carrée.

{ }

Niveau 1	→	Niveau 2	Niveau 1
[[*matrice*]]$_A$	→	[[*matrice*]]$_{EVec}$	[*vecteur*]$_{EVal}$

Accès clavier : (MTH) MATR (NXT) EGV

Indicateurs : Aucun

Remarques : Le vecteur résultant EVal contient les valeurs propres calculées. Les colonnes de la matrice EVec contiennent les vecteurs propres de droite correspondant aux éléments du vecteur EVal.

Les résultats calculés doivent aboutir à une réduction (dans la limite de la précision de calcul) :

$$\frac{|A \cdot EVec - EVec \cdot \mathrm{diag}(EVal)|}{n \cdot |A|}$$

où diag(*EVal*) désigne la matrice de $n \times n$ éléments diagonaux contenant les valeurs propres *EVal*.

EGV

Commande associée : EGVL

EGVL

Commande relative aux valeurs propres : Calcule les valeurs propres d'une matrice carrée.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]$_A$	→	[*vecteur*]$_{EVal}$

Accès clavier : (MTH) MATR (NXT) EGVL

Indicateurs : Aucun

Remarque : Le vecteur L obtenu contient les valeurs propres calculées.

Commande associée : EGV

ELSE

Commande ELSE : Commence une clause fausse dans une structure conditionnelle ou d'interception d'erreur.

Voir les rubriques IF et IFERR pour de plus amples informations sur la syntaxe.

Accès clavier :

(PRG) BRCH IF ELSE

(PRG) (NXT) ERROR IFERR ELSE

Remarque : Voir les rubriques IF et IFERR pour de plus amples informations.

Commandes associées : IF, IFERR, THEN, END

END

Commande END : Termine des structures conditionnelles, d'interception d'erreurs et en boucle infinie.

Voir les rubriques IF, CASE, IFERR, DO et WHILE pour de plus amples informations sur la syntaxe.

Accès clavier :

(PRG) BRCH IF END

(PRG) BRCH CASE END

(PRG) BRCH DO END

(PRG) BRCH WHILE END

(PRG) (NXT) ERROR IFERR END

Remarque : Voir les rubriques IF, CASE, IFERR, DO et WHILE pour de plus amples informations.

Commandes associées : IF, CASE, DO, ELSE, IFERR, REPEAT, THEN, UNTIL, WHILE,

ENDSUB

Commande de fin de sous-liste : Permet de connaître le nombre total de blocs contenus dans la liste utilisée par DOSUBS.

Accès clavier : (PRG) LIST PROC ENDS

Indicateurs : Aucun

Remarques : Le message d'erreur `Undefined Local Name` est renvoyé si cette commande est exécutée sans que DOSUBS ait été activée.

Exemple : Le programme suivant soustrait le nombre d'éléments d'une liste de chaque élément de cette liste :

```
« → a
 « a 1
  « ENDSUB -
```

ENDSUB

```
»
» DOSUBS
»
```

Commandes associées : DOSUBS, NSUB

ENG

Commande du mode ingénieur : Définit le mode ingénieur pour l'affichage des nombres, à savoir jusqu'à trois chiffres à gauche du séparateur décimal (virgule ou point) et un exposant multiple de trois. Le nombre total de chiffres significatifs affichés est de $n + 1$.

{ }

Niveau 1	→	Niveau 1
n	→	

Accès clavier : ◀ MODES FMT ENG

Indicateurs : Aucun

Remarques : Le mode ingénieur utilise $n + 1$ chiffres significatifs, où $0 \leq n \leq 11$. (Les valeurs de n se situant hors de cette fourchette sont arrondies au chiffre supérieur ou inférieur le plus proche.) Un nombre est affiché ou imprimé comme suit :

(*signe*) *mantisse* E (*signe*) *exposant*

où la mantisse se présente sous la forme (*nn*)*n*.(*n* ...) (avec un maximum de 12 chiffres au total) et où l'exposant contient de un à trois chiffres.

Un nombre ayant un exposant égal à −499 s'affiche automatiquement en mode scientifique.

Exemple : Le nombre 103.6 en mode ingénieur avec cinq chiffres significatifs (n=4) se présentera sous la forme `103.60E0`. Le même nombre avec un chiffre significatif (n=0) sera `100.E0`.

Commandes associées : FIX, SCI, STD

EQ→

Commande de séparation d'une équation : Sépare une équation en deux membres : gauche et droit.

{ }

Niveau 1	→	Niveau 2	Niveau 1
'$symb_1 = symb_2$'	→	'$symb_1$'	'$symb_2$'
z	→	*z*	0
'*nom*'	→	'*nom*'	0
x_unité	→	*x_unité*	0
'*symb*'	→	'*symb*'	0

Accès clavier : PRG TYPE NXT EQ→

Indicateurs : Aucun

Remarques : Si l'argument est une expression, il est traité comme une équation dont le membre droit égale zéro.

Commandes associées : ARRY→, DTAG, LIST→, OBJ→, STR→

EQNLIB

Commande d'ouverture de la bibliothèque d'équations : Lance l'application de la bibliothèque d'équations.

Accès clavier : ← EQ LIB EQLIB EQNLI

Indicateurs : Aucun

Remarques : La bibliothèque d'équations est un ensemble d'équations et de commandes utiles pour résoudre des problèmes scientifiques et techniques.

EQNLIB

Commandes associées : MSOLVR, SOLVEQN

ERASE

Commande d'effacement de PICT : Efface le contenu de *PICT*, en laissant un affichage *PICT* vierge de dimensions identiques.

Accès clavier :

◀ PICTURE EDIT NXT ERASE

◀ PICTURE ◀ CLEAR

◀ PLOT ERASE

Indicateurs : Aucun

Commande associée : DRAW

ERRM

Commande de message d'erreur : Renvoie une chaîne contenant le dernier message d'erreur du calculateur.

Niveau 1	→	Niveau 1
	→	*"message d'erreur"*

Accès clavier : PRG NXT ERROR ERRM

Indicateurs : Aucun

Remarques : ERRM renvoie la chaîne correspondant à une erreur générée par DOERR. Si l'argument pour DOERR était 0, la chaîne renvoyée par ERRM est vide.

Exemple : Le programme « IFERR + THEN ERRM END » renvoie le message "Bad Argument Type" au niveau 1 si des arguments

incorrects (par exemple un nombre complexe et un entier binaire) sont aux niveaux 1 et 2.

Commandes associées : DOERR, ERRN, ERR0

ERRN

Commande numéro d'erreur : Renvoie le numéro de la dernière erreur.

Niveau 1	→	Niveau 1
	→	#n_{erreur}

Accès clavier : PRG NXT ERROR ERRN

Indicateurs : Aucun

Remarques : Si la dernière erreur a été générée par DOERR avec un argument chaîne, ERRN renvoie # 70000h. Si elle a été générée par DOERR avec un entier binaire comme argument, ERRN renvoie cet entier binaire. (Si la dernière erreur a été générée par DOERR avec un nombre réel comme argument, ERRN renvoie l'entier binaire converti en nombre réel.)

Exemple : Le programme « IFERR + THEN ERRN END » renvoie # 202h au niveau 1 si des arguments incorrects (par exemple un nombre complexe et un entier binaire) sont aux niveaux 1 et 2.

Commandes associées : DOERR, ERRM, ERR0

ERR0

Commande d'effacement du dernier numéro d'erreur : Efface le dernier numéro d'erreur de telle sorte qu'une nouvelle exécution de ERRN renvoie # 0h, et efface le dernier message d'erreur.

Accès clavier : PRG NXT ERROR ERR0

Indicateurs : Aucun

Commandes associées : DOERR, ERRM, ERRN

EVAL

Commande d'évaluation d'un objet : Evalue un objet.

Niveau 1	→	Niveau 1
obj	→	(*voir ci-dessous*)

Accès clavier : EVAL

Indicateurs : Résultats numériques (−3)

Remarques : Le tableau suivant décrit les conséquences de l'évaluation selon les types d'objets :

Objet	Impact de l'évaluation
Nom local	Rappelle le contenu de la variable.
Nom global	*Appelle* le contenu de la variable : ■ Un nom est évalué. ■ Un programme est évalué. ■ Un répertoire devient le répertoire en cours. ■ Les autres objets sont placés dans la pile. S'il n'existe pas de variable pour un nom donné, l'évaluation renvoie ce nom dans la pile.
Programme	*Introduit* chaque objet du programme : ■ Les noms sont évalués (sauf s'ils sont entre guillemets). ■ Les commandes sont évaluées. ■ Les autres objets sont placés dans la pile.
Liste	*Introduit* chaque objet de la liste : ■ Les noms sont évalués. ■ Les commandes sont évaluées. ■ Les programmes sont évalués. ■ Les autres objets sont placés dans la liste.

EVAL

Objet	Impact de l'évaluation
Obj. identifié	Si l'identification concerne un port, l'objet spécifié est rappelé et évalué. Sinon, l'objet non identifié est placé dans la pile.
Exp. algébrique	*Introduit* chaque objet de l'expression algébrique : ■ Les noms sont évalués. ■ Les commandes sont évalués. ■ Les autres objets sont placés dans la pile.
Commande, fonction, nom XLIB	Evalue l'objet spécifié.
Autres objets	Place l'objet dans la pile.

Pour obtenir un résultat numérique à partir d'un argument symbolique, évaluez l'argument en mode Résultats numériques (indicateur −3 armé) ou exécutez →NUM sur cette fonction.

Commandes associées : →NUM, SYSEVAL

EXP

Fonction analytique de calcul d'exponentielle : Renvoie l'exponentielle, ou antilogarithme népérien, de l'argument, c'est-à-dire *e* élevé à la puissance donnée.

{ }

Niveau 1	→	Niveau 1
z	→	e^z
'*symb*'	→	'EXP(*symb*)'

Accès clavier : 

Indicateurs : Résultats numériques (−3)

Remarques : EXP utilise des constantes à précision étendue ainsi qu'un algorithme spécial pour calculer ses résultats avec une précision de 12 chiffres, et ce pour tous les arguments qui n'entraînent pas une erreur de dépassement de capacité négatif ou positif.

EXP fournit un résultat plus précis que e $\boxed{y^x}$. La différence de précision croît à mesure que z augmente. Par exemple :

z	**EXP(z)**	e^z
3	20.0855369232	20.0855369232
10	22026.4657948	22026.4657949
100	2.68811714182E43	2.68811714191E43
500	1.40359221785E217	1.40359221809E217
1000	1.97007111402E434	1.97007111469E434

Pour des arguments complexes,

$$e^{(x,y)} = e^x cosy + ie^x siny$$

Commandes associées : ALOG, EXPM, LN, LOG

EXPAN

Commande de développement : Récrit une expression algébrique ou une équation en développant les produits et puissances.

{ }

Niveau 1	→	Niveau 1
'*symb*$_1$'	→	'*symb*$_2$'
x	→	*x*
(*x*, *y*)	→	(*x*, *y*)

Accès clavier : ◀ SYMBOLIC EXPR

EXPAN

Indicateurs : Aucun

Exemples : 'A*(B+C)' EXPAN renvoie 'A*B+A*C'.

'A^(B+C)' EXPAN renvoie 'A^B*A^C'.

'X^5' EXPAN renvoie 'X*X^4'.

'(X+Y)^2' EXPAN renvoie 'X^2+2*X*Y+Y^2'.

Commandes associées : COLCT, ISOL, QUAD, SHOW

EXPFIT

Commande d'ajustement exponentiel : Stocke EXPFIT en tant que cinquième paramètre de la variable réservée ΣPAR, indiquant que les exécutions suivantes de LR doivent utiliser le modèle d'ajustement exponentiel pour la courbe.

Accès clavier : ◀ STAT ΣPAR MODL EXPFI

Indicateurs : Aucun

Remarque : LINFIT est le modèle par défaut de ΣPAR. Pour une description de ΣPAR, voir l'Annexe D, "Variables réservées".

Commandes associées : BESTFIT, LR, LINFIT, LOGFIT, PWRFIT

EXPM

Fonction analytique d'élévation à une puissance moins 1 : Renvoie $e^x - 1$.

{ }

Niveau 1	→	Niveau 1
x	→	$e^x - 1$
'*symb*'	→	'EXPM(*symb*)'

Accès clavier : (MTH) HYP (NXT) EXPM

Indicateurs : Résultats numériques (−3)

Remarques : Pour des valeurs de x proches de zéro, 'EXPM(*x*)' renvoie un résultat plus précis que 'EXP(*x*)−1'. (L'utilisation de EXPM permet d'avoir un argument et un résultat proches de zéro, et évite un résultat intermédiaire voisin de 1. Le calculateur peut exprimer des nombres allant jusqu'à 10^{-449} par rapport à zéro, mais seulement jusqu'à 10^{-11} par rapport à 1.)

Commandes associées : EXP, LNP1

EYEPT

Commande de définition du point de vue : Spécifie les coordonnées du point de vue dans un tracé en perspective.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
x_{point}	y_{point}	z_{point}	→	

Accès clavier : ◀ (PLOT) (NXT) 3D VPAR (NXT) EYEPT

Indicateurs : Aucun

Remarques : x_{point}, y_{point} et z_{point} sont des nombres réels qui définissent les coordonnées x, y et z du point de vue à partir duquel vous visualisez la vue volumique d'un tracé en trois dimensions. La coordonnée y doit toujours être inférieure d'une unité par rapport au point le plus proche de la vue volumique (y_{near} de YVOL). Ces coordonnées sont stockées dans la variable réservée *VPAR*.

Commandes associées : NUMX, NUMY, XVOL, XXRNG, YVOL, YYRNG, ZVOL

F0λ

Fonction relative au pouvoir émissif d'un corps noir : Renvoie la fraction du pouvoir émissif total d'un corps noir.

{ }

Niveau 2	Niveau 1	→	Niveau 1
y_{lambda}	x_T	→	$x_{pouvoir}$
y_{lambda}	'symb'	→	'F0λ(y_{lambda},symb)'
'symb'	x_T	→	'F0λ(symb,x_T)'
'$symb_1$'	'$symb_2$'	→	'F0λ($symb_1$,$symb_2$)'

Accès clavier : ◀ EQ LIB UTILS F0λ

Indicateurs : Résultats numériques (−3)

Remarques : F0λ calcule la fraction du pouvoir émissif total d'un corps noir à la température x_T entre les longueurs d'onde 0 et y_{lambda}. Si aucune unité n'est spécifiée, y_{lambda} est implicitement exprimée en mètres et x_T en Kelvin (K).

F0λ renvoie une fraction sans dimensions.

FACT

Fonction de calcul de factorielle (Gamma) : Cette fonction est fournie à des fins de compatibilité avec le HP 28. FACT est l'équivalent de ! (voir cette rubrique).

{ }

Niveau 1	→	Niveau 1
n	→	*n*!
x	→	Γ(*x*+1)
'symb'	→	'(symb)!'

Accès clavier : Aucun. A saisir.

FANNING

Fonction relative au facteur de friction Fanning : Calcule le facteur de friction Fanning de certains fluides en écoulement.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$x_{x/D}$	y_{Re}	→	$x_{fanning}$
$x_{x/D}$	'*symb*'	→	'FANNING($x_{x/D}$,*symb*)'
'*symb*'	y_{Re}	→	'FANNING(*symb*,y_{Re})'
'$symb_1$'	'$symb_2$'	→	'FANNING($symb_1$,$symb_2$)'

Accès clavier : ↩ EQ LIB UTILS FANNI

Indicateurs : Résultats numériques (−3)

Remarques : FANNING calcule le facteur de friction Fanning, qui corrige les effets de friction de certains fluides en écoulement présentant une température, une section, une vitesse et une viscosité constantes (dans un conduit par exemple). $x_{x/D}$ est la rugosité relative (rapport de la rugosité du conduit et de son diamètre). y_{Re} est le nombre de Reynolds. La fonction utilise des routines de calcul différentes pour un écoulement laminaire ($Re \leq 2100$) et un écoulement turbulent ($Re > 2100$). $x_{x/D}$ et y_{Re} doivent être des nombres réels ou des objets-unités réductibles à des nombres sans dimensions, et être supérieurs à zéro.

Commande associée : DARCY

FC?

Commande de détection de l'état d'un indicateur : Vérifie si l'indicateur système ou utilisateur désigné par $n_{\text{numéro indicateur}}$ est désarmé, et renvoie le résultat du test correspondant : 1 (vrai) s'il est désarmé, ou 0 (faux) s'il est armé.

{ }

Niveau 1	→	Niveau 1
$n_{\text{numéroindicateur}}$	→	0/1

Accès clavier :

PRG TEST NXT NXT FC?

◀ MODES FLAG FC?

Indicateurs : Aucun

Commandes associées : CF, FC?C, FS?, FS?C, SF

FC?C

Commande de détection de l'état d'un indicateur et de désarmement : Vérifie si l'indicateur système ou utilisateur désigné par $n_{\text{numéro indicateur}}$ est désarmé et renvoie le résultat du test correspondant : 1 (vrai) si l'indicateur est désarmé, ou 0 (faux) s'il est armé. A l'issue du test, désarme l'indicateur.

{ }

Niveau 1	→	Niveau 1
$n_{\text{numéroindicateur}}$	→	0/1

Accès clavier :

PRG TEST NXT NXT FC?C

◀ MODES FLAG FC?C

Indicateurs : Aucun

Exemple : Si l'indicateur −44 est armé, −44 FC?C renvoie 0 au niveau 1, puis désarme l'indicateur.

Commandes associées : CF, FC?, FS?, FS?C, SF

FFT

Commande de transformation de Fourier : Calcule la transformée de Fourier discrète uni- ou bidimensionnelle d'un tableau.

{ }

Niveau 1	→	Niveau 1
[*tableau*]$_1$	→	[*tableau*]$_2$

Accès clavier : MTH NXT FFT FFT

Indicateurs : Aucun

Remarques : Si l'argument est un vecteur de N éléments ou une matrice $N \times 1$ ou $1 \times N$, FFT calcule la transformée unidimensionnelle. Si l'argument est une matrice $M \times N$, FFT calcule la transformée bidimensionnelle. M et N doivent être des puissances entières de 2.

La transformée de Fourier discrète unidimensionnelle d'un vecteur de N éléments X est le vecteur de N éléments Y où :

$$Y_k = \sum_{n=0}^{N-1} X_n e^{-\frac{2\pi i k n}{N}}, \qquad i = \sqrt{-1}$$

pour $k = 0,\ 1,\ \ldots,\ N-1$.

La transformée de Fourier discrète bidimensionnelle d'une matrice $M \times N$ X est la matrice $M \times N$ Y où :

$$Y_{kl} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{mn} e^{-\frac{2\pi i k m}{M}} e^{-\frac{2\pi i l n}{N}}, \qquad i = \sqrt{-1}$$

pour $k = 0, 1, \ldots, M-1$ et $l = 0, 1, \ldots, N-1$.

La transformée de Fourier discrète et son inverse sont définis pour toute longueur de séquence positive. Cependant, le calcul peut être effectué très rapidement lorsque la longueur de la séquence est une puissance de deux ; les algorithmes résultants sont appelés transformation de Fourier rapide (FFT) et transformation de Fourier rapide inverse (IFFT).

La commande FFT utilise une arithmétique à 15 chiffres avec troncature et stockage intermédiaire, puis arrondit le résultat en précision à 12 chiffres.

Commande associée : IFFT

FINDALARM

Commande de recherche d'alarme : Renvoie l'index d'alarme n_{index} de la première alarme expirant après le délai spécifié.

Niveau 1	→	Niveau 1
date	→	n_{index}
{ *date heure* }	→	n_{index}
0	→	n_{index}

Accès clavier : ↩ TIME ALRM FINDA

Indicateurs : Format de date (−42)

Remarques : Si l'argument au niveau 1 est un nombre réel *date*, FINDALARM renvoie l'index de la première alarme qui expire après 12:00 AM le jour indiqué. Si l'argument est une liste { *date heure* }, elle renvoie l'index de la première alarme qui expire après cette heure et cette date. Si l'argument est le nombre réel 0, FINDALARM renvoie la première alarme *échue*.

Pour les trois arguments, FINDALARM renvoie 0 lorsqu'aucune alarme n'est détectée.

Commandes associées : DELALARM, RCLALARM, STOALARM

FINISH

Commande d'arrêt du mode serveur : Met fin au mode serveur Kermit sur une unité connectée au HP 48

Accès clavier : ◀ I/O SRVR FINIS

Indicateurs : Unité d'E-S (−33), Messages d'E-S (−39)

Remarques : FINISH est utilisée par une unité locale Kermit pour signifier à un serveur Kermit (connecté via le port série ou le port infrarouge) de quitter le mode serveur.

Commandes associées : BAUD, CKSM, KGET, PARITY, PKT, RECN, RECV, SEND, SERVER

FIX

Commande du mode fixe : Sélectionne le mode d'affichage Fix dans lequel les nombres sont arrondis à *n* décimales.

{ }

Niveau 1	→	Niveau 1
n	→	

Accès clavier : ◀ MODES FMT FIX

Indicateurs : Aucun

Remarques : Le mode Fix affiche *n* chiffres à droite du séparateur décimal (virgule ou point), où $0 \leq n \leq 11$. (Les valeurs de *n* en dehors de cette fourchette sont arrondis à l'entier le plus proche.) Un nombre est affiché ou imprimé comme suit :

(*signe*) *mantisse*

où la mantisse peut avoir n'importe quelle forme. Cependant, le calculateur affiche automatiquement un nombre en mode scientifique si l'une des propositions suivantes est vraie :

- Le nombre de chiffres à afficher dépasse 12.
- Une valeur non nulle au-delà de la *n*ième décimale sera cependant affichée 0.

Exemple : En mode Fix, le nombre 103.6 avec quatre décimales serait `103.6000`.

Commandes associées : FIX, SCI, STD

FLOOR

Fonction plancher : Renvoie le plus grand entier inférieur ou égal à l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	*n*
x_unité	→	*n_unité*
'*symb*'	→	'FLOOR(*symb*)'

Accès clavier : (MTH) REAL (NXT) (NXT) FLOOR

Indicateurs : Résultats numériques (−3)

Exemples : `3.2 FLOOR` renvoie 3; `-3.2 FLOOR` renvoie −4.

Commandes associées : CEIL, IP, RND, TRNC

FOR

Commande de structure en boucle finie : Commence les structures en boucle finie FOR ... NEXT et FOR ... STEP.

	Niveau 2	Niveau 1	→	Niveau 1
FOR	$x_{début}$	x_{fin}	→	
NEXT			→	
FOR	$x_{début}$	x_{fin}	→	
STEP		$x_{incrément}$	→	
STEP		$'symb_{incrément}'$	→	

Accès clavier :
Pour commencer une boucle finie :

PRG BRCH FOR FOR

Pour taper FOR NEXT

PRG BRCH ◀ FOR

Pour taper FOR STEP:

PRG BRCH ▶ FOR

Indicateurs : Aucun

Remarques : Les structures en boucle finie exécutent une commande ou une séquence de commandes un certain nombre de fois.

- Une boucle FOR ... NEXT exécute un segment de programme un certain nombre de fois, en utilisant une variable locale comme compteur de boucle. Cette variable peut figurer dans la boucle. La syntaxe est la suivante :

  $x_{début}$ x_{fin} FOR *compteur clause-de-la-boucle* NEXT

  FOR prend $x_{début}$ et x_{fin} dans la pile comme valeurs initiale et finale du compteur de boucle, puis crée la variable locale *compteur* comme compteur de boucle. La clause de la boucle est ensuite exécutée ; la variable *compteur* peut figurer dans la clause de la boucle ou être modifiée. NEXT incrémente de un le *compteur*, puis

vérifie si sa valeur est inférieure ou égale à x_{fin}. Si c'est le cas, la clause de la boucle est répétée (avec la nouvelle valeur de *compteur*).

A la fin de la boucle, la valeur de *compteur* est supprimée.

- FOR ... STEP fonctionne comme FOR ... NEXT, sauf qu'elle permet de spécifier une valeur d'incrémentation autre que 1. La syntaxe est la suivante :

  $x_{début}$ x_{fin} FOR *compteur clause-de-la-boucle* $x_{incrément}$ STEP

  FOR prend $x_{début}$ et x_{fin} dans la pile comme valeurs initiale et finale du compteur de la boucle, puis crée la variable locale *compteur* comme compteur de boucle. La clause de la boucle est ensuite exécutée ; la variable *compteur* peut figurer dans la clause de la boucle ou être modifiée. STEP prend la valeur de $x_{incrément}$ dans la pile et augmente le *compteur* en conséquence. Si l'argument de STEP est une expression algébrique ou un nom, il est automatiquement évalué à un nombre.

  La valeur d'incrément peut être positive ou négative. Si elle est positive, la boucle est exécutée à nouveau tant que le *compteur* est inférieur ou égal à x_{fin}. Sinon, la boucle est exécutée tant que le *compteur* est supérieur ou égal à x_{fin}.

  A la fin de la boucle, la valeur de *compteur* est supprimée.

Exemple : Le programme suivant fait la somme de tous les entiers impairs compris entre 1 et 100 :

```
« 0 1 100
  FOR I I + 2 STEP
»
```

Commandes associées : NEXT, START, STEP

FP

Fonction relative à la partie décimale d'un nombre : Renvoie la partie décimale de l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	*y*
x_unité	→	*y_unité*
'*symb*'	→	'FP(*symb*)'

Accès clavier : (MTH) REAL (NXT) FP

Indicateurs : Résultats numériques (−3)

Remarque : Le résultat possède le même signe que l'argument.

Exemples : `-32.3 FP` renvoie `-.3`; `32.3_m FP` renvoie `.3_m`.

Commande associée : IP

FREE

Commande de libération de carte RAM : Libère (rend *indépendante*) la RAM précédemment fusionnée située dans le port 1. FREE est fournie à des fins de compatibilité avec le HP 48SX, qui permettait aussi de fusionner la mémoire RAM dans le port 2 (voir FREE1).

Niveau 2	Niveau 1	→	Niveau 1
{ }	n_{port}	→	
{$nom_{sauvegarde}$ ⋯ $n_{bibliothèque}$ }	n_{port}	→	
$nom_{sauvegarde}$	n_{port}	→	
$n_{bibliothèque}$	n_{port}	→	

FREE

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : Le contenu précédent du port est placé en mémoire utilisateur. Si le niveau 2 spécifie des objets-sauvegardes ou bibliothèques, ils sont transférés du port 0 vers le port nouvellement libéré.

Commandes associées : FREE1, MERGE1

FREE1

Commande de libération de carte RAM : Libère (rend *indépendante*) la RAM précédemment fusionnée dans le port 1. Le contenu du port est placé en mémoire utilisateur. Si le niveau 1 spécifie des objets-sauvegardes ou bibliothèques, ceux-ci sont transférés du port 0 vers le port nouvellement libéré.

Niveau 1	→	Niveau 1
$\{nom_{sauvegarde} \cdots n_{bibliothèque}\}$	→	
$nom_{sauvegarde}$	→	
$n_{bibliothèque}$	→	

Accès clavier : ↰ LIBRARY FREE1

Indicateurs : Aucun

Remarques : La liste du niveau 1 peut être vide (dans ce cas, aucun objet n'est transféré vers la nouvelle RAM indépendante), ou elle peut contenir un nombre quelconque de noms de sauvegardes et de numéros de bibliothèques. Cependant, le niveau 1 ne peut pas être entièrement vide.

Commandes associées : FREE, MERGE, MERGE1

FREEZE

Commande pour geler l'affichage : Gèle la partie de l'affichage spécifiée par $n_{\text{zone d'affichage}}$, afin qu'elle reste inchangée jusqu'à ce qu'une touche soit activée.

{ }

Niveau 1	→	Niveau 1
$n_{\text{zoned'affichage}}$	→	

Accès clavier : PRG NXT OUT FREEZ

Indicateurs : Aucun

Remarques : Généralement, l'affichage de la pile est mis à jour dès que le calculateur est prêt pour la saisie de données. Par exemple, lorsque HALT interrompt un programme en cours d'exécution, ou lorsqu'un programme se termine, les messages affichés disparaissent. La commande FREEZE "gèle" une partie ou la totalité de l'affichage afin que son contenu soit maintenu tant que vous n'appuyez pas sur une touche. Ceci permet par exemple de conserver à l'écran un message même après interruption du programme.

$n_{\text{zone d'affichage}}$ est la somme des codes de valeur des zones à geler.

Zone d'affichage	Code de valeur
Zone d'état	1
Pile/ligne de commande	2
Zone des menus	4

Ainsi, 2 FREEZE gèle la zone de la pile/ligne de commande, 3 FREEZE gèle la zone d'état et la zone de la pile/ligne de commande, et 7 FREEZE gèle les trois zones.

Les valeurs de $n_{\text{zone d'affichage}} \geq 7$ ou ≤ 0 gèlent la totalité de l'affichage (équivalent de la valeur 7). Pour geler l'affichage graphique,

FREEZE

vous devez geler la zone d'état et celle de la pile/ligne de commande (en tapant 3), ou la totalité (en tapant 7).

Exemples :

Ce programme :

```
« "Prêt pour données" 1 DISP 1 FREEZE HALT »
```

affiche le contenu de la chaîne sur la ligne supérieure de l'affichage, puis gèle la zone d'état afin que la chaîne reste visible après exécution de HALT.

Ce programme :

```
« ( # 0d # 0d ) PVIEW 7 FREEZE »
```

sélectionne l'affichage graphique puis le gèle en totalité afin qu'il reste à l'écran une fois le programme terminé. (Si FREEZE n'est pas exécutée, l'affichage de la pile serait sélectionné à la fin du programme.)

Pour utiliser FREEZE avec PVIEW (ou un affichage graphique), vous devez entrer 3 ou 7.

Commandes associées : CLLCD, DISP, HALT

FS?

Commande de détection d'état d'un indicateur : Teste si l'indicateur système ou utilisateur spécifié par $n_{\text{numéro indicateur}}$ est armé, et renvoie un résultat de test correspondant : 1 (vrai) si l'indicateur est armé, ou 0 (faux) s'il est désarmé.

{ }

Niveau 1	→	Niveau 1
$n_{\text{numéroindicateur}}$	→	0/1

Accès clavier :

PRG TEST NXT NXT FS?

◀ MODES FLAG FS?

Indicateurs : Aucun

Commandes associées : CF, FC?, FC?C, FS?C, SF

FS?C

Commande de détection de l'état d'un indicateur et de désarmement : Teste si l'indicateur système ou utilisateur spécifié par $n_{\text{numéro indicateur}}$ est armé, et renvoie un résultat de test correspondant : 1 (vrai) si l'indicateur est armé ou 0 (faux) s'il est désarmé. Après le test, cette commande désarme l'indicateur.

{ }

Niveau 1	→	Niveau 1
$n_{\text{numéroindicateur}}$	→	0/1

Accès clavier :

PRG TEST NXT NXT FS?C

↩ MODES FLAG FS?C

Indicateurs : Aucun

Exemple : Si l'indicateur −44 est armé, −44 FS?C renvoie 1 au niveau 1, puis désarme l'indicateur −44.

Commandes associées : CF, FC?, FC?C, FS?, SF

FUNCTION

Commande du type de tracé Function : Sélectionne le type de tracé FUNCTION.

Accès clavier : ↩ PLOT PTYPE FUNC

Indicateurs : Indicateur de traçage simultané d'équations multiples (−28), Remplissage de courbe (−31)

Remarques : Lorsque le tracé est du type FUNCTION, la commande DRAW représente l'équation en cours sous la forme d'une fonction à valeurs réelles d'une variable réelle. L'équation en cours est spécifiée dans la variable réservée *EQ*. Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR* dont le format est le suivant :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé FUNCTION, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant les coordonnées de l'angle inférieur gauche de *PICT* (la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant les coordonnées de l'angle supérieur droit de *PICT* (la plage d'affichage). La valeur par défaut est (6.5,3.2).
- *indep* est le nom de la variable indépendante, ou une liste contenant son nom et deux nombres qui désignent les valeurs minimale et maximale de la variable indépendante (domaine de traçage). La valeur par défaut de *indep* est *X*.
- *res* est un nombre réel spécifiant l'intervalle (en unités-utilisateur) entre les points tracés de la variable indépendante, ou un entier binaire spécifiant cet intervalle en pixels. La valeur par défaut est 0, ce qui correspond à un intervalle de 1 pixel.
- *axes* est une liste contenant un ou plusieurs des éléments suivants : un nombre complexe spécifiant les coordonnées, en unités-utilisateur, de l'origine du tracé, une liste spécifiant l'intervalle de graduation, et deux libellés (chaînes) pour les axes horizontal et vertical. La valeur par défaut est (0,0).
- *ptype* est un nom de commande désignant le type de tracé. L'exécution de FUNCTION place le nom FUNCTION dans *PPAR*.

- *depend* est le nom du libellé de l'axe vertical. La valeur par défaut est *Y*.

L'équation en cours est tracée comme une fonction de la variable spécifiée dans *indep*. Les valeurs minimale et maximale de la variable indépendante (le domaine de traçage) peuvent être spécifiées dans *indep* ; sinon, les (x_{min}, y_{min}) et (x_{max}, y_{max}) (plage d'affichage) sont utilisées. Des lignes relient les points tracés, à moins que l'indicateur −31 soit armé.

Si *EQ* contient une expression ou un programme, celle-ci ou celui-ci est évalué en mode Résultats numériques pour chaque valeur de la variable indépendante afin de fournir les valeurs de la variable dépendante. Si *EQ* contient une équation, le traçage dépend de sa forme, comme l'indique le tableau ci-dessous.

Forme de l'équation en cours	Type de traçage
'*expr=expr*'	Chaque expression est tracée séparément. L'intersection des deux graphiques montre l'endroit où les expressions sont égales.
'*nom=expr*'	Seule l'expression est tracée.
'*indep=constante*'	Une ligne verticale est tracée.

Si l'indicateur −28 est armé, toutes les équations sont tracées simultanément.

Si la variable indépendante dans l'équation en cours représente un objet-unité, vous devez spécifier les unités en stockant un objet-unité dans la variable correspondante, au niveau du répertoire en cours. Par exemple, si l'équation en cours est `'X+3_m'`, et si vous souhaitez que *X* représente un certain nombre de pouces, vous allez stocker `1_in` (la partie numérique de l'objet-unité est ignorée) dans *X*. Pour chaque point tracé, la valeur numérique de la variable indépendante est associée à l'unité (dans notre exemple, des pouces) avant que l'équation en cours ne soit évaluée. Si le résultat est un objet-unité, seule la partie numérique est tracée.

Commandes associées : BAR, CONIC, DIFFEQ, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR,

FUNCTION

POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

GET

Commande d'extraction d'éléments : Extrait de la liste ou du tableau du niveau 2 (ou de la liste ou du tableau nommé) le nombre complexe ou réel z_{get} ou l'objet obj_{get} dont la position est spécifiée au niveau 1.

Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]	$n_{position}$	→	z_{get}
[[*matrice*]]	{ n_{ligne} m_{col} }	→	z_{get}
'$nom_{matrice}$'	$n_{position}$	→	z_{get}
'$nom_{matrice}$'	{ n_{ligne} m_{col} }	→	z_{get}
[*vecteur*]	$n_{position}$	→	z_{get}
[*vecteur*]	{ $n_{position}$ }	→	z_{get}
'$nom_{vecteur}$'	$n_{position}$	→	z_{get}
'$nom_{vecteur}$'	{ $n_{position}$ }	→	z_{get}
{ *liste* }	$n_{position}$	→	obj_{get}
{ *liste* }	{ $n_{position}$ }	→	obj_{get}
'nom_{liste}'	$n_{position}$	→	obj_{get}
'nom_{liste}'	{ $n_{position}$ }	→	obj_{get}

Accès clavier : (PRG) LIST ELEM GET

Indicateurs : Aucun

Remarque : Pour les matrices, $n_{position}$ est incrémenté dans l'ordre des *lignes*.

Exemples : [[2 3 7][3 2 9][2 1 3]] { 2 3 } GET renvoie 9.

[[2 3 7][3 2 9][2 1 3]] 8 GET renvoie 1.

{ A B C D E } { 1 } GET renvoie 'A'.

Commandes associées : GETI, PUT, PUTI

GETI

Commande d'extraction et d'incrémentation de l'index : Extrait de la liste ou du tableau de niveau 2 (ou de la liste ou du tableau nommé) le nombre réel ou complexe z_{get} ou l'objet obj_{get} dont la position est spécifiée au niveau 1, ainsi que l'argument de niveau 2 et la position suivante dans cet argument.

Niveau 2	Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
[[*matrice*]]	n_{pos1}	→	[[*matrice*]]	n_{pos2}	z_{get}
[[*matrice*]]	$\{ n_r\ m_c \}_1$	→	[[*matrice*]]	$\{ n_r\ m_c \}_2$	z_{get}
$'nom_{mtrc}'$	n_{pos1}	→	$'nom_{mtrc}'$	n_{pos2}	z_{get}
$'nom_{mtrc}'$	$\{ n_r\ m_c \}_1$	→	$'nom_{mtrc}'$	$\{ n_r\ m_c \}_2$	z_{get}
[*vect*]	n_{pos1}	→	[*vect*]	n_{pos2}	z_{get}
[*vect*]	$\{ n_{pos1} \}$	→	[*vect*]	$\{ n_{pos2} \}$	z_{get}
$'nom_{vect}'$	n_{pos1}	→	$'nom_{vect}'$	n_{pos2}	z_{get}
$'nom_{vect}'$	$\{ n_{pos1} \}$	→	$'nom_{vect}'$	$\{ n_{pos2} \}$	z_{get}
{ *liste* }	n_{pos1}	→	{ *list* }	n_{pos2}	obj_{get}
{ *liste* }	$\{ n_{pos1} \}$	→	{ *liste* }	$\{ n_{pos2} \}$	obj_{get}
$'nom_{liste}'$	n_{pos1}	→	$'nom_{liste}'$	n_{pos2}	obj_{get}
$'nom_{liste}'$	$\{ n_{pos1} \}$	→	$'nom_{liste}'$	$\{ n_{pos2} \}$	obj_{get}

Accès clavier : (PRG) LIST ELEM GETI

Indicateurs : Indicateur de bouclage d'index (−64)

L'indicateur de bouclage d'index est désarmé à chaque exécution de GETI *jusqu'à ce que* l'index de position revienne à la première position dans l'argument. A ce moment-là, l'indicateur est armé. L'exécution suivante de GETI désarme à nouveau l'indicateur.

Remarques : Pour les matrices, la position est incrémentée dans l'ordre des *lignes*.

GETI

Commandes associées : GET, PUT, PUTI

GOR

Commande OU avec des objets graphiques : Superpose $grob_1$ sur $grob_{cible}$ ou sur *PICT*, le pixel de l'angle supérieur gauche de $grob_1$ étant positionné aux coordonnées spécifiées dans $grob_{cible}$ ou dans *PICT*.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
$grob_{cible}$	{ #n #m }	$grob_1$	→	$grob_{résultat}$
$grob_{cible}$	(x,y)	$grob_1$	→	$grob_{résultat}$
PICT	{ #n #m }	$grob_1$	→	
PICT	(x,y)	$grob_1$	→	

Accès clavier : PRG GROB GOR

Indicateurs : Aucun

Remarques : GOR utilise un OU logique pour déterminer l'état (activé ou désactivé) de chaque pixel dans la zone de chevauchement de l'objet graphique.

Si l'argument de niveau 3 est un objet graphique quelconque autre que *PICT*, $grob_{résultat}$ est renvoyé dans la pile. Si l'argument de niveau 3 est *PICT*, aucun résultat n'est renvoyé dans la pile.

Toute partie de $grob_1$ qui dépasse $grob_{cible}$ ou *PICT* est tronquée.

Commandes associées : GXOR, REPL, SUB

GRAD

Commande du mode grades : Sélectionne le mode d'angle grades.

Accès clavier : ↩ MODES ANGL GRAD

Indicateurs : Aucun

Remarques : GRAD désarme l'indicateur −17 et arme l'indicateur −18, et affiche le témoin GRAD.

En mode d'angle grades, les arguments du type nombres réels qui représentent des angles sont interprétés en grades, et les résultats en nombres réels qui représentent des angles sont exprimés en grades.

Commandes associées : DEG, RAD

GRAPH

Commande d'environnement graphique : Sélectionne l'environnement Picture (sélectionne l'affichage graphique et active le curseur graphique ainsi que le menu Picture).

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarque : GRAPH est fournie à des fins de compatibilité avec la série HP 48S ; elle équivaut à la commande PICTURE (voir cette dernière).

Commandes associées : PICTURE, PVIEW, TEXT

GRIDMAP

Commande du type de tracé GRIDMAP : Sélectionne le type de tracé GRIDMAP.

Accès clavier : ◀ PLOT NXT 3D PTYPE GRID

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est GRIDMAP, la commande DRAW trace une représentation, sous forme de grille transformée (mappée), d'une fonction de deux vecteurs à deux variables. GRIDMAP nécessitent des valeurs se trouvant dans les variables réservées *EQ*, *VPAR* et *PPAR*.

VPAR a la forme suivante :

{ x_{left} x_{right} y_{near} y_{far} z_{low} z_{high} x_{min} x_{max} y_{min} y_{max} x_{eye} y_{eye} z_{eye} x_{step} y_{step} }.

Pour un tracé de type GRIDMAP, les éléments de *VPAR* sont utilisés comme suit :

- x_{left} et x_{right} sont des nombres réels spécifiant la largeur de la vue volumique.
- y_{near} et y_{far} sont des nombres réels spécifiant la profondeur de la vue volumique.
- z_{low} et z_{high} sont des nombres réels spécifiant la hauteur de la vue volumique.
- x_{min} et x_{max} sont des nombres réels spécifiant la largeur de la zone d'entrée. La valeur par défaut est (-1,1).
- y_{min} et y_{max} sont des nombres réels spécifiant la profondeur de la zone d'entrée. La valeur par défaut est (-1,1).
- x_{eye}, y_{eye} et z_{eye} sont des nombres réels spéficiant le point de vue.
- x_{step} et y_{step} sont des nombres réels qui déterminent le nombre de coordonnées x par rapport au nombre de coordonnées y tracées. Ils sont utilisables à la place de (ou avec) RES.

Les paramètres de traçage sont spécifiés dans la variable réservée *PAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé GRIDMAP, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) n'est pas utilisé.
- (x_{max}, y_{max}) n'est pas utilisé.
- *indep* est le nom de la variable indépendante. La valeur par défaut de *indep* est *X*.
- *res* est un nombre réel spécifiant l'intervalle (en unités-utilisateur) séparant deux valeurs tracées de la variable indépendante, ou un entier binaire indiquant cet intervalle en pixels. La valeur par défaut est 0, qui correspond à un intervalle de 1 pixel.
- *axes* n'est pas utilisé.
- *ptype* est un nom de commande désignant le type de tracé. L'exécution de la commande GRIDMAP place son nom dans *PPAR*.
- *depend* est le nom de la variable dépendante. La valeur par défaut est *Y*.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

→GROB

Commande de création d'objet graphique à partir de la pile

: Crée un objet graphique représentant l'objet du niveau 2, où l'argument $n_{taille\ car}$ spécifie la taille des caractères.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	$n_{taillecar}$	→	*grob*

Accès clavier : PRG GROB →GRO

Indicateurs : Aucun

→GROB

Remarques : $n_{\text{taille car}}$ peut prendre les valeurs 0, 1 (petit), 2 (moyen) ou 3 (grand). $n_{\text{taille}} = 0$ équivaut à $n_{\text{taille}} = 3$, sauf dans le cas des objets-unités et des objets algébriques, où 0 spécifie l'image EquationWriter.

Exemple : Ce programme :

```
« 'Y=3*X^2' 0 →GROB PICT STO { } PVIEW »
```

renvoie un objet graphique dans la pile, représentant une image EquationWriter de `'Y=3*X^2'`, puis stocke l'objet graphique dans *PICT* et l'affiche dans l'affichage graphique avec possibilité de défilement.

Commandes associées : →LCD, LCD→

GXOR

Commande OU exclusif pour des graphiques : Superpose $grob_1$ sur $grob_{\text{cible}}$ ou sur *PICT*, le pixel de l'angle supérieur gauche de $grob_1$ étant positionné aux coordonnées spécifiées dans $grob_{\text{cible}}$ ou *PICT*.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
$grob_{\text{cible}}$	{ #*n* #*m* }	$grob_1$	→	$grob_{\text{résultat}}$
$grob_{\text{cible}}$	(*x*,*y*)	$grob_1$	→	$grob_{\text{résultat}}$
PICT	{ #*n* #*m* }	$grob_1$	→	
PICT	(*x*,*y*)	$grob_1$	→	

Accès clavier : PRG GROB GXOR

Indicateurs : Aucun

Remarques : GXOR sert à créer des curseurs, par exemple pour que l'image du curseur apparaisse en noir sur un fond clair et en clair sur un fond noir. Une nouvelle exécution de GXOR sur la même image rétablit celle d'origine.

GXOR utilise un OU exclusif logique pour déterminer l'état des pixels (activés ou désactivés) dans la partie de chevauchement des objets graphiques de l'argument.

Toute partie de $grob_1$ qui dépasse $grob_{cible}$ ou *PICT* est tronquée.

Si l'argument de niveau 3 (l'objet graphique cible) est un objet graphique autre que *PICT*, $grob_{résultat}$ est renvoyé dans la pile. S'il correspond à *PICT*, aucun résultat n'est renvoyé dans la pile.

Exemple : Le programme

```
« ERASE PICT NEG PICT { # 0d #0d }
GROB 5 x 5 11A040A011 GXOR LASTARG GXOR »
```

active (assombrit) chaque pixel dans *PICT*, puis superpose un objet graphique de 5 x 5 sur *PICT* aux coordonnées `{ # 0d # 0d }`. Chaque pixel actif dans l'objet graphique 5 x 5 désactive (éclaircit) le pixel correspondant dans *PICT*. Vous rétablissez l'image originale en exécutant une nouvelle fois GXOR avec les mêmes arguments.

Commandes associées : GOR, REPL, SUB

---

*H

Commande de modification de la hauteur de l'échelle : Multiplie l'échelle verticale du tracé par un facteur x.

{ }

Niveau 1	→	Niveau 1
$x_{facteur}$	→	

Accès clavier : ◀ PLOT PPAR NXT *H

Indicateurs : Aucun

Remarques : L'exécution de *H modifie la plage d'affichage de l'axe y, c'est-à-dire les composants y_{min} et y_{max} des deux premiers nombres complexes de la variable réservée *PPAR*. L'origine du tracé (coordonnées en unités-utilisateur du pixel central) ne change pas.

*H

Commandes associées : AUTO, *W, YRNG

HALT

Commande d'interruption de programme : Arrête l'exécution du programme.

Accès clavier : PRG NXT RUN HALT

Indicateurs : Aucun

Remarques : L'exécution du programme est arrêtée au niveau de la commande HALT figurant dans le programme. Le témoin HALT est activé. Le programme reprend lorsque la commande CONT est exécutée (généralement à l'aide des touches ← CONT). L'exécution de KILL (par activation des touches PRG NXT RUN KILL) annule tous les programmes arrêtés.

Commandes associées : CONT, KILL

HEAD

Commande d'extraction du premier élément d'une liste : Renvoie le premier élément d'une liste ou d'une chaîne.

Niveau 1	→	Niveau 1
{ obj_1 ... obj_n }	→	obj_1
"*chaîne*"	→	"$élément_1$"

Accès clavier :

PRG LIST ELEM NXT HEAD

← CHARS NXT HEAD

Indicateurs : Aucun

Exemple : "Arrêt" HEAD renvoie "A".

Le programme suivant prend une liste de coordonnées { A B C } qui définissent un triangle rectangle et calcule la longueur de l'hypoténuse AC :

« DUP HEAD SWAP REVLIST HEAD - ABS »

Par exemple, saisir { (0,0) (0,3) (3,4) } renvoie 5.

Commande associée : TAIL

HEX

Commande de mode hexadécimal : Sélectionne la base hexadécimale pour les opérations sur des entiers binaires (la base par défaut est la base 10).

Accès clavier : (MTH) BASE HEX

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : Les entiers binaires doivent être accompagnés du préfixe #. Les entiers binaires saisis et renvoyés en base hexadécimale affichent automatiquement le suffixe h. Si la base en cours n'est pas hexadécimale, vous pouvez malgré tout entrer un nombre en hexadécimal en lui associant le suffixe h. Une fois saisi, il s'affiche dans la base en cours.

La base en cours n'affecte pas la représentation interne des entiers binaires en tant que nombres binaires non signés.

Commandes associées : BIN, DEC, OCT, RCWS, STWS

HISTOGRAM

Commande de type de tracé HISTOGRAM : Selectionne le type de tracé HISTOGRAM.

Accès clavier : ◀ PLOT NXT STAT PTYPE HISTO

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est HISTOGRAM, la commande DRAW crée un histogramme en utilisant les données d'une colonne de la matrice statistique en cours (variable réservée ΣDAT). La colonne est désignée par le premier paramètre figurant dans la variable réservée ΣPAR (par la commande XCOL). Les paramètres de traçage sont spécifiés dans la variable *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé HISTOGRAM, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant les coordonnées de l'angle inférieur gauche de *PICT* (angle inférieur gauche de la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant l'angle supérieur droit de *PICT* (angle supérieur droit de la plage d'affichage). La valeur par défaut est (6.5,3.2).
- *indep* est le nom d'un libellé pour l'axe horizontal, ou une liste contenant un nom et deux nombres qui représentent les valeurs minimale et maximale des données à tracer. La valeur par défaut de *indep* est *X*.
- *res* est un nombre réel spécifiant la taille de bloc, en unités-utilisateur, ou un entier binaire spécifiant la taille de bloc en pixels. La valeur par défaut est 0, correspondant à une taille de bloc égale à 1/13 de la différence entre les valeurs inférieure et supérieure spécifiées.
- *axes* est une liste contenant un ou plusieurs des éléments suivants, dans l'ordre mentionné : un nombre complexe spécifiant les coordonnées, en unités-utilisateur, de l'origine du tracé, une liste indiquant l'intervalle de graduation, et deux libellés (chaînes) pour les deux axes. La valeur par défaut est (0,0).

- *ptype* est un nom de commande désignant le type de tracé. L'exécution de la commande HISTOGRAM place le nom HISTOGRAM dans *PPAR*.
- *depend* est le libellé de l'axe vertical. La valeur par défaut est *Y*.

La fréquence des données est représentée par des barres, chacune correspondant à un ensemble de points de données. La base de chaque barre représente les valeurs des points de données, et la hauteur en indique le nombre. La largeur de chaque barre est spécifiée par *res*. Les valeurs globales inférieure et supérieure pour les données peuvent être spécifiées par *indep* ; sinon, les valeurs de $\langle x_{min}, y_{min} \rangle$ et $\langle x_{max}, y_{max} \rangle$ sont utilisées.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

HISTPLOT

Commande de tracé d'histogramme : Trace un histogramme des données d'une colonne de la matrice statistiques en cours (variable réservée *ΣDAT*).

Accès clavier : ◀ STAT PLOT HISTP

Indicateurs : Aucun

Remarques : La colonne de données à tracer est indiquée par XCOL et stockée dans le premier paramètre de la variable réservée *ΣPAR*. Si aucune colonne n'est spécifiée, la colonne 1 est sélectionnée par défaut. L'axe *y* est automatiquement mis à l'échelle et le type de tracé sélectionné est HISTOGRAM.

HISTPLOT trace des fréquences *relatives*, en utilisant 13 blocs comme nombre de partitions par défaut. La commande RES permet d'indiquer un nombre différent de blocs, en spécifiant la largeur de chacun d'eux. Pour tracer un histogramme à partir de fréquences *numériques*, vous devez stocker celles-ci dans *ΣDAT* et exécuter BINS suivie de BARPLOT.

Lorsque HISTPLOT est exécutée à partir d'un programme, l'affichage graphique qui montre le tracé résultant, ne subsiste pas à l'écran à

HISTPLOT

moins que vous n'exécutiez immédiatemment après une commande PICTURE, PVIEW (avec une liste vide) ou FREEZE.

Commandes associées : BARPLOT, BINS, FREEZE, PICTURE, PVIEW, RES, SCATRPLOT, XCOL

HMS+

Commande d'addition Heures-Minutes-Secondes : Renvoie la somme de deux nombres réels, les arguments et le résultat étant interprétés au format heures-minutes-secondes.

{ }

Niveau 2	Niveau 1	→	Niveau 1
HMS_1	HMS_2	→	$HMS_1 + HMS_2$

Accès clavier : ◀ TIME NXT HMS+

Indicateurs : Aucun

Remarques : Le format HMS (heure ou angle) est *H.MMSSs*, où :

- *H* est zéro ou plusieurs chiffres représentant la partie entière du nombre.
- *MM* sont deux chiffres représentant le nombre de minutes.
- *SS* sont deux chiffres représentant le nombre de secondes.
- *s* est zéro ou plusieurs chiffres (autant que le permet le mode d'affichage en cours) représentant les dixièmes de secondes.

Commandes associées : HMS→, →HMS, HMS–

HMS–

Commande de soustraction Heures-Minutes-Secondes : Renvoie la différence entre deux nombre réels, les arguments et le résultat étant interprétés au format heures-minutes-secondes.

{ }

Niveau 2	Niveau 1	→	Niveau 1
HMS_1	HMS_2	→	$HMS_1 - HMS_2$

Accès clavier : ◀ TIME NXT HMS-

Indicateurs : Aucun

Remarques : Le format HMS (heure ou angle) est *H.MMSSs*, où :

- *H* est zéro ou plusieurs chiffres représentant la partie entière du nombre.
- *MM* sont deux chiffres représentant le nombre de minutes.
- *SS* sont deux chiffres représentant le nombre de secondes.
- *s* est zéro ou plusieurs chiffres (autant que le permet le mode d'affichage en cours) représentant les dixièmes de secondes.

Commandes associées : HMS→, →HMS, HMS+

HMS→

Commande de conversion du format heures-minutes-secondes en décimal : Convertit un nombre réel au format heures-minutes-secondes en son équivalent décimal (heures ou degrés avec une décimale).

HMS→

{ }

Niveau 1	→	Niveau 1
HMS	→	*x*

Accès clavier : ◀ TIME NXT HMS→

Indicateurs : Aucun

Remarques : Le format HMS (heure ou angle) est *H.MMSSs*, où :

- *H* est zéro ou plusieurs chiffres représentant la partie entière du nombre.
- *MM* sont deux chiffres représentant le nombre de minutes.
- *SS* sont deux chiffres représentant le nombre de secondes.
- *s* est zéro ou plusieurs chiffres (autant que le permet le mode d'affichage en cours) représentant les dixièmes de secondes.

Commandes associées : →HMS, HMS+, HMS−

→HMS

Commande de conversion du format décimal en heures-minutes-secondes : Convertit un nombre réel avec décimale, représentant des heures ou des degrés, au format heures-minutes-secondes.

{ }

Niveau 1	→	Niveau 1
x	→	*HMS*

Accès clavier : ◀ TIME NXT HMS→

Indicateurs : Aucun

Remarques : Le format HMS (heure ou angle) est *H.MMSSs*, où :

- *H* est zéro ou plusieurs chiffres représentant la partie entière du nombre.
- *MM* sont deux chiffres représentant le nombre de minutes.
- *SS* sont deux chiffres représentant le nombre de secondes.
- *s* est zéro ou plusieurs chiffres (autant que le permet le mode d'affichage en cours) représentant les dixièmes de secondes.

Commandes associées : HMS→, HMS+, HMS−

HOME

Commande du répertoire HOME : Fait de *HOME* le répertoire en cours.

Accès clavier : ↷ HOME

Indicateurs : Aucun

Commandes associées : CRDIR, PATH, PGDIR, UPDIR

---

i

Fonction i : Renvoie la constante symbolique *i* ou sa représentation numérique (0, 1).

Niveau 1	→	Niveau 1
	→	'i'
	→	(0,1)

Accès clavier :

α ↶ I

MTH NXT CONS I

i

Indicateurs : Constantes symboliques (−2), Résultats numériques (−3)

L'évaluation de *i* renvoie sa représentation numérique si l'indicateur −2 ou −3 est armé ; sinon, elle renvoie la représentation symbolique.

Commandes associées : e, MAXR, MINR, π

IDN

Commande de matrice identité : Renvoie une matrice identité, c'est-à-dire une matrice carrée dont les éléments diagonaux égalent 1 et les éléments non diagonaux 0.

{ }

Niveau 1	→	Niveau 1
n	→	[[*R-matrice*$_{identité}$]]
[[*matrice*]]	→	[[*matrice*$_{identité}$]]
'nom'	→	

Accès clavier : (MTH) MATR MAKE IDN

Indicateurs : Aucun

Remarques : Le résultat est soit une nouvelle matrice carrée, soit une matrice carrée existante dont les éléments sont remplacés par ceux de la matrice identité en fonction de l'argument de niveau 1.

- **Création d'une nouvelle matrice :** Si l'argument est un nombre réel *n*, une nouvelle matrice identité réelle est renvoyée au niveau 1, avec le nombre de lignes et de colonnes égal à *n*.
- **Remplacement des éléments d'une matrice existante :** Si l'argument est une matrice carrée, une matrice identité de dimensions identiques est renvoyée. Si la matrice d'origine est complexe, la matrice résultante l'est aussi, avec les valeurs d'éléments diagonaux (1,0).

Si l'argument est un nom, celui-ci doit identifier une variable contenant une matrice carrée. Dans ce cas, les éléments de la matrice sont remplacés par ceux de la matrice identité (complexe si la matrice d'origine est complexe).

Commande associée : CON

IF

Commande de structure conditionnelle IF : Commence les structures conditionnelles IF ... THEN ... END et IF ... THEN ... ELSE ... END.

	Niveau 1	→	Niveau 1
IF		→	
THEN	*V/F*	→	
END		→	
		→	
IF		→	
THEN	*V/F*	→	
ELSE		→	
END		→	

Accès clavier : (PRG) BRCH IF IF

Indicateurs : Aucun

Remarques : Les *structures conditionnelles*, utilisées en combinaison avec des tests, permettent à un programme de prendre des décisions.

- IF ... THEN ... END exécute une séquence de commandes uniquement si un test renvoie un résultat différent de zéro (vrai). La syntaxe est la suivante :

  IF *clause-de-test* THEN *clause-vraie* END

IF

IF débute la clause de test, qui doit renvoyer un résultat dans la pile. THEN retire ce résultat. Si la valeur est différente de zéro, la clause vraie est exécutée. Sinon, le programme reprend après END.

- IF ... THEN ... ELSE ... END exécute une séquence de commandes si un test renvoie un résultat vrai (différent de zéro), ou une autre séquence de commandes si ce test renvoie un résultat faux (zéro). La syntaxe est la suivante :

  IF *clause-de-test* THEN *clause-vraie* ELSE *clause-fausse* END

  IF débute la clause de test, qui doit renvoyer un résultat dans la pile. THEN retire ce résultat. Si la valeur est différente de zéro, la clause vraie est exécutée. Sinon, la clause fausse est exécutée. Après exécution de la clause appropriée, le programme reprend après END.

La clause de test peut être une séquence de commandes (par exemple A B ≤) ou une expression algébrique (par exemple 'A≤B'). Si la clause de test est une expression algébrique, elle est automatiquement évaluée à un nombre (→NUM ou EVAL ne sont pas nécessaires).

Commandes associées : CASE, ELSE, END, IFERR, THEN

IFERR

Commande de structure conditionnelle d'interception d'erreurs : Débute les structures d'interception d'erreurs IFERR ... THEN ... END et IFERR ... THEN ... ELSE ... END.

Accès clavier : PRG NXT ERROR IFERR

Indicateurs : Derniers arguments (−55)

Remarques : Les structures d'*interception d'erreurs* permettent au programme de continuer à se dérouler après l'interception d'une erreur.

- IFERR ... THEN ... END exécute une séquence de commandes si une erreur se produit. La syntaxe de IFERR ... THEN ... END est la suivante :

  IFERR *clause-d'interception* THEN *clause-d'erreur* END

Si une erreur se produit durant l'exécution de la clause d'interception :

1. L'erreur est ignorée.
2. Le reste de la clause d'interception n'est pas pris en compte.
3. Le tampon des touches est effacé.
4. Si tout ou partie de l'affichage est gelé, cet état est annulé.
5. Si l'indicateur Derniers arguments est activé, les arguments pour la commande qui a provoqué l'erreur sont renvoyés dans la pile.
6. L'exécution du programme passe à la clause d'erreur.

Les commandes de la clause d'erreur ne sont exécutées que si une erreur se produit durant l'exécution de la clause d'interception.

- IFERR ... THEN ... ELSE ... END exécute une séquence de commandes si une erreur se produit ; sinon le programme passe à une autre séquence de commandes (clause de normalité). La syntaxe de IFERR ... THEN ... ELSE ... END est la suivante :

IFERR *clause-d'interception* THEN *clause-d'erreur* ELSE *clause-de-normalité* END

En cas d'erreur pendant l'exécution de la clause d'interception, les six événements mentionnés plus haut se produisent.

En l'absence d'erreur, l'exécution passe à la clause de normalité à la fin de la clause d'interception.

Exemple : Le programme suivant utilise IFERR de la même façon que le système linéraire intégré de résolution d'équations. Il prend un vecteur résultat et une matrice de coefficients, et renvoie pour les équations une solution par la méthode des moindres carrés.

```
« → a b
  « IFERR a b / THEN LSQ END
  »
»
```

Commandes associées : CASE, ELSE, END, IF, THEN

IFFT

Commande de transformation de Fourier inverse : Calcule l'inverse de la transformée de Fourier discrète uni- ou bidimensionnelle d'un tableau.

{ }

Niveau 1	→	Niveau 1
[*tableau*]$_1$	→	[*tableau*]$_2$

Accès clavier : (MTH) (NXT) FFT IFFT

Indicateurs : Aucun

Remarques : Si l'argument est un vecteur N ou une matrice $N \times 1$ ou $1 \times N$, IFFT calcule la transformée inverse unidimensionnelle. Si l'argument est une matrice $M \times N$, IFFT calcule la transformée inverse bidimensionnelle. M et N doivent être des puissances entières de 2.

L'inverse de la transformée de Fourier discrète unidimensionnelle d'un vecteur de N éléments Y est le vecteur de N éléments X où :

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} Y_k e^{\frac{2\pi i k n}{N}}, \qquad i = \sqrt{-1}$$

pour $n = 0,\ 1,\ \ldots,\ N-1$.

L'inserse de la transformée de Fourier discrète bidimensionnelle d'une matrice $M \times N$ Y est la matrice $M \times N$ X où :

$$X_{mn} = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} Y_{kl} e^{\frac{2\pi i k m}{M}} e^{\frac{2\pi i l n}{N}}, \qquad i = \sqrt{-1}$$

pour $m = 0,\ 1,\ \ldots,\ M-1$ and $n = 0,\ 1,\ \ldots,\ N-1$.

La transformée de Fourier discrète et son inverse sont définis pour toute longueur de séquence positive. Cependant, le calcul peut être effectué très rapidement lorsque la longueur de la séquence est une puissance de deux ; les algorithmes résultants sont appelés transformation de Fourier rapide (FFT) et transformation de Fourier rapide inverse (IFFT).

La commande FFT utilise une arithmétique à 15 chiffres avec troncature et stockage intermédiaire, puis arrondit le résultat avec une précision de 12 chiffres.

Commande associée : FFT

IFT

Commande IF-THEN : Exécute *obj* si *V/F* est différent de zéro, et l'ignore si *V/F* égale zéro.

Niveau 2	Niveau 1	→	Niveau 1
V/F	*obj*	→	*Cela dépend!*

Accès clavier : (PRG) BRCH (NXT) IFT

Indicateurs : Aucun

Remarque : IFT permet d'exécuter, en utilisant la syntaxe de la pile, le processus de test de la structure conditionnelle IF ... THEN ... END. La "clause vraie" est *obj* du niveau 1.

Exemple : « X 0 > "Positif" IFT » place "Positif" au niveau 1 si *X* contient un nombre réel positif.

Commande associée : IFTE

IFTE

Fonction IF-THEN-ELSE : Exécute *obj* au niveau 2 si *V/F* est différent de zéro, et exécute *obj* au niveau 1 si *V/F* égale zéro.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
V/F	obj_{vrai}	obj_{faux}	→	*Cela dépend!*

Accès clavier : PRG BRCH NXT IFTE

Indicateurs : Aucun

Remarques : IFTE permet d'exécuter, avec la syntaxe de la pile, le processus de test de la structure conditionnelle IF ... THEN ... ELSE ... END. La "clause vraie" est obj_{vrai} du niveau 2. La "clause fausse" est obj_{faux} du niveau 1.

IFTE est également possible dans des expressions algébriques, avec la syntaxe suivante :

'IFTE(*test,clause-vraie,clause-fausse*)'

Lorsqu'une expression algébrique contenant IFTE est évaluée, son premier argument *test* est évalué à un résultat de test. S'il renvoie un nombre réel différent de zéro, *clause-vraie* est évaluée. S'il renvoie zéro, *clause-fausse* est évaluée.

Exemples : La séquence de commandes X 0 ≥ "Positif" "Négatif" IFTE laisse "Positif" dans la pile si X contient un nombre réel non négatif, ou "Négatif" si X contient un nombre réel négatif.

L'expression algébrique 'IFTE(X≠0,SIN(X)/X,1)' renvoie la valeur de $\sin(x)/x$, même pour $x = 0$, ce qui normalement provoquerait le message d'erreur "Infinite Result".

Commande associée : IFT

IM

Fonction relative à la partie imaginaire d'un nombre : Renvoie la partie imaginaire de son argument (de type complexe).

{ }

Niveau 1	→	Niveau 1
x	→	0
(*x*, *y*)	→	*y*
[*R-tableau*]	→	[*R-tableau*]
[*C-tableau*]	→	[*R-tableau*]
'*symb*'	→	'IM(*symb*)'

Accès clavier : MTH NXT CMPL IM

Indicateurs : Résultats numériques (–3)

Remarques : Si l'argument est un tableau, IM renvoie un tableau réel, dont les éléments sont égaux aux parties imaginaires des éléments correspondants de l'argument. Si l'argument est un tableau réel, tous les éléments du tableau résultant égalent zéro.

Commandes associées : C→R, RE, R→C

INCR

Commande d'incrémentation : Prend une variable au niveau 1, ajoute 1, stocke la nouvelle valeur dans la variable d'origine, puis la renvoie au niveau 1.

{ }

Niveau 1	→	Niveau 1
'*nom*'	→	$x_{incrément}$

INCR

Accès clavier : ◀ MEMORY ARITH INCR

Indicateurs : Aucun

Remarque : La valeur de *nom* doit être un nombre réel.

Exemple : Si la valeur 35.7 est stockée dans *A*, 'A' INCR renvoie 36.7.

Commande associée : DECR

INDEP

Commande de définition de variable indépendante : Spécifie la variable indépendante et son domaine de traçage.

Niveau 2	Niveau 1	→	Niveau 1
	'global'	→	
	{ *global* }	→	
	{ *global* $x_{début}$ x_{fin} }	→	
	{ $x_{début}$ x_{fin} }	→	
$x_{début}$	x_{fin}	→	

Accès clavier : ◀ PLOT PPAR INDEP

Indicateurs : Aucun

Remarques : Le nom de la variable indépendante et son domaine de traçage sont stockés dans le troisième paramètre de la variable réservée *PPAR*, comme suit :

- Si l'argument est un nom de variable globale, ce nom remplace la variable indépendante qui figure dans *PPAR*.
- Si l'argument est une liste contenant un nom global, celui-ci remplace le nom de la variable indépendante mais le domaine de traçage existant ne change pas.
- Si l'argument est une liste contenant un nom global et deux nombres réels, cette liste remplace le contenu de la variable indépendante.

- Si l'argument est une liste contenant deux nombres réels, ou deux nombres réels des niveaux 1 et 2, ceux-ci spécifient un nouveau domaine de traçage, sans toucher au nom de la variable indépendante. (LASTARG renvoie une liste, même si les deux nombres ont été entrés séparément.)

L'entrée par défaut est X.

Commande associée : DEPND

INFORM

Commande de boîte de dialogue utilisateur : Crée un masque de saisie utilisateur (boîte de dialogue).

Niv 5	Niv 4	Niv 3	Niv 2	Niv 1 →	Niv 2	Niv 1
'titre'	{ s_1 s_2...s_n }	*format*	{ *réinit.* }	{ *init* } →	{ *val* }	*1*
'titre'	{ s_1 s_2...s_n }	*format*	{ *réinit.* }	{ *init* } →		0

Accès clavier : PRG NXT IN INFOR

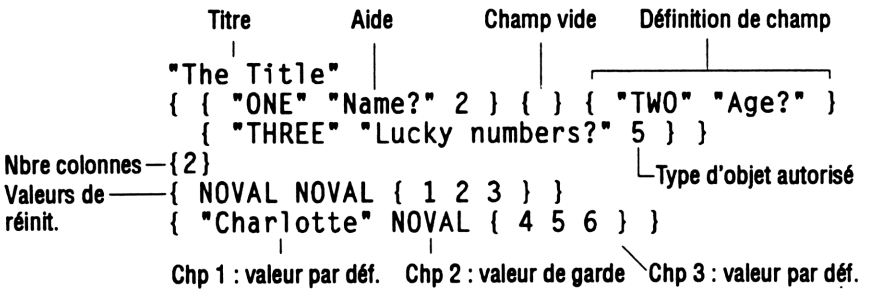
Indicateurs : Aucun

Remarques : INFORM crée une boîte de dialogue standard à partir des spécifications suivantes :

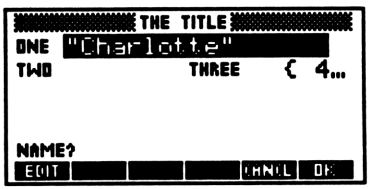
Variable	Fonction
"titre"	Titre, qui apparaît en haut de la boîte de dialogue.
$\{s_1\ s_2\ \ldots\ s_n\}$	Définitions des champs. Une définition de champ (s_x) peut avoir deux formats : *"libellé"*, libellé de champ, ou { *"libellé" "helpInfo" type$_0$ type$_1$ ... type$_n$* }, libellé de champ avec un texte d'aide facultatif qui apparaît en bas de l'écran, et une liste, facultative également, des types d'objets valides pour ce champ. Si les types d'objets ne sont pas spécifiés, tous les types sont valides. Pour de plus amples informations sur les types d'objets, voir la commande TYPE. Lors de la création d'une boîte de dialogue à plusieurs colonnes, vous pouvez les étendre en utilisant une liste vide comme définition de champ. Un champ situé à gauche d'un champ vide s'étend automatiquement afin d'occuper l'espace libre.
format	Informations sur le format du champ. Il s'agit du nombre de colonnes (*col*) ou d'une liste de la forme { *col tabs* } : *col* est le nombre de colonnes de la boîte de dialogue, et *tabs* spécifie éventuellement le nombre de tabulations entre les libellés et les champs en surbrillance. Cette liste peut être vide. Par défaut, *col* égale 1 et *tabs* égale 3.
{ *réinit.* }	Valeurs par défaut affichées lorsque la touche RESET est sélectionnée. Cette variable fournit les valeurs de réinitialisation de la liste dans le même ordre de spécification que celui des champs. Pour ne spécifier aucune valeur, utilisez la commande NOVAL afin de placer une valeur de garde. Cette liste peut être vide.
{ *init* }	Valeurs initiales affichées lorsque la boîte de dialogue apparaît. Cette variable fournit les valeurs initiales de la liste dans le même ordre de spécification que celui des champs. Pour ne spécifier aucune valeur, utilisez la commande NOVAL pour placer une valeur de garde. Cette liste peut être vide.

Si vous quittez la boîte de dialogue en sélectionnant OK ou ENTER, INFORM renvoie les valeurs de champs { *vals* } au niveau 2, et place 1 au niveau 1. (Si un champ est vide, NOVAL est renvoyée comme valeur de garde.) Si vous quittez la boîte de dialogue en sélectionnant CANCL ou CANCEL, INFORM renvoie 0.

Exemple : Les cinq lignes suivantes étant dans la pile :



appuyer sur INFOR affiche :



Commandes associées : CHOOSE, INPUT, NOVAL, TYPE

INPUT

Commande de saisie : Affiche un message invitant à saisir des données dans la ligne de commande et interdit l'accès aux opérations de la pile.

Niveau 2	Niveau 1	→	Niveau 1
" *message pile*"	" *message ligne-commande*"	→	" *résultat*"
" *message pile*"	{ $liste_{ligne-commande}$ }	→	" *résultat*"

Accès clavier : PRG NXT IN INPUT

Indicateurs : Aucun

Remarques : Lorsque la commande INPUT est exécutée, l'affichage de la pile est effacée et l'exécution du programme est suspendue pour permettre la saisie des données dans la ligne de commande. Le contenu de "*message pile*" s'affiche en haut de la pile. Selon l'argument du niveau 1, la ligne de commande peut aussi contenir une chaîne ou être vide. Lorsque vous appuyez sur ENTER, le programme reprend et renvoie le contenu de la ligne de commande au niveau 1 sous la forme d'une chaîne.

Dans sa forme générale, l'argument de niveau 1 pour INPUT est une liste spécifiant le contenu et l'interprétation de la ligne de commande. La liste peut comporter ***un ou plusieurs*** des paramètres suivants, ***dans n'importe quel ordre*** :

- "*message ligne-commande*", dont le contenu est placé dans la ligne de commande lorsque le programme s'interrompt.
- Soit un *nombre réel*, soit une *liste contenant deux nombres réels*, spécifiant la position initiale du curseur dans la ligne de commande :
  - Un nombre réel *n* au *n*ième caractère en partant de l'extrémité gauche de la première ligne. Une valeur ***positive*** de *n* spécifie le curseur d'insertion ; une valeur ***négative*** de *n* désigne le curseur de remplacement. 0 indique la fin de la chaîne de la ligne de commande.

- Une liste qui indique la position initiale du curseur (ligne et colonne) : le premier nombre de la liste désigne une ligne dans la ligne de commande (1 pour la première) ; le second correspond au nombre de caractères à partir de l'extrémité gauche de la ligne spécifiée. 0 indique la fin de la chaîne de la ligne de commande dans la ligne spécifiée. Un numéro de ligne positif représente le curseur d'insertion ; un numéro de ligne négatif correspond au curseur de remplacement.

- Un ou plusieurs des paramètres ALG, α ou V, saisis comme noms sans apostrophes :
  - ALG active le mode de saisie algébrique/de programme.
  - α ([α] [→] [A]) spécifie le verrouillage du calvier alpha.
  - V vérifie si les caractères dans la chaîne "résultat", sans les délimiteurs ", constituent un objet correct. Si ce n'est pas le cas, INPUT affiche l'avertissement "Invalid Syntax" et vous invite à ressaisir des données.

Vous pouvez spécifier un seul des paramètres de la liste du niveau 1. Les états par défaut de ces paramètres sont :

- Ligne de commande vide.
- Curseur d'insertion placé à la fin du message de la ligne de commande.
- Mode de saisie de programme.
- Absence de vérification de la syntaxe de la chaîne résultante.

Si vous spécifiez *uniquement* un message de ligne de commande pour l'argument de niveau 1, vous n'avez pas besoin de le placer dans une liste.

Commandes associées : PROMPT, STR→

INV

Fonction analytique d'inversion (1/x) : Renvoie la réciproque ou inverse d'une matrice.

{ }

Niveau 1	→	Niveau 1
z	→	*1/z*
[[*matrice*]]	→	[[*matrice*]] $^{-1}$
'*symb*'	→	'INV(*symb*)'
x_unité	→	*1/x_1/unité*

Accès clavier : 1/x

Indicateurs : Résultats numériques (−3)

Remarques : Pour un argument *complexe* (x, y), l'inverse est le nombre complexe $\left(\frac{x}{x^2+y^2}, \frac{-y}{x^2+y^2}\right)$.

Si les arguments sont des matrices, celles-ci doivent être carrées (réelles ou complexes). L'inverse calculé de la matrice A^{-1} satisfait $A \times A^{-1} = I_n$, où I_n est la matrice identité $n \times n$.

Commandes associées : SINV, /

IP

Fonction relative à la partie entière d'un nombre : Renvoie la partie entière de l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	*n*
x_unité	→	*n_unité*
'*symb*'	→	'IP(*symb*)'

Accès clavier : (MTH) REAL (NXT) IP

Indicateurs : Résultats numériques (−3)

Remarque : Le résultat a le même signe que l'argument.

Exemple : `32.3_m IP` renvoie `32_m`.

Commande associée : FP

IR

Commande de transmission série/infrarouge : Dirige les E-S et les sorties à imprimer vers le port série ("câble") ou infrarouge.

Accès clavier : (←)(I/O) IOPAR IR

Indicateurs : Unité d'E-S (−33), Unité d'impression (−34)

Remarques : Permet de basculer entre le mode de transmission infrarouge et par câble.

Pour de plus amples informations, reportez-vous à la variable réservée *IOPAR* (*paramètres d'E-S*) à l'Annexe D, "Variables réservées".

Commandes associées : BAUD, CKSM, PARITY, TRANSIO

ISOL

Commande d'isolation de variable : Renvoie une expression algébrique '*symb*$_2$' qui ré-arrange '*symb*$_1$' afin d'"isoler" la première occurrence de la variable *global*.

{}

Niveau 2	Niveau 1	→	Niveau 1
'*symb*$_1$'	'*global*'	→	'*symb*$_2$'

Accès clavier : ◀ SYMBOLIC ISOL

Indicateurs : Solution principale (−1), Résultats numériques (−3)

Lorsque l'indicateur −3 est armé, les résultats symboliques sont évalués à des nombres réels. Ainsi, le signe = est évalué. Si *global* ou toute autre variable de l'équation résultante est formelle, le message d'erreur "Undefined Name" est émis ; si *global* et toutes les autres variables ont des valeurs, un résultat numérique est renvoyé par le calcul *global* − *expression*. Ce résultat a une valeur limitée. En général, vous exécutez ISOL avec l'indicateur −3 désarmé.

Remarques : Le résultat '*symb*$_2$' est une équation de la forme '*global=expression*'. Si *global* apparaît plusieurs fois, '*symb*$_2$' est effectivement le membre droit d'une équation obtenue par ré-arrangement et résolution de '*symb*$_1$' en vue d'isoler la première occurrence de *global* dans le membre gauche de l'équation.

Si '*symb*$_1$' est une expression, elle est traitée comme le membre gauche d'une équation '*symb*$_1$=0'.

Si *global* apparaît dans l'argument d'une fonction au sein de '*symb*$_1$', cette fonction doit être *analytique* (fonction pour laquelle le HP 48 fournit un inverse). Ainsi, ISOL ne peut pas résoudre 'IP(X)=0' pour X, puisque IP n'a pas d'inverse.

Commandes associées : COLCT, EXPAN, QUAD, SHOW

KERRM

Commande d'affichage des messages d'erreur Kermit : Renvoie le texte du paquet d'erreur Kermit le plus récent.

Niveau 1	→	Niveau 1
	→	*"message-erreur"*

Accès clavier : ◀ I/O NXT KERR

Indicateurs : Aucun

Remarques : Si un transfert Kermit échoue après que l'unité Kermit connectée a envoyé un paquet d'erreur au HP 48, KERRM extrait et affiche le message d'erreur. (Les erreurs Kermit qui ne sont pas sous forme de paquet sont extraites par ERRM.)

Commandes associées : FINISH, KGET, PKT, RECN, RECV, SEND, SERVER

KEY

Commande de test des touches : Renvoie au niveau 1 un résultat de test, et en cas d'activation d'une touche, renvoie au niveau 2 la position de cette touche, désignée par le numéro de ligne et de colonne x_{nm}.

Niveau 1	→	Niveau 2	Niveau 1
	→	x_{nm}	1
	→		0

Accès clavier : PRG NXT IN KEY

Indicateurs : Aucun

KEY

Remarques : KEY renvoie un résultat faux (0) au niveau 1 jusqu'à ce qu'une touche soit actionnée. Lorsque vous appuyez sur une touche, KEY renvoie un résultat vrai (1) au niveau 1 et x_{nm} au niveau 2. Le résultat x_{nm} est un nombre à deux chiffres qui désigne la position (ligne et colonne) de la touche actionnée. Contrairement à WAIT, qui renvoie un nombre à trois chiffres identifiant les niveaux de clavier alpha et shifté, KEY renvoie la position de ***n'importe quelle*** touche actionnée, y compris ◀, ▶ et α.

Exemple : Le programme « DO UNTIL KEY END 71 SAME » renvoie 1 dans la pile si la touche ◀ est actionnée aussi longtemps que la boucle infinie s'exécute.

Commande associée : WAIT

KGET

Commande de transmission Kermit : Utilisée par une unité Kermit locale pour qu'un serveur Kermit transmette l'objet nommé.

Niveau 1	→	Niveau 1
'nom'	→	
"nom"	→	
{ nom_{ancien} $nom_{nouveau}$ }	→	
{ nom_1 ... nom_n }	→	
{{ nom_{ancien} $nom_{nouveau}$ } *nom* ... }	→	

Accès clavier : ◀ I/O SRVR KGET

Indicateurs : Unité d'E-S (−33), Ecrasement de RECV (−36), Messages d'E-S (−39)

L'indicateur Format de données des E-S (−35) affecte également KGET de la façon suivante :

- Si le serveur est un HP 48, l'indicateur −35 affecte KGET.

- Si le serveur n'est pas un HP 48 mais si le fichier à transférer provient d'un HP 48, l'indicateur −35 n'a aucun effet.
- Si le serveur n'est pas un HP 48 et si le fichier à transférer ne comporte pas l'en-tête *%%HP...;*, l'indicateur −35 spécifie la nécessité ou non d'analyser les données entrantes.

Remarques : Pour renommer un objet lorsque l'unité locale l'obtient, il suffit d'inclure l'ancien nom et le nouveau dans une liste imbriquée. Par exemple, `« AAA BBB » KGET` prend la variable nommée *AAA* et la renomme en *BBB*. `« AAA BBB » CCC » KGET` prend *AAA* comme *BBB*, et *CCC* sous son propre nom. (Si le nom d'origine n'est pas accepté par le HP 48, tapez-le sous la forme d'une chaîne.)

Commandes associées : BAUD, CKSM, FINISH, PARITY, RECN, RECV, SEND, SERVER, TRANSIO

KILL

Commande d'annulation de programme interrompu : Annule tous les programmes interrompus par HALT. (Les programmes interrompus sont généralement annulés par activation des touches [PRG] [NXT] `RUN KILL`.) Si la commande KILL est exécutée dans un programme, celui-ci est également annulé.

Accès clavier : [PRG] [NXT] `RUN KILL`

Indicateurs : Aucun

Remarques : Il est impossible de reprendre un programme annulé.

La commande KILL annule ***uniquement*** les programmes interrompus, ainsi que celui à partir duquel elle a été émise. Les commandes qui interrompent les programmes sont HALT et PROMPT.

Les programmes *suspendus* ne peuvent pas être annulés. Les commandes qui suspendent les programmes sont INPUT et WAIT.

Commandes associées : CONT, DOERR, HALT, PROMPT

LABEL

Commande de libellé des axes : Libelle les axes Dans *PICT* avec les noms de variables des axes x et y et les valeurs minimale et maximale de la plage d'affichage.

Accès clavier : ◀ PLOT NXT LABEL

Indicateurs : Aucun

Remarques : Le choix du nom de l'axe horizontal s'effectue selon l'ordre de priorité suivant :

1. Si le paramètre *axes* de la variable réservée *PPAR* est une liste, l'élément *axe-x* de cette liste est utilisé.
2. Si le paramètre *axes* n'est pas une liste, le nom de variable indépendante figurant dans *PPAR* est adopté.

Le choix du nom de l'axe vertical s'effectue selon l'ordre de priorité suivant :

1. Si le paramètre *axes* dans *PPAR* est une liste, l'élément *axe-y* de cette liste est utilisé.
2. Si *axes* n'est pas une liste, le nom de variable dépendante figurant dans PPAR est adopté.

Commandes associées : AXES, DRAW, DRAX

LAST

Commande de rappel des derniers arguments : Renvoie la copie des arguments de la dernière commande exécutée.

Accès clavier : Aucun. A saisir.

Remarque : LAST est fournit à des fins de compatibilité avec le HP 28S. LAST équivaut à LASTARG (voir cette dernière).

LASTARG

Commande de rappel des derniers arguments : Renvoie la copie des arguments de la dernière commande exécutée.

Niveau 1	→	Niveau n	...	Niveau 1
	→	obj_n	...	obj_1

Accès clavier :

(→) (ARG)

(PRG) (NXT) ERROR LASTA

Indicateurs : Derniers arguments (−55)

Remarques : Les objets sont renvoyés aux niveaux de la pile qu'ils occupaient à l'origine. Les commandes qui ne prennent pas d'arguments ne modifient pas les arguments sauvegardés en cours.

Lorsque LASTARG suit une commande qui évalue une expression algébrique ou un programme (comme ∂, $\int$, TAYLR, COLCT, DRAW, ROOT, ISOL, EVAL et →NUM), les derniers arguments sauvegardés proviennent de l'expression ou du programme évalué, non de la commande initiale.

Commande associée : LAST

LCD→

Commande de création d'un objet graphique à partir de l'affichage : Renvoie un objet graphique 131 × 64 représentant l'affichage de la pile et des menus.

Niveau 1	→	Niveau 1
	→	*grob*

LCD→

Accès clavier : PRG GROB NXT LCD→

Indicateurs : Aucun

Exemple : LCD→ PICT STO PICTURE renvoie au niveau 1 l'affichage en cours sous la forme d'un objet graphique, le stocke dans *PICT*, puis affiche l'image dans l'environnement Picture.

Commandes associées : →GROB, →LCD

→LCD

Commande d'insertion d'un objet graphique dans l'affichage de la pile : Affiche l'objet graphique du niveau 1, en positionnant le pixel supérieur gauche au niveau de l'angle supérieur gauche de l'affichage.

{ }

Niveau 1	→	Niveau 1
grob	→	

Accès clavier : PRG GROB NXT →LCD

Indicateurs : Aucun

Remarque : Si l'objet graphique dépasse les dimensions 131 × 56, il est tronqué.

Commandes associées : BLANK, →GROB, LCD→

LIBEVAL

Commande d'évaluation des fonctions de bibliothèque : Evalue des fonctions de bibliothèque non nommées.

{ }

Niveau 1	→	Niveau 1
#$n_{fonction}$	→	

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : L'utilisation de LIBEVAL avec des adresses aléatoires peut endommager la mémoire. #$n_{fonction}$ a la forme *lllfffh*, où *lll* est le numéro de bibliothèque et *fff* le numéro de fonction.

Commandes associées : EVAL, SYSEVAL

LIBS

Commande relative aux bibliothèques : Indique le titre, le numéro et le port de chaque bibliothèque associée au répertoire en cours.

Niveau 1 →	Niveau 1
→	{ "*titre*" n_{bib} n_{port} ... "*titre*" n_{bib} n_{port} }

Accès clavier : ◀ LIBRARY LIBS

Indicateurs : Aucun

Remarques : Le titre d'une bibliothèque prend souvent la forme *NOM-BIBLIOTHEQUE* : *Description*. Une bibliothèque sans titre est affichée sous la forme "".

Commandes associées : ATTACH, DETACH

LINE

Commande de tracé de lignes : Trace une ligne dans *PICT* entre les coordonnées des niveaux 1 et 2.

Niveau 2	Niveau 1	→	Niveau 1
(x_1, y_1)	(x_2, y_2)	→	
{ # n_1 # m_1 }	{ # n_2 # m_2 }	→	

Accès clavier : PRG PICT LINE

Indicateurs : Aucun

Exemple : Le programme

```
« (0,0) (2,3) LINE ( # 0d # 0d ) PVIEW 7 FREEZE »
```

trace une ligne dans *PICT* entre deux coordonnées exprimées en unités-utilisateur, affiche *PICT* avec les coordonnées en pixels (# 0d # 0d) dans l'angle supérieur gauche de l'affichage d'image puis gèle l'affichage.

Commandes associées : ARC, BOX, TLINE

ΣLINE

Commande relative à la formule d'un modèle de régression : Renvoie une expression représentant la droite d'ajustement la mieux adaptée en fonction du modèle statistique en cours, en utilisant *X* comme nom de variable indépendante et, pour la pente et l'intersection, des valeurs explicites extraites de la variable réservée *ΣPAR*.

Niveau 1	→	Niveau 1
	→	'$symb_{formule}$'

Accès clavier : ◀ STAT FIT ΣLINE

Indicateurs : Aucun

Remarque : Pour chaque modèle d'ajustement de courbe, le tableau suivant indique la forme de l'expression renvoyée par ΣLINE, où m est la pente, x la variable indépendante, et b l'intersection.

Modèle	Forme de l'expression
LINFIT	$mx + b$
LOGFIT	$m \ln(x) + b$
EXPFIT	be^{mx}
PWRFIT	bx^{m}

Exemple : Si le modèle en cours est EXPFIT, la pente 5 et l'intersection 3, ΣLINE renvoie `'3*EXP(5*X)'`.

Commandes associées : BESTFIT, COLΣ, CORR, COV, EXPFIT, LINFIT, LOGFIT, LR, PREDX, PREDY, PWRFIT, XCOL, YCOL

LINFIT

Commande d'ajustement linéaire : Stocke LINFIT dans le cinquième paramètre de la variable réservée ΣPAR, indiquant que les exécutions suivantes de LR doivent utiliser le modèle d'ajustement linéaire pour la courbe.

Accès clavier : ◀ STAT ΣPAR MODL LINFI

Indicateurs : Aucun

Remarques : LINFIT est le modèle par défaut dans ΣPAR. Pour une description de ΣPAR, voir l'Annexe D, "Variables réservées".

Commandes associées : BESTFIT, EXPFIT, LOGFIT, LR, PWRFIT

LININ

Fonction de test de linéarité : Teste, pour une variable donnée, la structure linéaire d'une équation algébrique.

{}

Niveau 2	Niveau 1	→	Niveau 1
'symb'	*'nom'*	→	0/1

Accès clavier : PRG TEST ↰ PREV LININ

Indicateurs : Aucun

Remarques : Si deux sous-expressions quelconques contenant une variable (*nom*) sont combinées uniquement par addition et soustraction, et une sous-expression quelconque contenant la variable est au maximum multipliée ou divisée par un autre facteur sans la variable, l'expression algébrique ('*symb*') est linéaire pour cette variable.

LININ renvoie 1 si l'expression algébrique est linéaire pour la variable et 0 dans le cas contraire.

Exemple :

```
'(X+1)*(Y^-2^Z)+(x/(3-Z^3)'
'X'
LININ
```

renvoie 1.

```
'(X^2-1)/(X+1)'
'X'
LININ
```

renvoie 0.

(Bien que cette équation donne lieu à une équation linéaire lorsqu'elle est factorisée, LININ la teste comme indiqué ci-dessus.)

LIST→

Commande de séparation d'une liste dans la pile : Prend une liste de *n* objets et les renvoie à des niveaux distincts, puis renvoie le nombre total d'objets au niveau 1.

Niveau 1	→	Niveau n+1 ...	Niveau 2	Niveau 1
{ obj_1 ... obj_n }	→	obj_1 ...	obj_n	*n*

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarque : La commande OBJ→ exécute également cette fonction. LIST→ est fournie à des fins de compatibilité avec le HP 28S.

Commandes associées : ARRY→, DTAG, EQ→, →LIST, OBJ→, STR→

→LIST

Commande de création d'une liste à partir de la pile : Prend *n* objets du niveau n+1 au niveau 2 et renvoie une liste de ces *n* objets.

Niveau n+1 ...	Niveau 2	Niveau 1	→	Niveau 1
obj_1 ...	obj_n	*n*	→	{ obj_1 ... obj_n }

Accès clavier : PRG TYPE →LIST

Indicateurs : Aucun

Exemple : Le programme

```
« DEPTH →LIST 'A' STO »
```

combine le contenu de la pile en une liste stockée dans la variable *A*.

→LIST

Commandes associées : →ARRY, LIST→, →STR, →TAG, →UNIT

ΣLIST

Commande de sommation des éléments d'une liste : Renvoie la somme des éléments d'une liste.

Niveau 1	→	Niveau 1
{ *liste* }	→	*sum*

Accès clavier : (MTH) LIST ΣLIST

Indicateurs : Aucun

Remarques : Les éléments de la liste doivent pouvoir s'additionner entre eux.

Exemples : { 5 8 2 } ΣLIST renvoie 15.

{ A B C 1 } ΣLIST renvoie 'A+B+C+1'.

Commandes associées : ΠLIST, STREAM

ΔLIST

Commande de calcul des différences dans une liste : Renvoie les différences du premier ordre d'une suite dans une liste.

Niveau 1	→	Niveau 1
{ *liste* }	→	{ *différences* }

Accès clavier : (MTH) LIST ΔLIST

Indicateurs : Aucun

Remarque : Les éléments adjacents de la liste doivent pouvoir se soustraire entre eux.

Exemples : ≪ 4 20 1 17 60 91 ≫ ΔLIST renvoie ≪ 16 -19 16 43 31 ≫.

≪ A B C 1 2 3 ≫ ΔLIST renvoie ≪ 'B-A' 'C-B' '1-C' 1 1 ≫.

≪ 'A+3' 'X/5' 'Y^4' ≫ ΔLIST renvoie
≪ 'X/5-(A+3)' 'Y^4-X/5' ≫.

Commandes associées : ΣLIST, ΠLIST, STREAM

ΠLIST

Commande de calcul du produit des éléments d'une liste :
Renvoie le produit des éléments d'une liste.

Niveau 1	→	Niveau 1
{ *liste* }	→	*produit*

Accès clavier : PRG LIST ΠLIST

Indicateurs : Aucun

Remarque : Les éléments de la liste doivent pouvoir se multiplier entre eux.

Exemples : ≪ 5 8 2 ≫ ΠLIST renvoie 80.

≪ A B C 1 ≫ ΠLIST renvoie 'A*B*C'.

Commandes associées : ΣLIST, ΔLIST, STREAM

LN

Fonction analytique logarithme népérien : Renvoie le logarithme népérien (base e) de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	ln z
'symb'	→	'LN(*symb*)'

Accès clavier : ↷ LN

Indicateurs : Solution principale (−1), Résultats numériques (−3), Exception de résultat infini (−22)

Remarques : Pour x=0 ou (0, 0), il se produit une exception de résultat infini (`Infinite Result`) ou, si l'indicateur −22 est armé, −MAXR est renvoyé.

L'inverse de EXP est une *relation*, non une fonction, puisque EXP envoie plusieurs arguments au même résultat. La relation inverse de EXP est exprimée par ISOL comme la *solution générale* :

```
'LN(Z)+2*π*i*n1'
```

La fonction LN est l'inverse d'une *partie* de EXP, partie définie par restriction du domaine de EXP de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points de ce domaine restreint de EXP constituent les *valeurs principales* de la relation inverse. LN dans sa totalité est appelée *détermination principale* de la relation inverse, et les points envoyés par LN à la frontière du domaine restreint de EXP constituent les *coupures* de LN.

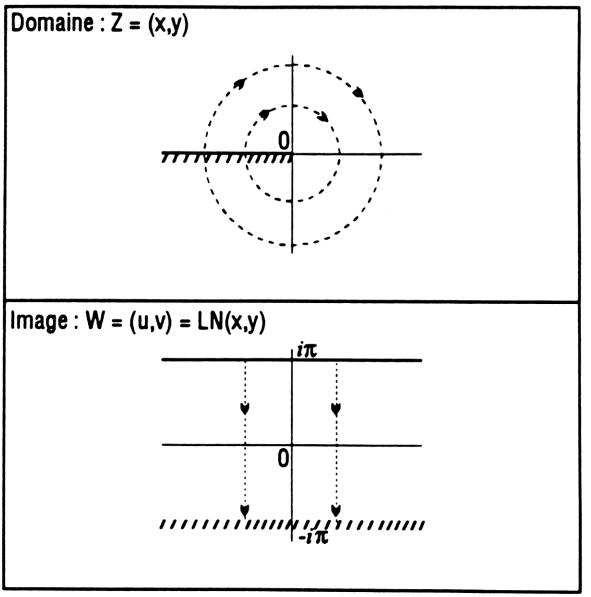
La détermination principale utilisée par le HP 48 pour LN a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse *à valeurs réelles* sont définis. La coupure pour la fonction de logarithme népérien à valeur complexe apparaît dans une zone où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Les graphiques ci-après montrent le domaine et l'image de LN. Le graphique représentant le domaine montre l'emplacement de la coupure : le trait plein marque un bord de la coupure, les hachures marquent l'autre bord. Le graphique représentant l'image indique l'endroit où chaque bord de la coupure est projeté par la fonction.

Ces graphiques montrent la relation inverse `'LN(Z)+2*π*i*n1'` pour *n1*=0. Pour d'autres valeurs de *n1*, la bande horizontale du graphique inférieur est translatée vers le haut (pour *n1* positif) ou vers le bas (pour *n1* négatif). La réunion de ces bandes couvre l'ensemble du plan complexe, qui représente le domaine de EXP.

Il suffit de regarder ces graphiques en inversant le domaine et l'image pour voir comment le domaine de EXP est restreint afin de permettre une *fonction* inverse. Considérez la bande verticale du graphique inférieur comme étant le domaine restreint `Z = (x,y)`. EXP projette ce domaine sur la totalité du plan complexe dans l'image `W = (u,v) = EXP(x,y)` du graphique supérieur.

Commandes associées : ALOG, EXP, ISOL, LNP1, LOG



LNP1

Fonction analytique logarithme népérien x Plus 1 : Renvoie ln (x + 1).

{ }

Niveau 1	→	Niveau 1
x	→	ln(x+1)
'*symb*'	→	'LNP1(*symb*)'

Accès clavier : (MTH) HYP (NXT) LNP1

Indicateurs : Résultats numériques (−3), Exception de résultat infini (−22)

Remarques : Pour des valeurs de x proches de zéro, 'LNP1(x)' renvoie un résultat plus précis que 'LN(x+1)'. L'utilisation de LNP1 permet d'avoir un argument et un résultat proches de zéro, et elle évite un résultat intermédiaire voisin de 1. Le calculateur peut exprimer des nombres allant jusqu'à 10^{-449} par rapport à zéro, mais seulement jusqu'à 10^{-11} par rapport à 1.

Pour des valeurs de $x < -1$, il se produit une exception du type Undefined Result. Pour $x=-1$, une exception Infinite Result survient ou, si l'indicateur −22 est armé, LNP1 renvoie −MAXR.

Commandes associées : EXPM, LN

LOG

Fonction analytique logarithme commun : Renvoie le logarithme commun (base 10) de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	log *z*
'*symb*'	→	'LOG(*symb*)'

Accès clavier : ↷ LOG

Indicateurs : Solution principale (−1), Résultats numériques (−3), Exception de résultat infini (−22)

Remarques : Pour x=0 or (0, 0), il se produit une exception `Infinite Result` ou, si l'indicateur −22 est armé (pas d'erreur), LOG renvoie −MAXR.

L'inverse de ALOG est une *relation*, non une fonction, puisque ALOG envoie plusieurs arguments au même résultat. La relation inverse de ALOG est exprimée par ISOL comme la *solution générale* :

```
'LOG(Z)+2*π*i*n1/2.30258509299'
```

La fonction LN est l'inverse d'une *partie* de ALOG, partie définie par restriction du domaine de ALOG de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points de ce domaine restreint de ALOG constituent les *valeurs principales* de la relation inverse. LOG dans sa totalité est appelée la *détermination principale* de la relation inverse, et les points envoyés par LOG à la frontière du domaine restreint de ALOG constituent les *coupures* de LOG.

La détermination principale utilisée par le HP 48 pour LOG(z) a été choisie car elle est analytique dans les régions où les arguments de la fonction inverse *à valeurs réelles* sont définis. La coupure pour la fonction LOG à valeur complexe apparaît dans une zone où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

LOG

Vous pouvez déterminer le graphique pour LOG(z) à partir du graphique de LN (voir cette dernière) et la relation $\log z = \ln z \,/\, \ln 10$.

Commandes associées : ALOG, EXP, ISOL, LN

LOGFIT

Commande d'ajustement logarithmique : Stocke LOGFIT dans le cinquième paramètre de la variable réservée ΣPAR, indiquant que les exécutions suivantes de LR doivent utiliser le modèle logarithmique d'ajustement pour la courbe.

Accès clavier : ◀ STAT ΣPAR MODL LOGFI

Indicateurs : Aucun

Remarques : LINFIT est le modèle par défaut dans ΣPAR. Pour une description de ΣPAR, voir l'Annexe D, "Variables réservées".

Commandes associées : BESTFIT, EXPFIT, LINFIT, LR, PWRFIT

LQ

Commande de factorisation LQ d'une matrice : Renvoie la factorisation LQ d'une matrice $n \times m$.

{ }

Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
[[*matrice*]] $_A$	→	[[*matrice*]] $_L$	[[*matrice*]] $_Q$	[[*matrice*]] $_P$

Accès clavier : MTH MATR FACTR LQ

Indicateurs : Aucun

Remarques : LQ factorise une matrice $m \times n$ A en trois matrices :

- L est une matrice trapézoïdale inférieure $m \times n$.
- Q est une matrice orthogonale $n \times n$.
- P est une matrice de permutation $m \times m$.

Où $P \times A = L \times Q$.

Commandes associées : LSQ, QR

LR

Commande de régression linéaire : Utilise le modèle statistique sélectionné pour calculer les coefficients de régression linéaire (intersection et pente) pour les variables dépendantes et indépendantes choisies dans la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 2	Niveau 1
	→	Intersection: x_1	Pente: x_2

Accès clavier : ↩ STAT FIT LR

Indicateurs : Aucun

Remarques : Les colonnes de données dépendantes et indépendantes sont spécifiées par les deux premiers éléments de la variable réservée ΣPAR, définis respectivement par XCOL et YCOL. (Les colonnes dépendantes et indépendantes par défaut sont 1 et 2.) Le modèle statistique sélectionné est le cinquième élément de ΣPAR. LR stocke l'intersection et la pente (non identifiés) respectivement dans les troisième et quatrième éléments de ΣPAR.

Les coefficients des modèles d'ajustement exponentiel (EXPFIT), logarithmique (LOGFIT) et de puissance (PWRFIT) sont calculés à l'aide de transformations qui permettent d'ajuster les données par régression linéaire standard. Les équations pour ces transformations sont indiquées dans le tableau suivant (b est l'intersection et m la pente). Le modèle logarithmique requiert des valeurs x positives (XCOL), le modèle exponentiel nécessite des valeurs y positives

LR

(YCOL), et le modèle de puissance requiert des valeurs x et y positives.

Pour une description de ΣPAR, voir l'Annexe D, "Variables réservées".

Equations de transformation

Modèle	Transformation
Logarithmique	$y = b + m \ln(x)$
Exponentiel	$\ln(y) = \ln(b) + mx$
Puissance	$\ln(y) = \ln(b) + m \ln(x)$

Commandes associées : BESTFIT, COLΣ, CORR, COV, EXPFIT, ΣLINE, LINFIT, LOGFIT, PREDX, PREDY, PWRFIT, XCOL, YCOL

LSQ

Commande de solution des moindres carrés : Renvoie la norme minimale, selon la méthode des moindres carrés, d'un système d'équations linéaires $A \times X = B$.

{ }

Niveau 2	Niveau 1	→	Niveau 1
[*tableau*]$_B$	[[*matrice*]]$_A$	→	[*tableau*]$_X$
[[*tableau*]]$_B$	[[*matrice*]]$_A$	→	[[*matrice*]]$_X$

Accès clavier :

(←) (SOLVE) SYS LSQ

(MTH) MATR LSQ

Indicateurs : Eléments "très petits" (−54)

Remarques : Si B est un vecteur, le vecteur résultat possède la norme euclidienne minimale $||X||$ parmi tous les vecteurs qui minimisent la norme euclidienne résiduelle $||A \times X - B||$. Si B est une matrice, chaque colonne de la matrice résultat X_i possède la norme euclidienne minimale $||X_i||$ parmi tous les vecteurs qui minimisent la norme euclidienne résiduelle $||A \times X_i - B_i||$.

Si A a moins de lignes que de colonnes (système d'équations sous-déterminé), il existe un nombre infini de solutions. LSQ renvoie la solution avec la longueur euclidienne minimale.

Si A a moins de colonnes que de lignes (système d'équations sur-déterminé), il se peut qu'il n'existe aucune solution satisfaisant à toutes les équations. LSQ renvoie la solution avec les résidus minimisés de $A \times X - B$.

Commandes associées : LQ, RANK, QR, /

LU

Commande de décomposition LU d'une matrice carrée: Renvoie la décomposition LU d'une matrice carrée.

{ }

Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
[[*matrice*]]$_A$	→	[[*matrice*]]$_L$	[[*matrice*]]$_U$	[[*matrice*]]$_P$

Accès clavier : (MTH) MATR FACTR LU

Indicateurs : Aucun

Remarques : Lors de la résolution d'un système d'équations exactement déterminé, de l'inversion d'une matrice carrée, ou du calcul du déterminant d'une matrice, le HP 48 factorise une matrice carrée en sa décomposition LU de Crout en utilisant un pivot partiel.

La décomposition LU de Crout de A est une matrice triangulaire inférieure L, une matrice triangulaire supérieure U contenant l'une de

ses diagonales, et une matrice de permutation P de façon que $P \times A = L \times U$. Le résultat satisfait à $P \times A \cong L \times U$.

Commandes associées : DET, INV, LSQ, /

MANT

Fonction mantisse : Renvoie la mantisse de l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	y_{mant}
'*symb*'	→	'MANT(*symb*)'

Accès clavier : MTH REAL NXT MANT

Indicateurs : Résultats numériques (−3)

Exemple : `-1.2E34 MANT` renvoie `1.2`.

Commandes associées : SIGN, XPON

↑MATCH

Commande de réécriture du bas vers le haut : Réécrit une expression.

Niv 2	Niv 1	→	Niv 2	Niv 1
'$symb_1$'	{ '$symb_{cfg}$' '$symb_{remp}$' }	→	'$symb_2$'	0/1
'$symb_1$'	{ '$symb_{cfg}$' '$symb_{remp}$' '$symb_{cond}$' }	→	'$symb_2$'	0/1

Accès clavier : ← SYMBOLIC NXT ↑MAT

Indicateurs : Aucun

Remarques : ↑MATCH réécrit des expressions ou des sous-expressions qui correspondent à une configuration spécifiée '$symb_{cfg}$'. Une condition facultative, '$symb_{cond}$', permet de restreindre le processus. Un résultat de test est également renvoyé, indiquant si l'exécution de la commande a donné lieu à une réécriture : 1 si c'est le cas, 0 dans le cas contraire.

La configuration '$symb_{cfg}$' et l'expression de remplacement '$symb_{repl}$' peuvent être des expressions normales ; par exemple, vous pouvez remplacer 'SIN(π/6)' par '1/2'. L'utilisation de "jokers" dans la configuration (pour correspondre à n'importe quelle sous-expression) et dans la chaîne de remplacement (pour représenter cette expression) est également possible. Un joker est un nom commençant par &, tel que '&A', utilisé pour remplacer 'SIN(&A+π)' par '−SIN(&A)'. Si une configuration contient plusieurs occurrences d'un joker donné, elles doivent correspondre à des sous-expressions identiques.

↑MATCH procède du bas vers le haut ; elle vérifie donc en premier les sous-expressions de plus bas niveau (celles qui sont au niveau inférieur d'imbrication). Cette méthode convient particulièrement aux simplifications. Une sous-expression simplifiée durant une exécution de ↑MATCH devient un argument plus simple de l'expression parente, de sorte que cette dernière peut être simplifiée par une nouvelle exécution de ↑MATCH. Une seule exécution de ↑MATCH peut simplifier plusieurs sous-expressions à condition qu'aucune ne soit la sous-expression d'une autre.

Exemples :

La séquence :

```
'SIN(π/6)' { 'SIN(π/6) '1/2' } ↑MATCH
```

renvoie '1/2' au niveau 2 et 1 (indiquant qu'un remplacement a eu lieu) au niveau 1.

La séquence :

```
'SIN(X+π)' { 'SIN(&A+π)' '−SIN(&A)' } ↑MATCH
```

renvoie '−SIN(X)' au niveau 2 et 1 au niveau 1.

La séquence :

↑MATCH

```
'W+√(SQ(5))' { '√(SQ&A))' '&A' '&A≥0' } ↑MATCH
```

renvoie 'W+5' au niveau 2 et 1 au niveau 1.

Commande associée : ↓MATCH

↓MATCH

Commande de réécriture du haut vers le bas : Réécrit une expression.

Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
'*symb*$_1$'	{ '*symb*$_{cfg}$' '*symb*$_{rem\,p}$' }	→	'*symb*$_2$'	0/1
'*symb*$_1$'	{ '*symb*$_{cfg}$' '*symb*$_{rem\,p}$' '*symb*$_{cond}$' }	→	'*symb*$_2$'	0/1

Accès clavier : ◀ SYMBOLIC NXT ↓MAT

Indicateurs : Aucun

Remarques : ↓MATCH réécrit des expressions ou des sous-expressions qui correspondent à une configuration spécifiée '*symb*$_{cfg}$'. Une condition facultative, '*symb*$_{cond}$', permet de restreindre le processus. Un résultat de test est également renvoyé, indiquant si l'exécution de la commande a donné lieu à une réécriture : 1 si c'est le cas, 0 dans le cas contraire.

La configuration '*symb*$_{cfg}$' et l'expression de remplacement '*symb*$_{repl}$' peuvent être des expressions normales ; ainsi, vous pouvez remplacer .5 par 'SIN(π/6)'. Vous pouvez aussi utiliser un "joker" dans la configuration (à la place d'une sous-expression quelconque) et dans la chaîne de remplacement (pour représenter l'expression). Un joker est un nom commençant par &, comme '&A', utilisé pour remplacer 'SIN(&A+&B)' par 'SIN(&A)*COS(&B)+COS(&A)*SIN(&B)'. Si un joker apparaît plusieurs fois dans une configuration, ses multiples occurrences doivent correspondre à des sous-expressions identiques.

↓MATCH procède du haut vers le bas ; elle vérifie donc en premier l'ensemble de l'expression. Cette méthode convient au développement

des termes. Une expression développée durant une exécution de ↓MATCH contient des sous-expressions supplémentaires, qui peuvent être développées par une nouvelle exécution de ↓MATCH. Une seule exécution de ↓MATCH peut développer plusieurs expressions, à condition qu'aucune ne soit la sous-expression d'une autre.

Exemples :

```
.5 { .5 'SIN(π/6)' } ↓MATCH
```

renvoie 'SIN(π/6)' au niveau 2 et 1 au niveau 1.

```
'SIN(U+V)' { 'SIN(&A+&B)'
'SIN(&A)*COS(&B)+COS(&A)*SIN(&B)' } ↓MATCH
```

renvoie 'SIN(U)*COS(V)+COS(U)*SIN(V)' au niveau 2 et 1 au niveau 1.

La séquence :

```
'SIN(5*Z)' { 'SIN(&A+&B)'
'Σ(K=0,&A,COMB(&A,K)*SIN(K*π)*
COS(&B^(&A-K)*SIN(&B)^K)'
'ABS(IP(&A))==&A' } ↓MATCH
```

renvoie

```
'Σ(K=0,5,COMB(5,K)*SIN(K*π)*COS(Z)^(5-K)*SIN(Z)^K)'
```

au niveau 2 et 1 au niveau 1.

Commande associée : ↑MATCH

MAX

Fonction maximum : Renvoie le plus grand des arguments.

MAX

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	max(*x*, *y*)
x	'*symb*'	→	'MAX(*x*, *symb*)'
'*symb*'	*x*	→	'MAX(*symb*, *x*)'
'$symb_1$'	'$symb_2$'	→	'MAX($symb_1$, $symb_2$)'
$x\_unité_1$	$y\_unité_2$	→	max($x\_unité_1$, $y\_unité_2$)

Accès clavier : MTH REAL MAX

Indicateurs : Résultats numériques (−3)

Exemples : 10 -23 MAX renvoie 10.

-10 -23 MAX renvoie -10.

1_m 9_cm MAX renvoie 1_m.

Commande associée : MIN

MAXR

Fonction nombre réel maximal : Renvoie la constante symbolique 'MAXR' ou sa représentation numérique 9.99999999999E499.

Niveau 1	→	Niveau 1
	→	'MAXR'
	→	9.99999999999E499

Accès clavier : MTH NXT CONS NXT MAXR

Indicateurs : Constantes symboliques (−2), Résultats numériques (−3)

MAXR renvoie sa représentation numérique si l'indicateur −2 ou −3 est armé ; sinon, elle renvoie sa représentation symbolique.

Remarque : MAXR est la plus grande valeur numérique représentable par le HP 48.

Commandes associées : e, i, MINR, π

MAXΣ

Commande de calcul des valeurs maximales : Calcule les valeurs maximales de chacune des m colonnes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	x_{max}
	→	$[x_{max1}\ x_{max2}\ \cdots\ x_{maxm}]$

Accès clavier : ◀ STAT 1VAR MAXΣ

Indicateurs : Aucun

Remarque : Les valeurs maximales sont renvoyées sous forme d'un vecteur de m nombres réels ou d'un seul nombre réel si $m = 1$.

Commandes associées : BINS, MEAN, MINΣ, SDEV, TOT, VAR

MCALC

Commande de déclaration de variable à valeur calculée : Déclare une variable comme valeur calculée (non définie par l'utilisateur) pour le Solver d'équations multiples.

MCALC

Niveau 1	→	Niveau 1
'nom'	→	
{ *liste* }	→	
"ALL"	→	

Accès clavier : ◀ EQ LIB MES MCAL

Indicateurs : Aucun

Remarque : MCALC déclare une seule variable, une liste de variables ou toutes les variables comme valeurs calculées.

Commande associée : MUSER

MEAN

Commande de calcul d'une moyenne : Renvoie la moyenne de chacune des m colonnes de coordonnées de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{moyenne}$
	→	$[\, x_{moyenne1}\ x_{moyenne2}\ \cdots\ x_{moyennem}\,]$

Accès clavier : ◀ STAT 1VAR MEAN

Indicateurs : Aucun

Remarques : La moyenne est renvoyée sous la forme d'un vecteur de m nombres réels, ou d'un seul nombre réel si $m = 1$. La moyenne est calculée selon la formule :

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$

où x_i est la $i$ème valeur de coordonnées d'une colonne, et n le nombre de points de données.

Commandes associées : BINS, MAXΣ, MINΣ, SDEV, TOT, VAR

MEM

Commande de vérification de la mémoire disponible : Renvoie le nombre d'octets de RAM disponible.

Niveau 1	→	Niveau 1
	→	*x*

Accès clavier : ◀ MEMORY MEM

Indicateurs : Aucun

Remarques : Le nombre renvoyé n'est qu'une approximation de la mémoire disponible, car les fonctions de récupération (LASTARG, ▶ UNDO, et ▶ CMD) utilisent ou libèrent des quantités variables de mémoire en fonction des opérations.

Avant de déterminer la quantité de mémoire disponible, MEM doit supprimer les objets placés en mémoire temporaire qui ne sont plus utilisés. Ce processus de nettoyage se déroule par ailleurs automatiquement lorsque la mémoire est saturée. Etant donné qu'il risque de ralentir le fonctionnement du calculateur à des moments inopportuns, vous avez la possibilité de le déclencher lorsque vous le souhaitez en exécutant MEM. Dans un programme, exécutez MEM DROP.

Commande associée : BYTES

MENU

Commande d'affichage d'un menu : Affiche un menu intégré ou un menu de bibliothèque, ou définit et affiche un menu personnalisé.

Niveau 1	→	Niveau 1
x_{menu}	→	
{ $liste_{définition}$ }	→	
'$nom_{définition}$'	→	
obj	→	

Indicateurs : Aucun

Remarques : Un menu intégré est spécifié par un nombre réel x_{menu}. Le format de x_{menu} est *mm.pp*, où *mm* est le numéro du menu et *pp* la page du menu. Si *pp* ne correspond pas à une page du menu spécifié, la première page s'affiche. Le tableau suivant récapitule les menus intégrés du HP 48 et les numéros correspondants.

No	Nom
0	Last menu
1	CST
2	VAR
3	MTH
4	MTH VECTR
5	MTH MATR
6	MTH MATR MAKE
7	MTH MATR NORM
8	MTH MATR FACTR
9	MTH MATR COL
10	MTH MATR ROW
11	MTH LIST
12	MTH HYP
13	MTH PROB
14	MTH REAL

No	Nom
15	MTH BASE
16	MTH BASE LOGIC
17	MTH BASE BIT
18	MATH BASE BYTE
19	MTH FFT
20	MTH CMPL
21	MTH CONS
22	PRG
23	PRG BRCH
24	PRG BRCH IF
25	PRG BRCH CASE
26	PRG BRCH START
27	PRG BRCH FOR
28	EDIT
29	PRG BRCH DO

No	Nom
30	SOLVE ROOT SOLVR
31	PRG BRCH WHILE
32	PRG TEST
33	PRG TYPE
34	PRG LIST
35	PRG LIST ELEM
36	PRG LIST PROC
37	PRG GROB
38	PRG PICT
39	PRG IN
40	PRG OUT
41	PRG RUN
42	UNITS (menu du catalogue des unités)
43	UNITS LENG
44	UNITS AREA
45	UNITS VOL
46	UNITS TIME
47	UNITS SPEED
48	UNITS MASS
49	UNITS FORCE
50	UNITS ENRG
51	UNITS POWR
52	UNITS PRESS
53	UNITS TEMP
54	UNITS ELEC
55	UNITS ANGL
56	UNITS LIGHT
57	UNITS RAD
58	UNITS VISC
59	UNITS
60	PRG ERROR IFERR
61	PRG ERROR

No	Nom
62	CHARS
63	MODES
64	MODES FMT
65	MODES ANGL
66	MODES FLAG
67	MODES KEYS
68	MODES MENU
69	MODES MISC
70	MEMORY
71	MEMORY DIR
72	MEMORY ARITH
73	STACK
74	SOLVE
75	SOLVE ROOT
76	SOLVE DIFFE
77	SOLVE POLY
78	SOLVE SYS
79	SOLVE TVM
80	SOLVE TVM SOLVR
81	PLOT
82	PLOT PTYPE
83	PLOT PPAR
84	PLOT 3D
85	PLOT 3D PTYPE
86	PLOT 3D VPAR
87	PLOT STAT
88	PLOT STAT PTYPE
89	PLOT STAT ΣPAR
90	PLOT STAT ΣPAR MODL
91	PLOT STAT DATA
92	PLOT FLAG
93	SYMBOLIC

MENU

No	Nom
94	TIME
95	TIME ALARM
96	STAT
97	STAT DATA
98	STAT ΣPAR
99	STAT ΣPAR MODL
100	STAT 1VAR
101	STAT PLOT
102	STAT FIT
103	STAT SUMS
104	IO
105	IO SRVR
106	IO IOPAR

No	Nom
107	IO PRINT
108	IO PRINT PRTPA
109	IO SERIA
110	LIBRARY
111	LIBRARY PORTS
112	LIBRARY PORTS :0:
113	EQ LIB
114	EQ LIB EQLIB
115	EQ LIB COLIB
116	EQ LIB MES
117	EQ LIB UTILS

Les menus des bibliothèques sont spécifiés de la même façon, le numéro de bibliothèque servant de numéro de menu.

Les menus personnalisés sont spécifiés par une liste de la forme *{ "libellé-objet" action-objet }* (voir l'Annexe D, "Variables réservées", pour de plus amples informations) ou par un nom contenant une liste (*'nom$_{définition}$'*). Ces arguments sont stockés dans la variable réservée *CST* pour afficher ensuite le menu personnalisé.

MENU prend *tout* objet comme argument correct et le stocke dans *CST*. Cependant, le calculateur ne peut créer un menu personnalisé *que* si une liste est stockée dans *CST*, ou un nom contenant une liste. Tout autre objet cause une erreur "Bad Argument Type" lorsque le calculateur tente d'afficher le menu personnalisé.

Exemples : 5 MENU affiche la première page du menu MTH MATR NORM.

48.02 MENU affiche la deuxième page du menu UNITS MASS.

{ A 123 "ABC" } MENU affiche le menu personnalisé défini par l'argument liste.

'MONMENU' MENU affiche le menu personnalisé défini par l'argument nom.

Commandes associées : RCLMENU, TMENU

MERGE

Commande de fusion de mémoire de carte RAM : Fusionne la mémoire de la carte RAM placée dans le port 1 avec la mémoire-utilisateur principale. Une fois fusionnée, la mémoire n'est plus indépendante.

Niveau 1	→	Niveau 1
1	→	

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarque : MERGE est fournie à des fins de compatibilité avec la série HP 48S (voir MERGE1).

Commandes associées : FREE1, MERGE1

MERGE1

Commande de fusion de mémoire de carte RAM : Fusionne la mémoire de la carte RAM placée dans le port 1 avec la mémoire-utilisateur principale. Une fois fusionnée, la mémoire n'est plus indépendante.

Accès clavier : ◀ LIBRARY MERG

Indicateurs : Aucun

Remarques : Si la carte RAM contient des objets-sauvegarde ou des bibliothèques, ceux-ci sont transférés dans le port 0 avant la fusion. Ces objets ne peuvent être qu'en mémoire indépendante (port 1 à 33) ou dans le port 0 dans le cas d'une fusion.

Il est impossible de fusionner des cartes de plus de 128 Ko ni de les enficher dans le port 1.

Commandes associées : FREE, FREE1, MERGE

MIN

Fonction minimum : Renvoie le plus petit des deux arguments.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	min(*x*, *y*)
x	'*symb*'	→	'MIN(*x*, *symb*)'
'*symb*'	*x*	→	'MIN(*symb*, *x*)'
'$symb_1$'	'$symb_2$'	→	'MIN($symb_1$, $symb_2$)'
$x\_unité_1$	$y\_unité_2$	→	min($x\_unité_1$, $y\_unité_2$)

Accès clavier : MTH REAL MIN

Indicateurs : Résultats numériques (−3)

Exemples : `10 23 MIN` renvoie `10`.

`-10 -23 MIN` renvoie `-23`.

`1_m 9_cm MIN` renvoie `9_cm`.

Commande associée : MAX

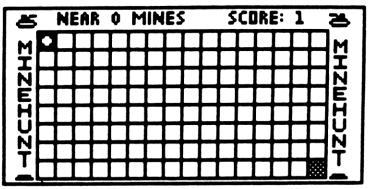
MINEHUNT

Commande du jeu Démineur : Lance le jeu du Démineur.

Accès clavier : ← EQ LIB UTILS MINE

Indicateurs : Aucun

Remarques : Dans ce jeu, vous partez de l'angle supérieur gauche d'une grille de 8 × 16 cases. Vous devez parvenir sans dommage jusqu'au coin inférieur droit, en évitant des mines invisibles placées sur le parcours. Le jeu vous indique le nombre de mines qui se trouvent sous les huit cases qui entourent votre position.



Utilisez les touches fléchées ou numériques pour vous déplacer d'une case à la fois. Les touches [7], [9], [1] et [3] vous permettent de vous déplacer en diagonale. Appuyez sur [CANCEL] pour quitter le jeu.

Pour interrompre une partie et la sauvegarder, appuyez sur [STO]. Vous créez ainsi une variable *MHpar* dans le répertoire en cours et le jeu se termine. Si *MHpar* existe lorsque vous commencez une nouvelle partie de Démineur, le jeu reprend là où vous y aviez mis fin et *MHpar* est supprimée.

Vous pouvez modifier le nombre de mines en créant une variable nommée *Nmines*, contenant le nombre souhaité. Celui-ci doit être un nombre réel (1 à 64). Si *Nmines* est négatif, les mines sont visibles durant le jeu.

MINIT

Commande d'initialisation du menu d'équations multiples : Crée la variable réservée *Mpar*.

Accès clavier : [↩] [EQ LIB] MES MINIT

Indicateurs : Aucun

Remarques : MINIT prend un ensemble d'équations stocké dans *EQ* et crée la variable réservée d'équations multiples *Mpar*. Pour de plus amples informations sur *Mpar*, voir l'Annexe D, "Variables réservées".

Commandes associées : MITM, MROOT, MSOLVR

MINR

Fonction nombre réel minimal : Renvoie la constante symbolique 'MINR' ou sa représentation numérique 1.00000000000E-499.

Niveau 1	→	Niveau 1
	→	'MINR'
	→	1.00000000000E-499

Accès clavier : MTH NXT CONS NXT MINR

Indicateurs : Constantes symboliques (−2), Résultats numériques (−3)

MINR renvoie sa représentation numérique si l'indicateur −2 ou −3 est armé ; sinon, elle renvoie sa représentation symbolique.

Remarque : MINR est la plus petite valeur numérique différente de zéro représentable par le HP 48.

Commandes associées : e, i, MAXR, π

MINΣ

Commande de calcul des valeurs minimales : Calcule les valeurs minimales de chacune des m colonnes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	x_{min}
	→	$[\ x_{min1}\ x_{min2}\ \cdots\ x_{minm}\ ]$

Accès clavier : ↶ STAT 1VAR MINΣ

Indicateurs : Aucun

Remarque : Les valeurs minimales sont renvoyées sous la forme d'un vecteur de m nombres réels, ou d'un seul nombre réel si $m = 1$.

Commandes associées : BINS, MAXΣ, MEAN, SDEV, TOT, VAR

MITM

Commande de modification du menu d'équations multiples : Modifie l'ordre et les titres dans le menu.

Niveau 2	Niveau 1	→	Niveau 1
"titre"	{ *liste* }	→	

Accès clavier : ↩ EQ LIB MES MITM

Indicateurs : Aucun

Remarques : *liste* contient les noms de variables dans l'ordre de votre choix. Utilisez "" pour insérer un libellé vierge. Vous devez inclure *toutes* les variables du menu d'origine (aucune autre).

Commande associée : MINIT

MOD

Fonction modulo : Renvoie un reste (modulo) défini comme suit :

$$x \bmod y = x - y \text{ floor } (x/y)$$

MOD

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	*x* mod *y*
x	'*symb*'	→	'MOD(*x*, *symb*)'
'*symb*'	*x*	→	'MOD(*symb*, *x*)'
'$symb_1$'	'$symb_2$'	→	'MOD($symb_1$, $symb_2$)'

Accès clavier : (MTH) REAL MOD

Indicateurs : Résultats numériques (−3)

Remarques : Mod (x, y) est périodique dans x avec la période y. Mod (x, y) se trouve dans l'intervalle [0, y) pour $y > 0$ et dans (y, 0] pour $y < 0$.

Commandes associées : FLOOR, /

MROOT

Commande de résolution d'équations multiples : Utilise le Solver d'équations multiples pour résoudre une ou plusieurs variables en utilisant l'ensemble d'équations stocké dans *Mpar*.

{ }

Niveau 1	→	Niveau 1
'*nom*'	→	*x*
"*ALL*"	→	

Accès clavier : (↶)(EQ LIB) MES MROO

Indicateurs : Aucun

Remarques : Résoud un ensemble d'équation pour une ou plusieurs variables en commençant par les valeurs-utilisateur, et laisse les valeurs calculées dans les variables. Aucun message d'état ne s'affiche. Etant donné un nom de variable, MROOT renvoie la valeur

calculée ; elle peut aussi prendre l'argument "ALL" (elle stocke une valeur calculée toutes les variables) et ne rien renvoyer dans la pile.

Commandes associées : MCALC, MUSER

MSGBOX

Commande de création d'une boîte de message : Crée une boîte de message utilisateur.

{ }

Niveau 1	→	Niveau 1
"*message*"	→	

Accès clavier : PRG NXT OUT MSGB

Indicateurs : Aucun

Remarques : MSGBOX affiche "*message*" sous la forme d'une boîte de message standard. Si le texte dépasse 75 caractères (y compris les espaces), les caractères en trop sont supprimés. Vous pouvez utiliser des espaces et des retours à la ligne (↵) pour contrôler le déroulement du texte et les coupures de ligne dans un message.

L'exécution du programme reprend lorsque la boîte de message est fermée, par activation de la touche OK ou CANCL.

Commandes associées : CHOOSE, INFORM, PROMPT

MSOLVR

Commande du Solver d'équations multiples : Utilise le menu de variables du Solver d'équations multiples pour l'ensemble d'équations défini par *Mpar*.

Accès clavier :

◀ EQ LIB MES MSOL

◀ EQ LIB EQLIB MSOL

Indicateurs : Aucun

Remarques : Le Solver d'équations multiples peut résoudre un ensemble de deux ou de plusieurs équations pour des variables inconnues, en calculant les racines de chaque équation, l'une après l'autre.

Le Solver utilise la liste des équations stockée dans *EQ*. Dans ce contexte, les "équations" incluent des programmes, des expressions et des noms de variables qui s'évaluent à une seule valeur. *EQ* doit contenir plusieurs équations, c'est-à-dire que l'application HP Solve doit inclure le libellé de menu NXEQ . Le Solver se sert de *EQ* pour créer une variable réservée *Mpar*, utilisée durant le processus de résolution. *Mpar* contient l'ensemble d'équations plus des informations complémentaires. Pour plus de détails sur *Mpar*, voir l'Annexe D, "Variables réservées".

Commandes associées : EQNLIB, SOLVEQN

MUSER

Commande de déclaration de variable-utilisateur : Déclare une variable comme étant définie par l'utilisateur pour le Solver d'équations multiples.

Niveau 1	→	Niveau 1
'nom'	→	
{ *liste* }	→	
"ALL"	→	

Accès clavier : ◀ EQ LIB MES MUSE

Indicateurs : Aucun

Remarque : MUSER déclare une seule variable, une liste de variables ou toutes les variables comme étant définies par l'utilisateur.

Commande associée : MCALC

NDIST

Commande de distribution normale : Renvoie la distribution de probabilité normale (courbe en forme de cloche) à x en se basant sur la moyenne m et la variance v de la distribution normale.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
m	*v*	*x*	→	*ndist(m,v,x)*

Accès clavier : MTH NXT PROB NXT NDIST

Indicateurs : Aucun

Remarque : NDIST est calculée avec la formule suivante :

$$ndist(m, v, x) = \frac{e^{-\frac{(x-m)^2}{2v}}}{\sqrt{2\pi v}}$$

Commande associée : UTPN

NEG

Fonction analytique d'opposé : Change le signe d'un objet (opposé).

{ }

Niveau 1	→	Niveau 1
z	→	$-z$
#n_1	→	#n_2
[*tableau*]	→	[$-$*tableau*]
'*symb*'	→	'$-$(*symb*)'
x_unité	→	$-$*x_unité*
$grob_1$	→	$grob_2$
$PICT_1$	→	$PICT_2$

Accès clavier :

+/-

MTH NXT CMPL NXT NEG

Indicateurs : Résultats numériques (−3), Taille de mot entier binaire (−5 à −10)

Remarques : L'opposé d'un tableau est un nouveau tableau contenant l'opposé de chacun des éléments d'origine. Dans le cas d'un nombre binaire, NEG prend son complément à deux (complément de chaque bit plus 1).

Dans le cas d'un objet graphique, l'objet résultant est "inversé" (les pixels inactifs sont activés, et inversement). Si l'argument est *PICT*, l'objet graphique stocké dans *PICT* est inversé.

Commandes associées : ABS, CONJ, NOT, SIGN

NEWOB

Commande de copie d'un objet : Crée une copie de l'objet spécifié.

Niveau 1	→	Niveau 1
obj	→	*obj*

Accès clavier : ◀ MEMORY NEWO

Indicateurs : Derniers arguments (−55)

Pour que NEWOB libère immédiatement la mémoire occupée par la l'original, il faut armer l'indicateur −55 afin que l'original ne soit pas sauvegardé comme dernier argument.

Remarques : NEWOB a deux fonctions principales :

- Supprimer une bibliothèque ou un objet-sauvegarde qui a été rappelé d'un port. NEWOB crée une copie distincte de l'objet en mémoire, permettant la suppression de l'original.
- Créer un nouvel objet à partir d'un autre objet composite plus volumineux (une liste par exemple) afin de récupérer la mémoire associée à ce dernier une fois qu'il n'est plus nécessaire.

Exemples : `:0:BKUP1 RCL NEWOB :0:BKUP1 PURGE` rappelle l'objet-sauvegarde *BKUP1* et le supprime.

`3 GET NEWOB` extrait le troisième élément d'une liste de la pile, libérant ainsi la mémoire occupée par la liste.

Commandes associées : MEM, PURGE

NEXT

Commande NEXT : Termine une structure en boucle finie.

Pour toute information sur la syntaxe, voir les rubriques FOR et START.

Accès clavier :

(PRG) BRCH START BRCH

(PRG) BRCH FOR NEXT

Remarque : Pour de plus amples informations, voir les commandes FOR et START.

Commandes associées : FOR, START, STEP

NEXT

Opération suivante : Renvoie, sans les exécuter, une ou deux étapes suivantes du programme.

Accès clavier : (PRG) (NXT) RUN NEXT

Indicateurs : Aucun

Commandes associées : SST, SST↓

NOT

Commande NOT : Renvoie le complément à un ou l'inverse logique de l'argument.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2
V/F	→	0/1
"*chaîne*$_1$"	→	"*chaîne*$_2$"
'*symb*'	→	'NOT *symb*'

Accès clavier :

PRG TEST NXT NOT

MTH BASE NXT LOGIC NOT

Indicateurs : Résultats numériques (−3), Taille de mot entier binaire (−5 à −10)

Remarques : Lorsque l'argument est un entier binaire ou une chaîne, NOT prend le complément de chaque bit de l'argument pour produire le résultat.

- Un entier binaire est traité comme une séquence de bits pour la totalité de la taille de mot en cours.
- Une chaîne est traitée comme une séquence de bits, à raison de 8 bits par caractère (utilisation de la version binaire du code de caractère).

Lorsque l'argument est un nombre réel ou symbolique, NOT effectue un test vrai/faux. Le résultat est 1 (vrai) si l'argument égale zéro, ou 0 (faux) si l'argument est différent de zéro. Ce test s'applique généralement à un résultat de test (V/F).

Si l'argument est un objet algébrique, le résultat est une expression algébrique de la forme 'NOT *symb*'. Exécutez →NUM (ou armez l'indicateur −3 avant l'exécution de NOT) afin de convertir le résultat algébrique en résultat numérique.

Commandes associées : AND, OR, XOR

NOVAL

Commande de valeur de garde/résultat pour INFORM : Place une valeur de garde pour les valeurs initiales et de réinitialisation dans les boîtes de dialogue utilisateur. Lorsqu'un champ est vide, NOVAL est renvoyée dans la pile.

Accès clavier : (PRG) (NXT) IN NOVA

Indicateurs : Aucun

Remarques : NOVAL sert à marquer un champ vide dans une boîte de dialogue utilisateur créée à l'aide de la commande INFORM. INFORM définit les champs de manière séquentielle. Si des valeurs par défaut sont utilisées pour ces champs, elles doivent être définies dans le même ordre que les champs. Pour ignorer certains champs (et donc ne pas leur attribuer de valeurs par défaut), utilisez la commande NOVAL.

Une fois INFORM terminée, NOVAL est renvoyée dans la pile (au niveau 2) lorsqu'un champ est vide et que la touche OK ou (ENTER) est activée.

Commande associée : INFORM

NSUB

Commande de position d'un bloc : Permet d'obtenir le numéro de bloc en cours (sous-liste) lors d'une itération d'un programme ou d'une commande réalisée au moyen de DOSUBS.

Accès clavier : (PRG) LIST PROC NSUB

Indicateurs : Aucun

Remarque : Le message d'erreur Undefined Local Name est renvoyé si cette commande est exécutée alors que DOSUBS n'est pas activée.

Commandes associées : DOSUBS, ENDSUB

NUM

Commande du numéro de caractère : Renvoie le code de caractère *n* du premier caractère de la chaîne.

{ }

Niveau 1	→	Niveau 1
" chaîne"	→	*n*

Accès clavier :

◀ CHARS NUM

PRG TYPE NXT NUM

Indicateurs : Aucun

Remarques : Les codes de caractères sont une extension de l'ISO 8859/1. Les codes 128 à 159 sont propres au HP 48.

Les tableaux suivants indiquent la relation entre les codes de caractères (résultats de NUM, arguments pour CHR) et les caractères (résultats de CHR, arguments pour NUM).

NUM

Codes de caractères (0 à 127)

NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR
0	■	32		64	@	96	'
1	■	33	!	65	A	97	a
2	■	34	"	66	B	98	b
3	■	35	#	67	C	99	c
4	■	36	$	68	D	100	d
5	■	37	%	69	E	101	e
6	■	38	&	70	F	102	f
7	■	39	'	71	G	103	g
8	■	40	(	72	H	104	h
9	■	41	)	73	I	105	i
10	■	42	*	74	J	106	j
11	■	43	+	75	K	107	k
12	■	44	,	76	L	108	l
13	■	45	-	77	M	109	m
14	■	46	.	78	N	110	n
15	■	47	/	79	O	111	o
16	■	48	0	80	P	112	p
17	■	49	1	81	Q	113	q
18	■	50	2	82	R	114	r
19	■	51	3	83	S	115	s
20	■	52	4	84	T	116	t
21	■	53	5	85	U	117	u
22	■	54	6	86	V	118	v
23	■	55	7	87	W	119	w
24	■	56	8	88	X	120	x
25	■	57	9	89	Y	121	y
26	■	58	:	90	Z	122	z
27	■	59	;	91	[	123	{
28	■	60	<	92	\	124	|
29	■	61	=	93	]	125	}
30	■	62	>	94	^	126	~
31	■	63	?	95	_	127	■

Codes de caractères (128 à 255)

NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR
128	∠	160		192	í	224	#
129	x̄	161	#	193	*	225	-
130	▽	162	¢	194	ó	226	+
131	∫	163	£	195	ƀ	227	*
132	∫	164	¤	196	#	228	#
133	Σ	165	¥	197	+	229	#
134	▶	166	¦	198	+	230	+
135	π	167	§	199	Ç	231	ç
136	∂	168	X	200	ú	232	#
137	≤	169	©	201	X	233	+
138	≥	170	ª	202	ñ	234	+
139	≠	171	«	203	Ñ	235	=
140	α	172	¬	204	X	236	+
141	→	173	-	205	*	237	#
142	←	174	®	206	X	238	#
143	↓	175	¯	207	X	239	X
144	↑	176	°	208	ð	240	ð
145	γ	177	±	209	+	241	+
146	δ	178	²	210	*	242	#
147	ε	179	³	211	*	243	#
148	η	180	¿	212	X	244	+
149	θ	181	µ	213	*	245	*
150	λ	182	¶	214	+	246	#
151	ρ	183	·	215	×	247	÷
152	σ	184	¿	216	+	248	+
153	τ	185	¹	217	ì	249	#
154	ω	186	º	218	≠	250	+
155	Δ	187	»	219	X	251	+
156	Π	188	¼	220	X	252	#
157	Ω	189	½	221	X	253	X
158	■	190	¾	222	Þ	254	Þ
159	∞	191	#	223	X	255	*

Commandes associées : CHR, POS, REPL, SIZE, SUB

→NUM

Commande d'évaluation à un nombre : Evalue un objet symbolique (autre qu'une liste) et renvoie un résultat numérique.

{ }

Niveau 1	→	Niveau 1
obj_{symb}	→	z

Accès clavier : ◀ →NUM

Indicateurs : Aucun

Remarques : →NUM évalue plusieurs fois un argument symbolique jusqu'à obtention d'un résultat numérique. Ce processus équivaut à évaluer l'argument symbolique en mode de Résultats numériques (indicateur −3 armé).

Commandes associées : EVAL, SYSEVAL

NUMX

Commande de définition du nombre de pas-X : Définit le nombre de pas-x pour chaque pas-y dans les tracés en perspective 3D.

{ }

Niveau 1	→	Niveau 1
n_x	→	

Accès clavier : ◀ PLOT NXT 3D VPAR NXT NUMX

Indicateurs : Aucun

Remarques : Le nombre de pas-x est le nombre de points tracés de la variable indépendante pour chaque point tracé de la variable dépendante. Il doit être égal à 2 ou plus. Cette valeur est stockée

dans la variable réservée VPAR. YSLICE est le seul type de tracé 3D à ne pas utiliser cette valeur.

Commande associée : NUMY

NUMY

Commande de définition du nombre de pas-Y : Définit le nombre de pas-y dans la vue volumique de tracés en perspective 3D.

{ }

Niveau 1	→	Niveau 1
n_y	→	

Accès clavier : ◀ PLOT NXT 3D VPAR NXT NUMY

Indicateurs : Aucun

Remarque : Le nombre de pas-y est le nombre de points de la variable dépendante tracés dans la vue volumique. Il doit être au moins égal à 2. Cette valeur est stockée dans la variable réservée VPAR.

Commande associée : NUMX

NΣ

Commande nombre de lignes : Renvoie le nombre de lignes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	n_{lignes}

NΣ

Accès clavier : ◀ STAT SUMS NΣ

Indicateurs : Aucun

Commandes associées : ΣX, ΣX*Y, ΣX^2, ΣY, ΣY^2

OBJ→

Commande de séparation d'un objet dans la pile : Décompose un objet en ses composants dans la pile. Pour certains types d'objets, le *nombre* de composants est renvoyé au niveau 1.

Niveau 1	→	Niveau n+1 ...	Niveau 2	Niveau 1
(*x*,*y*)	→		*x*	*y*
{ obj_1 ... obj_n }	→	obj_1	obj_n	*n*
[x_1 ... x_n]	→	x_1	x_n	{ *n* }
[[$x_{1\,1}$ ... $x_{m\,n}$]]	→	$x_{1\,1}$	$x_{m\,n}$	{ *m n* }
"*obj*"	→			*objet-évalué*
'*symb*'	→	arg_1 ... arg_n	*n*	'*fonction*'
x_unité	→		*x*	1_*unité*
:*id*:*obj*	→		*obj*	"*id*"

Accès clavier :

◀ CHARS NXT OBJ→

PRG TYPE OBJ→

Indicateurs : Aucun

Remarques : Si l'argument est un nombre complexe, une liste, un tableau ou une chaîne, OBJ→ fournit respectivement les mêmes fonctions que C→R, LIST→, ARRY→ et STR→. Pour des listes, OBJ→ renvoie aussi le nombre des éléments qu'elles contiennent. Si l'argument est un tableau, OBJ→ en renvoie les dimensions { *m n* }, *m* étant le nombre de lignes et *n* le nombre de colonnes.

En ce qui concerne les objets algébriques, OBJ→ renvoie les arguments de la fonction de plus haut niveau, autrement dit la moins imbriquée (arg_1 ... arg_n), le nombre de ses arguments n et son nom (*fonction*).

Si l'argument est une chaîne, la séquence d'objets qu'elle définit est exécutée.

Exemple : La séquence de commande `'∫ (0,1,SIN(X),X)' OBJ→` renvoie :

6:	0	premier argument
5:	1	deuxième argument
4:	'SIN(X)'	troisième argument
3:	'X'	quatrième argument
2:	4	nombre d'arguments pour ∫
1:	∫	nom de la fonction

Commandes associées : ARRY→, C→R, DTAG, EQ→, LIST→, R→C, STR→, →TAG

OCT

Commande de sélection de la base 8 : Sélectionne la base 8 pour les opérations sur des entiers binaires. (Par défaut, la base est décimale.)

Accès clavier : MTH BASE OCT

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : Les entiers binaires doivent être précédés du délimiteur #. Des entiers binaires saisis et renvoyés en base 8 sont automatiquement suivis de la lettre o. Si la base en vigueur n'est pas la base 8, vous devez faire suivre les nombres introduits en octal de la lettre o. Une fois saisis, ils sont affichés dans la base en cours.

La base en cours est sans effet sur la représentation interne des entiers binaires en tant que nombres binaires non signés.

OCT

Commandes associées : BIN, DEC, HEX, RCWS, STWS

OFF

Commande de mise hors tension : Met le calculateur hors tension.

Accès clavier : PRG NXT RUN NXT OFF

Indicateurs : Aucun

Remarques : Exécutée à partir d'un programme, cette commande permet la reprise de ce dernier lorsque vous rallumez le calculateur, faisant office de fonction de démarrage automatique programmable.

Commandes associées : CONT, HALT, KILL

OLDPRT

Commande de mappage des codes de caractères : Modifie la chaîne de mappage dans la variable réservée *PRTPAR* afin que le jeu de caractères étendu du HP 48 soit compatible avec celui de l'imprimante infrarouge HP 82240A.

Accès clavier : ← I/O PRINT PRTPA OLDPR

Indicateurs : Aucun

Remarques : Le jeu de caractères pour l'imprimante infrarouge HP 82240A ne correspond pas à celui du HP 48 :

- 24 caractères du jeu HP 48 ne sont pas disponibles avec l'imprimante infrarouge HP 82240A. (Dans le tableau fourni avec la description de la commande NUM, ces caractères correspondent aux codes 129, 130, 143-157, 159, 166, 169, 172, 174, 184 et 185.) A la place, l'imprimante HP 82240A imprime un ▒.
- De nombreux caractères du jeu étendu (codes 128 à 255) n'ont pas le même code. Par exemple, le caractère « a le code 171 dans le HP 48 et le code 146 dans l'imprimante infrarouge HP 82240A.

Pour utiliser la commande CHR afin d'imprimer les caractères du jeu étendu avec une imprimante infrarouge HP 82240A, exécutez d'abord OLDPRT. La chaîne de mappage modifiée par OLDPRT est le deuxième paramètre de *PRTPAR*. Cette chaîne (vide par défaut) modifie le code de caractère de chaque octet afin d'assurer la correspondance avec les codes de caractères de l'imprimante infrarouge HP 82240A.

Pour annuler le mappage des caractères, effacez le contenu de la variable *PRTPAR* ou tapez `'PRTPAR' 2 '' PUT`.

Si vous souhaitez imprimer une chaîne contenant des données graphiques, désactivez OLDPRT.

Commandes associées : CR, DELAY, PRLCD, PRST, PRSTC, PRVAR, PR1

OPENIO

Commande d'ouverture du port d'E-S : Ouvre le port série ou le port IR en utilisant les paramètres d'E-S de la variable réservée *IOPAR*.

Accès clavier : ← I/O NXT `SERIAL OPENI`

Indicateurs : Unité d'E-S (–33)

Remarques : Etant donné que toutes les commandes de protocole Kermit du HP 48 déclenchent automatiquement une commande OPENIO en premier, celle-ci n'est généralement pas requise. Vous pouvez cependant l'utiliser si une transmission d'E-S ne fonctionne pas. OPENIO est également nécessaire pour assurer la communication avec des unités qui interprètent un port fermé comme une interruption.

Par ailleurs, OPENIO est indispensable pour la réception automatique des données dans le tampon d'entrée, lorsque des commandes non Kermit sont exécutées. Si le port est fermé, les caractères entrants sont ignorés. S'il est ouvert, les caractères entrants sont automatiquement placés dans le tampon d'entrée. Ces caractères peuvent être détectés par BUFLEN et extraits à l'aide de SRECV.

OPENIO

Si le port est déjà ouvert, OPENIO n'a pas d'impact sur les données placées dans le tampon d'entrée. Cependant, si le port est fermé, l'exécution de OPENIO efface les données figurant dans le tampon.

Pour de plus amples informations, voir la description de *IOPAR* dans l'Annexe D, "Variables réservées".

Commandes associées : BUFLEN, CLOSEIO, SBRK, SRECV, STIME, XMIT

OR

Fonction OR : Renvoie le OU logique de deux arguments.

{ }

Niveau 2	Niveau 1	→	Niveau 1
#n_1	#n_2	→	#n_3
"*chaîne*$_1$"	"*chaîne*$_2$"	→	"*chaîne*$_3$"
V/F$_1$	*V/F*$_2$	→	0/1
V/F	'*symb*'	→	'*V/F* OR *symb*'
'*symb*'	*V/F*	→	'*symb* OR *V/F*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$ OR *symb*$_2$'

Accès clavier :

MTH BASE NXT LOGIC OR

PRG TEST NXT OR

Indicateurs : Résultats numériques (−3), Taille de mot entier binaire (−5 à −10)

Remarques : Lorsque les arguments sont des entiers binaires ou des chaînes, OR effectue une comparaison logique bit par bit (base 2).

- Un argument qui est un entier binaire est traité comme une séquence de bits sur la totalité de la taille du mot. Chaque bit résultant est déterminé par comparaison des bits correspondants

(bit_1 et bit_2) dans les deux arguments, comme le montre le tableau suivant.

bit_1	bit_2	bit_1 **OR** bit_2
0	0	0
0	1	1
1	0	1
1	1	1

- Si l'argument est une chaîne, il est traité comme une séquence de bits à raison de 8 bits par caractère (version binaire du code de caractère). Les deux chaînes formant les arguments doivent avoir la même longueur.

Lorsque les arguments sont des nombres réels ou des objets symboliques, OR effectue simplement un test vrai/faux. Le résultat est 1 (vrai) si l'un des arguments, ou les deux, sont différents de zéro ; il est égal à 0 (faux) si les deux arguments égalent zéro. Ce test est généralement effectué pour comparer deux résultats de test.

Si un argument ou les deux sont des objets algébriques, le résultat est un objet algébrique de la forme '$symb_1$ OR $symb_2$'. Exécutez →NUM (ou armez l'indicateur −3 avant d'exécuter OR) afin d'obtenir un résultat numérique.

Commandes associées : AND, NOT, XOR

ORDER

Commande de classement des variables : Tri les variables dans le répertoire en cours (affiché dans le menu VAR) selon l'ordre spécifié.

Niveau 1	→	Niveau 1
{ $global_1$ ... $global_n$ }	→	

Accès clavier : ◀ MEMORY DIR ORDER

ORDER

Indicateurs : Aucun

Remarques : Les noms qui apparaissent en premier dans la liste seront les premiers à figurer dans le menu VAR. Les variables non spécifiées dans la liste sont placées après les variables classées.

Si la liste inclut le nom d'un sous-répertoire de grande taille, il se peut que la mémoire soit insuffisante pour l'exécution de ORDER.

Commande associée : VARS

OVER

Commande de copie au niveau 1 : Renvoie la copie d'un objet de niveau 2 au niveau 1 de la pile.

Niveau 2	Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
obj_1	obj_2	→	obj_1	obj_2	obj_1

Accès clavier : ◀ STACK OVER

Indicateurs : Aucun

Commandes associées : PICK, ROLL, ROLLD, ROT, SWAP

PARAMETRIC

Commande de type de tracé Parametric : Sélectionne le type de tracé PARAMETRIC.

Accès clavier : ◀ PLOT PTYPE PARA

Indicateurs : Indicateur de traçage simultané (−28), Remplissage de courbe (−31)

Remarques : Lorsque le type de tracé est PARAMETRIC, la commande DRAW représente l'équation en cours par une fonction à valeurs complexes à une variable réelle. L'équation en cours est

spécifiée dans la variable réservée *EQ*. Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour les tracés PARAMETRIC, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant les coordonnées de l'angle inférieur gauche de *PICT* (angle inférieur gauche de la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant les coordonnées de l'angle supérieur droit de *PICT* (angle supérieur droit de la plage d'affichage). La valeur par défaut est (6.5,3.2).
- *indep* est une liste contenant le nom de la variable indépendante, et deux nombres spécifiant les valeurs minimale et maximale de cette variable (plage de traçage). La valeur par défaut est *X*. Si *X* n'est pas modifié ni inclus dans une liste avec une plage de traçage, les valeurs dans (x_{min}, y_{min}) et (x_{max}, y_{max}) sont utilisées comme plage de traçage, ce qui généralement produit des résultats non significatifs.
- *res* est un nombre réel désignant l'intervalle, en unités-utilisateur, séparant deux valeurs de la variable indépendante. La valeur par défaut est 0, ce qui équivaut à un intervalle de 1/130 de la différence entre les valeurs inférieure et supérieure de *indep* (domaine de traçage).
- *axes* est une liste contenant un ou plusieurs des éléments suivants, dans l'ordre mentionné : un nombre complexe spécifiant, en unités-utilisateur, les coordonnées de l'origine du tracé, une liste précisant l'intervalle de graduation, et deux libellés (chaînes) pour les deux axes. La valeur par défaut est (0,0).
- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de la commande PARAMETRIC place le nom PARAMETRIC dans *PPAR*.
- *depend* est le libellé de l'axe vertical. La valeur par défaut est *Y*.

Le contenu de *EQ* doit être une expression ou un programme, mais il ne peut pas être une équation. Il est évalué pour chaque valeur de la variable indépendante. Les résultats, qui doivent être des nombres

complexes, fournissent les coordonnées des points à tracer. Des lignes relient les points tracés, à moins que l'indicateur −31 ne soit armé.

Si l'indicateur −28 est armé, toutes les équations sont tracées simultanément.

Pour consulter un exemple de tracé PARAMETRIC, reportez-vous au chapitre 23 du *Manuel d'utilisation du HP 48*.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

PARITY

Commande de parité : Définit la valeur de parité dans la variable réservée *IOPAR*.

{ }

Niveau 1	→	Niveau 1
$n_{\text{parité}}$	→	

Accès clavier : ◀ I/O IOPAR PARIT

Indicateurs : Aucun

Remarques : Les valeurs autorisées sont indiquées ci-dessous. Lorsque la valeur est négative, le HP 48 ne contrôle pas la parité des octets reçus durant les transferts Kermit ou via SRECV. Toutefois, la parité continue d'être utilisée pendant la transmission des données.

Valeur *n*	Signification
0	pas de parité (valeur par défaut)
1	parité impaire
2	parité paire
3	marque
4	espace

Pour de plus amples informations, voir la variable réservée *IOPAR* (*paramètres d'E-S*)) dans l'Annexe D, "Variables réservées".

Commandes associées : BAUD, CKSM, TRANSIO

PARSURFACE

Commande du type de tracé PARSURFACE : Sélectionne le type de tracé PARSURFACE.

Accès clavier : ← PLOT NXT 3D PTYPE PARSU

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est PARSURFACE, la commande DRAW trace la représentation graphique d'une fonction de 3 vecteurs à deux variables. PARSURFACE extrait ses valeurs des variables réservées *EQ*, *VPAR* et *PPAR*.

VPAR comporte les éléments suivants :

{ x_{left} x_{right} y_{near} y_{far} z_{low} z_{high} x_{min} x_{max} y_{min} y_{max} x_{eye} y_{eye} z_{eye} x_{step} y_{step} }

Pour le type de tracé PARSURFACE, les éléments de *VPAR* sont utilisés comme suit :

- x_{left} et x_{right} sont des nombres réels spécifiant la largeur de la vue volumique.
- y_{near} et y_{far} sont des nombres réels spécifiant la profondeur de la vue volumique.

- z_{low} et z_{high} sont des nombres réels spécifiant la hauteur de la vue volumique.
- x_{min} et x_{max} sont des nombres réels qui désignent la largeur de la zone d'entrée. La valeur par défaut est ⟨−1, 1⟩.
- y_{min} et y_{max} sont des nombres réels qui désignent la profondeur de la zone d'entrée. La valeur par défaut est ⟨−1, 1⟩.
- x_{eye}, y_{eye} et z_{eye} sont des nombres réels qui spécifient le point de vue.
- x_{step} et y_{step} sont des nombres réels qui spécifient le nombre de coordonnées x par rapport au nombre de coordonnées y tracées.

Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, dont la forme est la suivante :

{ ⟨x_{min}, y_{min}⟩ ⟨x_{max}, y_{max}⟩ *indep res axes ptype depend* }

Pour le type de tracé PARSURFACE, les éléments de *PPAR* sont utilisés comme suit :

- ⟨x_{min}, y_{min}⟩ n'est pas utilisé.
- ⟨x_{max}, y_{max}⟩ n'est pas utilisé.
- *indep* désigne la variable indépendante. La valeur par défaut de *indep* est *X*.
- *res* n'est pas utilisé.
- *axes* n'est pas utilisé.
- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de PARSURFACE place le nom PARSURFACE dans *ptype*.
- *depend* est le nom de la variable dépendante. La valeur par défaut est *Y*.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

PATH

Commande d'indication du chemin d'accès : Renvoie une liste indiquant le chemin d'accès au répertoire en cours.

Niveau 1 →	Niveau 1
→	{ HOME *nom-répertoire*$_1$ ... *nom-répertoire*$_n$ }

Accès clavier : ◀ MEMORY DIR PATH

Indicateurs : Aucun

Remarques : Le premier répertoire est toujours *HOME* et le dernier le répertoire en cours.

Si un programme doit accéder à un répertoire spécifique, cela peut être fait en évaluant une liste de répertoires, telle que celle créée préalablement par PATH.

Commandes associées : CRDIR, HOME, PGDIR, UPDIR

PCOEF

Commande de calcul des coefficients d'un polynôme unitaire : Renvoie les coefficients d'un polynôme unitaire (polynôme avec un coefficient dominant égal à 1) ayant des racines données.

{ }

Niveau 1	→	Niveau 1
[*tableau*]$_{racines}$	→	[*tableau*]$_{coefficients}$

Accès clavier : ◀ SOLVE POLY PCOEF

Indicateurs : Aucun

PCOEF

Remarques : L'argument doit être un tableau réel ou complexe de longueur n contenant les racines du polynôme. Le résultat est un vecteur réel ou complexe de longueur $n+1$ contenant les coefficients, classés dans l'ordre décroissant des monômes, avec un coefficient dominant égal à 1.

Exemple : Calculez le polynôme ayant les racines 2, −3, 4, −5 :

`[ 2 -3 4 -5 ]` PCOEF renvoie `[ 1 2 -25 -26 120 ]`, représentant le polynôme $x^4 + 2x^3 - 25x^2 - 26x + 120$.

Commandes associées : PEVAL, PROOT

PCONTOUR

Commande du type de tracé PCONTOUR : Sélectionne le type de tracé PCONTOUR.

Accès clavier : ◀ PLOT NXT 3D PTYPE PCON

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est PCONTOUR, la commande DRAW trace un contour d'une fonction scalaire à deux variables. PCONTOUR extrait ses valeurs des variables réservées *EQ*, *VPAR* et *PPAR*.

VPAR est composée des éléments suivants :
$\{ x_{left}\ x_{right}\ y_{near}\ y_{far}\ z_{low}\ z_{high}\ x_{min}\ x_{max}\ y_{min}\ y_{max}\ x_{eye}\ y_{eye}\ z_{eye}\ x_{step}\ y_{step} \}$

Pour le type de tracé PCONTOUR, les éléments de *VPAR* sont utilisés comme suit :

- x_{left} et x_{right} sont des nombres réels spécifiant la largeur de la vue volumique.
- y_{near} et y_{far} sont des nombres réels spécifiant la profondeur de la vue volumique.
- z_{low} et z_{high} sont des nombres réels spécifiant la hauteur de la vue volumique.
- x_{min} et x_{max} ne sont pas utilisés.

- y_{min} et y_{max} ne sont pas utilisés.
- x_{eye}, y_{eye} et z_{eye} sont des nombres réels qui désignent le point de vue.
- x_{step} et y_{step} sont des nombres réels qui définissent le nombre de coordonnées x par rapport au nombre coordonnées y tracées.

Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé PCONTOUR, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) n'est pas utilisé.
- (x_{max}, y_{max}) n'est pas utilisé.
- ***indep*** est le nom de la variable indépendante. La valeur par défaut de ***indep*** est X.
- *res* n'est pas utilisé.
- *axes* n'est pas utilisé.
- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de PCONTOUR place le nom PCONTOUR dans *ptype*.
- *depend* est le nom de la variable dépendante. La valeur par défaut est Y.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

PCOV

Commande de covariance de population : Renvoie la covariance de population des colonnes de données dépendantes et indépendantes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{\text{covariancep}}$

Accès clavier : ◀ STAT FIT NXT PCOV

Indicateurs : Aucun

Remarques : Les colonnes sont spécifiées par les deux premiers éléments de la variable réservée ΣPAR, définis respectivement par XCOL et YCOL . Si ΣPAR n'existe pas, PCOV la crée et définit les éléments avec leurs valeurs par défaut (1 et 2).

La covariance de population est calculée à l'aide de la formule suivante :

$$\frac{1}{n}\sum_{k=1}^{n}(x_{kn_1} - \overline{x_{n_1}})(x_{kn_2} - \overline{x_{n_2}})$$

où x_{kn_1} est la $k$ième valeur de coordonnées dans la colonne n_1, x_{kn_2} est la $k$ième valeur de coordonnées dans la colonne n_2, $\overline{x_{n_1}}$ est la moyenne des données dans la colonne n_1, $\overline{x_{n_2}}$ est la moyenne des données dans la colonne n_2, et n est le nombre de points de données.

Commandes associées : COLΣ, CORR, COV, PREDX, PREDY, XCOL, YCOL

PDIM

Commande de redimensionnement de PICT : Remplace *PICT* par un *PICT* vierge ayant les dimensions spécifiées.

{ }

Niveau 2	Niveau 1	→	Niveau 1
(x_{min}, y_{min})	(x_{max}, y_{max})	→	
$\#n_{largeur}$	$\#m_{hauteur}$	→	

Accès clavier : PRG PICT PDIM

Indicateurs : Aucun

Remarques : Si les arguments sont des nombres complexes, PDIM modifie la taille de *PICT* et les arguments deviennent les nouvelles valeurs de (x_{min}, y_{min}) et (x_{max}, y_{max}) dans la variable réservée *PPAR*. Ainsi, l'échelle du tracé suivant reste inchangée. Si les arguments sont des entiers binaires, *PPAR* ne change pas, de sorte que l'échelle du tracé suivant *est* modifiée.

PICT ne peut pas être inférieur à 131 pixels de large × 64 pixels de haut, ni dépasser 2048 pixels de large (la hauteur étant illimitée).

Commandes associées : PMAX, PMIN

PERM

Fonction de calcul de permutations : Renvoie le nombre de permutations possibles de n éléments pris par groupes de m à la fois.

PERM

{ }

Niveau 2	Niveau 1	→	Niveau 1
n	*m*	→	$P_{n,m}$
'*symb*$_n$'	*m*	→	'PERM(*symb*$_{n,m}$)'
n	'*symb*$_m$'	→	'PERM(*n*, *symb*$_m$)'
'*symb*$_n$'	'*symb*$_m$'	→	'PERM(*symb*$_n$, *symb*$_m$)'

Accès clavier : MTH NXT PROB PERM

Indicateurs : Résultats numériques (−3)

Remarques : La formule servant à calculer $P_{n,m}$ est la suivante :

$$P_{n,m} = \frac{n!}{(n-m)!}$$

Les arguments n et m doivent être inférieurs à 10^{12}.

Commandes associées : COMB, !

PEVAL

Commande d'évaluation de polynôme : Evalue un polynôme de degré n en x.

{ }

Niveau 2	Niveau 1	→	Niveau 1
[*tableau*]$_{coefficients}$	*x*	→	*p*(*x*)

Accès clavier : ← SOLVE POLY PEVAL

Indicateurs : Aucun

Remarques : Les arguments doivent être un tableau de longueur $n+1$ contenant les coefficients du polynôme, classés par ordre décroissant des monônes, ainsi que la valeur x à laquelle le polynôme doit être évalué.

Exemple : Evaluez le polynôme $x^4 + 2x^3 - 25x^2 - 26x + 120$ à $x = 8$:

`[ 1 2 -25 -26 120 ] 8` renvoie `3432`.

Commandes associées : PCOEF, PROOT

PGDIR

Commande de suppression d'un répertoire : Supprime le répertoire nommé (qu'il soit vide ou non).

{ }

Niveau 1	→	Niveau 1
'global'	→	

Accès clavier : ◀ MEMORY `DIR PGDIR`

Indicateurs : Aucun

Commandes associées : CLVAR, CRDIR, HOME, PATH, PURGE, UPDIR

PICK

Commande de copie d'objet : Copie le contenu d'un niveau spécifié dans le niveau 1.

Niv n+1..	Niv 2	Niv 1	→	Niv n+1..	Niv 2	Niv 1
obj_n ..	obj_1	*n*	→	obj_n ..	obj_1	obj_n

Accès clavier : ◀ STACK `PICK`

PICK

Indicateurs : Aucun

Commandes associées : DUP, DUPN, DUP2, OVER, ROLL, ROLLD, ROT, SWAP

PICT

Commande PICT : Place le nom PICT dans la pile.

Niveau 1	→	Niveau 1
	→	*PICT*

Accès clavier : (PRG) PICT PICT

Indicateurs : Aucun

Remarques : *PICT* désigne un emplacement de stockage dans la mémoire du calculateur contenant l'objet graphique en cours. La commande PICT permet d'accéder au contenu de cet emplacement comme s'il s'agissait d'une variable. Cependant, *PICT n'est pas* une variable au sens où on l'entend dans le HP 48 : son nom ne peut pas être mis entre apostrophes, et seuls des objets graphiques peuvent y être stockés.

Si un objet graphique inférieur à 131 × 64 pixels (largeur × hauteur) est placé dans *PICT*, il est agrandi aux dimensions 131 × 64. Un objet graphique d'une hauteur illimitée et au maximum de 2048 pixels de large peut être stocké dans *PICT*.

Exemples : PICT RCL renvoie l'objet graphique en cours dans la pile.

GRAPHIC 131 × 64 PICT STO stocke un objet graphique dans *PICT*, et en fait l'objet en cours.

Commandes associées : GOR, GXOR, NEG, PICTURE, PVIEW, RCL, REPL, SIZE, STO, SUB

PICTURE

Commande d'environnement Picture : Sélectionne l'environnement Picture (sélectionne l'affichage graphique et active le curseur graphique ainsi que le menu Picture).

Accès clavier : ◀ PICTURE

Indicateurs : Aucun

Remarques : Lorsque PICTURE est exécutée depuis un programme, elle suspend l'exécution de ce dernier jusqu'à ce que vous appuyiez sur CANCEL.

Exemple : Le programme

```
« "Appuyez sur CANCEL pour revenir■à la pile" 1 DISP
3 WAIT PICTURE »
```

affiche un message pendant 3 secondes, puis sélectionne l'environnement Picture. (Le caractère ■ dans le programme indique un saut de ligne.)

Commandes associées : PICTURE, PVIEW, TEXT

PINIT

Commande d'initialisation de port : Initialise tous les ports actifs. N'affecte pas les données déjà stockées dans un port.

Accès clavier : ◀ LIBRARY NXT PINIT

Indicateurs : Aucun

Commandes associées : Aucune

Remarques : PINIT est particulièrement utile lorsque vous vous servez d'une carte enfichable pouvant contenir plusieurs ports. Elle stocke puis supprime un objet dans chaque port (partition de 128 Ko), accessible au moment de l'exécution de la commande. Ceci a pour effet d'initialiser chaque port sans affecter les données déjà stockées.

PIXOFF

Commande de désactivation des pixels : Désactive, dans *PICT*, le pixel spécifié par les coordonnées indiquées.

Niveau 1	→	Niveau 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	

Accès clavier : PRG PICT NXT PIXOF

Indicateurs : Aucun

Commandes associées : PIXON, PIX?

PIXON

Commande d'activation des pixels : Active, dans *PICT*, le pixel spécifié par les coordonnées indiquées.

Niveau 1	→	Niveau 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	

Accès clavier : PRG PICT NXT PIXON

Indicateurs : Aucun

Commandes associées : PIXOFF, PIX?

PIX?

Commande de test sur l'état des pixels : Teste si le pixel spécifié dans *PICT* est allumé ou non ; renvoie 1 (vrai) s'il l'est et 0 (faux) dans le cas contraire.

Niveau 1	→	Niveau 1
(*x*, *y*)	→	0/1
{ #*n* #*m* }	→	0/1

Accès clavier : PRG PICT NXT PIX?

Indicateurs : Aucun

Commandes associées : PIXON, PIXOFF

PKT

Commande d'envoi de paquets : Utilisée pour envoyer des "paquets" de commandes à un serveur Kermit (et en recevoir les données demandées).

{}

Niveau 2	Niveau 1	→	Niveau 1
"*données*"	"*type*"	→	"*réponse*"

Accès clavier : ← I/O SRVR PKT

Indicateurs : Unité d'E-S (−33), Messages d'E-S (−39)

L'indicateur Format de données des E-S (−35) peut être déterminant si le serveur renvoie plusieurs paquets.

Remarques : Pour envoyer des objets HP 48, utilisez SEND.

PKT

PKT permet l'envoi de commandes supplémentaires à un serveur Kermit. Pour de plus amples informations, reportez-vous à l'ouvrage *Using MS-DOS Kermit*, par Christine M. Gianone, Digital Press, 1990; ou à l'ouvrage *KERMIT, A File Transfer Protocol*, par Frank da Cruz, Digital Press, 1987, notamment au chapitre 11, "The Client/Server Model".

Les données du paquet, leur type et la réponse à la transmission du paquet ont la forme d'une chaîne. La commande PKT procède d'abord à un échange de paquet I (*initialisation*) avec le serveur Kermit, puis elle envoie un paquet constitué en fonction de ses arguments "données" et "type". La réponse à PKT est soit une chaîne contenant un message d'accusé de réception (éventuellement vide), soit un paquet "erreur" (voir KERRM).

Pour l'argument *type*, seule la première lettre est significative.

Exemples : Une commande PKT destinée à envoyer une demande de répertoire générique a la forme `"R" "G" PKT`.

Pour envoyer un paquet de *commandes hôte*, utilisez une commande du système d'exploitation du serveur pour la chaîne *données* et `"C"` pour la chaîne *type*. Par exemple, `"'ABC' PURGE" "C" PKT` sur un HP 48 local demande à un serveur HP 48 de supprimer la variable *ABC/*

Commandes associées : CLOSEIO, KERRM, SERVER

PMAX

Commande de coordonnées maximales de PICT : Spécifie `(x, y)` comme coordonnées de l'angle supérieur droit de la zone d'affichage.

{}

Niveau 1	→	Niveau 1
(*x*, *y*)	→	

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarque : Le nombre complexe ‹x, y› est stocké comme deuxième élément de la variable réservée *PPAR*.

Commandes associées : PDIM, PMIN, XRNG, YRNG

PMIN

Commande de coordonnées minimales de PICT : Spécifie ‹x, y› comme coordonnées de l'angle inférieur gauche de la zone d'affichage.

{ }

Niveau 1	→	Niveau 1
(x,y)	→	

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarque : Le nombre complexe ‹x, y› est stocké comme premier élément de la variable réservée *PPAR*.

Commandes associées : PDIM, PMAX, XRNG, YRNG

POLAR

Commande du type de tracé Polar : Sélectionne le type de tracé POLAR.

Accès clavier : ↶ PLOT PTYPE POLAR

Indicateurs : Indicateur de traçage simultané d'équations multiples (−28), Remplissage de courbe (−31)

Remarques : Lorsque le type de tracé est POLAR, la commande DRAW trace l'équation en cours selon le système de coordonnées polaires, où la variable indépendante est l'angle polaire et la variable

dépendante le rayon. L'équation en cours est spécifiée dans la variable réservée *EQ*.

Les paramètres de traçage sont indiqués dans la variable réservée *PPAR*, dont la forme est la suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé POLAR, les éléments de *PPAR* sont utilisés comme suit :

- (x_{min}, y_{min}) est un nombre complexe représentant l'angle inférieur gauche de *PICT* (angle inférieur gauche de la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe représentant l'angle supérieur droit de *PICT* (angle supérieur droit de la plage d'affichage. La valeur par défaut est (6.5,3.2).
- *indep* est le nom de la variable indépendante, ou une liste contenant son nom et deux nombres désignant les valeurs inférieure et supérieure de la variable indépendante (domaine de traçage). La valeur par défaut de *indep* est *X*.
- *res* est un nombre réel spécifiant l'intervalle, en unités-utilisateur, séparant deux valeurs de la variable indépendante. La valeur par défaut est 0, ce qui équivaut à un intervalle de 2 degrés, 2 grades ou $\pi/90$ radians.
- *axes* est une liste contenant un ou plusieurs des éléments suivants, dans l'ordre cité : un nombre complexe spécifiant les coordonnées, en unités- utilisateur, de l'origine du tracé, une liste indiquant l'intervalle de graduation, et les libellés (chaînes) des deux axes. La valeur par défaut est (0,0).
- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de POLAR place le nom POLAR dans *ptype*.
- *depend* est le libellé de l'axe vertical. La valeur par défaut est *Y*.

L'équation en cours est tracée comme une fonction de la variable spécifiée dans *indep*. Les valeurs minimale et maximale de la variable indépendante (domaine de traçage) peuvent figurer dans *indep* ; sinon, la valeur minimale par défaut est 0 et la valeur maximale correspond à un cercle complet selon le mode d'angle en cours (360 degrés, 400 grades ou 2π radians). Des lignes relient les points tracés, à moins que l'indicateur −31 ne soit armé.

Si l'indicateur −28 est armé, toutes les équations sont tracées simultanément.

Si *EQ* contient une expression ou un programme, elle ou il est évalué en mode Résultats numériques pour chaque valeur de la variable indépendante de façon à fournir les valeurs de la variable dépendante. Si *EQ* contient une équation, le traçage dépend de sa forme.

Forme de l'équation en cours	Type de traçage
'*expr=expr*'	Chaque expression est tracée séparément. L'intersection des deux graphiques montre l'endroit où les expressions sont égales.
'*nom=expr*'	Seule l'expression est tracée.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

POS

Commande de localisation : Renvoie la position d'une sous-chaîne dans une chaîne ou d'un objet dans une liste.

Niveau 2	Niveau 1	→	Niveau 1
"*chaîne*"	"*sous-chaîne*"	→	*n*
{ *liste* }	*obj*	→	*n*

Accès clavier :

◀ CHARS POS

PRG LIST ELEM POS

POS

Indicateurs : Aucun

Remarques : En cas de non correspondance pour *obj* ou *sous-chaîne*, POS renvoie zéro.

Commandes associées : CHR, NUM, REPL, SIZE, SUB

PREDV

Commande relative à une valeur y prévue : Renvoie la valeur prévue de la variable dépendante $y_{\text{dépendante}}$, en fonction de la valeur de la variable indépendante $x_{\text{indépendante}}$, du modèle statistique sélectionné et des coefficients de régression en cours dans la variable réservée ΣPAR.

{ }

Niveau 1	→	Niveau 1
$x_{\text{indépendante}}$	→	$y_{\text{dépendante}}$

Accès clavier : Aucun. A saisir.

Remarques : Fournie à des fins de compatibilité avec le HP 28, PREDV est l'équivalent de PREDY (voir cette dernière).

PREDX

Commande relative à une valeur x prévue : Renvoie la valeur prévue de la variable indépendante $x_{\text{indépendante}}$, en fonction de la valeur de la variable dépendante $x_{\text{dépendante}}$, du modèle statistique sélectionné et des coefficients de régression en cours dans la variable réservée ΣPAR.

{ }

Niveau 1	→	Niveau 1
$y_{dépendante}$	→	$x_{indépendante}$

Accès clavier : ↩ STAT FIT PREDX

Indicateurs : Aucun

Remarques : La valeur est prévue au moyen des coefficients de régression les plus récents calculés avec LR, et stockés dans la variable réservée ΣPAR. Pour le modèle statistique linéaire, l'équation employée est la suivante :

$$y_{dependante} = (mx_{independante}) + b$$

où m est la pente (troisième élément dans ΣPAR) et b l'intersection (quatrième élément dans ΣPAR).

Pour les autres modèles statistiques, les équations utilisées par PREDX sont indiquées à la rubrique de la commande LR.

Si PREDX est exécutée sans avoir au préalable généré des coefficients de régression dans ΣPAR, une valeur par défaut de zéro est utilisée pour les deux coefficients et une erreur est renvoyée.

Exemple : Etant donné cinq colonnes de données dans ΣDAT, la séquence de commandes :

```
2 XCOL 5 YCOL LOGFIT LR 23 PREDX
```

définit la colonne 2 comme celle de la variable indépendante, la colonne 5 comme celle de la variable dépendante, et sélectionne le modèle statistique logarithmique. Elle exécute ensuite LR, générant ainsi les coefficients de régression de pente et d'intersection, et les stocke dans ΣPAR. Puis, étant donné la valeur dépendante 23, elle renvoie la valeur indépendante prévue en fonction des coefficients de régression et du modèle statistique.

Commandes associées : COLΣ, CORR, COV, EXPFIT, ΣLINE, LINFIT, LOGFIT, LR, PREDY, PWRFIT, XCOL, YCOL

PREDY

Commande relative à une valeur y prévue : Renvoie la valeur prévue de la variable dépendante $y_{\text{dépendante}}$, en fonction de la valeur de la variable indépendante $x_{\text{indépendante}}$, du modèle statistique sélectionné et des coefficients de régression en cours dans la variable réservée ΣPAR.

{ }

Niveau 1	→	Niveau 1
$x_{\text{indépendante}}$	→	$y_{\text{dépendante}}$

Accès clavier : ◀ STAT FIT PREDY

Indicateurs : Aucun

Remarques : La valeur est prévue au moyen des coefficients de régression les plus récents calculés avec LR, et stockés dans la variable réservée ΣPAR. Pour le modèle statistique linéaire, l'équation employée est la suivante :

$$y_{dependante} = (mx_{independante}) + b$$

où m est la pente (troisième élément dans ΣPAR) et b l'intersection (quatrième élément dans ΣPAR).

Pour les autres modèles statistiques, les équations utilisées par PREDY sont indiquées à la rubrique de la commande LR.

Si PREDY est exécutée sans avoir au préalable généré des coefficients de régression dans ΣPAR, une valeur par défaut de zéro est utilisée par les deux coefficients de régression ; dans ce cas, PREDY renvoie 0 pour les modèles statistiques LINFIT et LOGFIT, et une erreur pour les modèles EXPFIT et PWRFIT.

Exemple : Etant donné quatre colonnes de données dans ΣDAT, la séquence de commandes :

```
2 XCOL 4 YCOL PWRFIT LR 11 PREDY
```

définit la colonne 2 comme celle de la variable indépendante, la colonne 4 comme celle de la variable dépendante, et sélectionne le modèle statistique de puissance. Elle exécute ensuite LR, générant

ainsi les coefficients de régression de pente et d'intersection, et les stocke dans ΣPAR. Puis, étant donné la valeur indépendante 11, elle renvoie une valeur dépendante prévue en fonction des coefficients de régression et du modèle statistique.

Commandes associées : COLΣ, CORR, COV, EXPFIT, ΣLINE, LINFIT, LOGFIT, LR, PREDX, PWRFIT, XCOL, YCOL

PRLCD

Commande d'impression de l'affichage : Imprime l'affichage en cours, pixel par pixel (à l'exception des témoins).

Accès clavier : ← I/O PRINT PRLCD

Indicateurs : Unité d'impression (−34), Unité d'E-S (−33), Saut de ligne (−38)

Si l'indicateur −34 est armé (sortie d'imprimante acheminée vers le port série), l'indicateur −33 doit être désarmé.

L'indicateur −38 doit être désarmé.

Remarques : La largeur de l'image imprimée de l'affichage est moindre avec PRLCD qu'avec une commande d'impression telle que PR1. La différence vient de l'espacement des caractères. Sur l'affichage, un seul espace sépare les caractères, et PRLCD l'imprime tel quel. En revanche, des commandes d'impression comme PR1 impriment deux espaces entre des caractères adjacents.

Exemple : La séquence ERASE DRAW PRLCD efface *PICT*, trace l'équation en cours puis imprime le contenu de l'affichage graphique.

Commandes associées : CR, DELAY, OLDPRT, PRST, PRSTC, PRVAR, PR1

PROMPT

Commande d'invite : Affiche le contenu de "*message*" dans la zone d'état, et arrête l'exécution du programme.

{ }

Niveau 1	→	Niveau 1
"*message*"	→	

Accès clavier : PRG NXT IN NXT PROM

Indicateurs : Aucun

Remarque : PROMPT équivaut à 1 DISP 1 FREEZE HALT.

Commandes associées : CONT, DISP, FREEZE, HALT, INFORM, INPUT, MSGBOX

PROOT

Commande de calcul des racines d'un polynôme : Renvoie toutes les racines d'un polynôme de degré n ayant des coefficients réels ou complexes.

{ }

Niveau 1	→	Niveau 1
[*tableau*]$_{coefficients}$	→	[*tableau*]$_{racines}$

Accès clavier : ← SOLVE POLY PROOT

Indicateurs : Exception de résultat infini (−22)

Remarques : Pour un polynôme du $n^{ième}$ ordre, l'argument doit être un tableau réel ou complexe de longueur $n+1$ contenant les coefficients, classés par ordre décroissant des monônes. Le résultat

est un vecteur réel ou complexe de longueur n contenant les racines calculées.

PROOT interprète les coefficients dominants égaux à zéro de manière restrictive. Lorsqu'un coefficient dominant est voisin de zéro, une racine du polynôme est proche de l'infini : ainsi, si l'indicateur −22 est désarmé (état par défaut), PROOT renvoie une erreur de résultat infini lorsque le coefficient dominant est zéro. Si l'indicateur −22 est armé, PROOT renvoie une racine de (MAXREAL,0) pour chaque zéro dominant dans un tableau contenant des coefficients réels, et une racine de (MAXREAL,MAXREAL) pour chaque zéro dominant dans un tableau contenant des coefficients complexes.

Exemple : Calculez les racines du polynôme $x^4 + 2x^3 - 25x^2 - 26x + 120$:

`[ 1 2 -25 -26 120 ] PROOT` renvoie `[ 2 -3 4 -5 ]`.

Commandes associées : PCOEF, PEVAL

PRST

Commande d'impression de la pile : Imprime tous les objets de la pile, en commençant par celui du plus haut niveau.

Accès clavier : ⬅ I/O PRINT PRST

Indicateurs : Impression en double interligne (−37), Unité d'impression (−34), Unité d'E-S (−33), Saut de ligne (−38)

Si l'indicateur −34 est armé (sortie d'imprimante acheminée vers le port série), l'indicateur −33 doit être désarmé.

Lorsque l'indicateur −38 est armé, *aucun* saut de ligne n'est ajouté à la fin des lignes imprimées. Généralement, l'indicateur −38 doit être désarmé pour l'exécution de PRST. PRST ne modifie pas la pile.

Remarques : Les objets sont imprimés au format multiligne. Reportez-vous à la commande PR1 pour une description de ce format.

Commandes associées : CR, DELAY, OLDPRT, PRLCD, PRSTC, PRVAR, PR1

PRSTC

Commande d'impression de la pile (format compacté) : Imprime dans un format compacté tous les objets de la pile, en commençant par celui du plus haut niveau.

Accès clavier : ← I/O PRINT PRSTC

Indicateurs : Impression en double interligne (−37), Unité d'impression (−34), Unité d'E-S (−33), Saut de ligne (−38)

Si l'indicateur −34 est armé (sortie d'imprimante acheminée vers le port série), l'indicateur −33 doit être désarmé.

Lorsque l'indicateur −38 est armé, *aucun* saut de ligne n'est ajouté à la fin des lignes imprimées. Généralement, l'indicateur −38 doit être désarmé pour l'exécution de PRSTC.

Remarques : Le format compacté d'impression est identique au format compacté pour l'affichage. Les objets multilignes sont tronqués et n'apparaissent que sur une ligne. PRSTC ne modifie pas la pile.

Commandes associées : CR, DELAY, OLDPRT, PRLCD, PRST, PRVAR, PR1

PRVAR

Commande d'impression de variables : Recherche les variables spécifiées dans le répertoire ou le port en cours et en imprime le nom et le contenu.

Niveau 1	→	Niveau 1
'nom'	→	
{ nom_1 nom_2 ... }	→	
$:n_{port}$ *:'global'*	→	

Accès clavier : ← I/O PRINT PRVAR

Indicateurs : Impression en double interligne (−37), Unité d'impession (−34), Unité d'E-S (−33), Saut de ligne (−38)

Si l'indicateur −34 est armé (sortie d'imprimante acheminée vers le port série), l'indicateur −33 doit être désarmé.

Lorsque l'indicateur −38 est armé, *aucun* saut de ligne n'est ajouté à la fin des lignes imprimées. Généralement, l'indicateur −38 doit être désarmé pour l'exécution de PRVAR.

Remarque : Les objets sont imprimés en format multiligne (voir la commande PR1 pour une description de ce format).

Commandes associées : CR, DELAY, OLDPRT, PR1, PRLCD, PRST, PRSTC

PR1

Commande d'impression de l'objet du niveau 1 : Imprime un objet en format multiligne.

Accès clavier : ↩ I/O PR1

Indicateurs : Impression en double interligne (−37), Unité d'impression (−34), Unité d'E-S (−33), Saut de ligne (−38)

Si l'indicateur −34 est armé (sortie d'imprimante acheminée vers le port série), l'indicateur −33 doit être désarmé.

Remarques : Tous les objets, à l'exception des chaînes, sont imprimés avec leurs délimiteurs d'identification. Les chaînes sont imprimées sans les délimiteurs de début et de fin ". PR1 ne modifie pas la pile.

Le format d'impression multiligne est le même que le format d'affichage multiligne, à quelques exceptions près :

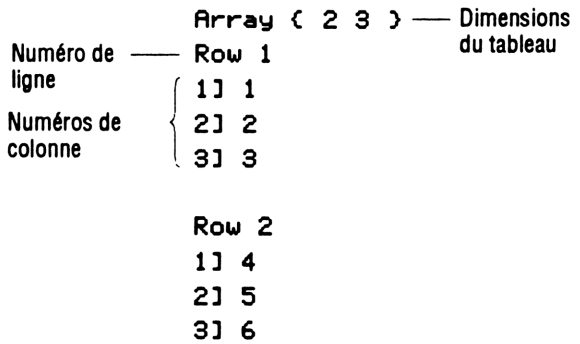
- Les chaînes et les noms de plus de 24 caractères se poursuivent sur la ligne suivante.
- Les parties réelles et imaginaires des nombres complexes sont imprimées sur des lignes séparées si elles ne tiennent pas sur une même ligne.

PR1

- Les tableaux sont imprimés avec un en-tête numéroté pour chaque ligne et un numéro de colonne devant chaque élément. Par exemple, le tableau 2 × 3

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

s'imprime comme suit :



Commandes associées : CR, DELAY, OLDPRT, PRLCD, PRST, PRSTC, PRVAR

PSDEV

Commande d'écart standard de population : Calcule l'écart standard de population de chacune des m colonnes de valeurs de coordonnées dans la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{écartps}$
	→	$[\ x_{écartps1}\ x_{écartps2}\ \cdots\ x_{écartpsm}\ ]$

Accès clavier : ◀ STAT 1VAR NXT PSDEV

Indicateurs : Aucun

Remarques : PSDEV renvoie un vecteur de m nombres réels, ou un nombre réel simple si $m = 1$. L'écart standard de population est calculé selon la formule :

$$\sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - \overline{x})^2}$$

où x_k est $k$ème valeur de coordonnées d'une colonne, $\overline{x}$ est la moyenne des données de cette colonne, et n le nombre de points de données.

Commandes associées : MEAN, PCOV, PVAR, SDEV, TOT, VAR

PURGE

Commande de suppression : Supprime les variables nommées ou les sous-répertoires vides du répertoire en cours.

Niveau 1	→	Niveau 1
'global'	→	
{ $global_1$ ... $global_n$ }	→	
PICT	→	
$:n_{port}$ $:nom\_svgrde_{JJ=JJJJ??:??n}$ $port_{??:??n}$ $bibl._{JJ}$	→	

Accès clavier : ◀ PURGE

Indicateurs : Aucun

Remarques : Exécutée dans un programme, PURGE ne sauvegarde pas ses arguments en vue de leur récupération au moyen de LASTARG.

Pour effacer le contenu d'un répertoire nommé avant de le supprimer, utilisez PGDIR.

PURGE

Pour préparer facilement une liste des variables à supprimer, utilisez VARS.

La suppression de *PICT* remplace l'objet graphique en cours par un autre de dimensions 0 × 0.

Si une liste d'objets (avec des noms globaux, des objets-sauvegarde, des bibliothèques ou *PICT*) à supprimer contient un objet incorrect, les objets précédant celui-ci sont supprimés, puis un message `Bad Argument Type` s'affiche.

Pour supprimer un objet-sauvegarde ou bibliothèque, associez à son numéro ou à son nom le numéro de port adéquat (: n_{port}), qui doit être compris entre 0 et 33. (Une bibliothèque ne peut être effacée que de la mémoire RAM.) Pour un objet-sauvegarde, le numéro de port est remplaçable par le joker & ; dans ce cas le HP 48 cherche l'objet-sauvegarde nommé dans les ports 33 à 0, puis dans la mémoire principale.

Il est possible de supprimer des objets-bibliothèques en RAM, mais non ceux qui sont en ROM (cartes d'application et cartes RAM protégées en écriture). Pour supprimer un objet-bibliothèque du répertoire *HOME*, vous devez au préalable le dissocier.

Vous ne pouvez pas supprimer un objet-bibliothèque ni un objet-sauvegarde tant qu'il est référencé en interne par des pointeurs de pile (par exemple un objet se trouvant dans la pile, dans une variable locale, dans la pile LAST, ou dans une pile interne de renvoi). Vous obtiendriez le message d'erreur `Object in Use`. Pour éviter ce problème, utilisez NEWOB avant la suppression (voir cette dernière).

Commandes associées : CLEAR, CLUSR, CLVAR, NEWOB, PGDIR

PUT

Commande de remplacement d'éléments : Dans le tableau ou la liste au niveau 3, PUT remplace l'objet dont la position est spécifiée au niveau 2 par z_{put} ou obj_{put} ; si le tableau ou la liste n'est pas nommé, cette commande renvoie le nouveau tableau ou la nouvelle liste.

	Niveau 3	Niveau 2	→	Niveau 1	Niveau 1
[[*matrice*]]$_1$	$n_{position}$	z_{put}	→	[[*matrice*]]$_2$	
[[*matrice*]]$_1$	{ n_{ligne} m_{col} }	z_{put}	→	[[*matrice*]]$_2$	
'$nom_{matrice}$'	$n_{position}$	z_{put}	→		
'$nom_{matrice}$'	{ n_{ligne} m_{col} }	z_{put}	→		
[*vecteur*]$_1$	$n_{position}$	z_{put}	→	[*vecteur*]$_2$	
[*vecteur*]$_1$	{ $n_{position}$ }	z_{put}	→	[*vecteur*]$_2$	
'$nom_{vecteur}$'	$n_{position}$	z_{put}	→		
'$nom_{vecteur}$'	{ $n_{position}$ }	z_{put}	→		
{ *liste* }$_1$	$n_{position}$	obj_{put}	→	{ *list* }$_2$	
{ *liste* }$_1$	{ $n_{position}$ }	obj_{put}	→	{ *list* }$_2$	
'nom_{liste}'	$n_{position}$	obj_{put}	→		
'nom_{liste}'	{ $n_{position}$ }	obj_{put}	→		

Accès clavier : (PRG) LIST ELEM PUT

Indicateurs : Aucun

Remarques : Pour des matrices, $n_{position}$ désigne séquentiellement les numéros de lignes.

Si l'argument du niveau 3 est un nom, PUT modifie le tableau ou la liste nommé et ne renvoie rien dans la pile.

Exemples : La séquence de commandes :
`[[ 2 3 4 ][ 4 1 2 ]] < 1 3 > 96` PUT renvoie
`[[ 2 3 96 ][ 4 1 2 ]]`.

La séquence de commandes `[[ 2 3 4 ][ 4 1 2 ]] 5 96` PUT renvoie
`[[ 2 3 4 ][ 4 96 2 ]]`.

PUT

La séquence de commandes { A B C D E } { 3 } 'Z' PUT renvoie { A B Z D E }.

Commandes associées : GET, GETI, PUTI

PUTI

Commande de remplacement d'un élément et d'incrémentation de l'index : Dans un tableau ou une liste de niveau 3, remplace l'objet dont la position est spécifiée au niveau 2 par z_{put} ou obj_{put}, en renvoyant le nouveau tableau ou la nouvelle liste *et* la nouvelle position dans ceux-ci.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
[[*matrice*]]$_1$	n_{pos1}	z_{put}	→	[[*matrix*]]$_2$	n_{pos2}
[[*matrice*]]$_1$	{ n_r m_c }$_1$	z_{put}	→	[[*matrice*]]$_2$	{ n_r m_c }$_2$
'$nom_{matrice}$'	n_{pos1}	z_{put}	→	'$nom_{matrice}$'	n_{pos2}
'$nom_{matrice}$'	{ n_r m_c }$_1$	z_{put}	→	'$nom_{matrice}$'	{ n_r m_c }$_2$
[*vecteur*]$_1$	n_{pos1}	z_{put}	→	[*vecteur*]$_2$	n_{pos2}
[*vecteur*]$_1$	{ n_{pos1} }	z_{put}	→	[*vecteur*]$_2$	{ n_{pos2} }
'$nom_{vecteur}$'	n_{pos}	z_{put}	→	'$nom_{vecteur}$'	n_{pos2}
'$nom_{vecteur}$'	{ n_{pos1} }	z_{put}	→	'$nom_{vecteur}$'	{ n_{pos2} }
{ *list* }$_1$	n_{pos1}	obj_{put}	→	{ *liste* }$_2$	n_{pos2}
{ *list* }$_1$	{ n_{pos1} }	obj_{put}	→	{ *liste* }$_2$	{ n_{pos2} }
'nom_{liste}'	n_{pos1}	obj_{put}	→	'nom_{liste}'	n_{pos2}
'nom_{liste}'	{ n_{pos1} }	obj_{put}	→	'nom_{liste}'	{ n_{pos2} }

Accès clavier : PRG LIST ELEM PUTI

Indicateurs : Indicateur de bouclage d'index (−64)

L'indicateur de bouclage d'index est désarmé à chaque exécution de PUTI *jusqu'à* ce que l'index revienne à la première position dans le tableau ou la liste ; à ce moment-là, l'indicateur est armé. L'exécution suivante de PUTI le désarme de nouveau.

Remarques : Pour des matrices, l'incrémentation s'effectue dans l'ordre des *lignes*.

Contrairement à PUT, PUTI renvoie un tableau ou une liste nommé (au niveau 2). Ceci permet une nouvelle exécution de PUTI à la position suivante de cet objet nommé.

Exemple : Le programme suivant utilise PUTI et l'indicateur −64 pour remplacer par X les éléments A, B et C de la liste.

```
« { A B C } DO 'X' PUTI UNTIL -64 FS? END »
```

Commandes associées : GET, GETI, PUT

PVAR

Commande de calcul de variance de population : Calcule la variance de population des valeurs de coordonnées dans chacune des m colonnes de la matrice statistique en cours (ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{\text{variancep}}$
	→	$[\, x_{\text{variancep1}} \cdots x_{\text{variancepm}} \,]$

Accès clavier : ◀ STAT 1VAR NXT PVAR

Indicateurs : Aucun

Remarques : La variance de population (égale au carré de l'écart standard de population) est renvoyée sous la forme d'un vecteur de m nombres réels, ou d'un seul nombre réel si $m = 1$. Les variances de population sont calculées selon la formule :

$$\frac{1}{n}\sum_{k=1}^{n}(x_k - \overline{x})^2$$

où x_k est la $k$ième valeur de coordonnées dans une colonne, $\overline{x}$ est la moyenne des données de cette colonne, et n le nombre de points de données.

PVAR

Commandes associées : MEAN, PCOV, PSDEV, SDEV, VAR

PVARS

Commande relative aux variables associées à un port :
Renvoie une liste des objets-sauvegarde ($:n_{port}:nom$) et des objets-bibliothèque ($:n_{port}:n_{bibliothèque}$) pour le port spécifié. Renvoie aussi la taille de la mémoire disponible (en cas de RAM) ou le type de mémoire.

{ }

Niveau 1	→	Niveau 2	Niveau 1
n_{port}	→	{ $:n_{port}$ $:nom_{sauvegarde}$ ... }	*mémoire*
n_{port}	→	{ $:n_{port}$ $:n_{bibliothèque}$ ... }	*mémoire*

Accès clavier : ◀ LIBRARY PVARS

Indicateurs : Aucun

Remarques : Le numéro de port, n_{port}, doit être compris entre 0 et 33.

- Si $n_{port} = 0$, *mémoire* désigne le nombre d'octets de RAM principale disponible.
- Si le port contient une RAM indépendante, *mémoire* indique le nombre d'octets de RAM disponible dans ce port.
- Si le port contient une RAM fusionnée, *mémoire* indique "SYSRAM".
- Si le port contient une ROM, *mémoire* indique "ROM".
- Si le port est vide, le message Port Not Available s'affiche.

Commandes associées : PVARS, VARS

PVIEW

Commande de visualisation de PICT : Affiche *PICT* avec les coordonnées spécifiées dans l'angle supérieur gauche de l'affichage graphique.

Niveau 1	→	Niveau 1
(*x*, *y*)	→	
{ #*n* #*m* }	→	
{ }	→	

Accès clavier :

PRG NXT OUT PVIEW

PRG PICT NXT PVIEW

Indicateurs : Aucun

Remarques : *PICT* doit occuper tout l'affichage au moment de l'exécution de PVIEW. Ainsi, si une position autre que l'angle supérieur gauche de *PICT* est spécifiée, *PICT* doit être suffisamment grand pour occuper un rectangle qui s'étend sur 131 pixels à droite et 64 pixels vers le bas.

Si PVIEW est exécutée depuis un programme avec des coordonnées comme argument (au lieu d'une liste vide), l'affichage graphique reste visible jusqu'à ce que le clavier soit à nouveau prêt pour la saisie (par exemple jusqu'à la fin de l'exécution du programme). Cependant, la commande FREEZE permet de geler l'affichage jusqu'à activation d'une touche.

Si PVIEW est exécutée avec une liste *vide* comme argument, *PICT* est centré dans l'affichage, et le mode de défilement est activé. Dans ce cas, l'affichage graphique reste à l'écran jusqu'à activation de la touche CANCEL.

PVIEW *n*'active *pas* le curseur graphique ni le menu Picture. Pour ce faire, exécuter PICTURE.

Exemple : Le programme

PVIEW

```
« ( # 0d # 0d ) PVIEW  7 FREEZE »
```

affiche *PICT* dans l'affichage graphique avec les coordonnées (# 0d # 0d) dans l'angle supérieur gauche de l'affichage, puis le gèle jusqu'à ce qu'une touche soit activée.

Commandes associées : FREEZE, PICTURE, TEXT

PWRFIT

Commande d'ajustement de puissance : Stocke PWRFIT dans le cinquième paramètre de la variable réservée ΣPAR, indiquant que les exécutions suivantes de LR doivent utiliser le modèle d'ajustement de puissance pour la courbe.

Accès clavier : ◀ STAT ΣPAR MODL PWRFI

Indicateurs : Aucun

Remarque : LINFIT est le modèle par défaut dans ΣPAR. Pour une description de ΣPAR, voir l'Annexe D, "Variables réservées".

Commandes associées : BESTFIT, EXPFIT, LINFIT, LOGFIT, LR

PX→C

Commande de conversion de pixels en unités-utilisateur : Convertit en unités-utilisateur des coordonnées exprimées en pixels.

Niveau 1	→	Niveau 1
{ #*n* #*m* }	→	(*x*, *y*)

Accès clavier : PRG PICT NXT PX→C

Indicateurs : Aucun

Remarques : Les coordonnées en unités-utilisateur sont fournies par les paramètres (x_{min}, y_{min}) et (x_{max}, y_{max}) de la variable réservée *PPAR*. Les coordonnées correspondent au centre géométrique du pixel.

Commande associée : C→PX

→Q

Commande de conversion en fraction rationnelle : Renvoie une fraction rationnelle de l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	'*a/b*'
(*x*,*y*)	→	'*a/b*+*c/d**i'
'$symb_1$'	→	'$symb_2$'

Accès clavier : ◀ SYMBOLIC NXT →Q

Indicateurs : Format d'affichage numérique (−45 à −50)

Remarques : La fraction rationnelle obtenue est la meilleure approximation possible, étant donné qu'il peut en exister plusieurs cohérentes par rapport à l'argument. →Q calcule un quotient d'entiers compatible avec l'argument dans la limite du nombre de décimales spécifié par le format d'affichage.

→Q agit aussi sur des nombres qui font partie d'expressions algébriques ou d'équations.

Exemple : 'Y+2.5' →Q renvoie 'Y+5/2'.

Commandes associées : →Qπ, /

→Qπ

Commande de conversion en fraction rationnelle par π : Renvoie une fraction rationnelle de l'argument *ou* une fraction rationnelle de l'argument avec mise en facteur de π, de façon à fournir la fraction comportant le plus petit dénominateur.

{ }

Niveau 1	→	Niveau 1
x	→	'*a*/*b**π'
x	→	'*a*/*b*'
'*symb*$_1$'	→	'*symb*$_2$'
(*x*,*y*)	→	'*a*/*b**π+*c*/*d**π*i'
(*x*,*y*)	→	'*a*/*b*+*c*/*d**i'

Accès clavier : ◀ SYMBOLIC NXT →Qπ

Indicateurs : Format d'affichage numérique (−45 à −50)

Remarques : →Qπ calcule et compare deux quotients (fractions rationnelles) : le quotient de l'argument et celui de l'argument divisé par π. Elle renvoie ensuite la fraction comportant le plus petit dénominateur. Si l'argument a été divisé par π, π est un facteur dans le résultat.

La fraction rationnelle obtenue est la meilleure approximation possible puisqu'il peut en exister plusieurs cohérentes par rapport à l'argument. →Qπ calcule un quotient d'entiers compatible avec l'argument dans la limite du nombre de décimales spécifié par le format d'affichage.

→Qπ agit aussi sur des nombres appartenant à des expressions algébriques ou à des équations.

En cas d'argument complexe, la partie réelle ou imaginaire (ou les deux) peut avoir π comme facteur.

Exemple : En mode fixe avec quatre décimales, `6.2832 →Qπ` renvoie `'2*π'`. Cependant, en mode standard, `6.2832 →Qπ` renvoie `3927/625`.

Commandes associées : →Q, /, π

QR

Commande de factorisation QR d'une matrice : Renvoie la factorisation QR d'une matrice $n \times m$.

{ }

Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
[[*matrice*]]$_A$	→	[[*matrice*]]$_Q$	[[*matric*]]$_R$	[[*matrice*]]$_P$

Accès clavier : (MTH) MATR FACTR QR

Indicateurs : Aucun

Remarques : QR factorise une matrice $m \times n$ A en trois matrices :

- Q est une matrice orthogonale $m \times m$.
- R est une matrice trapézoïdale supérieure $m \times n$.
- P est une matrice de permutation $n \times n$.

Où $A \times P = Q \times R$.

Commandes associées : LQ, LSQ

QUAD

Commande de résolution d'équation quadratique : Résoud un objet algébrique '*symb*$_1$' pour la variable *global*, et renvoie une expression '*symb*$_2$' représentant la solution.

{ }

Niveau 2	Niveau 1	→	Niveau 1
'*symb*$_1$'	'*global*'	→	'*symb*$_2$'

Accès clavier : ◀ (SYMBOLIC) QUAD

Indicateurs : Solution principale (−1)

QUAD

Remarques : QUAD calcule l'approximation polynômiale du second degré de Taylor de '*symb*$_1$' afin de la convertir en forme quadratique. La solution '*symb*$_2$' est exacte si '*symb*$_1$' est du second degré ou moins dans *global*.

Etant donné que QUAD évalue '*symb*$_1$', les variables dans '*symb*$_1$' autres que *global* ne doivent pas exister dans le répertoire en cours si elles doivent rester dans la solution comme variables formelles.

Généralement, QUAD ne fonctionne pas si *global* a besoin d'unités pour satisfaire l'équation.

Exemple : `'A*X^2+B*X+C=0' 'X' QUAD` renvoie
`'X=(-B+±1*√(B^2-4*(A*2/2)*C))/(2*(A*2/2))'`

réductible à la solution quadratique suivante :

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Commandes associées : COLCT, EXPAN, ISOL, SHOW

QUOTE

Fonction de mise entre apostrophes d'un argument : Renvoie l'argument sans évaluation :

{ }

Niveau 1	→	Niveau 1
'*symb*'	→	'*symb*'
obj	→	*obj*

Accès clavier : ⬅ SYMBOLIC NXT NXT QUOT

Indicateurs : Aucun

Remarques : Dans le cas d'une expression algébrique, les arguments d'une fonction sont évalués avant la fonction elle-même. Par exemple, lorsque `'SIN(X)'` est évaluée, le nom X est évalué en premier, puis le résultat est laissé dans la pile comme argument pour SIN.

Ce processus pose un problème pour les fonctions qui nécessitent des arguments symboliques. Ainsi, la fonction $\int$ requiert comme un de ses arguments un nom spécifiant la variable d'intégration. Si l'évaluation d'une expression intégrale entraînait l'évaluation du nom, le résultat de l'évaluation serait laissé dans la pile pour $\int$, à la place du nom. Pour éviter ce problème, le HP 48 met automatiquement (et de façon transparente) ce type d'arguments entre apostrophes. Lorsque l'argument entre apostrophes est évalué, il est renvoyé sans apostrophes.

Si une fonction-utilisateur prend des arguments symboliques, ceux-ci doivent être mis entre apostrophes à l'aide de QUOTE, comme le montre l'exemple suivant.

Exemple : La fonction-utilisateur suivante *ArcLen* calcule la longueur de l'arc d'une fonction :

```
«
 → start end expr var
 «
  start end
  expr var ∂ SQ 1 + ∫
  var ∫
 »
»
```

ENTER ' ArcLen STO

Pour utiliser cette fonction dans une expression algébrique, les arguments symboliques doivent être mis entre apostrophes :

```
'ArcLen(0,π,QUOTE(SIN(X),QUOTE(X))'
```

Commandes associées : APPLY, | (Remplacement)

RAD

Commande de définition du mode radians : Sélectionne le mode d'angle radians.

Accès clavier :

◀ RAD

◀ MODES ANGL RAD

Indicateurs : Aucun

Remarques : RAD arme l'indicateur −17, désarme l'indicateur −18 et affiche le témoin RAD.

En mode d'angle radians, les arguments du type nombres réels représentant des angles sont interprétés en radians, et les résultats en nombres réels représentant des angles sont exprimés en radians.

Commandes associées : DEG, GRAD

RAND

Commande de génération de nombres aléatoires : Renvoie un nombre pseudo-aléatoire obtenu à partir d'un générateur et met à jour ce dernier.

Niveau 1	→	Niveau 1
	→	$x_{\text{aléatoire}}$

Accès clavier : MTH NXT PROB RAND

Indicateurs : Aucun

Remarques : Le HP 48 utilise une méthode de congruence linéaire et un générateur pour obtenir un nombre aléatoire $x_{\text{aléatoire}}$ compris entre $0 \leq x < 1$. Chaque exécution de RAND renvoie une valeur calculée qui sert de générateur pour le nombre suivant. (Utilisez RDZ pour modifier le générateur.)

Commandes associées : COMB, PERM, RDZ, !

RANK

Commande de calcul du rang d'une matrice : Renvoie le rang d'une matrice rectangulaire.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]	→	n_{rang}

Accès clavier : (MTH) MATR NORM (NXT) RANK

Indicateurs : Eléments très petits (−54)

Remarques : Le rang est déterminé par calcul des valeurs singulières de la matrice et par comptage du nombre des valeurs non négligeables. Si toutes les valeurs singulières sont égales à zéro, RANK renvoie zéro. Sinon, la commande analyse l'indicateur −54 comme suit :

- Si l'indicateur −54 est désarmé (par défaut), RANK compte toutes les valeurs singulières calculées inférieures ou égales à 1.E−14 fois la taille de la valeur singulière calculée la plus grande.
- Si l'indicateur −54 est armé, RANK compte toutes les valeurs singulières calculées différentes de zéro.

Commandes associées : LQ, LSQ, QR

RANM

Commande de création d'une matrice aléatoire : Renvoie une matrice aux dimensions spécifiées contenant des entiers aléatoires dans la plage −9 à 9.

Niveau 1	→	Niveau 1
{ *m n* }	→	[[*matrice aléatoire*]]$_{m \times n}$
[[*matrice*]]$_{m \times n}$	→	[[*matrice aléatoire*]]$_{m \times n}$

Accès clavier : (MTH) MATR MAKE RANM

Indicateurs : Aucun

Remarques : La probabilité qu'il y ait un chiffre particulier différent de zéro est de 0,05. La probabilité d'avoir 0 est de 0,1.

Commandes associées : RAND, RDZ

RATIO

Fonction préfixe de division : Forme préfixe de / (division) généré par l'application EquationWriter.

{ }

Niveau 2	Niveau 1	→	Niveau 1
z_1	z_2	→	$z_1 \ / \ z_2$
[*tableau*]	[[*matrice*]]	→	[[*tableau* x $matrice^{-1}$]]
[*tableau*]	*z*	→	[*tableau/z*]
z	'*symb*'	→	'*z/symb*'
'*symb*'	*z*	→	'*symb/z*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ / $symb_2$'
$\#n_1$	n_2	→	$\#n_3$
n_1	$\#n_2$	→	$\#n_3$
$\#n_1$	$\#n_2$	→	$\#n_3$
$x\_unité_1$	$y\_unité_2$	→	$(x/y)\_unité_1 / unité_2$
x	*y_unité*	→	*(x/y)_1/unité*
x_unité	*y*	→	*(x/y)_unité*
'*symb*'	*x_unité*	→	'*symb/x_unité*'
x_unité	'*symb*'	→	'*x_unité/symb*'

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : RATIO est identique à / (division), à ceci près que, en syntaxe algébrique, RATIO est une fonction *préfixe*, alors que / est une fonction *infixe*. Par exemple, `'RATIO(A,2)'` est l'équivalent de `'A/2'`.

RATIO est générée en interne par l'application EquationWriter lorsque ▲ commence un numérateur. Elle n'ajoute aucune fonctionnalité à / et n'apparaît en clair que dans la chaîne laissée par EquationWriter dans la pile lorsque vous appuyez sur → `" "` ou que le calculateur n'a plus assez de mémoire.

Commande associée : /

RCEQ

Commande de rappel d'une équation de EQ : Renvoie le contenu non évalué de la variable réservée *EQ* à partir du répertoire en cours.

Niveau 1	→	Niveau 1
	→	obj_{EQ}

Accès clavier :

◀ PLOT EQ

◀ PLOT NXT 3D EQ

◀ PLOT ▶ EQ (mode de saisie de programme)

◀ PLOT NXT 3D ▶ EQ (mode de saisie de programme)

Indicateurs : Aucun

Remarques : Pour rappeler le contenu de *EQ* à partir d'un répertoire parent (lorsque *EQ* n'existe pas dans le répertoire actif), vous devez évaluer le nom *EQ*.

Commande associée : STEQ

RCI

Commande de multiplication des lignes par une constante : Multiplie la ligne *n* d'une matrice (ou l'élément *n* d'un vecteur) par une constante $x_{facteur}$, et renvoie la matrice modifiée.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	$x_{facteur}$	$n_{numéroligne}$	→	[[*matrice*]]$_2$
[*vecteur*]$_1$	$x_{facteur}$	$n_{numéroélément}$	→	[*vecteur*]$_2$

Accès clavier : (MTH) MATR ROW RCI

Indicateurs : Aucun

Remarque : RCI arrondit le numéro de ligne à l'entier le plus proche, et traite les arguments vecteurs comme des vecteurs-colonnes.

Commande associée : RCIJ

RCIJ

Commande de multiplication-addition de lignes : Multiplie la ligne i d'une matrice par une constante $x_{facteur}$, ajoute ce produit à la ligne j de la matrice, et renvoie la matrice modifiée. Ou bien, multiplie l'élément i d'un vecteur par une constante $x_{facteur}$, ajoute ce produit à l'élément j du vecteur et renvoie le vecteur modifié.

{ }

Niveau 4	Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	$x_{facteur}$	n_{lignei}	n_{lignej}	→	[[*matrice*]]$_2$
[*vecteur*]$_1$	$x_{facteur}$	$n_{élémenti}$	$n_{élémentj}$	→	[*vecteur*]

Accès clavier : (MTH) MATR ROW RCIJ

Indicateurs : Aucun

Remarque : RCIJ arrondit les numéros de lignes à l'entier le plus proche, et traite les arguments vecteurs comme des vecteurs-colonnes.

Commande associée : RCI

RCL

Commande de rappel : Renvoie le contenu non évalué d'une variable spécifiée ou d'un objet enfiché.

Niveau 1	→	Niveau 1
'nom'	→	*obj*
PICT	→	*grob*
$:n_{port}$ $:n_{bibliothèque}$	→	*obj*
$:n_{port}$ $:nom_{sauvegarde}$	→	*obj*

Accès clavier : ▶ RCL

Indicateurs : Aucun

Remarques : RCL explore le chemin d'accès en cours, en commençant par le répertoire actif. Pour explorer un autre chemin, spécifiez { *chemin nom* }, où *chemin* est le nouveau chemin d'accès à la variable *nom*. Le sous-répertoire désigné par *chemin* ne devient pas le sous-répertoire actif (contrairement à ce qui se produit avec EVAL).

Pour rappeler un objet-bibliothèque ou sauvegarde, associez au numéro du premier ou au nom du deuxième le numéro de port approprié (n_{port}), qui doit se situer entre 0 et 33. (Le rappel d'une bibliothèque ne s'effectue que depuis une RAM.) Le rappel d'un objet-sauvegarde place une copie de son *contenu* dans la pile, non l'objet lui-même.

Pour rechercher un objet-sauvegarde, remplacez le numéro de port par le joker & : le HP 48 le cherche (dans l'ordre) dans les ports 33 à 0, puis dans la mémoire principale.

Commande associée : STO

RCLALARM

Commande de rappel d'alarme : Rappelle une alarme spécifiée.

{ }

Niveau 1	→	Niveau 1
n_{index}	→	{ *date heure* obj_{action} $x_{répéter}$ }

Accès clavier : ◀ TIME ALRM RCLAL

Indicateurs : Aucun

Remarques : obj_{action} correspond à l'exécution de l'alarme. Si aucune exécution n'est spécifiée, obj_{action} est par défaut une chaîne vide.

$x_{répéter}$ est l'intervalle de répétition, exprimée en tops d'horloge, 1 top égalant 1/8192 seconde. Si aucun intervalle n'est précisé, la valeur par défaut est 0.

Commandes associées : DELALARM, FINDALARM, STOALARM

RCLF

Commande de rappel d'indicateurs : Renvoie une liste contenant deux entiers binaires de 64 bits représentant respectivement les états des 64 indicateurs système et utilisateur.

Niveau 1	→	Niveau 1
	→	{ #$n_{système}$ #$n_{utilisateur}$ }

Accès clavier : ◀ MODES FLAG NXT RCLF

Indicateurs : Taille de mot entier binaire (−5 à −10)

RCLF

La taille de mot en cours doit être de 64 bits (taille par défaut) pour que soient rappelés les états des 64 indicateurs utilisateur et des 64 indicateurs système. Si par exemple la taille de mot est 32, RCLF renvoie deux entiers binaires de 32 bits.

Remarques : Un bit égal à 1 indique que l'indicateur correspondant est armé ; un bit égal à 0 indique qu'il est désarmé. Le bit le plus à droite (poids faible) de $\#n_{système}$ et $\#n_{utilisateur}$ indique l'état de l'indicateur système -1 et de l'indicateur utilisateur $+1$.

Utilisée avec STOF, RCLF permet à un programme qui modifie l'état d'un ou de plusieurs indicateurs durant son exécution de le récupérer ensuite.

Commande associée : STOF

RCLKEYS

Commande de rappel des définitions de touches : Renvoie les définitions de touches effectuées par l'utilisateur. Un S est inclus si les définitions standard sont actives (non supprimées) pour les touches non réaffectées.

Niveau 1	→	Niveau 1
	→	{ obj_1 $x_{touche1}$ ... obj_n $x_{touchen}$ }
	→	{ S obj_1 $x_{touche1}$ ... obj_n $x_{touchen}$ }

Accès clavier : ↩ MODES KEYS RCLK

Indicateurs : Verrouillage mode utilisateur (-61) et Mode utilisateur (-62).

Remarques : L'argument x_{touche} est un nombre réel de la forme $lc.s$ qui désigne la touche par son numéro de ligne l, son numéro de colonne c et le code shift s. Pour les valeurs admissibles de s, reportez-vous à la commande ASN.

Commandes associées : ASN, DELKEYS, STOKEYS

RCLMENU

Commande de rappel du numéro de menu : Renvoie le numéro du menu en cours.

Niveau 1	→	Niveau 1
	→	x_{menu}

Accès clavier : ◀ MODES MENU RCLM

Indicateurs : Aucun

Remarques : x_{menu} a la forme *mm.pp*, où *mm* est le numéro du menu et *pp* la page du menu (voir la commande MENU pour connaître la liste des menus intégrés du HP 48 et les numéros correspondants).

Lorsque le menu en cours est un menu-utilisateur (créé par TMENU), RCLMENU renvoie `0.01` (en mode fixe à deux décimales), indiquant "Last menu" (dernier menu).

Exemple : Si la troisième page du menu PRG DSPL est active, RCLMENU renvoie `13.03` (mode fixe à deux décimales).

Commandes associées : MENU, TMENU

RCLΣ

Commande de rappel de la matrice statistique : Renvoie la matrice statistique en cours (le contenu de la variable réservée ΣDAT) du répertoire actif.

Niveau 1	→	Niveau 1
	→	*obj*

RCLΣ

Accès clavier :

◀ PLOT NXT STAT ΣDAT

◀ STAT DATA ΣDAT

◀ PLOT NXT STAT ▶ ΣDAT (mode saisie de programme)

◀ STAT DATA ▶ ΣDAT (mode saisie de programme)

Indicateurs : Aucun

Remarques : Pour rappeler *ΣDAT* à partir d'un répertoire parent (lorsque *ΣDAT* n'existe pas dans le répertoire en cours), évaluez le nom *ΣDAT*.

Commandes associées : CLΣ, STOΣ, Σ+, Σ−

RCWS

Commande de rappel de la taille de mot : Renvoie la taille de mot en cours en bits (1 à 64).

Niveau 1	→	Niveau 1
	→	*n*

Accès clavier : MTH BASE NXT RCWS

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Commandes associées : BIN, DEC, HEX, OCT, STWS

RDM

Commande de redimensionnement de tableaux : Réorganise les éléments de l'argument conformément aux dimensions spécifiées.

Niveau 2	Niveau 1	→	Niveau 1
[*vecteur*]$_1$	{ $n_{éléments}$ }	→	[*vecteur*]$_2$
[*vecteur*]	{ n_{lignes} m_{cols} }	→	[[*matrice*]]
[[*matrice*]]	{ $n_{éléments}$ }	→	[*vecteur*]
[[*matrice*]]$_1$	{ n_{lignes} m_{cols} }	→	[[*matrice*]]$_2$
'*global*'	{ $n_{éléments}$ }	→	
'*global*'	{ n_{lignes} m_{cols} }	→	

Accès clavier : (MTH) MATR MAKE RDM

Indicateurs : Aucun

Remarques : Si la liste contient un seul nombre $n_{éléments}$, le résultat est un vecteur de n éléments. Si la liste contient deux nombres n_{lignes} et m_{cols}, le résultat est une matrice $n \times m$.

Les éléments pris dans le vecteur ou la matrice argument conservent le même ordre des lignes dans le vecteur résultant ou la matrice. Si le tableau est redimensionné de façon à contenir moins d'éléments que l'ancien, le surplus est supprimé. Dans le cas contraire, les éléments manquants sont complétés par des zéros (ou par `(0,0)` si l'argument est un tableau complexe).

Si le vecteur ou la matrice argument est spécifié par *global*, le résultat remplace le contenu de la variable.

Exemples : `[ 2 4 6 8 ] ( 2 2 )` RDM renvoie `[[ 2 4 ][ 6 8 ]]`.

`[[ 2 3 4 ][ 1 6 9 ]] 8` RDM renvoie `[ 2 3 4 1 6 9 0 0 ]`.

Commande associée : TRN

RDZ

Commande de définition d'un générateur : Utilise un nombre réel $x_{générateur}$ comme générateur de nombres aléatoires pour la commande RAND.

{ }

Niveau 1	→	Niveau 1
$x_{générateur}$	→	

Accès clavier : MTH NXT PROB RDZ

Indicateurs : Aucun

Remarque : Si l'argument est 0, une valeur aléatoire basée sur l'horloge système est utilisée comme générateur.

Commandes associées : COMB, PERM, RAND, !

RE

Fonction d'extraction de la partie réelle d'un nombre : Renvoie la partie réelle de l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	*x*
x_unité	→	*x*
(*x*, *y*)	→	*x*
[*R-tableau*]	→	[*R-tableau*]
[*C-tableau*]	→	[*R-tableau*]
'*symb*'	→	'RE(*symb*)'

Accès clavier : MTH NXT CMPL RE

Indicateurs : Résultats numériques (−3)

Remarques : Si l'argument est un vecteur ou une matrice, RE renvoie un tableau réel, dont les éléments égalent les parties réelles des éléments correspondants de l'argument.

Commandes associées : C→R, IM, R→C

RECN

Commande de réception d'un objet renommé : Prépare le HP 48 à recevoir un fichier d'une autre unité Kermit et à le stocker dans une variable spécifiée.

{ }

Niveau 1	→	Niveau 1
'nom'	→	
"nom"	→	

Accès clavier : ◀ I/O NXT RECN

Indicateurs : Unité d'E-S (−33), Format de données des E-S (−35), Ecrasement de RECV (−36), Messages d'E-S (−39)

Lorsqu'un HP 48 envoie un objet, il lui ajoute automatiquement un en-tête indiquant au HP 48 récepteur s'il convient d'utiliser le mode ASCII ou binaire. L'indicateur −35 n'a d'effet que si cet en-tête n'existe pas.

Remarque : RECN est identique à RECV, sauf que le nom utilisé pour le stockage des données est spécifié dans la pile.

Commandes associées : BAUD, CKSM, CLOSEIO, FINISH, KERRM, KGET, PARITY, RECV, SEND, SERVER, TRANSIO

RECT

Commande de mode rectangulaire : Sélectionne le mode rectangulaire.

Accès clavier :

(MTH) VECTR (NXT) RECT

(↩)(MODES) ANGL RECT

Indicateurs : Aucun

Remarque : RECT désarme les indicateurs −15 et −16, et désactive les témoins R∠Z et R∠∠.

En mode rectangulaire, les vecteurs sont affichés comme des composants rectangulaires. Ainsi, un vecteur 3D s'affiche sous la forme [*X Y Z*].

Commandes associées : CYLIN, SPHERE

RECV

Commande de réception d'un objet : Donne instruction au HP 48 de rechercher un fichier nommé en provenance d'une autre unité Kermit. Le fichier reçu est stocké dans une variable nommée par l'émetteur.

Accès clavier : (↩)(I/O) RECV

Indicateurs : Unité d'E-S (−33), Format de données des E-S (−35), Ecrasement de RECV (−36), Messages d'E-S (−39)

Lorsqu'un HP 48 envoie un objet, il lui ajoute automatiquement un en-tête qui indique au HP 48 récepteur s'il convient d'utiliser le mode ASCII ou binaire. L'indicateur −35 n'a d'effet que si cet en-tête n'existe pas.

Remarque : Comme le HP 48 ne recherche pas systématiquement les fichiers Kermit entrants, vous devez utiliser RECV pour lui en donner l'instruction.

Commandes associées : BAUD, CKSM, FINISH, KGET, PARITY, RECN, SEND, SERVER, TRANSIO

REPEAT

Commande de répétition : Commence une clause de boucle dans la structure en boucle infinie WHILE ... REPEAT ... END.

Pour plus d'informations sur la syntaxe, voir la commande WHILE.

Accès clavier : PRG BRCH WHILE REPEA

Remarque : Pour de plus amples informations, voir la commande WHILE.

Commandes associées : END, WHILE

REPL

Commande de remplacement : Remplace une partie de l'objet cible du niveau 3 par l'objet du niveau 1, en commençant à la position spécifiée au niveau 2.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	$n_{position}$	[[*matrice*]]$_2$	→	[[*matrice*]]$_3$
[[*matrice*]]$_1$	{ n_{ligne} n_{col} }	[[*matrice*]]$_2$	→	[[*matrice*]]$_3$
[*vecteur*]$_1$	$n_{position}$	[*vecteur*]$_2$	→	[*vecteur*]$_3$
{ $liste_{cible}$ }	$n_{position}$	{ $liste_1$ }	→	{ $liste_{résultat}$ }
"$chaîne_{cible}$"	$n_{position}$	"$chaîne_1$"	→	"$chaîne_{résultat}$"
$grob_{cible}$	{#*n* #*m*}	$grob_1$	→	$grob_{résultat}$
$grob_{cible}$	(*x*,*y*)	$grob_1$	→	$grob_{résultat}$
PICT	{#*n* #*m*}	$grob_1$	→	
PICT	(*x*,*y*)	$grob_1$	→	

Accès clavier :

← CHARS REPL

PRG LIST REPL

PRG GROB REPL

REPL

(MTH) MATR MAKE (NXT) REPL

Indicateurs : Aucun

Remarques : Pour des tableaux, $n_{position}$ respecte l'ordre des lignes. Pour des matrices, $n_{position}$ spécifie le nouvel emplacement de l'élément supérieur gauche de la matrice de remplacement.

Pour des objets graphiques, l'angle supérieur gauche de $grob_1$ est positionné aux coordonnées $\langle x,y \rangle$ (unités-utilisateur) ou $\langle$ #n #m $\rangle$ (pixels). A partir de là, l'objet écrase la partie rectangulaire de $grob_{cible}$ ou de *PICT*. Si $grob_1$ s'étend au-delà de $grob_{cible}$ ou de *PICT* dans l'une ou l'autre direction, il est tronqué. Si les coordonnées spécifiées se trouvent hors de l'objet cible, celui-ci ne change pas.

Exemples : [[1 1 1 1] [1 1 1 1] [1 1 1 1]]
6 [[2 2] [2 2]] REPL renvoie
[[1 1 1 1] [1 2 2 1] [1 2 2 1]].

« A B C D E » 2 « F G » REPL renvoie « A F G D E ».

"ABCDE" 5 "FG" REPL renvoie "ABCDFG".

ERASE PICT (0,0) # 5d # 5d BLANK NEG REPL remplace une partie de *PICT* par un objet graphique 5 x 5, dont chaque pixel est activé (sombre) et dont l'angle supérieur gauche est situé au point (0,0) dans *PICT*.

Commandes associées : CHR, GOR, GXOR, NUM, POS, SIZE, SUB

RES

Commande de résolution : Spécifie la résolution des tracés mathématiques et statistiques, à savoir l'intervalle qui sépare les valeurs de la variable indépendante utilisée pour générer le tracé.

{ }

Niveau 1	→	Niveau 1
$n_{intervalle}$	→	
$\#n_{intervalle}$	→	

Accès clavier : ◀ PLOT PPAR RES

Indicateurs : Aucun

Remarques : Un nombre réel $n_{intervalle}$ spécifie l'intervalle en unités-utilisateur. Un entier binaire $\#n_{intervalle}$ le spécifie en pixels.

La résolution est stockée dans le quatrième élément de *PPAR* (0 est la valeur par défaut). La valeur par défaut est interprétée comme indiqué dans le tableau suivant :

RES

Type de tracé	Intervalle par défaut
BAR	10 pixels (largeur de barre = colonnes de 10 pixels)
DIFFEQ	sans limite : la taille du pas n'est pas limitée
FUNCTION	1 pixel (trace un point dans chaque colonne de pixels)
CONIC	1 pixel (trace un point dans chaque colonne de pixels)
TRUTH	1 pixel (trace un point dans chaque colonne de pixels)
GRIDMAP	RES non applicable
HISTOGRAM	10 pixels (largeur de bloc = colonnes de 10 pixels)
PARAMETRIC	[fourchette de la variable indépendante en unités-utilisateur]/130
PARSURFACE	RES non applicable
PCONTOUR	RES non applicable
POLAR	2°, 2 grades ou $\pi/90$ radians
SCATTER	RES non applicable
SLOPEFIELD	RES non applicable
WIREFRAME	RES non applicable
YSLICE	1 pixel (trace un point dans chaque colonne de pixels)

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

RESTORE

Commande de restauration de HOME : Remplace le répertoire en cours *HOME* par la copie de sauvegarde spécifiée (:n_{port}:$nom_{sauvegarde}$), préalablement créée au moyen de ARCHIVE.

{}

Niveau 1	→	Niveau 1
:n_{port} :$nom_{sauvegarde}$	→	
sauvegarde	→	

Accès clavier : ◀ MEMORY NXT RESTO

Indicateurs : Aucun

Remarques : Le numéro de port spécifié doit être compris entre 0 et 33. Les ports 1 et 2 doivent être configurés comme RAM indépendante (voir FREE).

Pour restaurer un répertoire *HOME* sauvegardé sur un système distant à l'aide de :ES:*nom* ARCHIVE, placez l'objet-sauvegarde lui-même dans la pile et exécutez RESTORE.

Exemple : Pour restaurer un répertoire *HOME* sauvegardé dans le fichier *AUG1* sur un système distant, exécutez 'AUG1' SEND sur le système distant, puis exécutez la séquence suivante sur le HP 48 :

```
RECV 'AUG1' RCL RESTORE
```

Commande associée : ARCHIVE

REVLIST

Commande d'inversion de liste : Inverse l'ordre des éléments d'une liste.

Niveau 1	→	Niveau 1
{ obj_n ... obj_1 }	→	{ obj_1 ... obj_n }

Accès clavier :

PRG LIST PROC REVLI

MTH LIST REVLI

Indicateurs : Aucun

Commande associée : ↑SORT

RKF

Commande de résolution d'un problème à valeur initiale (méthode Runge-Kutta-Fehlberg) : Calcule la solution d'un problème à valeur initiale pour une équation différentielle, en utilisant la méthode Runge-Kutta-Fehlberg (4,5).

Niveau 3	Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
{ *liste* }	x_{tol}	x_{Tfinal}	→	{ *liste* }	x_{tol}
{ *liste* }	{ x_{tol} x_{hpas} }	x_{Tfinal}	→	{ *liste* }	x_{tol}

Accès clavier : ↶ SOLVE DIFFE RKF

Indicateurs : Aucun

Remarques : RKF résout y'(t)=f(t,y), où $y(t_0)=y_0$. Les arguments et les résultats se présentent comme suit :

- { *liste* } contient trois éléments placés dans l'ordre ci-après : la variable indépendante (t), la variable solution (y) et le membre droit de l'équation différentielle (ou une variable contenant l'expression).
- x_{tol} définit la tolérance d'erreur absolue. En cas d'utilisation d'une liste, la première valeur est la tolérance d'erreur absolue et la seconde est la taille du pas initial possible.
- x_{Tfinal} spécifie la valeur finale de la variable indépendante.

RKF appelle RKFSTEP autant de fois que nécessaire pour progresser de la valeur initiale à x_{Tfinal}.

Exemple : Résolvez le problème à valeur initiale suivant pour y(8), étant donné y(0) = 0:

$$y' = \frac{1}{(1+t^2)} - 2y^2 = f(t,y)$$

1. Stockez la valeur initiale de la variable indépendante, 0, dans T.
2. Stockez la valeur initiale de la variable dépendante, 0, dans Y.
3. Stockez l'expression, $\frac{1}{(1+t^2)} - 2y^2$, dans F.
4. Entrez une liste contenant ces trois éléments : `{ T Y F }`.
5. Indiquez la tolérance. Utilisez la précision décimale estimée pour choisir une tolérance : 0.00001.
6. Entrez la valeur finale pour la variable indépendante : 8.

   La pile doit se présenter ainsi :

   ```
   { T Y F }
   .00001
   8
   ```

7. Appuyez sur RKF . (Le calcul dure un certain temps.)
   La variable T contient à présent 8, et Y contient la valeur .123077277659.

La réponse réelle est .123076923077 ; la comparaison des résultats montre une erreur d'environ .00000035, qui satisfait à la tolérance spécifiée.

Commandes associées : RKFERR, RKFSTEP, RRK, RRKSTEP, RSBERR

RKFERR

Commande d'estimation d'erreur pour la méthode Runge-Kutta-Fehlberg : Renvoie l'estimation d'erreur absolue pour un pas donné *h* lors de la résolution d'un problème à valeur initiale pour une équation différentielle.

Niveau 2	Niveau 1	→	Niveau 4	Niveau 3	Niveau 2	Niveau 1
{ *liste* }	*h*	→	{ *liste* }	*h*	y_{delta}	*err*

Accès clavier : ↩ SOLVE DIFFE RKFE

Indicateurs : Aucun

Remarques : Les arguments et les résultats se présentent comme suit :

- { *liste* } contient trois éléments placés dans l'ordre ci-après : la variable indépendante (t), la variable solution (y) et le membre droit de l'équation différentielle (ou une variable contenant l'expression).
- *h* est un nombre réel désignant le pas.
- y_{delta} affiche le changement de solution pour le pas spécifié.
- *err* affiche l'erreur absolue pour ce pas. Une erreur zéro indique que la méthode Runge-Kutta-Fehlberg a échoué et que la méthode d'Euler a été utilisée à la place.

L'erreur absolue est la valeur absolue de l'erreur estimée pour un problème scalaire et la norme infinie (norme de ligne) du vecteur d'erreur estimé pour un problème vectoriel. (Cette valeur est une borne supérieure de l'erreur faite sur n'importe quelle composante de la solution.)

Commandes associées : RKF, RKFSTEP, RRK, RRKSTEP, RSBERR

RKFSTEP

Commande du pas de la solution suivante pour RKF : Calcule le pas suivant ($h_{suivant}$) de la solution d'un problème à valeur initiale pour une équation différentielle.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
{ *liste* }	x_{tol}	h	→	{ *liste* }	x_{tol}	$h_{suivant}$

Accès clavier : ↩ SOLVE DIFFE RKFS

Indicateurs : Aucun

Remarques : Les arguments et les résultats se présentent comme suit :

- { *liste* } contient trois éléments placés dans l'ordre ci-après : la variable indépendante (t), la variable solution (y) et le membre droit de l'équation différentielle (ou une variale contenant l'expression).
- x_{tol} définit la valeur de tolérance.
- h spécifie le pas initial possible.
- $h_{suivant}$ désigne le pas suivant possible.

Les variables indépendante et solution doivent contenir des valeurs. RKFSTEP fait progresser ces variables au point suivant jusqu'à la solution.

Si la tolérance globale d'erreur n'est pas satisfaite, le pas réel utilisé par RKFSTEP est inférieur à la valeur entrée h. Si une tolérance globale d'erreur stricte oblige RKFSTEP à réduire la taille du pas au point de faire échouer la méthode Runge-Kutta-Fehlberg, RKFSTEP applique la méthode d'Euler pour calculer le pas suivant de la solution, et considère que la tolérance d'erreur est respectée. La méthode Runge-Kutta-Fehlberg échoue si la variable indépendante en cours est égale à zéro et la taille du pas à $\leq 1.3 \times 10^{-498}$, ou si la variable est différente de zéro et la taille du pas égale à 1.3×10^{-10} fois sa grandeur.

Commandes associées : RKF, RKFERR, RRK, RRKSTEP, RSBERR

RL

Commande de rotation vers la gauche : Effectue la rotation d'un entier binaire d'un bit vers la gauche.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : MTH BASE NXT BIT RL

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : Le bit le plus à gauche de #n_1 devient le bit le plus à droite de #n_2.

Commandes associées : RLB, RR, RRB

RLB

Commande de rotation vers la gauche d'un octet : Effectue la rotation d'un entier binaire d'un octet vers la gauche.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : MTH BASE NXT BYTE RLB

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : L'octet le plus à gauche de #n_1 devient l'octet de plus à droite de #n_2. RLB équivaut à huit exécutions de RL.

Commandes associées : RL, RR, RRB

RND

Fonction d'arrondi : Arrondit un objet au nombre spécifié de décimales ou de chiffres significatifs, ou l'ajuste au format d'affichage en cours.

{ }

Niveau 2	Niveau 1	→	Niveau 1
z_1	$n_{arrondi}$	→	z_2
z	'$symb_{arrondi}$'	→	'RND(z,$symb_{arrondi}$)'
'$symb$'	$n_{arrondi}$	→	'RND($symb$,$n_{arrondi}$)'
'$symb_1$'	'$symb_{arrondi}$'	→	'RND($symb_1$,$symb_{arrondi}$)'
[$tableau_1$]	$n_{arrondi}$	→	[$tableau_2$]
$x\_unité$	$n_{arrondi}$	→	$y\_unité$
$x\_unité$	'$symb_{arrondi}$'	→	'RND($x\_unité$,$symb_{arrondi}$)'

Accès clavier : MTH REAL NXT NXT RND

Indicateurs : Résultats numériques (−3)

Remarques : $n_{arrondi}$ (ou $symb_{arrondi}$ si l'indicateur −3 est armé) contrôle la façon dont l'argument du niveau 2 est arrondi, comme suit :

$n_{arrondi}$ ou $symb_{arrondi}$	Effet sur l'argument de niveau 2
0 à 11	Arrondi à n décimales.
−1 à −11	Arrondi à n chiffres significatifs.
12	Arrondi au format d'affichage en cours.

RND

Pour des nombres complexes et des tableaux, chaque nombre réel est arrondi. Pour des objets-unités, la partie numérique de l'objet est arrondie.

Exemples : (4.5792,8.1275) 2 RND renvoie (4.58,8.13).

[2.34907 3.96351 2.73453] -2 RND renvoie [2.3 4 2.7].

Commande associée : TRNC

RNRM

Commande de calcul de la norme de ligne : Renvoie la norme de ligne (norme infinie) d'un tableau.

{ }

Niveau 1	→	Niveau 1
[*tableau*]	→	$x_{normeligne}$

Accès clavier : (MTH) MATR NORM RNRM

Indicateurs : Aucun

Remarques : La norme de ligne est la valeur maximale (pour toutes les lignes) des sommes des valeurs absolues de tous les éléments d'une ligne. Pour un vecteur, elle est la valeur absolue maximale de ses éléments.

Commandes associées : CNRM, CROSS, DET, DOT

ROLL

Commande de permutation d'objets : Déplace le contenu d'un niveau spécifié vers le niveau 1 et remonte la partie de la pile située sous le niveau spécifié.

Niv n+1 ... Niv 2	Niv 1	→	Niv n ... Niv 2	Niv 1
obj_n ... obj_1	*n*	→	obj_{n-1} ... obj_1	obj_n

Accès clavier : ↶ STACK ROLL

Indicateurs : Aucun

Remarque : 3 ROLL est l'équivalent de ROT.

Commandes associées : OVER, PICK, ROLLD, ROT, SWAP

ROLLD

Commande d'abaissement d'un niveau : Déplace le contenu du niveau 1 vers un niveau spécifié et fait descendre la portion de la pile située au-dessous du niveau spécifié.

Niv n+1 ... Niv 2	Niv 1	→	Niv n	Niv n-1 ... Niv 1
obj_n ... obj_1	*n*	→	obj_1	obj_n ... obj_2

Accès clavier : ↶ STACK ROLLD

Indicateurs : Aucun

Commandes associées : OVER, PICK, ROLL, ROT, SWAP

ROOT

Commande de calcul de racine : Renvoie un nombre réel x_{racine} correspondant à une valeur de la variable spécifiée *global* pour laquelle le programme ou l'objet algébrique indiqué s'évalue à une valeur proche de zéro ou une valeur extrême locale.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
≪ *programme* ≫	'*global*'	*éval.*	→	x_{racine}
≪ *programme* ≫	'*global*'	{ *éval.* }	→	x_{racine}
'*symb*'	'*global*'	*éval.*	→	x_{racine}
'*symb*'	'*global*'	{ *éval.* }	→	x_{racine}

Accès clavier : ◀ SOLVE ROOT ROOT

Indicateurs : Aucun

Remarques : ROOT est la forme programmable de l'application SOLVE.

éval. est une estimation initiale de la solution. ROOT génère une erreur si elle ne parvient pas à trouver une solution, et renvoie le message `Bad Guess(es)` si une ou plusieurs estimations figurent hors du domaine de l'équation, ou le message `Constant?` si l'équation renvoie la même valeur à chaque point d'échantillonnage. ROOT *ne* renvoie *pas* de messages lorsqu'une racine est trouvée.

ROT

Commande de permutation d'objets : Effectue une permutation circulaire des trois premiers objets de la pile, l'objet du 3 passant au niveau 1.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
obj_3	obj_2	obj_1	→	obj_2	obj_1	obj_3

Accès clavier : ⬐ STACK ROT

Indicateurs : Aucun

Remarques : ROT est l'équivalent de 3 ROLL.

Commandes associées : OVER, PICK, ROLL, ROLLD, SWAP

→ROW

Commande de conversion de matrice en lignes : Eclate une matrice en une série de vecteurs-lignes et renvoie les vecteurs ainsi que le nombre de lignes, ou transforme un vecteur en ses éléments et renvoie ces derniers ainsi que leur nombre.

{ }

Niveau 1	→	Niveau n+1 ...	Niveau 2	Niveau 1
[[*matrice*]]	→	[*vecteur*]$_{ligne1}$	[*vecteur*]$_{lignen}$	$n_{nbrelignes}$
[*vecteur*]	→	*élément*$_1$	*élément*$_n$	$n_{nbreéléments}$

Accès clavier : MTH MATR ROW →ROW

Indicateurs : Aucun

Commandes associées : →COL, COL→, ROW→

ROW+

Commande d'insertion de ligne : Insère un tableau dans une matrice (ou un ou plusieurs nombres dans un vecteur) à la position indiquée par n_{index}, et renvoie la matrice modifiée (ou le vecteur).

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	[*matrice*]$_2$	n_{index}	→	[[*matrice*]]$_3$
[[*matrice*]]$_1$	[*vecteur*]$_{ligne}$	n_{index}	→	[[*matrice*]]$_2$
[*vecteur*]$_1$	$n_{élément}$	n_{index}	→	[*vecteur*]$_2$

Accès clavier : (MTH) MATR ROW ROW+

Indicateurs : Aucun

Remarques : Le tableau inséré doit avoir le même nombre de colonnes que le tableau cible.

n_{index} est arrondi à l'entier le plus proche. Le tableau initial est redimensionné de façon à inclure les nouvelles colonnes ou les nouveaux éléments, et les éléments situés au niveau du point d'insertion ou au-dessous sont décalés vers le bas.

Commandes associées : COL−, COL+, ROW−, RSWP

ROW−

Commande de suppression de ligne : Supprime la ligne *n* d'une matrice (ou l'élément *n* d'un vecteur) et renvoie la matrice modifiée (ou le vecteur) ainsi que la ligne supprimée (ou l'élément).

{ }

Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
[[*matrice*]]$_1$	n_{ligne}	→	[[*matrice*]]$_2$	[*vecteur*]$_{ligne}$
[*vecteur*]$_1$	$n_{élément}$	→	[*vecteur*]$_2$	*élément*$_n$

Accès clavier : (MTH) MATR ROW ROW−

Indicateurs : Aucun

Remarque : n_{ligne} ou $n_{élément}$ est arrondi à l'entier le plus proche.

Commandes associées : COL−, COL+, ROW+, RSWP

ROW→

Commande de conversion de lignes en matrice : Construit une matrice à partir d'une série de vecteurs-lignes et d'un nombre de lignes, ou génère un vecteur à partir d'une séquence de nombres et d'un nombre d'éléments.

Niveau$_{n+1}$...	Niveau 2	Niveau 1	→	Niveau 1
[*vecteur*]$_{ligne1}$	[*vecteur*]$_{lignen}$	$n_{nbrelignes}$	→	[[*matrice*]]
élément$_1$	*élément*$_n$	$n_{nbreéléments}$	→	[*vecteur*]$_{colonne}$

Accès clavier : (MTH) MATR ROW ROW+

Indicateurs : Aucun

Commandes associées : →COL, COL→, →ROW

RR

Commande de rotation vers la droite : Effectue la rotation d'un entier binaire d'un bit vers la droite.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : (MTH) BASE (NXT) BIT RR

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarque : Le bit le plus à droite de #n_1 devient le bit le plus à gauche de #n_2.

Commandes associées : RL, RLB, RRB

RRB

Commande de rotation vers la droite d'un octet : Effectue la rotation d'un entier binaire d'un octet vers la droite.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : (MTH) BASE (NXT) BYTE RRB

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12).

Remarque : L'octet le plus à droite de #n_1 devient l'octet le plus à gauche de #n_2. RRB équivaut à huit exécutions de RR.

Commandes associées : RL, RLB, RR

RREF

Commande de forme réduite échelonnée d'une matrice : Calcule la forme réduite échelonnée d'une matrice rectangulaire.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]$_A$	→	[[*matrice*]]$_R$

Accès clavier : (MTH) MATR FACTR RREF

Indicateurs : Eléments "très petits" (−54)

Remarques : Convertit une matrice en sa forme réduite échelonnée. La forme linéaire échelonnée étant principalement utilisée pour l'étude de systèmes d'équations linéaires, RREF ignore les très petites valeurs lorsque l'indicateur système −54 est désarmé.

Commande associée : LU

RRK

Commande de résolution pour des valeurs initiales (Rosenbrock, Runge-Kutta) : Calcule la solution d'un problème à valeur initiale pour une équation différentielle avec des dérivées partielles connues.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
{ *liste* }	x_{tol}	x_{Tfinal}	→	{ *liste* }	x_{tol}
{ *liste* }	{ x_{tol} x_{hpas} }	x_{Tfinal}	→	{ *liste* }	x_{tol}

Accès clavier : ↩ (SOLVE) DIFFE RRK

Indicateurs : Aucun

RRK

Remarques : RRK résout y'(t)=f(t,y), où $y(t_0)=y_0$. Les arguments et les résultats se présentent comme suit :

- { *liste* } contient cinq éléments, placés dans l'ordre ci-après :
  - La variable indépendante (t).
  - La variable solution (y).
  - Le membre droit de l'équation différentielle (ou une variable contenant l'expression).
  - La dérivée partielle de y'(t) par rapport à la variable solution (ou une variable contenant l'expression).
  - La dérivée partielle de y'(t) par rapport à la variable indépendante (ou une variable contenant l'expression).
- x_{tol} définit la valeur de tolérance. En cas d'utilisation d'une liste, la première valeur est la tolérance, la seconde est la taille du pas initial possible.
- x_{Tfinal} spécifie la valeur finale de la variable indépendante.

RRK appelle RKFSTEP de façon répétée à mesure qu'elle progresse de la valeur initiale jusqu'à x_{Tfinal}.

Exemple : Résolvez le problème à valeur initiale suivant pour y(8), étant donné y(0) = 0:

$$y' = \frac{1}{(1+t^2)} - 2y^2 = f(t,y)$$

La dérivée de la fonction par rapport à y ($\partial f/\partial y$) est -4y, et la dérivée de la fonction par rapport à t ($\partial f/\partial t$) est $\frac{-2t}{(1+t^2)^2}$.

1. Stockez la valeur initiale de la variable indépendante, 0, dans T.
2. Stockez la valeur initiale de la variable dépendante, 0, dans Y.
3. Stockez l'expression, $\frac{1}{(1+t^2)} - 2y^2$, dans F.
4. Stockez $\partial f/\partial y$, -4y, dans FY.
5. Stockez $\partial f/\partial t$, $\frac{-2t}{(1+t^2)^2}$, dans FT.
6. Entrez ces cinq éléments dans une liste : { T Y F FY FT }.
7. Indiquez la tolérance. Utilisez la précision décimale estimée pour choisir une tolérance : 0.00001.
8. Entrez la valeur finale de la variable indépendante : 8.

La pile doit se présenter ainsi :

```
{ T Y F FY FT}
.00001
8
```

9. Appuyez sur RRK . (Le calcul dure un certain temps.)
   La variable T contient à présent 8, et Y contient la valeur .123077277659.

La réponse réelle est .123076923077. La comparaison des résultats montre une erreur d'environ .00000035, ce qui satisfait à la tolérance spécifiée.

Commandes associées : RKF, RKFERR, RKFSTEP, RRKSTEP, RSBERR

RRKSTEP

Commande du pas suivant de la solution et méthode utilisée (RKF ou RRK) :

Calcule le pas suivant (h_{suiv}) de la solution d'un problème à valeur initiale pour une équation différentielle, et affiche la méthode utilisée pour parvenir au résultat.

Niv 4	Niv 3	Niv 2	Niv 1	→	Niv 4	Niv 3	Niv 2	Niv 1
{ *liste* }	x_{tol}	*h*	*dern*	→	{ *liste* }	x_{tol}	h_{suiv}	*courant*

Accès clavier : ↶ SOLVE DIFFE RRKS

Indicateurs : Aucun

Remarques : Les arguments et les résultats se présentent comme suit :

- { *liste* } contient cinq éléments, placés dans l'ordre ci-après :
  - La variable indépendante (t).
  - La variable solution (y).

- Le membre droit de l'équation différentielle (ou une variable contenant l'expression).
- La dérivée partielle de y'(t) par rapport à la variable solution (ou une variable contenant l'expression).
- La dérivée partielle de y'(t) par rapport à la variable indépendante (ou une variable contenant l'expression).

- x_{tol} est la valeur de tolérance.
- h spécifie le pas initial possible.
- *dern* spécifie la dernière méthode utilisée (RKF = 1, RRK = 2). Si vous utilisez RRKSTEP pour la première fois, entrez 0.
- *courant* affiche la méthode en cours utilisée pour arriver au pas suivant.
- h_{suiv} est le pas suivant possible.

Les variables indépendante et solution doivent contenir des valeurs. RRKSTEP fait progresser ces variables au point suivant jusqu'à la solution.

Le pas réel utilisé par RRKSTEP est inférieur à la valeur entrée h si la tolérance globale d'erreur n'est pas satisfaite. Si une tolérance globale d'erreur stricte oblige RRKSTEP à réduire la taille du pas de telle sorte que la méthode Runge-Kutta-Fehlberg ou Rosenbrock échoue, RRKSTEP applique la méthode d'Euler pour calculer le pas suivant de la solution et considère que la tolérance d'erreur est satisfaite. La méthode Rosenbrock échoue si la variable indépendante en cours est zéro et le pas égal à $\leq 2.5 \times 10^{-499}$, ou si la variable est différente de zéro et le pas égal à 2.5×10^{-11} fois sa grandeur. La méthode Runge-Kutta-Fehlberg échoue si la variable indépendante en cours est zéro et le pas égal à $\leq 1.3 \times 10^{-498}$, ou si la variable est différente de zéro et le pas égal à 1.3×10^{-10} fois sa grandeur.

Commandes associées : RKF, RKFERR, RKFSTEP, RRK, RSBERR

RSBERR

Commande d'estimation d'erreur pour la méthode Rosenbrock : Renvoie une estimation d'erreur pour un pas donné *h* lors de la résolution d'un problème à valeur initiale pour une équation différentielle.

Niv 2	Niv 1	→	Niv 4	Niv 3	Niv 2	Niv 1
{ *liste* }	*h*	→	{ *liste* }	*h*	y_{delta}	*err*

Accès clavier : ◀ SOLVE DIFFE RSBER

Indicateurs : Aucun

Remarques : Les arguments et les résultats se présentent comme suit :

- { *liste* } contient cinq éléments placés dans l'ordre ci-après :
  - La variable indépendante (t).
  - La variable solution (y).
  - Le membre droit de l'équation différentielle (ou une variable contenant l'expression).
  - La dérivée partielle de y'(t) par rapport à la variable solution (ou une variable contenant l'expression).
  - La dérivée partielle de y'(t) par rapport à la variable indépendante (ou une variable contenant l'expression).
- *h* est un nombre réel spécifiant le pas initial.
- y_{delta} affiche le changement de la solution.
- *err* affiche l'erreur absolue pour ce pas. L'erreur ***absolue*** est la valeur absolue de l'erreur estimée pour un problème scalaire et la norme infinie (norme de ligne) du vecteur d'erreur estimé pour un problème vectoriel. (Cette valeur est une borne supérieure de l'erreur faite sur n'importe quelle composante de la solution.) Une erreur zéro indique que la méthode Rosenbrock a échoué et que la méthode d'Euler a été appliquée à la place.

Commandes associées : RKF, RKFERR, RKFSTEP, RRK, RRKSTEP

RSD

Commande de calcul des résidus : Calcule le résidu **B** − **AZ** des tableaux **B**, **A** et **Z**.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[*vecteur*]$_B$	[[*matrice*]]$_A$	[*vecteur*]$_Z$	→	[*vecteur*]$_{B-AZ}$
[[*matrice*]]$_B$	[[*matrice*]]$_A$	[[*matrice*]]$_Z$	→	[[*matrice*]]$_{B-AZ}$

Accès clavier : (MTH) MATR (NXT) RSD

Indicateurs : Aucun

Remarques : **A**, **B** et **Z** font l'objet des restrictions suivantes :

- **A** doit être une matrice.
- Le nombre de colonnes de **A** doit être égal au nombre d'éléments de **Z** si **Z** est un vecteur, ou au nombre de lignes de **Z** si **Z** est une matrice.
- Le nombre de lignes de **A** doit être égal au nombre d'éléments de **B** si **B** est un vecteur, ou au nombre de lignes de **B** si **B** est une matrice.
- **B** et **Z** doivent tous deux être des vecteurs ou des matrices.
- **B** et **Z** doivent avoir le même nombre de colonnes s'ils sont des matrices.

RSD sert généralement à calculer une correction de **Z**, lorsque **Z** a été obtenu comme approximation de la solution **X** pour le système d'équations **AX** = **B**.

RSWP

Commande de permutation de ligne : Permute les lignes i et j d'une matrice et renvoie la matrice modifiée, ou permute les éléments i et j d'un vecteur et renvoie le vecteur modifié.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	$n_{\text{ligne}i}$	$n_{\text{ligne}j}$	→	[[*matrice*]]$_2$
[*vecteur*]$_1$	$n_{\text{élément}i}$	$n_{\text{élément}j}$	→	[*vecteur*]$_2$

Accès clavier : (MTH) MATR ROW (NXT) RSWP

Indicateurs : Aucun

Remarques : Les numéros de lignes sont arrondis à l'entier le plus proche. Les arguments vecteurs sont traités comme des vecteurs-colonnes.

Commandes associées : CSWP, ROW+, ROW−

R→B

Commande de conversion de réel en binaire : Convertit un entier réel positif en son équivalent entier binaire.

{ }

Niveau 1	→	Niveau 1
n	→	#*n*

Accès clavier : (MTH) BASE R→B

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

R→B

Remarques : Pour toute valeur de $n \leq 0$, le résultat est # 0. Pour toute valeur de $n \geq 1.84467440738E19$ (base 10), le résultat est # FFFFFFFFFFFFFFFF (base 16).

Commande associée : B→R

R→C

Commande de conversion de réel en complexe : Combine deux nombres réels ou deux tableaux réels en un nombre ou un tableau complexe.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	(*x*,*y*)
[*R-tableau*$_1$]	[*R-tableau*$_2$]	→	[*C-tableau*]

Accès clavier :

PRG TYPE NXT R→C

MTH NXT CMPL R→C

Indicateurs : Aucun

Remarques : L'argument du niveau 2 représente le ou les éléments réels du résultat complexe. L'argument du niveau 1 représente le ou les éléments imaginaires du résultat complexe.

Les arguments tableaux doivent avoir les mêmes dimensions.

Commandes associées : C→R, IM, RE

R→D

Fonction de conversion de radians en degrés : Convertit un nombre réel exprimé en radians en son équivalent en degrés.

{ }

Niveau 1	→	Niveau 1
x	→	(180/π) *x*
'*symb*'	→	'R→D(*symb*)'

Accès clavier : (MTH) REAL (NXT) (NXT) R→D

Indicateurs : Résultats numériques (−3)

Remarques : Cette fonction opère indépendamment du mode d'angle.

Commande associée : D→R

SAME

Commande de comparaison d'objets : Compare deux objets, et renvoie un résultat vrai (1) s'ils sont identiques, et un résultat faux (0) s'ils ne le sont pas.

Niveau 2	Niveau 1	→	Niveau 1
obj_1	obj_2	→	0/1

Accès clavier : (PRG) TEST (NXT) SAME

Indicateurs : Aucun

Remarques : SAME est en fait identique à == pour tous les types d'objet à l'exception des expressions algébriques, des noms et de certaines unités (pour les expressions algébriques et les noms, ==

SAME

renvoie une expression qui peut être évaluée pour produire un résultat de test en fonction de valeurs numériques).

Exemples : « A B » (4,5) SAME renvoie 0.

« A B » « B A » SAME renvoie 0.

"CHATS" "CHATS" SAME renvoie 1.

Commandes associées : TYPE, ==

SBRK

Commande d'interruption série : Interrompt une transmission ou une réception en série.

Accès clavier : ◀ I/O NXT SERIA SBRK

Indicateurs : Unité d'E-S (−33)

Remarque : SBRK est en principe utilisé lorsqu'un problème a lieu dans une transmission de données en série.

Commandes associées : BUFLEN, SRECV, STIME, XMIT

SCALE

Commande de définition de l'échelle du tracé : Ajuste les deux premiers paramètres de *PPAR*, (x_{min}, y_{min}) et (x_{max}, y_{max}), de manière à ce que $x_{échelle}$ et $y_{échelle}$ soient les nouvelles échelles horizontale et verticale du tracé, et à ce que le point central ne change pas.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$x_{échelle}$	$y_{échelle}$	→	

Accès clavier : ◀ PLOT PPAR NXT SCALE

Indicateurs : Aucun

Remarques : L'échelle de chaque axe est le nombre d'unités-utilisateur par graduation. L'échelle par défaut employée sur les deux axes est 1 unité par graduation.

Commandes associées : AUTO, CENTR, *H, *W

SCATRPLOT

Commande de tracé de diagramme en nuage de points : Trace le diagramme en nuage de points de données (x, y) issus des colonnes spécifiées de la matrice statistique en cours (variable réservée ΣDAT).

Accès clavier : ◀ STAT PLOT SCATR

Indicateurs : Aucun

Remarques : Les colonnes de données tracées sont spécifiées par XCOL et YCOL, et sont stockées comme les deux premiers paramètres dans la variable réservée ΣPAR. S'il n'y a pas de colonnes de données spécifiées, les colonnes 1 (indépendante) et 2 (dépendante) sont sélectionnées par défaut. L'axe y est mis à l'échelle automatiquement et le type de tracé est défini comme SCATTER.

Lorsque SCATRPLOT est exécuté à partir d'un programme, l'affichage résultant n'est pas maintenu, sauf si PICTURE ou PVIEW est exécuté ensuite.

Si PICTURE *est* exécuté, l'utilisation de STATL dans l'environnement Picture trace une ligne d'ajustement des données au modèle statistique en cours.

Exemple : Le programme suivant trace le diagramme en nuage de points des données des colonnes 3 et 4 de ΣDAT, le superpose au modèle de regression le mieux adapté et affiche le tracé :

```
« 3 XCOL 4 YCOL SCATRPLOT BESTFIT ΣLINE STEQ
  FUNCTION DRAW ( # 0d # 0d ) PVIEW 7 FREEZE »
```

Commandes associées : BARPLOT, PICTURE, HISTPLOT, PVIEW, SCLΣ, XCOL, YCOL

SCATTER

Commande de type de tracé SCATTER : Définit le type de tracé SCATTER.

Accès clavier : ◀ PLOT NXT STAT PTYPE SCATT

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est SCATTER, la commande DRAW trace des points en prenant des coordonnées x et y dans deux colonnes de la matrice statistique en cours (variable réservée ΣDAT). Les colonnes sont spécifiées par le premier et le deuxième paramètre dans la variable réservée ΣPAR (à l'aide des commandes XCOL et YCOL). Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, qui se présente sous la forme suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé SCATTER, les éléments de *PPAR* sont utilisés de la manière suivante :

- (x_{min}, y_{min}) est un nombre complexe spécifiant l'angle inférieur gauche de *PICT* (l'angle inférieur gauche de la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe spécifiant l'angle supérieur gauche de *PICT* (l'angle supérieur droit de la plage d''affichage). La valeur par défaut est (6.5,3.2).
- *indep* est un nom spécifiant la variable indépendante. La valeur par défaut de *indep* est X.
- *res* n'est pas utilisé.
- *axes* est une liste contenant un ou plusieurs des éléments suivants, dans l'ordre : un nombre complexe spécifiant les coordonnées en unités-utilisateur de l'origine du tracé, une liste spécifiant l'intervalle de graduation et deux chaînes spécifiant le libellé des axes horizontal et vertical. La valeur par défaut est (0,0).
- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de la commande SCATTER place le nom SCATTER dans *ptype*.
- *depend* est un nom spécifiant la variable dépendante. La valeur par défaut est Y.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE

SCHUR

Commande de décomposition de Schur d'une matrice carrée : Renvoie la décomposition de Schur d'une matrice carrée.

{ }

Niveau 1	→	Niveau 2	Niveau 1
[[*matrice*]]$_A$	→	[[*matrice*]]$_Q$	[[*matrice*]]$_T$

Accès clavier : (MTH) MATR FACTR SCHUR

Indicateurs : Aucun

Remarques : SCHUR décompose A en deux matrices Q et T:

- Si A est une matrice complexe, Q est une matrice unitaire et T est une matrice triangulaire supérieure.
- Si A est une matrice réelle, Q est une matrice orthogonale et T est une matrice quasi triangulaire supérieure (T est triangulaire supérieure par blocs, avec des blocs diagonaux 1×1 ou 2×2 ; les blocs 2×2 ayant des valeurs propres complexes conjuguées).

Dans les deux cas, $A \cong Q \times T \times \mathrm{TRN}(Q)$.

Commandes associées : LQ, LU, QR, SVD, SVL, TRN

SCI

Commande de mode scientifique : Définit le mode scientifique pour l'affichage des nombres, à savoir un chiffre à gauche du séparateur décimal et *n* chiffres significatifs à droite.

{ }

Niveau 1	→	Niveau 1
n	→	

Accès clavier : ↩ MODES FMT SCI

Indicateurs : Aucun

Remarques : Le mode Scientific est équivalent à la notation scientifique utilisant $n + 1$ chiffres significatifs, où $0 \leq n \leq 11$ (les valeurs de *n* sortant de cette plage sont arrondies à l'entier le plus proche). En mode Scientific, les nombres sont affichés et imprimés comme suit :

(*signe*) *mantisse* E (*signe*) *exposant*

où la mantisse est de la forme *n*.(*n* ...) et comporte entre zéro et 11 décimales, et où l'exposant comporte entre un et trois chiffres.

Exemple : Le nombre 103.6 en mode Scientific à quatre décimales apparaît sous la forme `1.0360E2`.

Commandes associées : ENG, FIX, STD

SCLΣ

Commande d'ajustement de l'échelle : Ajuste (x_{min}, y_{min}) et (x_{max}, y_{max}) dans *PPAR* de manière à ce qu'un tracé en nuage de points ultérieur remplisse exactement *PICT*.

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est SCATTER, la commande AUTO incorpore les fonctions de SCLΣ. En outre, la commande SCATRPLOT exécute automatiquement AUTO pour atteindre le même résultat. SCLΣ est inclus dans le HP 48 aux fins de compatibilité avec le HP 28.

Commandes associées : AUTO, SCATRPLOT

SCONJ

Commande de stockage de conjugué : Conjugue le contenu d'un objet nommé.

{ }

Niveau 1	→	Niveau 1
'nom'	→	

Accès clavier : ◀ (MEMORY) ARITH (NXT) SCON

Indicateurs : Aucun

Remarques : L'objet nommé doit être un nom, un tableau ou un objet algébrique. Pour plus d'informations sur la conjugaison, voir CONJ.

Commandes associées : CONJ, SINV, SNEG

SDEV

Commande d'écart standard : Calcule l'écart standard sur échantillon de chacune des m colonnes de valeurs de coordonnées dans la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{écarts}$
	→	$[\ x_{écarts1}\ x_{écarts2}\ \cdots\ x_{écartsm}\ ]$

Accès clavier : ◀ STAT 1VAR SDEV

Indicateurs : Aucun

Remarques : SDEV renvoie un vecteur de m nombres réels, ou un nombre réel unique si $m = 1$. Les écarts standards (la racine carrée des variances) sont calculés à l'aide de la formule suivante :

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

où x_i est la $i$ème valeur de coordonnées d'une colonne, $\overline{x}$ est la moyenne des données de cette colonne et n est le nombre de points de données.

Commandes associées : MAXΣ, MEAN, MINΣ, PSDEV, PVAR, TOT, VAR

SEND

Commande d'envoi d'un objet : Envoie une copie des objets nommés à une unité Kermit.

Niveau 1	→	Niveau 1
'nom'	→	
{ nom_1 ... nom_n }	→	
{{ nom_{ancien} $nom_{nouveau}$ } *nom* ... }	→	

Accès clavier : ◀ I/O SEND

Indicateurs : Unité d'E-S (−33), Format de données des E-S (−35), Messages d'E-S (−39)

Si l'indicateur −35 est désarmé (transfert ASCII), le paramétrage de la traduction a également un effet.

Remarques : Les données sont toujours envoyées d'une unité Kermit locale, mais peuvent être réceptionnées par une autre unité Kermit locale (qui doit exécuter RECV ou RECN) ou un serveur Kermit.

Pour renommer un objet lors de l'envoi, incluez l'ancien nom et le nouveau dans une liste imbriquée.

Exemples : « AAA BBB » SEND envoie la variable nommée *AAA* en changeant son nom en *BBB*.

« AAA BBB » CCC » SEND envoie *AAA* sous la forme *BBB* et envoie *CCC* sous son propre nom (si le nouveau nom n'est pas autorisé sur le HP 48, il suffit de l'entrer sous forme de chaîne).

Commandes associées : BAUD, CLOSEIO, CKSM, FINISH, KERRM, KGET, PARITY, RECN, RECV, SERVER, TRANSIO

SEQ

Commande de calcul séquentiel : Renvoie une liste de résultats générés par une exécution répétée de $obj_{exéc}$ en employant *index* sur la plage $x_{début}$ à x_{fin}, par incrément de x_{incr}.

{ }

Niv 5	Niv 4	Niv 3	Niv 2	Niv 1	→	Niv 1
$obj_{exéc}$	*index*	$x_{début}$	x_{fin}	x_{incr}	→	{ *liste* }

Accès clavier : PRG LIST PROC NXT SEQ

Indicateurs : Aucun

Remarques : $obj_{exéc}$ est en principe un programme ou un objet algébrique constituant une fonction d'*index*, mais peut en réalité être un objet quelconque. *index* doit être un nom global ou local. Les objets restants peuvent être différents éléments qui s'évalueront en nombres réels.

L'action de SEQ pour des entrées arbitraires est équivalente à celle du programme suivant :

$x_{début}$ x_{fin} FOR *index* $obj_{exéc}$ EVAL x_{incr} STEP n → LIST

où n est le nombre de nouveaux objets laissés dans la pile par la boucle FOR ... STEP. Notez qu'*index* devient une variable locale quel que soit son type d'origine.

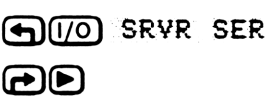
Exemple : `'n^2' 'n' 1 4 1` renvoie { 1 4 9 16 }.

Commandes associées : DOSUBS, STREAM

SERVER

Commande de mode serveur : Sélectionne le mode serveur Kermit.

Accès clavier :



Indicateurs : Unité d'E-S (−33), Format de données des E-S (−35), Ecrasement de RECV (−36), Messages d'E-S (−39)

Remarques : Un serveur Kermit (une unité Kermit en mode serveur) traite passivement les demandes qui lui sont envoyées par l'unité Kermit locale. Le serveur reçoit des données en réponse à SEND, transmet des données en réponse à KGET, met fin au mode serveur en réponse à FINISH ou à LOGOUT, et transmet un listage de répertoire en réponse à une demande de répertoire générique.

Le mode serveur prend en charge les paquets de commandes hôte Kermit. Vous pouvez ainsi, par exemple, utiliser un PC pour taper une entrée dans la ligne de commande du HP 48 (caractéristique utile lors du test de programmes). Voici comment procéder :

1. Configurez le HP 48 pour un transfert de données vers un ordinateur, comme décrit dans "Transfert de données entre le HP 48 et un ordinateur", au chapitre 27 du *Manuel d'utilisation du HP 48* .
2. Exécutez SERVER pour placer le HP 48 en mode serveur.
3. Sur le PC, tapez REMOTE HOST, et un maximum de 89 caractères à entrer dans la ligne de commande du HP 48.
4. Appuyez sur Entrée pour transmettre et exécuter les commandes. Le HP 48 exécute les commandes transmises, puis renvoie à l'écran du PC le contenu résultant de la pile, tel qu'il l'afficherait normalement.

Si vous vous servez d'un PC pour écrire des programmes pour le HP 48, vous devez inclure l'en-tête *%%HP...;* dans le programme. Reportez-vous aux explications données sur le mode ASCII, au chapitre 27 du *Manuel d'utilisation du HP 48.*

Commandes associées : BAUD, CKSM, FINISH, KERRM, KGET, PARITY, PKT, RECN, RECV, SEND, TRANSIO

SF

Commande d'armement de l'indicateur : Arme l'indicateur utilisateur ou système spécifié.

{ }

Niveau 1	→	Niveau 1
$n_{numéro-d'indicateur}$	→	

Accès clavier :

PRG TEST NXT NXT SF

◀ MODES FLAG SF

Indicateurs : Aucun

Remarques : Les indicateurs utilisateur sont numérotés de 1 à 64. Les indicateurs système sont numérotés de −1 à −64. Reportez-vous à l'annexe C, "Indicateurs système" qui fournit la liste des indicateurs système du HP 48 et des numéros correspondants.

Commandes associées : CF, FC?, FC?C, FS?, FS?C

SHOW

Commande d'affichage de variables : Renvoie '$symb_2$', ce qui est équivalent à '$symb_1$' à ceci près que toutes les références implicites à un *nom* de variable sont rendues explicites.

Niveau 2	Niveau 1	→	Niveau 1
'$symb_1$'	'*nom*'	→	'$symb_2$'
'$symb_1$'	{ nom_1 nom_2 ... }	→	'$symb_2$'

Accès clavier : ◀ SYMBOLIC SHOW

Indicateurs : Résultats numériques (−3)

Remarques : Si l'argument de niveau 1 est une liste, SHOW évalue toutes les variables globales de '*symb*$_1$ ' *non* contenues dans cette liste.

Exemple : Si 7 est stocké dans C et 5 est stocké dans D,

```
'X-Y+2*C+3*D' { X Y } SHOW
```

renvoie 'X-Y+14+15'.

Commandes associées : COLCT, EXPAN, ISOL, QUAD

SIDENS

Commande de calcul de la densité intrinsèque du silicium

: Calcule la densité intrinsèque du silicium en fonction de la température, x_T.

{ }

Niveau 1	→	Niveau 1
x_T	→	$x_{densité}$
x_unité	→	*x_1/cm*^{3}
'*symb*'	→	'SIDENS(*symb*)'

Accès clavier : ◀ EQ LIB UTILS SIDEN

Indicateurs : Résultats numériques (−3)

Remarques : Si x_T est un objet-unité, il doit se réduire à une température simple, et la densité est renvoyée sous forme d'objet-unité avec des unités de 1/cm^3.

Si x_T est un nombre réel, ses unités sont censées être K, et la densité est renvoyée sous forme de nombre réel avec des unités implicites de 1/cm^3.

x_T doit être compris entre 0 et 1685 K.

SIGN

Fonction signe : Renvoie le signe d'un argument nombre réel, le signe de la partie numérique d'un argument objet-unité, ou le vecteur-unité dans la direction d'un argument nombre complexe.

{ }

Niveau 1	→	Niveau 1
z_1	→	z_2
x_unité	→	x_{signe}
'*symb*'	→	'SIGN(*symb*)'

Accès clavier :

(MTH) REAL (NXT) SIGN (renvoie le signe d'un nombre)

(MTH) (NXT) CMPL (NXT) SIGN (renvoie le vecteur-unité d'un nombre complexe)

Indicateurs : Résultats numériques (−3)

Remarques : Dans le cas des nombres réels et des objets-unités, la commande renvoie +1 pour des arguments positifs, −1 pour des arguments négatifs et 0 pour l'argument 0.

Pour un argument complexe :

$$\text{SIGN}(x+iy) = \frac{x}{\sqrt{x^2+y^2}} + \frac{iy}{\sqrt{x^2+y^2}}$$

Exemples : 32_ft SIGN renvoie 1.

(1,1) SIGN renvoie (.707106781187,.707106781187).

Commandes associées : ABS, MANT, XPON

SIMU

Commande de traçage simultané : Active et désactive le traçage simultané.

Accès clavier : ◀ PLOT NXT FLAG SIMU

Indicateurs : Traçage simultané (−28)

Remarques : SIMU devient SIMU■ lorsque l'indicateur −28 est armé (traçage simultané activé).

Si le calculateur se trouve en mode de saisie de programme, l'utilisation de la touche de menu copie les numéros des indicateurs AXES, CNCT et SIMU dans la ligne de commande. En appuyant sur ◀ ou ▶, les numéros d'indicateurs y sont copiés en premier, suivis de SF ou de CF.

Commandes associées : AXES, CF, SF

SIN

Fonction analytique sinus : Renvoie le sinus de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	sin *z*
'*symb*'	→	'SIN(*symb*)'
x_unité$_{\text{angulaire}}$	→	sin (*x_unité*$_{\text{angulaire}}$)

Accès clavier : SIN

Indicateurs : Résultats numériques (−3), Mode d'angle (−17, −18)

Remarques : Pour des arguments réels, le mode d'angle en cours détermine les unités du nombre, sauf si des unités angulaires sont spécifiées.

Pour des arguments complexes, $(x + iy) = \sin x \cosh y + i \cos x \sinh y$.

SIN

Si l'argument de SIN est un objet-unité, l'unité angulaire spécifiée remplace le mode d'angle pour déterminer le résultat. L'intégration et la différenciation, en revanche, conservent toujours le mode d'angle. Par conséquent, pour intégrer ou différencier correctement des expressions contenant SIN avec un objet-unité, il faut opter pour le mode radians (car il s'agit d'un mode "neutre").

Commandes associées : ASIN, COS, TAN

SINH

Fonction analytique sinus hyperbolique : Renvoie le sinus hyperbolique de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	sinh *z*
'*symb*'	→	'SINH(*symb*)'

Accès clavier : (MTH) HYP SINH

Indicateurs : Résultats numériques (−3)

Remarques : Pour des arguments complexes, $\sinh(x + iy) = \sinh x \cos y + i \cosh x \sin y$.

Commandes associées : ASINH, COSH, TANH

SINV

Commande de stockage d'inverses : Remplace le contenu de la variable nommée par son inverse.

{}

Niveau 1	→	Niveau 1
'nom'	→	

Accès clavier : ◀ MEMORY ARITH NXT SINV

Indicateurs : Aucun

Remarques : L'objet nommé doit être un nombre, une matrice, un objet algébrique ou un objet-unité. Pour plus d'informations sur les réciproques, voir INV.

Commandes associées : INV, SCONJ, SNEG

SIZE

Commande de taille : Renvoie le nombre de caractères d'une chaîne, le nombre d'éléments d'une liste, les dimensions d'un tableau, le nombre d'objets d'un objet-unité ou d'un objet algébrique, ou les dimensions d'un objet graphique.

SIZE

Niveau 1	→	Niveau 2	Niveau 1
"*chaîne*"	→		*n*
{ *liste* }	→		*n*
[*vecteur*]	→		{ *n* }
[[*matrice*]]	→		{ *n m* }
'*symb*'	→		*n*
grob	→	$\#n_{largeur}$	$\#m_{hauteur}$
PICT	→	$\#n_{largeur}$	$\#m_{hauteur}$
unité_x	→		*n*

Accès clavier :

(←)(CHARS) SIZE

(PRG) LIST ELEM SIZE

(PRG) GROB (NXT) SIZE

Indicateurs : Aucun

Remarques : La taille d'une unité est calculée comme suit : le scalaire (+1), le caractère de soulignement (+1), chaque nom d'unité (+1), l'opérateur ou l'exposant (+1), et chaque préfixe (+2).

Tout type d'objet non énuméré ci-dessus renvoie une valeur de 1.

Commandes associées : CHR, NUM, POS, REPL, SUB

SL

Commande de décalage vers la gauche : Décale un entier binaire d'un bit vers la gauche.

{ }

Niveau 1	→	Niveau 1
$\#n_1$	→	$\#n_2$

Accès clavier : (MTH) BASE (NXT) BIT SL

Indicateurs : Taille de mot entier binaire (−5 à −10), Base d'entier binaire (−11, −12)

Remarques : Le bit de poids fort est décalé vers la gauche et perdu, alors que le bit de poids faible est régénéré sous forme de zéro. SL est équivalent à une multiplication binaire par 2, tronquée à la taille de mot en cours.

Commandes associées : ASR, SLB, SR, SRB

SLB

Commande de décalage d'un octet vers la gauche : Décale un entier binaire d'un octet vers la gauche.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : (MTH) BASE (NXT) BYTE SLB

Indicateurs : Taille d'entier binaire (−5 à −10), Base d'entier binaire (−11, −12)

Remarques : L'octet de poids fort est décalé vers la gauche et perdu, alors que l'octet de poids faible est régénéré sous forme de zéro. SLB est équivalent à une multiplication par 2^8 (ou à l'exécution de SL huit fois), tronquée à la taille du mot en cours.

Commandes associées : ASR, SL, SR, SRB

SLOPEFIELD

Commande de type de tracé SLOPEFIELD : Définit le type de tracé SLOPEFIELD.

Accès clavier : ◀ PLOT NXT 3D PTYPE SLOPE

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est SLOPEFIELD, la commande DRAW trace une représentation sous forme de pente d'une fonction scalaire avec deux variables. SLOPEFIELD requiert des valeurs dans les variables réservées *EQ*, *VPAR* et *PPAR*.

VPAR se présente sous la forme suivante :

{ x_{left} x_{right} y_{near} y_{far} z_{low} z_{high} x_{min} x_{max} y_{min} y_{max} x_{eye} y_{eye} z_{eye} x_{step} y_{step} }

Pour le type de tracé SLOPEFIELD, les éléments de *VPAR* sont utilisés comme suit :

- x_{left} et x_{right} sont des nombres réels qui spécifient la largeur de la vue volumique.
- y_{near} et y_{far} sont des nombres réels qui spécifient la profondeur de la vue volumique.
- z_{low} et z_{high} sont des nombres réels qui spécifient la hauteur de la vue volumique.
- x_{min} et x_{max} ne sont pas utilisés.
- y_{min} et y_{max} ne sont pas utilisés.
- x_{eye}, y_{eye}, et z_{eye} sont des nombres réels qui spécifient le point dans l'espace à partir duquel le graphique est visualisé.
- x_{step} et y_{step} sont des nombres réels qui définissent le nombre de coordonnées x par rapport au nombre de coordonnées y tracées.

Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, qui se présente sous la forme suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé SLOPEFIELD, les éléments de *PPAR* sont employés comme suit :

- (x_{min}, y_{min}) n'est pas utilisé.

- (x_{max}, y_{max}) n'est pas utilisé.
- *indep* est un nom spécifiant la variable indépendante. La valeur par défaut de *indep* est X.
- *res* n'est pas utilisé.
- *axes* n'est pas utilisé.
- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de la commande SLOPEFIELD place le nom de commande SLOPEFIELD dans *ptype*.
- *depend* est un nom spécifiant la variable dépendante. La valeur par défaut est Y.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, TRUTH, WIREFRAME, YSLICE

SNEG

Commande de stockage d'opposé : Remplace le contenu d'une variable par son opposé.

{ }

Niveau 1	→	Niveau 1
'nom'	→	

Accès clavier : ◀ MEMORY ARITH NXT SNEG

Indicateurs : Aucun

Remarques : L'objet nommé doit être un nombre, un tableau, un objet algébrique, un objet-unité ou un objet graphique. Pour plus d'informations sur les opposés, reportez-vous à NEG.

Commandes associées : NEG, SCONJ, SINV

SNRM

Commande de norme spectrale : Renvoie la norme spectrale d'un tableau.

{ }

Niveau 1	→	Niveau 1
[*tableau*]	→	$x_{norme-spect}$

Accès clavier : (MTH) MATR NORM SNRM

Indicateurs : Aucun

Remarque : Pour un vecteur, la norme spectrale est égale à sa longueur euclidienne et, pour une matrice, à sa plus grande valeur singulière.

Commandes associées : ABS, CNRM, COND, RNRM, SRAD, TRACE

SOLVEQN

Commande de lancement du Solver d'équations : Lance le Solver d'équations multiples pour un ensemble d'équations donné.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
n	*m*	*0/1*	→	

Accès clavier : ↶ (EQ LIB) EQLIB SOLVE

Indicateurs : Système d'unités (60), Emploi des unités (61)

Remarques : SOLVEQN configure et lance le Solver approprié pour l'ensemble d'équations spécifié, contournant les catalogues de la bibliothèque d'équations. Il définit *EQ* (et *Mpar* si plusieurs

équations sont à résoudre), définit les options d'unités en fonction des indicateurs 60 et 61, et lance le Solver approprié.

SOLVEQN utilise des numéros de sujet et de titre (niveaux 3 et 2) et une option "PICT" (niveau 1), et ne renvoie rien. Les numéros de sujet et de titre sont énumérés au chapitre 4. Si l'option "PICT" est 0, *PICT* n'est pas affecté, autrement, l'image de l'équation est copiée dans *PICT*.

Commandes associées : EQNLIB, MSOLVER

SORT

Commande de tri par ordre croissant : Trie les éléments d'une liste par ordre croissant.

Niveau 1	→	Niveau 1
{ *liste* }$_1$	→	{ *liste* }$_2$

Accès clavier :

PRG LIST PROC NXT SORT

MTH LIST SORT

Indicateurs : Aucun

Remarques : Les éléments de la liste peuvent être des nombres réels, des chaînes, des listes, des noms, des entiers binaires, ou des objets-unités. Toutefois les éléments de la liste doivent tous être du même type. Les chaînes et les noms sont triés par numéro de code de caractère. Les listes de listes sont triées en fonction du premier élément de chaque liste.

Pour trier dans l'ordre inverse, utilisez SORT REVLIST.

Commandes associées : REVLIST

SPHERE

Commande de mode sphérique : Active le mode sphérique.

Accès clavier :

(MTH) VECTR (NXT) SPHER

(↶)(MODES) ANGL SPHER

Indicateurs : Aucun

Remarques : SPHERE active les indicateurs −15 et −16, et affiche le témoin R∠∠.

En mode sphérique, les vecteurs sont affichés en tant que composants polaires. Par conséquent, un vecteur 3D s'affiche sous la forme [r ∠θ ∠ϕ].

Commandes associées : CYLIN, RECT

SQ

Fonction analytique d'élévation au carré : Renvoie le carré de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	z^2
x_unité	→	x^2_*unité*^{2}
[[*matrice*]]	→	[[*matrice* x *matrice*]]
'*symb*'	→	'SQ(*symb*)'

Accès clavier : (↶)(x^2)

Indicateurs : Résultats numériques (−3)

Remarques : Le carré d'un argument complexe (x, y) est le nombre complexe (x^2-y^2, $2xy$).

Les arguments de matrice doivent être des carrés.

Commandes associées : √, ^

SR

Commande de décalage vers la droite : Décale un entier binaire d'un bit vers la droite.

{ }

Niveau 1	→	Niveau 1
$\#n_1$	→	$\#n_2$

Accès clavier : MTH BASE NXT BIT SR

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : Le bit de poids faible est décalé vers la droite et perdu, alors que le bit de poids fort est régénéré sous forme de zéro. SR est équivalent à une division binaire par 2.

Commandes associées : ASR, SL, SLB, SRB

SRAD

Commande de rayon spectral d'une matrice : Renvoie le rayon spectral d'une matrice carrée.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]$_{n \times n}$	→	$x_{rayonspectral}$

SRAD

Accès clavier : (MTH) MATR NORM SRAD

Indicateurs : Aucun

Remarques : Le rayon spectral d'une matrice est une mesure de la taille de la matrice, et il est égal à la valeur absolue de la plus grande valeur propre de la matrice.

Commandes associées : COND, SNRM, TRACE

SRB

Commande de décalage d'un octet vers la droite : Décale un entier binaire d'un octet vers la droite.

{ }

Niveau 1	→	Niveau 1
#n_1	→	#n_2

Accès clavier : (MTH) BASE (NXT) BYTE SRB

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12)

Remarques : L'octet de poids faible est décalé vers la droite et perdu, alors que l'octet de poids fort est régénéré sous forme de zéro. SRB est équivalent à une division binaire par 2^8 (ou à l'exécution de SR huit fois).

Commandes associées : ASR, SL, SLB, SR

SRECV

Commande de réception série : Lit jusqu'à *n* caractères dans le tampon d'entrée en série et les renvoie sous forme de chaîne, avec un chiffre indiquant si des erreurs se sont produites.

{}

Niveau 1	→	Niveau 2	Niveau 1
n	→	*'chaîne'*	0/1

Accès clavier : ◀ I/O NXT SERIA SRECV

Indicateurs : Unité d'E-S (−33)

Remarques : SRECV n'emploie pas le protocole Kermit.

Si *n* caractères ne sont pas reçus dans le délai spécifié par STIME (par défaut, 10 secondes) SRECV stoppe la réception, renvoyant un 0 au niveau 1 et autant de caractères qu'il en avait été reçu au niveau 2.

Si la sortie de niveau 2 issue de BUFLEN est employée en tant qu'entrée pour SRECV, SRECV ne devra pas attendre la réception de plus de caractères. En revanche, il renvoie les caractères déjà présents dans le tampon d'entrée.

Si vous souhaitez accumuler des octets avant d'exécuter SRECV, vous devez tout d'abord ouvrir le port à l'aide d'OPENIO (s'il n'est pas encore ouvert).

SRECV peut détecter trois types d'erreur pendant la lecture du tampon d'entrée :

- Erreurs de trame et engorgement de l'UART (dans les deux cas, message d'erreur "Receive Error" dans ERRM).
- Des dépassements de capacité du tampon d'entrée (message d'erreur "Receive Buffer Overflow" dans ERRM).
- Des erreurs de parité (message d'erreur "Parity Error" dans ERRM).

SRECV renvoie 0 si une erreur est détectée pendant la lecture du tampon d'entrée, ou 1 si aucune erreur n'est détectée.

SRECV

Les erreurs de parité n'arrêtent pas le flux de données dans le tampon d'entrée. En revanche, si une erreur de parité se produit, SRECV arrête de lire des données après avoir rencontré un caractère présentant une erreur.

En cas d'erreurs de trame, d'engorgement et de dépassement de capacité, tous les caractères reçus par la suite sont ignorés, jusqu'à ce que l'erreur soit supprimée. SRECV ne détecte et ne supprime aucun de ces types d'erreur avant de tenter de lire l'octet où l'erreur a eu lieu. Etant donné qu'avec ces trois erreurs, l'octet où l'erreur s'est produite et tous les octets suivants sont ignorés, le tampon d'entrée sera vide une fois que tous les octets corrects auront été lus. Par conséquent, SRECV détecte et supprime ces erreurs lorsqu'il tente de lire un octet dans un tampon d'entrée vide.

Notez que BUFLEN supprime également les erreurs précitées. Par conséquent, SRECV ne peut pas détecter un dépassement de capacité du tampon d'entrée après l'exécution de BUFLEN, sauf si de nouveaux caractères ont été reçus après celle-ci (provoquant un autre dépassement). En outre, SRECV ne peut pas détecter les erreurs de trame et d'engorgement de l'UART supprimées par BUFLEN. Pour savoir à quel niveau les erreurs sur les données se sont produites, enregistrez le nombre de caractères renvoyé par BUFLEN (ce qui donne le nombre de caractères corrects reçus) parce que, dès que l'erreur est supprimée, de nouveaux caractères peuvent pénétrer dans le tampon d'entrée.

Exemple : Si 10 octets corrects ont été reçus et suivis d'une erreur de trame, une commande SRECV recevant l'ordre de lire 10 octets ne signale *pas* d'erreur. Ce n'est que lorsque SRECV tente de lire l'octet ayant provoqué l'erreur qu'il renvoie un 0. De même, si le tampon d'entrée a été saturé, SRECV n'indique pas d'erreur avant de tenter de lire le premier octet perdu à cette occasion.

Commandes associées : BUFLEN, CLOSEIO, OPENIO, SBRK, STIME, XMIT

SST

Opération d'exécution pas à pas d'un programme : Renvoie et exécute l'étape suivante d'un programme. Si l'étape suivante est un sous-programme, exécute le sous-programme en une étape unique.

Accès clavier : PRG NXT RUN SST

Indicateurs : Aucun

Remarque : SST n'est pas programmable.

Commandes associées : NEXT (opération), SST↓

SST↓

Opération d'exécution pas à pas d'un sous-programme : Renvoie et exécute l'étape suivante d'un programme ou d'un sous-programme. Si l'étape suivante est un sous-programme, renvoie et exécute la première étape de ce sous-programme.

Accès clavier : PRG NXT RUN SST↓

Indicateurs : Aucun

Remarque : SST↓ n'est pas programmable.

Commandes associées : NEXT (opération), SST

START

Commande de début de la structure en boucle finie : Commence les structures en boucle finie START ... NEXT et START ... STEP.

START

	Niveau 2	Niveau 1	→	Niveau 1
START	$x_{début}$	x_{fin}	→	
NEXT			→	
STEP		$x_{incrément}$	→	
STEP		$'symb_{incrément}'$	→	

Accès clavier : (PRG) BRCH START START

Indicateurs : Aucun

Remarques : Les *structures en boucle finie* exécutent une commande ou une séquence de commandes un nombre de fois spécifié.

- START ... NEXT exécute une partie d'un programme un nombre de fois spécifié. La syntaxe est la suivante :

  $x_{début}$ x_{fin} START *clause-de-la-boucle* NEXT

  START prend deux nombres ($x_{début}$ et x_{fin}) dans la pile et les stocke en tant que valeurs de début et de fin d'un compteur de boucle. La clause de la boucle est ensuite exécutée. NEXT incrémente le compteur de 1 et vérifie si sa valeur est inférieure ou égale à x_{fin}. Si tel est le cas, la clause de la boucle est réexécutée. Notez que la clause de la boucle est toujours exécutée au moins une fois.

- START ... STEP fonctionne exactement comme START ... NEXT, à ceci près qu'il peut employer une valeur d'incrément différente de 1. La syntaxe est la suivante :

  $x_{début}$ x_{fin} START *clause-de-la-boucle* $x_{incrément}$ STEP

  START prend deux nombres ($x_{début}$ et x_{fin}) dans la pile et les stocke en tant que valeurs de début et de fin du compteur de boucle. La clause de la boucle est ensuite exécutée. STEP prend $x_{incrément}$ dans la pile et incrémente le compteur de cette valeur. Si l'argument de STEP est une expression algébrique ou un nom, il est automatiquement évalué en nombre.

  La valeur de l'incrément peut être positive ou négative :

  - Si elle est positive, la boucle est réexécutée lorsque le compteur est inférieur ou égal à x_{fin}.

- Si elle est négative, la boucle est réexécutée lorsque le compteur est supérieur ou égal à x_{nn}.

Commandes associées : FOR, NEXT, STEP

STD

Commande de mode Standard : Définit le mode standard pour l'affichage des nombres.

Accès clavier : ↩ MODES FMT STD

Indicateurs : Aucun

Remarques : L'exécution de STD produit le même effet que de désarmer les indicateurs −49 et −50.

Le format Standard (ANSI Minimal BASIC Standard X3J2) produit les résultats suivants lors de l'affichage ou de l'impression d'un nombre :

- Les nombres pouvant être représentés exactement sous forme d'entiers à 12 chiffres ou moins sont affichés sans séparateur décimal ni exposant. Le zéro est affiché sous la forme 0.
- Les nombres pouvant être représentés exactement avec 12 chiffres ou moins, mais pas sous forme d'entiers, sont affichés avec un séparateur décimal mais sans exposant. Les zéros initiaux, à gauche du séparateur décimal et les zéros finals, à droite du séparateur décimal, sont omis.
- Tous les autres nombres sont affichés en notation scientifique (voir SCI) avec à la fois un séparateur décimal (avec un nombre à gauche) et un exposant (comportant de 1 à 3 chiffres). Il n'y a ni zéro initiaux, ni zéros finals.

Dans les objets algébriques, les entiers inférieurs à 10^3 sont toujours affichés en mode Standard.

Exemple : Le tableau suivant montre des exemples de nombres affichés en mode Standard :

STD

Nombre	Affichage	Représentable avec 12 chiffres ?
10^{11}	100000000000	Oui (entier)
10^{12}	1.E12	Non
10^{-11}	.000000000001	Oui
1.2×10^{-11}	1.23E-11	Non
12.345	12.345	Oui

Commandes associées : ENG, FIX, SCI

STEP

Commande STEP : Définit la valeur de l'incrément (pas), et termine la structure en boucle finie.

Reportez-vous aux rubriques des commandes FOR et START pour plus d'informations sur la syntaxe.

Accès clavier :

(PRG) BRCH FOR STEP

(PRG) BRCH START STEP

Remarque : Reportez-vous aux rubriques FOR et START pour un complément d'information.

Commandes associées : FOR, NEXT, START

STEQ

Commande de stockage dans EQ : Stocke un objet dans la variable réservée *EQ*, dans le répertoire en cours.

{ }

Niveau 1	→	Niveau 1
obj	→	

Accès clavier : Cette commande doit être tapée, mais vous pouvez stocker un objet dans *EQ* avec :

◀ PLOT ◀ EQ

◀ PLOT NXT 3D ◀ EQ

Indicateurs : Aucun

Commandes associées : RCEQ

STIME

Commande de définition de la temporisation des transferts en série : Spécifie le délai d'attente pour la réception série (SRECV) et la transmission série (XMIT).

{ }

Niveau 1	→	Niveau 1
x_{secondes}	→	
0	→	

Accès clavier : ◀ I/O NXT SERIA STIME

Indicateurs : Aucun

STIME

Remarques : La valeur de *x* est interprétée comme positive de 0 à 25,4 secondes. Si aucune valeur n'est fournie, le délai par défaut est de 10 secondes. Si *x* est 0, il n'y a pas de temporisation, et l'unité reste donc en attente indéfiniment, ce qui risque de décharger les piles.

STIME n'est pas utilisé pour la temporisation Kermit .

Commandes associées : BUFLEN, CLOSEIO, SBRK, SRECV, XMIT

STO

Commande de stockage : Stocke un objet dans une variable ou un objet spécifié.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	*'nom'*	→	
grob	*PICT*	→	
obj	$:n_{port}$ $:nom_{sauvegarde}$	→	
obj	*'nom(index)'*	→	
sauvegarde	n_{port}	→	
bibliothèque	n_{port}	→	
bibliothèque	$:n_{port}$ $:n_{bibliothèque}$	→	

Accès clavier : STO

Indicateurs : Aucun

Remarques : Le stockage d'un objet graphique dans *PICT* en fait l'objet graphique en cours.

Pour créer un objet-sauvegarde, enregistrez *obj* dans l'emplacement de sauvegarde voulu (identifié sous la forme $:n_{port}:nom_{sauvegarde}$). STO n'"écrase" pas d'objet de sauvegarde existant.

Pour stocker des objets-sauvegarde et des objets-bibliothèque, spécifiez un numéro de port (de 0 à 33). Les ports 1 et 2 doivent être

configurés sous forme de RAM indépendante, étant donné que les objets-sauvegarde et les objets-bibliothèque peuvent uniquement être stockés dans ce type de mémoire (voir la rubrique FREE).

Pour utiliser un objet-bibliothèque, l'objet doit se trouver dans un port et il doit être associé. Un objet-bibliothèque d'une carte d'application (ROM) est automatiquement placé dans un port (de 1 à 33), mais un objet-bibliothèque copié dans la RAM (par l'intermédiaire de PC Link, par exemple) doit être stocké dans un port à l'aide de STO.

Une fois l'objet-bibliothèque stocké dans un port, il faut l'associer à son répertoire pour pouvoir l'utiliser. Pour rendre "associable" une bibliothèque stockée, éteignez et rallumez le calculateur (reportez-vous à la rubrique ATTACH). Cette opération (stocker un objet-bibliothèque, puis éteindre et rallumer le calculateur) cause une *interruption du système*, ce qui efface la pile, la pile LAST et toutes les variables locales, et affiche le menu MATH.

STO permet également de remplacer un seul élément dans un tableau (ou une liste) stocké dans une variable. Spécifiez la variable au niveau 1 sous la forme '*nom*(*index*)', ce qui constitue une fonction-utilisateur ayant *index* pour argument. L' *index* peut être *n* ou *n,m*, où *n* spécifie la position de la ligne dans un vecteur ou une liste, et *n,m* spécifie la position de la ligne et de la colonne dans une matrice.

Exemple : `'A+B+C+D' 'SUMAD' STO` stocke l'expression A+B+C+D dans la variable *SUMAD*.

`5 'A(3)' STO` stocke l'entier 5 dans le troisième élément d'une liste ou un vecteur *A*.

`2 'A(3,5)' STO` stocke l'entier 2 dans l'élément de la troisième ligne et cinquième colonne de la matrice *A*.

Commandes associées : DEFINE, RCL, →

STOALARM

Commande de stockage des alarmes : Stocke une alarme dans la liste des alarmes système et renvoie son numéro d'index.

Niveau 1	→	Niveau 1
x_{heure}	→	n_{index}
{ *date heure* }	→	n_{index}
{ *date heure* obj_{action} }	→	n_{index}
{ *date heure* obj_{action} $x_{répéter}$ }	→	n_{index}

Accès clavier : ◀ TIME ALRM STOAL

Indicateurs : Format de date (−42), Alarmes répétitives non reprogrammées (−43), Alarmes ayant reçu un accusé de réception sauvegardées (−44)

Remarques : Si l'argument est un nombre réel x_{heure}, la date de l'alarme est par défaut la date système en cours .

Si obj_{action} est une chaîne, l'alarme est une alarme de rendez-vous, et la chaîne constitue le message d'alarme. Si obj_{action} est un quelconque autre type d'objet, l'alarme est une alarme de contrôle, et l'objet est exécuté lorsqu'elle arrive à échéance.

$x_{répéter}$ est l'intervalle de répétition exprimé en tops d'horloge, où 8192 tops d'horloge égalent 1 seconde.

n_{index} est un entier réel identifiant l'alarme en fonction de sa position chronologique dans la liste d'alarmes système.

Exemple : Avec l'indicateur −42 désarmé, la commande suivante :

```
{ 11.06 15.2530 RUN 491520 } STOALARM
```

définit une alarme de contrôle répétitive pour le 6 novembre de l'année spécifiée en cours, à 3:25:30 de l'après-midi. L'action de l'alarme consiste à exécuter la variable *RUN*. L'intervalle de répétition est de 491520 tops d'horloge (1 minute).

Commandes associées : DELALARM, FINDALARM, RCLALARM

STOF

Commande de stockage des indicateurs : Définit les états des indicateurs système, ou des indicateurs système et utilisateur.

Niveau 1	→	Niveau 1
$\#n_{système}$	→	
{ $\#n_{système}$ $\#n_{utilisateur}$ }	→	

Accès clavier : ◀ MODES FLAG NXT STOF

Indicateurs : Taille de mot entier binaire (−5 à −10)

La taille de mot en cours doit être de 64 bits (la taille de mot par défaut) pour stocker tous les indicateurs. Par exemple, l'exécution de STOF avec un entier binaire de 32 bits stocke uniquement les indicateurs −1 à −32 et *désarme* les autres indicateurs système.

Remarques : Avec l'argument $\#n_{système}$, STOF définit uniquement les états des indicateurs système (−1 à −64). Avec l'argument { $\#n_{système}$ $\#n_{utilisateur}$ }, STOF définit à la fois les états des indicateurs système et utilisateur.

Un bit de valeur 1 arme l'indicateur correspondant, un bit de valeur 0 désarme l'indicateur correspondant. Le bit le plus à droite (de poids faible) de $\#n_{système}$ et $\#n_{utilisateur}$ correspond respectivement aux états de l'indicateur système −1 et de l'indicateur utilisateur +1. Si $\#n_{système}$ ou $\#n_{utilisateur}$ contient moins de 64 bits, les bits de poids fort non spécifiés sont considérés comme ayant la valeur 0.

STOF peut conserver l'état des indicateurs avant qu'un programme ne s'exécute et ne les modifie. RCLF peut ensuite les rappeler après l'exécution du programme.

Commandes associées : RCLF

STOKEYS

Commande de stockage de l'affectation des touches : Définit plusieurs touches sur le clavier utilisateur en leur affectant des objets.

Niveau 1	→	Niveau 1
{ obj_1 $x_{touche1}$ ... obj_n $x_{touchen}$ }	→	
{ S obj_1 $x_{touche1}$ ... obj_n $x_{touchen}$ }	→	
'S'	→	

Accès clavier : ◀ (MODES) KEYS STOK

Indicateurs : Verrouillage mode Utilisateur (−61), Mode Utilisateur (−62).

Remarques : x_{touche} est un nombre réel ayant la forme *lc.s*, spécifiant la touche par son numéro de ligne *l*, son numéro de colonne *c* et son codeshift *s* (pour les valeurs admissibles, reportez-vous à ASN.)

Le paramètre de liste ou l'argument S initial facultatif rend à toutes les touches dépourvues de définition utilisateur leur affectation de *touche standard* sur le clavier utilisateur. Cette particularité ne s'applique que lorsque toutes les définitions de touche standard ont été supprimées (pour le clavier utilisateur) par la commande 'S' DELKEYS (voir DELKEYS).

Si l'argument *obj* est le nom 'SKEY', la définition standard de la touche est restaurée.

Commandes associées : ASN, DELKEYS, RCLKEYS

STO+

Commande d'addition avec un objet stocké : Ajoute un nombre ou un autre objet au contenu d'une variable spécifiée.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	'*nom*'	→	
'*nom*'	*obj*	→	

Accès clavier : ↩ MEMORY ARITH STO+

Indicateurs : Aucun

Remarques : L'objet présent dans la pile et l'objet stocké dans la variable doivent être compatibles pour l'addition. STO+ peut additionner toute combinaison d'objets se prêtant à cette opération dans la pile (reportez-vous à la rubrique +).

Utiliser STO+ pour ajouter deux tableaux (où *obj* est un tableau et *nom* est le nom global d'un tableau) requiert moins de mémoire que de recourir à la pile dans le même but.

Commandes associées : STO–, STO*, STO/, +

STO–

Commande de soustraction avec un objet stocké : Calcule la différence entre un nombre (ou un autre objet) et le contenu d'une variable spécifiée, et stocke la nouvelle valeur dans la variable spécifiée.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	'*nom*'	→	
'*nom*'	*obj*	→	

STO−

Accès clavier : ◀ MEMORY ARITH STO-

Indicateurs : Aucun

Remarques : L'objet présent dans la pile et l'objet stocké dans la variable doivent être compatibles pour la soustraction. STO− peut soustraire toute combinaison d'objets se prêtant à cette opération dans la pile (reportez-vous à la rubrique −).

Utiliser STO− pour soustraire deux tableaux (où *obj* est un tableau et *nom* est le nom global d'un tableau) requiert moins de mémoire que de recourir à la pile dans le même but.

Commandes associées : STO+, STO*, STO/, −

STO*

Commande de multiplication avec un objet stocké : Multiplie le contenu d'une variable spécifiée par un nombre ou un autre objet.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	'*nom*'	→	
'*nom*'	*obj*	→	

Accès clavier : ◀ MEMORY ARITH STO*

Indicateurs : Aucun

Remarques : L'objet présent dans la pile et l'objet stocké dans la variable doivent être compatibles pour la multiplication. Lors de la multiplication de deux tableaux, le résultat dépend de l'ordre des arguments. Le nouvel objet de la variable nommée est le tableau de niveau 2 multiplié par le tableau de niveau 1. Les tableaux doivent être conformes pour la multiplication.

Utiliser STO* pour multiplier deux tableaux, ou un nombre et un tableau (où *obj* est un tableau ou un nombre et *nom* est le nom global

d'un tableau) requiert moins de mémoire que de recourir à la pile dans le même but.

Commandes associées : STO+, STO−, STO/, *

STO/

Commande de division avec un objet stocké : Calcule le quotient d'un nombre (ou d'un autre objet) et du contenu d'une variable spécifiée, et stocke la nouvelle valeur dans la variable spécifiée.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj	'*nom*'	→	
'*nom*'	*obj*	→	

Accès clavier : ↩ MEMORY ARITH STO/

Indicateurs : Aucun

Remarques : Le nouvel objet de la variable spécifiée est l'objet de niveau 2 divisé par l'objet de niveau 1.

L'objet présent dans la pile et l'objet stocké dans la variable doivent être compatibles pour la division. Si les deux objets sont des tableaux, le diviseur (niveau 1) doit être une matrice carrée, et le dividende (niveau 2) doit comporter le même nombre de colonnes que le diviseur.

Utiliser STO/ pour diviser un tableau par un autre ou pour diviser un tableau par un nombre (où *obj* est un tableau ou un nombre et *nom* est le nom global d'un tableau) requiert moins de mémoire que de recourir à la pile dans le même but.

Commandes associées : STO+, STO−, STO*, /

STOΣ

Commande de stockage statistique : Stocke *obj* dans la variable réservée ΣDAT.

{ }

Niveau 1	→	Niveau 1
obj	→	

Accès clavier : Cette commande doit être tapée, mais vous pouvez stocker un objet dans ΣDAT en utilisant au choix :

◀ PLOT NXT STAT ◀ ΣDAT ou

◀ STAT DATA ◀ ΣDAT

Indicateurs : Aucun

Remarques : STOΣ accepte tout objet et le stocke dans ΣDAT. Toutefois, si l'objet n'est pas une matrice ou le nom d'une variable contenant une matrice, une erreur `Invalid Σ Data` se produit lors de l'exécution ultérieure d'une commande statistique.

Commandes associées : CLΣ, RCLΣ, Σ+, Σ−

STREAM

Commande d'application récursive : Place les deux premiers éléments de la liste dans la pile et exécute *obj*. Place alors l'élément suivant (éventuel) dans la pile et réexécute *obj* en employant le résultat précédent et le nouvel élément. Répète ce processus jusqu'à ce que la liste soit épuisée, et renvoie le résultat final.

Niveau 2	Niveau 1	→	Niveau 1
{ *liste* }	*obj*	→	*resultat*

Accès clavier : PRG LIST PROC STREA

Indicateurs : Aucun

Remarques : STREAM est en principe conçu pour qu'*obj* soit un programme ou une commande nécessitant deux arguments et renvoyant un seul résultat.

Exemples : ‹ 1 2 3 4 5 › «*» STREAM renvoie 120.

«+» STREAM est équivalent à ΣLIST.

Commandes associées : DOSUBS

STR→

Commande d'évaluation d'une chaîne : Evalue le texte d'une chaîne comme si le texte était entré à partir de la ligne de commande.

{ }

Niveau 1	→	Niveau 1
"obj"	→	*objet évalué*

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : OBJ→ inclut également cette fonction. STR→ est fournie à des fins de compatibilité avec le HP 28S.

Commandes associées : ARRY→, DTAG, EQ→, LIST→, OBJ→, →STR

→STR

Commande de conversion d'un objet en chaîne : Confère à un objet quelconque une forme de chaîne.

Niveau 1	→	Niveau 1
obj	→	"*obj*"

Accès clavier :

◀ CHARS NXT →STR

PRG TYPE →STR

Indicateurs : Taille de mot entier binaire (−5 à −10), Base entier binaire (−11, −12), Format d'affichage numérique (−49, −50)

Remarques : La forme interne en pleine précision d'un nombre n'est pas nécessairement représentée dans la chaîne de résultat. Pour faire en sorte que →STR conserve la pleine précision d'un nombre, sélectionnez le format d'affichage Standard ou une taille de mot de 64 bits (ou les deux) avant d'exécuter →STR.

La chaîne de résultat inclut l'objet entier, même si la forme affichée de l'objet est trop grande pour tenir dans l'affichage.

Si l'objet-argument est affiché normalement sur plusieurs lignes, la chaîne de résultat contient des caractères de retour à la ligne (le caractère 10) à la fin de chaque ligne. Les retours à la ligne sont signalés par le caractère ▪.

Si l'objet-argument est déjà une chaîne, →STR renvoie la chaîne.

Exemple : →STR peut créer des affichages particuliers pour libeller le résultat d'un programme ou fournir des messages pour la saisie. La séquence

```
"Resultat = " SWAP →STR + 1 DISP 1 FREEZE
```

affiche `Résultat =` *objet* de la ligne 1 de l'affichage, où *objet* est la représentation sous forme de chaîne d'un objet extrait du niveau 1.

Commandes associées : →ARRY, →LIST, STR→, →TAG, →UNIT

STWS

Commande de définition de la taille de mot : Définit la taille de mot entier binaire en cours comme étant *n* bits, où *n* est une valeur comprise entre 1 et 64 (par défaut 64).

{ }

Niveau 1	→	Niveau 1
n	→	
#*n*	→	

Accès clavier : (MTH) BASE (NXT) STWS

Indicateurs : Taille de mot entier binaire (−5 à −10), Base d'entier binaire (−11, −12)

Remarques : Les valeurs de *n* inférieures à 1 ou supérieures à 64 sont considérées respectivement comme étant 1 ou 64.

Si la taille de mot est inférieure à un entier entré dans la ligne de commande, les bits de poids *fort* ne sont pas affichés lors de l'entrée. Les bits tronqués sont cependant présents de manière interne (sauf s'ils dépassent 64), mais ils ne sont pas employés pour des calculs, et sont perdus lorsqu'une commande emploie l'entrée binaire comme argument.

Les résultats dépassant la taille de mot donnée sont eux aussi tronqués à la taille de mot.

Commandes associées : BIN, DEC, HEX, OCT, RCWS

SUB

Commande de sous-ensemble : Renvoie la partie d'une chaîne ou d'une liste définie par des positions spécifiées, ou renvoie la partie rectangulaire d'un objet graphique ou de *PICT* définie par des coordonnées en pixels.

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
[[*matrice*]]$_1$	$n_{posdedébut}$	$n_{posdefin}$	→	[[*matrice*]]$_2$
[[*matrice*]]$_1$	{ n_{ligne} n_{col} }	$n_{posdefin}$	→	[[*matrice*]]$_2$
[[*matrice*]]$_1$	$n_{posdedébut}$	{ n_{ligne} n_{col} }	→	[[*matrice*]]$_2$
[[*matrice*]]$_1$	{ n_{ligne} n_{col} }	{ n_{ligne} n_{col} }	→	[[*matrice*]]$_2$
"*chaîne*$_{cible}$"	$n_{posdedébut}$	$n_{posdefin}$	→	"*chaîne*$_{résult}$"
{ *liste*$_{cible}$ }	$n_{posdedébut}$	$n_{posdefin}$	→	{ *liste*$_{résult}$ }
grob$_{cible}$	{ #n_1 #m_1 }	{ #n_2 #m_2 }	→	*grob*$_{résult}$
grob$_{cible}$	(x_1 , y_1)	(x_2 , y_2)	→	*grob*$_{résult}$
PICT	{ #n_1 #m_1 }	{ #n_2 #m_2 }	→	*grob*$_{résult}$
PICT	(x_1 , y_1)	(x_2 , y_2)	→	*grob*$_{résult}$

Accès clavier :

◀ CHARS SUB

PRG LIST SUB

PRG GROB SUB

MTH MATR MAKE NXT SUB

Indicateurs : Aucun

Remarques : Si $n_{posdefin}$ est inférieur à $n_{posdedébut}$, SUB renvoie une chaîne ou une liste vide. Les valeurs de *n* inférieures à 1 sont traitées comme 1 ; les valeurs de *n* dépassant la longueur de la chaîne ou de la liste sont réduites à cette longueur.

Pour les objets graphiques, une coordonnée en unités-utilisateur inférieure à la coordonnée en unités-utilisateur minimale de l'objet graphique est traitée comme ce minimum. Une coordonnée en pixels ou en unités-utilisateur supérieure à la coordonnée en pixels ou en

unités-utilisateur maximale de l'objet graphique est traitée comme ce maximum.

Exemples : `{ A B C D E } 2 4 SUB` renvoie `{ B C D }`.

`"ABCDE" 0 10 SUB` renvoie `"ABCDE"`.

`PICT { # 10d # 20d } { # 20d # 40d } SUB` renvoie `GRAPHIC 11 x 21`.

Commandes associées : CHR, GOR, GXOR, NUM, POS, REPL, SIZE

SVD

Commande de décomposition en valeurs singulières : Renvoie la décomposition en valeurs singulières d'une matrice $m \times n$.

{ }

Niveau 1	→	Niveau 3	Niveau 2	Niveau 1
[[*matrice*]]$_A$	→	[[*matrice*]]$_U$	[[*matrice*]]$_V$	[*vecteur*]$_S$

Accès clavier : (MTH) MATR FACTR SVD

Indicateurs : Aucun

Remarques : SVD décompose A en 2 matrices et un vecteur. U est une matrice orthogonale $m \times m$, V est une matrice orthogonale $n \times n$ et S est un vecteur réel, tel que $A = U \times diag(S) \times V$. S a la longueur MIN(m,n) et contient les valeurs singulières de A par ordre décroissant. La matrice $diag(S)$ est une matrice diagonale $m \times n$ contenant les valeurs singulières S.

Les résultats calculés doivent être réduits (dans les limites de la précision liée aux calculs) :

$$\frac{|A - U \cdot \mathrm{diag}(S) \cdot V|}{\min(m,n) \cdot |A|}$$

où diag(S) est la matrice diagonale $m \times n$ contenant les valeurs singulières S.

SVD

Commandes associées : DIAG→, MIN, SVL

SVL

Commande de calcul des valeurs singulières : Renvoie les valeurs singulières d'une matrice $m \times n$.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]	→	[*vecteur*]

Accès clavier : (MTH) MATR FACTR (NXT) SVL

Indicateurs : Aucun

Remarques : SVL renvoie un vecteur réel contenant les valeurs singulières d'une matrice $m \times n$ par ordre décroissant. Le vecteur a la longueur MIN(m,n).

Commandes associées : MIN, SVD

SWAP

Commande de permutation d'objets : Permute les deux premiers objets présents dans la pile.

Niveau 2	Niveau 1	→	Niveau 2	Niveau 1
obj_1	obj_2	→	obj_2	obj_1

Accès clavier : (←) (SWAP)

Indicateurs : Aucun

Commandes associées : DUP, DUPN, DUP2, OVER, PICK, ROLL, ROLLD, ROT

SYSEVAL

Commande d'évaluation des objets système : Evalue des objets système non nommés spécifiés par leur adresse en mémoire.

{ }

Niveau 1	→	Niveau 1
#$n_{adresse}$	→	

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarques : Une utilisation de SYSEVAL avec des adresses aléatoires peut altérer la mémoire.

Exemple : Affichez la lettre de la version d'un HP 48 en exécutant `#30794h SYSEVAL` . La version A, par exemple, affiche `"HPHP48-A"` .

Commandes associées : EVAL, LIBEVAL

%T

Fonction de pourcentage du total : Renvoie le pourcentage que représente l'argument de niveau 1 par rapport à celui de niveau 2.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	*100y/x*
x	*'symb'*	→	*'%T(x, symb)'*
'symb'	*x*	→	*'%T(symb, x)'*
'symb$_1$'	*'symb$_2$'*	→	*'%T(symb$_1$, symb$_2$)'*
x_unitée$_1$	*y_unité$_2$*	→	*100y_unité$_2$/x_unité$_1$*
x_unité	*'symb'*	→	*'%T(x_unité, symb)'*
'symb'	*x_unité*	→	*'%T(symb, x_unité)'*

Accès clavier : (MTH) REAL %T

Indicateurs : Résultats numériques (−3)

Remarques : Si les deux arguments sont des objets-unités, les unités doivent être cohérentes entre elles.

Les dimensions d'un objet-unité sont éliminées du résultat, ***mais les unités font partie du calcul.***

Pour un complément d'informations sur l'utilisation des unités de température avec des fonctions arithmétiques, reportez-vous à la rubrique +.

Exemple : `1_m 500_cm %T` renvoie `500`, parce que 500 cm représentent 500 % de 1 m.

`100_K 50_K %T` renvoie `50`.

Commandes associées : %, %CH

→TAG

Commande de création d'un objet identifié : Combine des objets des niveaux 1 et 2 pour créer un objet identifié (libellé).

Niveau 2	Niveau 1	→	Niveau 1
obj	"*id*"	→	:*id*:*obj*
obj	'*nom*'	→	:*nom*:*obj*
obj	*x*	→	:*x*:*obj*

Accès clavier : PRG TYPE →TAG

Indicateurs : Aucun

Remarques : L'argument "*id*" est une chaîne de moins de 256 caractères.

Commandes associées : →ARRY, DTAG, →LIST, OBJ→, →STR, →UNIT

TAIL

Commande d'extraction des derniers éléments : Renvoie tous les éléments d'une liste ou d'une chaîne à l'exception du premier.

Niveau 1	→	Niveau 1
{ obj_1 ... obj_n }	→	{ obj_2 ... obj_n }
"$chaîne_1$"	→	"$chaîne_2$"

Accès clavier :

PRG LIST ELEM NXT TAIL

◀ CHARS NXT TAIL

TAIL

Indicateurs : Aucun

Exemple : "cas" TAIL renvoie "as".

Commandes associées : HEAD

TAN

Fonction analytique tangente : Renvoie la tangente de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	tan *z*
'*symb*'	→	'TAN(*symb*)'
x_unité$_{\text{angulaire}}$	→	tan (*x_unité*$_{\text{angulaire}}$)

Accès clavier : TAN

Indicateurs : Résultats numériques (−3), mode d'angle (−17, −18), exception de résultat infini (−22)

Remarques : Pour des arguments réels, le mode d'angle en cours détermine l'interprétation du nombre en tant qu'angle, sauf si les unités angulaires sont spécifiées.

Pour un argument réel constituant un entier impair multiple de 90 en mode Degrés, une exception "Infinite Result" se produit. Si l'indicateur −22 est armé (pas d'erreur), le signe du résultat (MAXR) correspond à celui de l'argument.

Pour des arguments complexes,

$$\tan(x+iy) = \frac{(\sin x)(\cos x) + i(\sinh y)(\cosh y)}{\sinh^2 y + \cos^2 x}$$

Si l'argument de TAN est un objet-unité, l'unité angulaire spécifiée remplace le mode d'angle pour déterminer le résultat. L'intégration et la différenciation, en revanche, observent toujours le mode d'angle. Par conséquent, pour intégrer ou différencier correctement des

expressions contenant TAN avec un objet-unité, il faut opter pour le mode radians (car il s'agit d'un mode "neutre").

Commandes associées : ATAN, COS, SIN

TANH

Fonction analytique tangente hyperbolique : Renvoie la tangente hyperbolique de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	tanh *z*
'*symb*'	→	'TANH(*symb*)'

Accès clavier : (MTH) HYP TANH

Indicateurs : Résultats numériques (−3)

Remarque : Pour des arguments complexes,

$$\tanh(x+iy) = \frac{\sinh 2x + i\,\sin 2y}{\cosh 2x + \cos 2y}$$

Commandes associées : ATANH, COSH, SINH

TAYLR

Commande de calcul du polynôme de Taylor : Calcule le polynôme de Taylor de *n*ième ordre de '*symb*' dans la variable *global*.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
'*symb*'	'*global*'	n_{ordre}	→	'$symb_{Taylor}$'

Accès clavier : ◀ SYMBOLIC TAYLR

Indicateurs : Aucun

Remarques : Le polynôme est calculé au point *global* = 0 (appelé série de MacLaurin).

TAYLR renvoie toujours un résultat symbolique, quel que soit l'état de l'indicateur de résultats numériques (−3).

Exemple : La séquence de commande `'1+SIN(X)^2' 'X' 5 TAYLR` renvoie `'1+X^2-8/4!*X^4'`.

Commandes associées : ∂, $\int$, Σ

TDELTA

Fonction de calcul d'écart de température : Calcule une modification de la température.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	x_{delta}
x_unité1	*y_unité2*	→	$x\_unité1_{delta}$
x_unité	*'symb'*	→	'TDELTA(*x_unité,symb*)'
'symb'	*y_unité*	→	'TDELTA(*symb,y_unité*)'
$'symb_1'$	$'symb_2'$	→	'TDELTA($symb_1$,$symb_2$)'

Accès clavier : ◀ EQ LIB UTILS NXT TDELT

Indicateurs : Résultats numériques (−3)

Remarques : TDELTA soustrait deux points sur une échelle de température, produisant un *incrément* de température (non une température réelle). *x* ou *x_unité1* est la température finale, et *y* ou *y_unité2* est la température initiale. Si des objets-unités sont donnés, l'incrément est renvoyé sous forme d'objet-unité ayant les mêmes unités que *x_unité1*. Si des nombres réels sont donnés, l'incrément est renvoyé en tant que nombre réel.

Commande associée : TINC

TEACH

Fonction de chargement du répertoire EXAMPLES : Crée un sous-répertoire EXAMPLES (EXAM) dans le répertoire HOME et y charge des exemples de programmation, de tracé de graphiques et de résolution d'équations intégrés au HP 48.

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

Remarque : Les éléments stockés dans le sous-répertoire EXAMPLES sont supprimés lorsque CLTEACH est exécuté.

Commande associée : CLTEACH

TEXT

Commande d'affichage de la pile : Affiche la pile.

Accès clavier : PRG NXT OUT TEXT

Indicateurs : Aucun

Remarque : TEXT permet de basculer de l'affichage graphique à l'affichage de la pile. TEXT ne met pas à jour l'affichage de la pile.

Exemple : La séquence de commandes DRAW 5 WAIT TEXT sélectionne l'affichage graphique et trace le contenu de la variable réservée *EQ* (ou de la variable réservée *ΣDAT*), puis, au bout de 5 secondes, rappelle l'affichage de la pile.

Commandes associées : PICTURE, PVIEW

THEN

Commande THEN : Débute la clause vraie dans une structure conditionnelle ou d'interception d'erreurs.

Reportez-vous aux rubriques IF et IFERR pour plus d'informations sur la syntaxe.

Accès clavier :

PRG NXT ERROR IFERR THEN

PRG BRCH CASE THEN

PRG BRCH IF THEN

Remarque : Reportez-vous aux rubriques IF et IFERR pour un complément d'informations.

Commandes associées : CASE, ELSE, END, IF, IFERR

TICKS

Commande heure système : Renvoie l'heure système sous la forme d'un entier binaire, en unité de 1/8192 ième de seconde.

Niveau 1	→	Niveau 1
	→	#n_{heure}

Accès clavier : ◀ TIME TICKS

Indicateurs : Aucun

Remarque : TICKS permet de calculer le temps écoulé.

Exemple : Si le résultat d'une exécution antérieure de TICKS se trouve au niveau 1, `TICKS SWAP - B→R 8192 /` renvoie un nombre réel dont la valeur correspond au temps écoulé en secondes entre les deux exécutions.

Commande associée : TIME

TIME

Commande heure système : Renvoie l'heure système sous la forme HH.MMSSs.

Niveau 1	→	Niveau 1
	→	*heure*

Accès clavier : ◀ TIME TIME

Indicateurs : Aucun

Remarques : *heure* se présente sous la forme *HH.MMSSs*, où *HH* représente les heures, *MM* les minutes, *SS* les secondes et *s* correspond à zéro ou plusieurs chiffres (autant qu'en permet le mode

TIME

d'affichage en cours) représentant des fractions de secondes. L'*heure* est toujours renvoyée en format 24 heures, quel que soit l'état de l'indicateur du format de l'horloge (−41).

Commandes associées : DATE, TICKS, TSTR

→TIME

Commande de définition de l'heure système : Définit l'heure du système.

{ }

Niveau 1	→	Niveau 1
heure	→	

Accès clavier : ↩ TIME →TIM

Indicateurs : Aucun

Remarques : *Heure* doit être au format *HH.MMSSs*, où *HH* représente les heures, *MM* les minutes, *SS* les secondes et *s* correspond à zéro ou plusieurs chiffres (autant qu'en permet le mode d'affichage en cours) représentant des fractions de secondes. *Heure* doit être au format 24 heures.

Exemple : `13.3341 →TIME` définit l'heure système à 1:33:41 de l'après-midi.

Commandes associées : CLKADJ, →DATE

TINC

Commande d'incrément de température : Calcule un incrément de température.

{ }

Niveau 2	Niveau 1	→	Niveau 1
$x_{initial}$	y_{delta}	→	x_{final}
x_unité1	*y_unité2*$_{delta}$	→	*x_unité1*$_{final}$
x_unité	'*symb*'	→	'TINC(*x_unité,symb*)'
'*symb*'	*y_unité*$_{delta}$	→	'TINC(*symb,y_unité*$_{delta}$)'
'*symb*$_1$'	'*symb*$_2$'	→	'TINC(*symb*$_1$,*symb*$_2$')'

Accès clavier : ◀ EQ LIB UTILS NXT TINC

Indicateurs : Résultats numériques (−3)

Remarques : TINC ajoute un *incrément* de température (pas une température réelle) à un point situé sur une échelle de température. Utilisez un incrément négatif pour soustraire l'incrément de la température. $x_{initial}$ ou *x_unité1* est la température initiale, et y_{delta} ou *y_unité2*$_{delta}$ est l'incrément de température. La température renvoyée est la température finale résultante. Si des objets-unités sont donnés, la température finale est renvoyée en tant qu'objet-unité avec les mêmes unités que *x_unité1*. Si des nombres réels sont donnés, la température finale est renvoyée en tant que nombre réel.

Commande associée : TDELTA

TLINE

Commande de changement d'état de la ligne : Dans *PICT*, TLINE allume ou éteint (en fonction de leur état antérieur) les pixels de la ligne définie par les coordonnées spécifiées.

Niveau 2	Niveau 1	→	Niveau 1
(x_1, y_1)	(x_2, y_2)	→	
{ #n_1 #m_1 }	{ #n_2 #m_2 }	→	

Accès clavier : PRG PICT TLINE

Indicateurs : Aucun

Exemple : Le programme suivant allume et éteint 10 fois les pixels de la ligne définie par les coordonnées en unités-utilisateur (1,1) et (9,9). Chaque état est maintenu pendant 0,25 seconde.

```
«
 ERASE 0 10 XRNG 0 10 YNRG
 ( # 0d # 0d ) PVIEW
 «
  1 10 START
   (1,1) (9,9) TLINE
   .25 WAIT
  NEXT
 »
»
```

Commandes associées : ARC, BOX, LINE

TMENU

Commande d'affichage d'un menu temporaire : Affiche un menu intégré, un menu de bibliothèques ou un menu-utilisateur.

Niveau 1	→	Niveau 1
x_{menu}	→	
{ $liste_{définition}$ }	→	
'$nom_{définition}$'	→	

Accès clavier : ◀ MODES MENU TMEN

Indicateurs : Aucun

Remarques : TMENU fonctionne exactement comme MENU, sauf pour les menus-utilisateur (spécifiés par une liste ou par le nom d'une variable qui contient une liste). Ces menus sont affichés comme des menus personnalisés et fonctionnent comme tels, mais ils ne sont pas stockés dans la variable réservée *CST*. Par conséquent, un menu défini et affiché par TMENU ne peut pas être réaffiché en évaluant *CST*.

Reportez-vous à la rubrique MENU pour trouver la liste des menus intégrés du HP 48 et des numéros de menu correspondants (x_{menu}).

Examples: `7 TMENU` affiche la première page du menu MTH MATR.

`48.02 TMENU` affiche la deuxième page du menu UNITS MASS.

`768 TMENU` affiche la première page de commandes de la bibliothèque 768.

`{ A 123 "ABC" } TMENU` affiche le menu personnalisé défini par l'argument liste.

`'MONMENU' TMENU` affiche le menu personnalisé défini par l'argument nom.

Commandes associées : MENU, RCLMENU

TOT

Commande de sommation : Calcule la somme de chacune des m colonnes de valeurs de coordonnées figurant dans la matrice de statistiques en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	x_{somme}
	→	$[\ x_{somme1}\ x_{somme2}\ \cdots\ x_{sommem}\ ]$

Accès clavier : ◀ STAT 1VAR TOT

Indicateurs : Aucun

Remarque : Les sommes sont renvoyées sous la forme d'un vecteur de m nombres réels, ou d'un nombre réel unique si $m = 1$.

Commandes associées : MAXΣ, MINΣ, MEAN, PSDEV, PVAR, SDEV, VAR

TRACE

Commande de calcul de la trace d'une matrice : Renvoie la trace d'une matrice carrée.

{ }

Niveau 1	→	Niveau 1
$[[\ matrice\ ]]_{nxn}$	→	x_{trace}

Accès clavier : MTH MATR NORM NXT TRACE

Indicateurs : Aucun

Remarques : La trace d'une matrice carrée est la somme de ses éléments diagonaux.

TRANSIO

Commande de traduction des E-S : Spécifie l'option de traduction des caractères. Ces traductions ne concernent que les transferts Kermit ASCII et les fichiers imprimés via le port série.

{}

Niveau 1	→	Niveau 1
n_{option}	→	

Accès clavier : (↩)(I/O) IOPAR TRAN

Indicateurs : Aucun

Remarques : Les valeurs autorisées pour *n* sont les suivantes :

n	Effet
0	Pas de traduction
1	Traduit le caractère 10 (saut de ligne uniquement) en caractères 10 et 13 (saut de ligne avec retour chariot, protocole Kermit) et inversement (valeur par défaut)
2	Traduit les caractères 128 à 159 (80 à 9F en hexadécimal)
3	Traduit tous les caractères (128 à 255)

Commandes associées : BAUD, CKSM, PARITY

TRN

Commande de transposition d'une matrice : Renvoie la transposée (conjuguée) d'une matrice.

{ }

Niveau 1	→	Niveau 1
[[*matrice*]]	→	[[*matrice*]]$_{\text{transposée}}$
'*nom*'	→	

Accès clavier : (MTH) MATR MAKE TRN

Indicateurs : Aucun

Remarques : TRN remplace une matrice $\mathbf{A}$ $n \times m$ par une matrice $\mathbf{A}^T$ $m \times n$, où :

$\mathbf{A}_{ij}{}^T = \mathbf{A}_{ji}$ pour des matrices réelles

$\mathbf{A}_{ij}{}^T = \text{CONJ}(\mathbf{A}_{ji})$ pour des matrices complexes

Si la matrice est spécifiée par *nom*, $\mathbf{A}^T$ remplace $\mathbf{A}$ dans *nom*.

Exemple : `[[ 2 3 1 ][ 4 6 9 ]]` TRN renvoie `[[ 2 4 ][ 3 6 ][ 1 9 ]]`.

Commandes associées : CONJ

TRNC

Fonction de troncature : Tronque un objet à un nombre spécifié de décimales ou de chiffres significatifs, ou en fonction du format d'affichage en cours.

{ }

Niveau 2	Niveau 1	→	Niveau 1
z_1	$n_{troncature}$	→	z_2
z_1	'$symb_{troncature}$'	→	'TRNC(z_1 ,$symb_{troncature}$)'
'$symb_1$'	$n_{troncature}$	→	'TRNC($symb_1$,$n_{troncature}$)'
'$symb_1$'	'$symb_{troncature}$'	→	'TRNC($symb_1$,$symb_{troncature}$)'
[*tableau*]$_1$	$n_{troncature}$	→	[*tableau*]$_2$
x_unité	$n_{troncature}$	→	*y_unité*
x_unité	'$symb_{troncature}$'	→	'TRNC(*x_unité*,$symb_{troncature}$)'

Accès clavier : MTH REAL NXT NXT TRNC

Indicateurs : Résultats numériques (−3)

Remarques : $n_{troncature}$ (ou 'symb$_{troncature}$' si l'indicateur −3 est armé) contrôle la troncature de l'argument de niveau 2 de la manière suivante :

$n_{troncature}$	Effet sur l'argument de Niveau 2
0 à 11	tronqué à *n* décimales
−1 à −11	tronqué à *n* chiffres significatifs
12	tronqué au format d'affichage en cours

Pour les nombres complexes et les tableaux, chaque élément nombre réel est tronqué. Pour des objets-unités, seule la partie numérique est tronquée.

Exemples : (4.5792,8.1275) 2 TRNC renvoie (4.57,8.12).

[2.34907 3.96351 2.73453] -2 TRNC renvoie
[2.3 3.9 2.7].

Commande associée : RND

TRUTH

Commande de type de tracé Truth : Définit le type de tracé TRUTH.

Accès clavier : ◀ PLOT PTYPE TRUTH

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est TRUTH, la commande DRAW trace l'équation en cours comme une fonction booléenne de deux variables réelles. L'équation en cours est spécifiée dans la variable réservée *EQ*. Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, qui se présente sous la forme suivante :

$$\{\ (x_{min},\ y_{min})\ (x_{max},\ y_{max})\ \textit{indep res axes ptype depend}\ \}$$

Pour le type de tracé TRUTH, les éléments de *PPAR* sont utilisés de la manière suivante :

- (x_{min}, y_{min}) est un nombre complexe spécifiant l'angle inférieur gauche de *PICT* (l'angle inférieur gauche de la plage d'affichage). La valeur par défaut est (-6.5,-3.1).
- (x_{max}, y_{max}) est un nombre complexe spécifiant l'angle supérieur droit de *PICT* (l'angle supérieur droit de la plage d'affichage). La valeur par défaut est (6.5,3.2).
- *indep* est un nom spécifiant la variable indépendante sur l'axe horizontal, ou une liste contenant ce nom et deux nombres spécifiant les valeurs minimale et maximale de la variable indépendante (domaine de traçage horizontal). La valeur par défaut est *X*.
- *res* est un nombre réel spécifiant l'intervalle (coordonnées en unités-utilisateur) entre les valeurs tracées de la variable indépendante sur l'axe *horizontal*, ou un entier binaire spécifiant l'intervalle en pixels. La valeur par défaut est 0, soit un intervalle de 1 pixel.
- *axes* est une liste contenant un ou plusieurs des éléments suivants, dans l'ordre indiqué : un nombre complexe spécifiant les coordonnées en unités-utilisateur de l'origine du tracé, une liste spécifiant l'intervalle de graduation et deux chaînes spécifiant des libellés pour les axes horizontal et vertical. La valeur par défaut est (0,0).

- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de la commande TRUTH place le nom TRUTH dans *ptype*.
- *depend* est un nom spécifiant la variable indépendante sur l'axe vertical, ou une liste contenant ce nom et deux nombres spécifiant les valeurs minimale et maximale pour la variable indépendante (domaine de traçage vertical). La valeur par défaut est Y.

Le contenu de *EQ* doit être une expression ou un programme, et ne peut pas être une équation. Il est évalué pour chaque pixel de la région de traçage. Les valeurs minimale et maximale des variables indépendantes (domaines de traçage) peuvent être spécifiées dans *indep* et *depend*; autrement, les valeurs présentes dans $\langle x_{min}, y_{min} \rangle$ et $\langle x_{max}, y_{max} \rangle$ (plage d'affichage) sont employées. Le résultat de chaque évaluation doit être un nombre réel. Si le résultat est zéro, l'état du pixel n'est pas modifié. Si le résultat est différent de zéro, le pixel est activé (assombri).

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, WIREFRAME, YSLICE

TSTR

Commande de conversion date et heure en chaîne : Renvoie une chaîne obtenue à partir de la date et de l'heure.

{ }

Niveau 2	Niveau 1	→	Niveau 1
date	*heure*	→	"*JOUR DATE HEURE*"

Accès clavier : ↩ TIME NXT NXT TSTR

Indicateurs : Format de date (−42), Format d'horloge (−41)

Remarques : La chaîne est sous la forme "*JOUR DATE HEURE*", où *JOUR* est une abréviation en trois lettres du jour de la semaine

correspondant à l'argument *date* et *heure*, *DATE* est l'argument *date* au format en cours et *HEURE* est l'argument *heure* au format en cours.

Exemple : Avec les indicateurs −42 et −41 désarmés, `2.061990 14.55 TSTR` renvoie `"TUE 02/06/90 02:55:00P"`.

Commandes associées : DATE, TICKS, TIME

TVARS

Commande d'affichage des variables d'un type donné : Affiche la liste de toutes les variables globales présentes dans le répertoire en cours qui contiennent des objets ayant les types spécifiés.

Niveau 1	→	Niveau 1
n_{type}	→	{ *global* ... }
{ n_{type} ... }	→	{ *global* ... }

Accès clavier : ◀ MEMORY DIR TVARS

Indicateurs : Aucun

Remarques : Si le répertoire en cours ne contient pas de variables ayant les types spécifiés, TVARS renvoie une liste vide.

Pour connaître les numéros des types d'objets, reportez-vous à la rubrique TYPE.

Commandes associées : PVARS, TYPE, VARS

TVM

Commande de menu TVM : Affiche le menu du Solver TVM.

Accès clavier : Cette commande doit être tapée, mais vous pouvez également accéder au menu à l'aide de ◀ SOLVE TVM SOLVR.

Indicateurs : Aucun

Commandes associées : AMORT, TVMBEG, TVMEND, TVMROOT

TVMBEG

Commande de paiement en début de période : Spécifie que, dans les calculs TVM, les paiements sont effectués au début des périodes de composition.

Accès clavier : Cette commande doit être tapée, mais vous pouvez contrôler le mode de début/fin avec ◀ SOLVE TVM BEG.

Indicateurs : Aucun

Commandes associées : AMORT, TVM, TVMEND, TVMROOT

TVMEND

Commande de paiement en fin de période : Spécifie que, dans les calculs TVM, les paiements sont effectués à la fin des périodes de composition.

Accès clavier : Cette commande doit être tapée, mais vous pouvez contrôler le mode de début/fin avec ◀ SOLVE TVM BEG.

Indicateurs : Aucun

Commandes associées : AMORT, TVM, TVMBEG, TVMROOT

TVMROOT

Commande de calcul TVM : Calcule la solution pour la variable TVM spécifiée en employant les valeurs stockées dans les variables TVM restantes.

{ }

Niveau 1	→	Niveau 1
'variable TVM'	→	$x_{variableTVM}$

Accès clavier : ◀ SOLVE TVM TVMR

Indicateurs : Aucun

Commandes associées : AMORT, TVM, TVMBEG, TVMEND

TYPE

Commande de type : Renvoie le numéro de type d'un objet.

Niveau 1	→	Niveau 1
obj	→	n_{type}

Accès clavier :

PRG TYPE NXT NXT TYPE

PRG TEST NXT TYPE

Indicateurs : Aucun

Remarques : Le tableau suivant présente la liste des types d'objets et des numéros correspondants.

Numéros des types d'objets

Objet	Numéro
Objets utilisateur :	
Nombre réel	0
Nombre complexe	1
Chaîne de caractères	2
Tableau réel	3
Tableau complexe	4
Liste	5
Nom global	6
Nom local	7
Programme	8
Objet algébrique	9
Entier binaire	10

Numéros des types d'objets (suite)

Objet	Numéro
Objet graphique	11
Objet identifié	12
Objet-unité	13
Nom XLIB	14
Répertoire	15
Bibliothèque	16
Objet-sauvegarde	17
Commandes intégrées :	
Fonction intégrée	18
Commande intégrée	19
Objets système :	
Binaire système	20
Réel étendu	21
Complexe étendu	22
Tableau lié	23
Caractère	24
Objet code	25
Données de bibliothèque	26
Objet externe	27-31

La commande TYPE du HP 28S renvoie le numéro 8 pour les fonctions et les commandes (numéros 18 et 19 sur le HP 48).

Commandes associées : SAME, TVARS, VTYPE, ==

UBASE

Fonction de conversion en unités de base SI : Convertit un objet-unité en unités de base SI.

{ }

Niveau 1	→	Niveau 1
x_unité	→	*y_unité-base*
'*symb*'	→	'UBASE(*symb*)'

Accès clavier : ◀ UNITS UBASE

Indicateurs : Résultats numériques (−3)

Exemple : 30_knot UBASE renvoie 15.4333333333_m/s.

Commandes associées : CONVERT, UFACT, →UNIT, UVAL

UFACT

Commande de factorisation d'unités : Factorise l'unité de niveau 1 dans l'expression-unité de l'objet-unité de niveau 2.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_1 _*unité*$_1$	x_2 _*unité*$_2$	→	x_3 _*unité*$_2$ * *unité*$_3$

Accès clavier : ◀ UNITS UFACT

Indicateurs : Aucun

Remarques : UFACT est équivalent à la séquence suivante :

```
OBJ→ 3 ROLLD / OVER / UBASE *
```

Exemple : 1_W 1_N UFACT renvoie 1_N*m/s.

Commandes associées : CONVERT, UBASE, →UNIT, UVAL

→UNIT

Commande de création d'un objet-unité à partir de la pile : Crée un objet-unité à partir d'un nombre réel et de la partie unité d'un objet-unité.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y_unité*	→	*x_unité*

Accès clavier :

PRG TYPE →UNIT

◀ UNITS →UNIT

Indicateurs : Aucun

Remarques : →UNIT ajoute des unités à un nombre réel, combinant le nombre et la partie unité d'un objet-unité (la partie numérique de l'objet-unité est ignorée). →UNIT est l'inverse de OBJ→ appliquée à un objet-unité.

Commandes associées : →ARRY, →LIST, →STR, →TAG

UNTIL

Commande UNTIL : Débute la clause de test dans une structure en boucle infinie DO ... UNTIL ... END.

Reportez-vous à la rubrique DO pour plus d'informations sur la syntaxe.

Accès clavier : PRG BRCH DO UNTIL

Remarque : Reportez-vous à la rubrique DO pour un complément d'informations.

Commandes associées : DO, END

UPDIR

Commande d'accès au répertoire supérieur : Accède au répertoire parent, puis devient le répertoire en cours.

Accès clavier : ◀ UP

Indicateurs : Aucun

Remarques : UPDIR n'a pas d'effet si le répertoire en cours est *HOME*.

Commandes associées : CRDIR, HOME, PATH, PGDIR

UTPC

Commande de distribution Khi carré à droite : Renvoie la probabilité utpc(n, x) qu'une variable aléatoire Khi carré soit supérieure à x, où n est le nombre de degrés de liberté de la distribution.

{ }

Niveau 2	Niveau 1	→	Niveau 1
n	*x*	→	*utpc(n, x)*

Accès clavier : MTH NXT PROB NXT UTPC

Indicateurs : Aucun

Remarques : Les équations de définition sont les suivantes :

- Pour x ≥ 0 :

$$utpc(n,x) = \left[\frac{1}{2^{\frac{n}{2}}\Gamma\left(\frac{n}{2}\right)}\right]\int_x^\infty t^{\frac{n}{2}-1}e^{-\frac{t}{2}}dt$$

- Pour x < 0 :

$$utpc(n,x) = 1$$

UTPC

Pour toute valeur z, $\Gamma\left(\frac{z}{2}\right) = \left(\frac{z}{2} - 1\right)!$, où ! est la commande factorielle du HP 48 .

La valeur n est arrondie à l'entier le plus proche et, une fois arrondie, doit être positive.

Commandes associées : UTPF, UTPN, UTPT

UTPF

Commande de distribution F de Snedecor à droite : Renvoie la probabilité utpf(n_1 , n_2 , x) qu'une variable aléatoire F de Snedecor soit supérieure à x, où n_1 et n_2 sont les degrés de liberté du numérateur et du dénominateur de la distribution F.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
n_1	n_2	x	→	$utpf(n_1, n_2, x)$

Accès clavier : MTH NXT PROB NXT UTPF

Indicateurs : Aucun

Remarques : Les équations de définition pour utpf(n_1 , n_2 , x) sont les suivantes :

- Pour $x \geq 0$:

$$\left(\frac{n_1}{n_2}\right)^{\frac{n_1}{2}} \left[\frac{\Gamma\left(\frac{n_1+n_2}{2}\right)}{\Gamma\left(\frac{n_1}{2}\right)\Gamma\left(\frac{n_2}{2}\right)}\right] \int_x^{\infty} t^{\frac{(n_1-2)}{2}} \left[1 + \left(\frac{n_1}{n_2}\right) t\right]^{-\frac{(n_1+n_2)}{2}} dt$$

- Pour $x < 0$:

$$utpf(n_1, n_2, x) = 1$$

Pour toute valeur z, $\Gamma\left(\frac{z}{2}\right) = \left(\frac{z}{2} - 1\right)!$, où ! est la commande factorielle du HP 48.

Les valeurs n_1 et n_2 sont arrondies aux entiers les plus proches et, une fois arrondies, doivent être positives.

Commandes associées : UTPC, UTPN, UTPT

UTPN

Commande de distribution normale à droite : Renvoie la probabilité utpn(m, v, x) qu'une variable aléatoire normale soit supérieure à x, où m et v sont respectivement la moyenne et la variance de la distribution normale.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
m	*v*	*x*	→	*utpn(m, v, x)*

Accès clavier : MTH NXT PROB NXT UTPN

Indicateurs : Aucun

Remarques : Pour tout x et m, et pour $v > 0$, l'équation de définition est la suivante :

$$utpn(m, v, x) = \left[\frac{1}{\sqrt{2\pi v}}\right] \int_x^{\infty} e^{-\frac{(t-m)^2}{2v}} dt$$

Pour $v = 0$, UTPN renvoie 0 pour $x \geq m$, et 1 pour $x < m$.

Commandes associées : UTPC, UTPF, UTPT

UTPT

Commande de distribution t de Student à droite : Renvoie la probabilité utpt(n, x) qu'une variable aléatoire t de Student soit supérieure à x, où n est le nombre de degrés de liberté de la distribution.

{ }

Niveau 2	Niveau 1	→	Niveau 1
n	x	→	$utpt(n, x)$

Accès clavier : MTH NXT PROB NXT UTPT

Indicateurs : Aucun

Remarque : Pour tout x, l'équation de définition est la suivante :

$$utpt(n,x) = \left[\frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)\sqrt{n\pi}}\right]\int_x^\infty \left(1+\frac{t^2}{n}\right)^{-\frac{n+1}{2}} dt$$

Pour toute valeur z, $\Gamma\left(\frac{z}{2}\right) = \left(\frac{z}{2}-1\right)!$, où ! est la commande factorielle du HP 48 .

La valeur n est arrondie à l'entier le plus proche et, une fois arrondie, doit être positive.

Commandes associées : UTPC, UTPF, UTPN

UVAL

Fonction d'élimination de la partie unité : Renvoie la partie numérique d'un objet-unité.

UVAL *Elim. unité* **Function**

Niveau 1	→	Niveau 1
x_unité	→	*x*
'*symb*'	→	'UVAL(*symb*)'

Accès clavier : ◀ (UNITS) UVAL

Indicateurs : Résultats numériques (−3)

Commandes associées : CONVERT, UBASE, UFACT, →UNIT

VAR

Commande de calcul de variance : Calcule la variance sur échantillon des valeurs de coordonnées dans chacune des m colonnes de la matrice de statistiques en cours (ΣDAT).

Niveau 1	→	Niveau 1
	→	$x_{variance}$
	→	$[\, x_{variance1} \cdots x_{variancem} \,]$

Accès clavier : ◀ (STAT) 1VAR (NXT) VAR

Indicateurs : Aucun

Remarques : La variance (égale au carré de l'écart type) est renvoyée sous la forme d'un vecteur de m nombres réels, ou d'un seul nombre réel si $m = 1$. Les variances sont calculées à l'aide de la formule suivante :

VAR

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Où x_i est la $i$ème valeur de coordonnée d'une colonne, $\bar{x}$ est la moyenne des données de cette colonne et n est le nombre de points de données.

Commandes associées : MAXΣ, MEAN, MINΣ, PSDEV, PVAR, SDEV, TOT

VARS

Commande d'affichage des variables : Renvoie la liste de tous les noms de variables figurant dans le menu VAR (répertoire en cours).

Niveau 1	→	Niveau 1
	→	{ *global*$_1$ ... *global*$_n$ }

Accès clavier : ◀ MEMORY DIR VARS

Indicateurs : Aucun

Commandes associées : ORDER, PVARS, TVARS

VERSION

Commande d'affichage de la version du logiciel : Affiche la version du logiciel et le copyright.

Niveau 1	→	Niveau 2	Niveau 1
	→	"*numéro de version*"	"*copyright*"

Accès clavier : Aucun. A saisir.

Indicateurs : Aucun

VTYPE

Commande d'affichage du type de variable : Renvoie le numéro de type de l'objet contenu dans la variable nommée.

{ }

Niveau 1	→	Niveau 1
'*nom*'	→	n_{type}
$:n_{port}:nom_{sauvegarde}$	→	n_{type}
$:n_{port}:n_{bibliothèque}$	→	n_{type}

Accès clavier : PRG TYPE NXT NXT VTYPE

Indicateurs : Aucun

Remarques : Si la variable nommée n'existe pas, VTYPE renvoie −1.

Pour connaître les numéros de types d'objets, reportez-vous à la rubrique TYPE.

Commande associée : TYPE

→V2

Commande de combinaison en vecteur/nombre complexe : Convertit deux nombres figurant dans la pile en un vecteur à deux éléments ou en un nombre complexe.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	[*x y*]
x	*y*	→	[*x* ∠ *y*]
x	*y*	→	(*x*, *y*)
x	*y*	→	(*x*, ∠*y*)

Accès clavier : (MTH) VECTR →V2

Indicateurs : Mode complexe (−19), Mode de coordonnées (−16)

Remarques : Le résultat renvoyé dépend du paramétrage des indicateurs −16 et −19, comme indiqué dans le tableau suivant :

	Ind. −19 désarmé	Ind. −19 armé
Indicateur −16 désarmé (mode rectangulaire)	[x y]	(x, y)
Indicateur −16 armé (mode polaire)	[x ∠y]	(x, ∠y)

Exemples : Les indicateurs −19 et −16 étant désarmés, 2 3 →V2 renvoie [2 3].

Les indicateurs −19 et −16 étant armés (mode polaire/sphérique), 2 3 →V2 renvoie (2,∠3).

Commandes associées : V→, →V3

→V3

Commande de combinaison en un vecteur à 3 éléments :
Convertit trois nombres en un vecteur à 3 éléments.

{ }

Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
x_1	x_2	x_3	→	[x_1 x_2 x_3]
x_1	x_{theta}	x_z	→	[x_1 ∠x_{theta} x_z]
x_1	x_{theta}	x_{phi}	→	[x_1 ∠x_{theta} ∠x_{phi}]

Accès clavier : (MTH) VECTR →V3

Indicateurs : Mode de coordonnées (−15 et −16)

Remarques : Le résultat renvoyé dépend du mode de coordonnées utilisé, comme indiqué dans le tableau suivant :

Mode	Résultat
Rectangulaire (Ind. −16 désarmé)	[x_1 x_2 x_3]
Polaire/Cylindrique (Ind. −15 désarmé et −16 armé)	[x_1 x∠$_{theta}$ x_z]
Polaire/Sphérique (Ind. −15 et −16 armés)	[x_1 x∠$_{theta}$ x∠$_{phi}$]

Exemples : L'indicateur −16 étant désarmé (mode rectangulaire), 1 2 3 →V3 renvoie [1 2 3].

Les indicateurs −15 et −16 étant respectivement désarmé et armé (mode polaire/cylindrique), 1 2 3 →V3 renvoie [1 ∠2 3].

Les indicateurs −15 et −16 étant armés (mode polaire/sphérique), 1 2 3 →V3 renvoie [1 ∠2 ∠3].

Commandes associées : V→, →V2

V→

Commande d'éclatement d'un vecteur/nombre complexe dans la pile : Sépare un vecteur ou un nombre complexe de manière à obtenir ses éléments constitutifs.

{ }

Niv 1	→	Niv n .. Niv 3	Niv 2	Niv 1
[x y]	→		x	y
[x_r ∠y_{theta}]	→		x_r	y_{theta}
[x_1 x_2 x_3]	→	x_1	x_2	x_3
[x_1 ∠x_{theta} x_z]	→	x_1	x_{theta}	x_z
[x_1 ∠x_{theta} ∠x_{phi}]	→	x_1	x_{theta}	x_{phi}
[x_1 x_2 ... x_n]	→	x_1 ... x_{n-2}	x_{n-1}	x_n
(x, y)	→		x	y
(x_r, ∠y_{theta})	→		x_r	y_{theta}

Accès clavier : (MTH) VECTR V→

Indicateurs : Mode de coordonnées (−15 et −16)

Les éléments du nombre complexe ou du vecteur de l'argument sont convertis à partir de leurs valeurs en mode rectangulaire (la forme sous laquelle le nombre complexe ou le vecteur est stocké de manière interne) en mode de coordonnées en cours avant d'être renvoyés vers la pile. Ainsi, les valeurs des éléments renvoyées dans la pile correspondent-elles toujours aux valeurs des éléments *affichées* du vecteur ou du nombre complexe de l'argument.

Remarques : Pour les vecteurs comptant quatre éléments ou plus, V→ s'exécute *indépendamment* du mode de coordonnées, et renvoie toujours les éléments du vecteur dans la pile tels qu'ils sont stockés de manière interne (sous forme rectangulaire). Par conséquent, V→ est équivalent à OBJ→ pour des vecteurs comptant quatre éléments ou plus.

Exemples : L'indicateur −16 étant désarmé (mode rectangulaire), (2,3) V→ renvoie 2 au niveau 2 et 3 au niveau 1.

Les indicateurs −15 et −16 étant respectivement désarmé et armé (mode polaire/cylindrique), [2 ∠7 4] V→ renvoie 2 au niveau 3, 7 au niveau 2 et 4 au niveau 1.

[9 7 5 3] V→ renvoie 9 au niveau 4, 7 au niveau 3, 5 au niveau 2 et 3 au niveau 1, indépendamment de l'état des indicateurs −15 et −16.

Commandes associées : →V2, →V3

---

*W

Commande de multiplication de la largeur : Multiplie l'échelle horizontale d'un tracé par $x_{facteur}$.

{ }

Niveau 1	→	Niveau 1
$x_{facteur}$	→	

Accès clavier : ◀ PLOT PPAR NXT *W

Accès clavier : Aucun

Remarques : L'exécution de *W modifie la plage d'affichage de l'axe x (x_{min} et x_{max} dans la variable réservée *PPAR*). Le centre du tracé (les coordonnées en unités-utilisateur du pixel central) n'est pas modifié.

Commandes associées : AUTO, *H, XRNG

WAIT

Commande d'attente : Suspend l'exécution du programme pendant un délai spécifié, ou jusqu'à l'utilisation d'une touche.

{ }

Niveau 1	→	Niveau 1
x	→	
0	→	x_{touche}
−1	→	x_{touche}

Accès clavier : PRG NXT IN WAIT

Accès clavier : Aucun

Remarques : La fonction de WAIT dépend de l'argument, comme suit :

- L' argument x interrompt l'exécution du programme pendant x secondes.
- L'argument 0 suspend l'exécution du programme jusqu'à l'utilisation d'une touche correcte (voir ci-dessous). WAIT renvoie alors x_{touche}, qui définit où la touche utilisée se trouve sur le clavier, et reprend l'exécution du programme.

  x_{touche} est un nombre à trois chiffres qui identifie l'emplacement d'une touche sur le clavier. Reportez-vous à la rubrique ASN pour la description du format de x_{touche}.
- L'argument −1 fonctionne comme l'argument 0, à ceci près que le menu spécifié en cours est également affiché.

←, →, α, α←, et α→ ne constituent pas en soi des touches correctes.

Les arguments 0 ou −1 n'affectant pas l'affichage, les messages persistent même si le clavier est prêt pour une entrée (FREEZE n'est pas indispensable).

En principe, la commande MENU ne met pas à jour les touches de menu jusqu'à l'interruption ou l'arrêt d'un programme. WAIT, employée avec l'argument −1, permet à MENU de s'exécuter plus tôt,

pour afficher le menu pendant que le programme est suspendu dans l'attente de l'utilisation d'une touche.

Exemples : Le programme suivant :

```
« "Appuyez sur [1] pour additionner■
 Appuyez sur une touche quelconque pour soustraire"
1 DISP 0 WAIT IF 82.1 SAME THEN + ELSE - END »
```

affiche le message d'invite et interrompt l'exécution du programme jusqu'à l'utilisation d'une touche. Si la touche 1 (emplacement 82.1) est actionnée, deux nombres figurant dans la pile sont additionnés. Si une autre touche est actionnée, deux nombres figurant dans la pile sont soustraits.

Le programme suivant :

```
« { ADD { } { } { } { } SUB } MENU "Appuyez sur [ADD]
pour additionner■ Appuyez sur [SUB] pour soustraire"
1 DISP -1 WAIT IF 11.1 SAME THEN + ELSE - END »
```

crée un menu personnalisé avec des libellés ADD et SUB et un message d'invite. L'exécution de -1 WAIT affiche le menu personnalisé (notez qu'il n'est pas actif) et suspend l'exécution pour la saisie au clavier. Si la touche de menu ADD (emplacement 11.1) est actionnée, deux nombres figurant dans la pile sont additionnés. Si une autre touche est utilisée, deux nombres figurant dans la pile sont soustraits.

Commande associée : KEY

WHILE

Commande de structure en boucle infinie WHILE : Débute la structure en boucle infinie WHILE ... REPEAT ... END.

WHILE

	Niveau 1	→	Niveau 1
WHILE		→	
REPEAT	*V/F*	→	
END		→	

Accès clavier : PRG BRCH WHILE WHILE

Accès clavier : Aucun

Remarques : WHILE ... REPEAT ... END évalue plusieurs fois un test et exécute une clause de boucle si le test est vrai. La clause de test étant exécutée avant la clause de la boucle, cette dernière n'est jamais exécutée si le test est faux. La syntaxe est la suivante :

WHILE *clause-de-test* REPEAT *clause-de-la-boucle* END

La clause de test est exécutée et doit renvoyer un résultat de test dans la pile. REPEAT prend la valeur dans la pile. Si la valeur est différente de zéro, l'exécution se poursuit par la clause de la boucle ; sinon elle reprend à la suite de END.

Commandes associées : DO, END, REPEAT

WIREFRAME

Commande de type de tracé WIREFRAME : Définit le type de tracé WIREFRAME.

Accès clavier : ↩ PLOT NXT 3D PTYPE WIREF

Accès clavier : Aucun

Remarques : Lorsque le type de tracé est WIREFRAME, la commande DRAW trace une vue en perspective du graphe d'une fonction scalaire à deux variables. WIREFRAME requiert des valeurs dans les variables réservées *EQ*, *VPAR* et *PPAR*.

VPAR se présente sous la forme suivante :

$\{ x_{left}\ x_{right}\ y_{near}\ y_{far}\ z_{low}\ z_{high}\ x_{min}\ x_{max}\ y_{min}\ y_{max}\ x_{eye}\ y_{eye}\ z_{eye}\ x_{step}\ y_{step} \}$

Pour le type de tracé WIREFRAME, les éléments de *VPAR* sont utilisés de la manière suivante :

- x_{left} et x_{right} sont des nombres réels qui spécifient la largeur de la vue volumique.
- y_{near} et y_{far} sont des nombres réels qui spécifient la profondeur de la vue volumique.
- z_{low} et z_{high} sont des nombres réels qui spécifient la hauteur de la vue volumique.
- x_{min} et x_{max} ne sont pas utilisés.
- y_{min} et y_{max} ne sont pas utilisés.
- x_{eye}, y_{eye}, et z_{eye} sont des nombres réels qui spécifient le point dans l'espace à partir duquel le graphe est visualisé.
- x_{pas} et y_{pas} sont des nombres réels qui définissent le nombre de coordonnées x par rapport au nombre de coordonnées y tracées.

Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, qui se présente sous la forme suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé WIREFRAME, les éléments de *PPAR* sont utilisés de la manière suivante :

- (x_{min}, y_{min}) n'est pas utilisé.
- (x_{max}, y_{max}) n'est pas utilisé.
- *indep* est un nom spécifiant la variable indépendante. La valeur par défaut de *indep* est *X*.
- *res* n'est pas utilisé.
- *axes* n'est pas utilisé.
- *ptype* est un nom spécifiant le type de tracé. L'exécution de la commande WIREFRAME place le nom de commande WIREFRAME dans *ptype*.
- *depend* est un nom spécifiant la variable dépendante. La valeur par défaut est *Y*.

WIREFRAME

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, YSLICE

WSLOG

Commande d'affichage du journal de démarrage à chaud :
Renvoie quatre chaînes enregistrant la date, l'heure et la cause des quatre événements de démarrage à chaud les plus récents.

Niveau 1	→	Niveau 4 ... Niveau 1
	→	"jrl_4" ... "jrl_1"

Accès clavier : Aucun. A saisir.

Accès clavier : Format de date (−42)

Remarques : Chaque chaîne "jrl_n" se présente sous la forme "*code−date heure*". Le tableau suivant résume les valeurs de *code* autorisées et la description correspondante.

Code	Description
0	Le journal de démarrage à chaud a été effacé en appuyant sur ON SPC puis sur ON pour activer le calculateur. ON SPC place le HP 48 en "mode Coma" (très faible consommation et *horloge système arrêtée*). En appuyant sur ON le journal est effacé et le système démarre à chaud.
1	Le système d'interruption a détecté au niveau des contacts que les piles étaient très faibles (différent d'une tension faible du système), et a placé le calculateur en "mode Veille" (*horloge du système en fonctionnement*). Lorsque ON est utilisé après rétablissement de la tension des piles, le système démarre à chaud et place un 1 dans le journal.
2	Incident matériel en cours de transmission IR (expiration du délai).
3	Exécution jusqu'à l'adresse 0.
4	L'heure système est altérée.
5	Une activation en mode Veille (par exemple, ON, Alarm) n'a pas détecté de modification de l'état du port, mais certaines modifications des données sur l'une des cartes ou les deux.
6	Inutilisé.
7	Un mot à 5 quartets (mot de test CMOS) dans la RAM était altéré (ce mot est vérifié lors de chaque interruption, mais il est utilisé uniquement en tant qu'indicateur de RAM potentiellement altérée).

Code	Description
8	L'une des anomalies suivantes relatives à la configuration des unités a été détectée : ■ Le système d'interruption a détecté que l'une des cinq unités n'était pas configurée. ■ Au cours d'un démarrage à chaud, une chaîne d'identification d'unité inattendue a été rencontrée pendant une tentative de configuration de 3 (Port1, Port2, Xtra) des 5 unités. ■ Comme précédemment, mais détection au cours d'une activation en mode Veille.
9	La liste des alarmes est altérée.
A	Inutilisé.
B	Le module carte a été retiré (ou la carte a bougé).
C	Une réinitialisation matérielle s'est produite (par exemple, décharge électrostatique ou réinitialisation par l'utilisateur).
D	Un sous-programme de traitement d'erreurs système (RPL) prévu n'a pas été détecté dans la chaîne d'exécution.
E	La table de configuration est altérée (somme de contrôle incorrecte pour des données de la table).
F	La carte RAM système a été retirée.

La partie *date heure* du journal peut être affichée sous la forme `00 ... 0000` pour l'une des trois raisons suivantes :

- L'heure système était altérée lorsque cette partie a été enregistrée.
- Cette partie elle-même est altérée (somme de contrôle incorrecte).
- Moins de quatre démarrages à chaud ont eu lieu depuis le dernier effacement du journal.

ΣX

Commande de sommation des valeurs x : Calcule la somme des valeurs figurant dans la colonne de variables indépendantes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	x_{somme}

Accès clavier : ◀ STAT SUMS ΣX

Indicateurs : Aucun

Remarques : La colonne de variables indépendantes est spécifiée par XCOL et est stockée en tant que premier paramètre dans la variable réservée ΣPAR. Le numéro de la colonne de variables indépendantes par défaut est 1.

Commandes associées : NΣ, XCOL, ΣX*Y, ΣX^2, ΣY, ΣY^2

ΣX^2

Commande de sommation des carrés des valeurs x : Calcule la somme des carrés des valeurs figurant dans la colonne de variables indépendantes de la matrice statistique en cours (variable réservée ΣDAT).

Niveau 1	→	Niveau 1
	→	x_{somme}

Accès clavier : ◀ STAT SUMS ΣX^2

Indicateurs : Aucun

ΣX^2

Remarques : La colonne de variables indépendantes est spécifiée par XCOL et est stockée en tant que premier paramètre dans la variable réservée ΣPAR. Le numéro de la colonne de variables indépendantes par défaut est 1.

Commandes associées : NΣ, ΣX, XCOL, ΣX*Y, ΣY, ΣY^2

XCOL

Commande de définition de la colonne de variables indépendantes : Spécifie la colonne de variables indépendantes de la matrice statistique en cours (variable réservée ΣDAT).

{ }

Niveau 1	→	Niveau 1
n_{col}	→	

Accès clavier : ↩ STAT ΣPAR XCOL

Indicateurs : Aucun

Remarques : Le numéro de la colonne de variables indépendantes est stocké en tant que premier paramètre dans la variable réservée ΣPAR. Le numéro de la colonne de variables indépendantes par défaut est 1.

XCOL accepte un nombre réel non entier et le stocke dans ΣPAR, mais les commandes ultérieures utilisant la spécification XCOL dans ΣPAR provoquent une erreur.

Commandes associées : BARPLOT, BESTFIT, COLΣ, CORR, COV, EXPFIT, HISTPLOT, LINFIT, LOGFIT, LR, PREDX, PREDY, PWRFIT, SCATRPLOT, YCOL

XMIT

Commande de transmission en série : Envoie une chaîne en série sans utiliser le protocole Kermit, et renvoie un chiffre unique qui indique si la transmission a réussi.

{}

Niveau 1	→	Niveau 2	Niveau 1
"*chaîne*"	→		1
"*chaîne*"	→	"*sous-chaîne*$_{\text{nontransmis}}$"	0

Accès clavier : ◀ I/O NXT SERIA XMIT

Indicateurs : Unité d'E-S (−33)

Remarques : XMIT est utile pour communiquer avec des unités non Kermit telles que des imprimantes RS-232.

Si la transmission réussit, XMIT renvoie un 1. Si la transmission échoue, XMIT renvoie la partie non transmise de la chaîne et un 0. Utilisez ERRM pour obtenir le message d'erreur.

Après avoir reçu une commande XOFF (la *régulation de la transmission* dans la variable réservée *IOPAR* étant activée), XMIT arrête de transmettre et attend une commande XON. XMIT reprend la transmission si un XON est reçu avant que le délai défini par STIME soit expiré ; sinon, XMIT prend fin, renvoie un 0 et stocke "Timeout" dans ERRM.

Commandes associées : BUFLEN, SBRK, SRECV, STIME

XOR

Fonction OU exclusif : Renvoie le OU exclusif logique de deux arguments.

{ }

Niveau 2	Niveau 1	→	Niveau 1
#n_1	#n_2	→	#n_3
"*chaîne*$_1$"	"*chaîne*$_2$"	→	"*chaîne*$_3$"
V/F$_1$	*V/F*$_2$	→	0/1
V/F	'*symb*'	→	'*V/F* XOR *symb*'
'*symb*'	*V/F*	→	'*symb* XOR *V/F*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$ XOR *symb*$_2$'

Accès clavier :

MTH BASE NXT LOGIC XOR

PRG TEST NXT XOR

Indicateurs : Résultats numériques (−3), taille de mot entier binaire (−5 à −10)

Remarques : Lorsque les arguments sont des entiers binaires ou des chaînes, XOR effectue une comparaison logique bit par bit (base 2) :

- Les arguments entiers binaires sont traités comme des séquences de bits d'une longueur égale à la taille de mot en cours. Chaque bit du résultat est déterminé en comparant les bits corerspondants (*bit*$_1$ et *bit*$_2$) des deux arguments, comme indiqué dans le tableau suivant.

bit$_1$	*bit*$_2$	*bit*$_1$ **XOR** *bit*$_2$
0	0	0
0	1	1
1	0	1
1	1	0

- Les arguments chaînes sont traités comme des séquences de bits, en utilisant 8 bits par caractère (c-à-d, en utilisant la version binaire du code du caractère). Les deux arguments chaînes doivent être de la même longueur.

Lorsque les arguments sont des nombres réels ou symboliques, XOR effectue un test vrai/faux. Le résultat est 1 (vrai) si l'un des arguments, mais pas les deux, est différent de zéro, il est 0 (faux) si les deux arguments sont différents de zéro ou égaux à zéro. Ce test est en général effectué pour comparer deux résultats de test.

Si l'un des deux arguments ou les deux sont des objets algébriques, le résultat est une expression algébrique se présentant sous la forme '$symb_1$ XOR $symb_2$'. Exécutez →NUM (ou armez l'indicateur −3 avant d'exécuter XOR) pour produire un résultat numérique à partir du résultat algébrique.

Commandes associées : AND, NOT, OR

XPON

Fonction exposant : Renvoie l'exposant de l'argument.

{ }

Niveau 1	→	Niveau 1
x	→	n_{expos}
'*symb*'	→	'XPON(*symb*)'

Accès clavier : (MTH) REAL (NXT) XPON

Indicateurs : Résultats numériques (−3)

Examples: 1.2E34 XPON renvoie 34.

12.4E3 XPON renvoie 4.

'A*1E34 XPON renvoie 'XPON(A*1E34)'.

Commandes associées : MANT, SIGN

XRECV

Commande de réception XModem : Prépare le HP 48 à recevoir un objet via XModem. L'objet reçu est stocké dans le nom de variable donné.

Niveau 1	→	Niveau 1
'nom'	→	

Accès clavier : ◀ I/O NXT XRECV

Indicateurs : Unité d'E-S (−33), Ecrasement de RECV (−36)

Remarques : Le transfert débute plus rapidement si vous lancez l'émetteur XModem *avant* d'exécuter XRECV.

Les noms d'objet incorrects provoquent une erreur. Si l'indicateur −36 est désarmé, les noms d'objets déjà employés causent eux aussi une erreur.

Si vous transférez des données entre deux HP 48, l'exécution de `{AAA BBB CCC} XRECV` reçoit *AAA*, *BBB* et *CCC*. Vous devez également utiliser une liste du côté émetteur (`{ AAA BBB CCC} XSEND`, par exemple).

Commandes associées : BAUD, RECV, RECN, SEND, XSEND

XRNG

Commande de définition de la plage d'affichage de l'axe x : Spécifie la plage d'affichage de l'axe x.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_{min}	x_{max}	→	

Accès clavier : ◀ PLOT PPAR XRNG

Indicateurs : Aucun

Remarques : La plage d'affichage de l'axe x est stockée dans la variable réservée *PPAR* sous la forme x_{min} et x_{max} dans les nombres complexes (x_{min}, y_{min}) et (x_{max}, y_{max}). Ces nombres complexes sont les deux premiers éléments de *PPAR* et spécifient les coordonnées des angles inférieur gauche et supérieur droit des plages d'affichage.

Les valeurs par défaut de x_{min} et x_{max} sont respectivement -6.5 et 6.5.

Commandes associées : AUTO, PDIM, PMAX, PMIN, YRNG

XROOT

Commande d'extraction de la xième racine de y : Calcule la $x$ième racine d'un nombre réel.

{ }

Niveau 2	Niveau 1	→	Niveau 1
y	*x*	→	$\sqrt[x]{y}$
'*symb*$_1$'	'*symb*$_2$'	→	'XROOT(*symb*$_2$, *symb*$_1$)'
'*symb*'	*x*	→	'XROOT(*x*, *symb*)'
y	'*symb*'	→	'XROOT(*symb*, *y*)'
y_unité	*x*	→	$\sqrt[x]{y}$_unité$^{1/x}$
y_unité	'*symb*'	→	'XROOT(*symb*, *y_unité*)'

Accès clavier : ▶ $\sqrt[x]{y}$

Indicateurs : Résultats numériques (−3)

Remarques : Notez que tandis que la syntaxe de la *pile* est *y x* XROOT (la racine est le deuxième argument), la syntaxe *algébrique* est XROOT(x, y) (la racine est le premier argument) pour être homogène avec l'application EquationWriter.

XROOT est équivalent à $y^{1/x}$, mais avec une plus grande précision.

XROOT

Si $y < 0$, x doit être un entier.

Commande associée : ^

XSEND

Commande d'émission XModem : Envoie une copie de l'objet nommé via XModem.

Niveau 1	→	Niveau 1
'nom'	→	

Accès clavier : ◀ I/O NXT XSEN

Indicateurs : Unité d'E-S (−33)

Remarques : Un HP 48 récepteur doit exécuter XRECV pour recevoir un objet via XModem.

Pour débuter le transfert plus rapidement, lancez le récepteur XModem *après* avoir exécuté XSEND. En outre, si le modem récepteur est configuré de façon à *ne pas* effectuer de contrôle par redondance cyclique (si possible), vous éviterez le délai de 30 à 60 secondes d'attente avant le début du transfert.

Si vous transférez des données entre deux HP 48, l'exécution de `« AAA BBB CCC » XSEND` envoie *AAA*, *BBB* et *CCC*. Vous devez également utiliser une liste du côté récepteur (`« AAA BBB CCC » XRECV`, par exemple).

Commandes associées : BAUD, RECN, RECV, SEND, XRECV

XVOL

Commande de définition des coordonnées volumiques x : Définit la largeur de la vue volumique dans la variable réservée *VPAR*.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_{left}	x_{right}	→	

Accès clavier : ◀ PLOT NXT 3D VPAR XVOL

Indicateurs : Aucun

Remarques : x_{left} et x_{right} définissent les coordonnées x de la vue volumique utilisée dans des tracés 3D. Ces valeurs sont stockées dans la variable réservée *VPAR*. Reportez-vous à l'annexe D, "Variables réservées," pour complément d'informations sur *VPAR*.

Commandes associées : EYEPT, XXRNG, YVOL, YYRNG, ZVOL

XXRNG

Commande de définition de la plage x de la zone d'entrée (domaine) : Spécifie la plage x d'une zone d'entrée (domaine) pour les tracés GRIDMAP et PARSURFACE.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_{min}	x_{max}	→	

Accès clavier : ◀ PLOT NXT 3D VPAR XXRN

Indicateurs : Aucun

XXRNG

Remarques : x_{min} et x_{max} sont des nombres réels qui définissent les coordonnées x de la zone d'entrée. Ces valeurs sont stockées dans la variable réservée *VPAR*. Reportez-vous à l'annexe D, "Variables réservées," pour complément d' informations sur *VPAR*.

Commandes associées : EYEPT, NUMX, NUMY, XVOL, YVOL, YYRNG, ZVOL

ΣX*Y

Commande de sommation de x fois y : Calcule la somme des produits de chacune des valeurs correspondantes figurant dans les colonnes de variables indépendantes et dépendantes de la matrice statistique en cours (variable réservée *ΣDAT*).

Niveau 1	→	Niveau 1
	→	x_{somme}

Accès clavier : ↶ STAT SUMS ΣX*Y

Indicateurs : Aucun

Remarques : La colonne de variables indépendantes est spécifiée par XCOL et est stockée en tant que premier paramètre dans la variable réservée *ΣPAR*. Le numéro de la colonne de variables indépendantes par défaut est 1.

La colonne de variables dépendantes est spécifiée par YCOL et est stockée en tant que deuxième paramètre dans la variable réservée *ΣPAR*. Le numéro de la colonne de variables dépendantes par défaut est 2.

Commandes associées : NΣ, ΣX, XCOL, ΣX^2, ΣY, ΣY^2

ΣY

Commande de sommation des valeurs y : Calcule la somme des valeurs figurant dans la colonne de variables dépendantes de la matrice statistique en cours (variable réservée *ΣDAT*).

Niveau 1	→	Niveau 1
	→	x_{somme}

Accès clavier : ◀ STAT SUMS ΣY

Indicateurs : Aucun

Remarques : La colonne des variables dépendantes est spécifiée par YCOL, et est stockée en tant que deuxième paramètre dans la variable réservée *ΣPAR*. Le numéro de la colonne de variables dépendantes par défaut est 2.

Commandes associées : NΣ, ΣX, XCOL, ΣX*Y, ΣX^2, YCOL, ΣY^2

ΣY^2

Commande de sommation des carrés des valeurs y : Calcule la somme des carrés des valeurs figurant dans la colonne de variables dépendantes de la matrice statistique en cours (variable réservée *ΣDAT*).

Niveau 1	→	Niveau 1
	→	x_{somme}

Accès clavier : ◀ STAT SUMS ΣY^2

Indicateurs : Aucun

ΣY^2

Remarques : La colonne des variables dépendantes est spécifiée par YCOL. Le numéro de la colonne des variables par défaut est 2.

Commandes associées : NΣ, ΣX, XCOL, ΣX*Y, ΣX^2, YCOL, ΣY

YCOL

Commande de définition de la colonne de variables dépendantes
: Spécifie la colonne de variables dépendantes de la matrice statistique en cours (variable réservée ΣDAT).

{ }

Niveau 1	→	Niveau 1
n_{col}	→	

Accès clavier : ↩ STAT ΣPAR YCOL

Indicateurs : Aucun

Remarques : Le numéro de la colonne de variables dépendantes est stocké en tant que deuxième paramètre dans la variable réservée ΣPAR. Le numéro de la colonne de variables dépendantes par défaut est 2.

YCOL accepte un nombre réel non entier et le stocke dans ΣPAR, mais les commandes ultérieures qui utilisent la spécification YCOL dans ΣPAR provoquent une erreur.

Commandes associées : BARPLOT, BESTFIT, COLΣ, CORR, COV, EXPFIT, HISTPLOT, LINFIT, LOGFIT, LR, PREDX, PREDY, PWRFIT, SCATRPLOT, XCOL

YRNG

Commande de définition de la plage d'affichage de l'axe y : Spécifie la plage d'affichage de l'axe y.

{ }

Niveau 2	Niveau 1	→	Niveau 1
y_{min}	y_{max}	→	

Accès clavier : ◀ PLOT PPAR YRNG

Indicateurs : Aucun

Remarques : La plage d'affichage de l'axe y est stockée dans la variable réservée *PPAR* sous la forme y_{min} et y_{max} dans les nombres complexes (x_{min}, y_{min}) et (x_{max}, y_{max}). Ces nombres complexes sont les deux premiers éléments de *PPAR* et spécifient les coordonnées des angles inférieur gauche et supérieur droit des plages d'affichage.

Les valeurs par défaut de y_{min} et y_{max} sont respectivement −3.1 et 3.2.

Commandes associées : AUTO, PDIM, PMAX, PMIN, XRNG

YSLICE

Commande de tracé Y-Slice : Définit le type de tracé YSLICE.

Accès clavier : ◀ PLOT NXT 3D PTYPE YSLIC

Indicateurs : Aucun

Remarques : Lorsque le type de tracé est YSLICE, la commande DRAW trace une vue en sections d'une fonction scalaire de deux variables. YSLICE requiert des valeurs dans les variables réservées *EQ*, *VPAR* et *PPAR*.

VPAR se présente sous la forme suivante :

YSLICE

{ x_{left} x_{right} y_{near} y_{far} z_{low} z_{high} x_{min} x_{max} y_{min} y_{max} x_{eye} y_{eye} z_{eye} x_{step} y_{step} }

Pour le type de tracé YSLICE, les éléments de *VPAR* sont utilisés de la manière suivante :

- x_{left} et x_{right} sont des nombres réels qui spécifient la largeur de la vue volumique.
- y_{near} et y_{far} sont des nombres réels qui spécifient la profondeur de la vue volumique.
- z_{low} et z_{high} sont des nombres réels qui spécifient la hauteur de la vue volumique.
- x_{min} et x_{max} ne sont pas utilisés.
- y_{min} et y_{max} ne sont pas utilisés.
- x_{eye}, y_{eye} et z_{eye} sont des nombres réels qui spécifient le point dans l'espace à partir duquel le graphique est vu.
- x_{step} détermine l'intervalle séparant les valeurs x tracées au sein de chaque section.
- y_{pas} détermine le nombre de sections à tracer.

Les paramètres de traçage sont spécifiés dans la variable réservée *PPAR*, qui se présente sous la forme suivante :

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *indep res axes ptype depend* }

Pour le type de tracé YSLICE, les éléments de *PPAR* sont utilisés de la manière suivante :

- (x_{min}, y_{min}) n'est pas utilisé.
- (x_{max}, y_{max}) n'est pas utilisé.
- *indep* est un nom spécifiant la variable indépendante. La valeur par défaut de *indep* est *X*.
- *res* est un nombre réel spécifiant l'intervalle, en coordonnées exprimées en unités-utilisateur, qui sépare les valeurs tracées de la variable indépendante, ou un entier binaire spécifiant l'intervalle en pixels. La valeur par défaut est 0, soit un intervalle de 1 pixel.
- *axes* n'est pas utilisé.

- *ptype* est un nom de commande spécifiant le type de tracé. L'exécution de la commande YSLICE place YSLICE dans *ptype*.
- *depend* est un nom spécifiant la variable dépendante. La valeur par défaut est *Y*.

Commandes associées : BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME

YVOL

Commande de coordonnées volumiques y : Définit la profondeur de la vue volumique dans la variable réservée *VPAR*.

{ }

Niveau 2	Niveau 1	→	Niveau 1
y_{near}	y_{far}	→	

Accès clavier : ◀ PLOT NXT 3D VPAR YVOL

Indicateurs : Aucun

Remarques : Les variables y_{near} et y_{far} sont des nombres réels qui définissent les coordonnées y de la vue volumique utilisée dans des tracés 3D. y_{near} doit être inférieur à y_{far}. Ces valeurs sont stockées dans la variable réservée *VPAR*.

Commandes associées : EYEPT, XVOL, XXRNG, YYRNG, ZVOL

YYRNG

Commande de définition de la plage y de la zone d'entrée (domaine) : Spécifie la plage y d'une zone d'entrée (domaine) pour les tracés GRIDMAP et PARSURFACE.

{ }

Niveau 2	Niveau 1	→	Niveau 1
y_{near}	y_{far}	→	

Accès clavier : ◀ PLOT NXT 3D VPAR YYRN

Indicateurs : Aucun

Remarques : Les variables $y_{y\ near}$ et $y_{y\ far}$ sont des nombres réels qui définissent les coordonnées y de la zone d'entrée. Ces valeurs sont stockées dans la variable réservée VPAR.

Commandes associées : EYEPT, XVOL, XXRNG, YVOL, ZVOL

ZFACTOR

Fonction de facteur Z de compressibilité du gaz : Calcule le facteur de correction de compressibilité du gaz pour le comportement non idéal d'un hydrocarbure.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_{Tr}	y_{Pr}	→	$x_{Zfacteur}$
x_{Tr}	'*symb*'	→	'ZFACTOR(x_{Tr},*symb*)'
'*symb*'	y_{Pr}	→	'ZFACTOR(*symb*,y_{Pr})'
'$symb_1$'	'$symb_2$'	→	'ZFACTOR($symb_1$,$symb_2$)'

Accès clavier : ◀ EQ LIB UTILS ZFACT

Indicateurs : Résultats numériques (−3)

Remarques : x_{Tr} est la température réduite : le rapport de la température réelle (T) à la température pseudocritique (T_c) (calculez le rapport en employant des températures absolues). x_{Tr} doit être comprise entre 1,05 et 3,0.

y_{Pr} est la pression réduite : le rapport de la pression réelle (P) à la pression pseudocritique (P_c). y_{Pr} doit être comprise entre 0 et 30.

x_{Tr} et y_{Pr} doivent être des nombres réels ou des objets-unités qui se réduisent à des nombres sans dimension.

ZVOL

Commande de définition des coordonnées volumiques Z : Définit la hauteur de la vue volumique dans la variable réservée *VPAR*.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x_{low}	x_{high}	→	

Accès clavier : ◀ PLOT NXT 3D VPAR ZVOL

Indicateurs : Aucun

Remarques : x_{low} et x_{high} sont des nombres réels qui définissent les coordonnées z de la vue volumique utilisée dans des tracés 3D. Ces valeurs sont stockées dans la variable réservée *VPAR*.

Commandes associées : EYEPT, XVOL, XXRNG, YVOL, YYRNG

+

Fonction analytique d'addition : Renvoie la somme des arguments.

Niveau 2	Niveau 1	→	Niveau 1
z_1	z_2	→	z_1+z_2
[*tableau*]$_1$	[*tableau*]$_2$	→	[*tableau*]$_{1\,??+???2}$
z	'*symb*'	→	'*z*+(*symb*)'
'*symb*'	*z*	→	'*symb*+*z*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$+*symb*$_2$'
{ *liste*$_1$ }	{ *liste*$_2$ }	→	{ *liste*$_1$ *liste*$_2$ }
obj$_A$	{ *obj*$_1$ ... *obj*$_n$ }	→	{ *obj*$_A$ *obj*$_1$ ... *obj*$_n$ }
{ *obj*$_1$ ... *obj*$_n$ }	*obj*$_A$	→	{ *obj*$_1$ ... *obj*$_n$ *obj*$_A$ }
"*chaîne*$_1$"	"*chaîne*$_2$"	→	"*chaîne*$_1$ *chaîne*$_2$"
obj	"*chaîne*"	→	"*obj chaîne*"
"*chaîne*"	*obj*	→	"*chaîne obj*"
#*n*$_1$	*n*$_2$	→	#*n*$_3$
n$_1$	#*n*$_2$	→	#*n*$_3$
#*n*$_1$	#*n*$_2$	→	#*n*$_3$
x$_1$_*unité*$_1$	*y*_*unité*$_2$	→	(*x*$_2$+*y*)_*unité*$_2$
'*symb*'	*x*_*unité*	→	'*symb*+*x*_*unité*'
*x*_*unité*	'*symb*'	→	'*x*_*unité*+*symb*'
grob$_1$	*grob*$_2$	→	*grob*$_3$

Accès clavier : ⊞ +

Indicateurs : Résultats numériques (−3), Taille de mot entier binaire (−5 à −10)

Remarques : La somme d'un nombre réel a et d'un nombre complexe (x, y) est le nombre complexe $(x+a, y)$.

La somme de deux nombres complexes (x_1, y_1) et (x_2, y_2) est le nombre complexe (x_1+x_2, y_1+y_2).

La somme d'un tableau réel et d'un tableau complexe est un tableau complexe, où chaque élément x du tableau réel est traité comme

un élément complexe (x, 0). Les tableaux doivent avoir les mêmes dimensions.

La somme d'un entier binaire et d'un nombre réel est un nombre binaire correspondant à la somme des deux arguments, tronquée en fonction de la taille de mot en cours. (Le nombre réel est converti en entier binaire avant l'addition.)

La somme de deux entiers binaires est tronquée en fonction de la taille de mot entier binaire en vigueur.

La somme de deux objets-unités est un objet-unité ayant les mêmes dimensions que l'argument de niveau 1. Les unités des deux arguments doivent être compatibles.

La somme de deux objets graphiques équivaut au résultat d'une opération logique OR, sauf que les deux objets *doivent* avoir les mêmes dimensions.

Il existe une ambiguïté avec certaines unités de température. Lorsque °C ou °F représente un niveau de température (valeur lue sur un thermomètre), cette dernière est une unité avec une constante additive : 0 °C = 273.15 K, et 0 °F = 459.67 °R. Mais lorsque °C ou °F représente une *différence* de température, celle-ci est une unité sans constante additive 1 °C = 1 K et 1 °F = 1 °R.

Le calculateur suppose que les unités de température simples x_°C et x_°F représentent des niveaux lorsqu'elles sont utilisées comme arguments des fonctions <, >, ≤, ≥, == et ≠. Ainsi, pour effectuer l'opération, le calculateur convertit d'abord les valeurs Celsius en kelvins et les valeurs Fahrenheit en Rankines. (Pour d'autres fonctions ou pour des unités de température *composées*, comme x_°C/min, il suppose que les unités représentent des différences de température, de sorte qu'aucune constante additive n'est impliquée et, par conséquent, aucune conversion n'a lieu.)

Les opérateurs arithmétiques +, −, %CH et %T traitent les températures comme des différences, sans constante additive, mais les deux arguments doivent être des valeurs absolues (K et °R), des °C ou des °F. Aucune autre combinaison n'est permise.

Exemples : `{ 1 2 3 }{ A B C }` + renvoie `{ 1 2 3 A B C }`.

`5_ft 9_in` + renvoie `69_in`.

`[[ 0 1 ][ 1 3 ]] [[ 2 1 ][ 0 1 ]]` + renvoie `[[ 2 2 ][ 1 4 ]]`.

+

'PREMIER' 'SECOND' + renvoie 'PREMIER+SECOND'.

Commandes associées : −, *, /, =

−

Fonction analytique de soustraction : Renvoie la différence entre des arguments : l'objet du niveau 1 est soustrait du niveau 2.

{ }

Niveau 2	Niveau 1	→	Niveau 1
z_1	z_2	→	$z_1 - z_2$
[*tableau*]$_1$	[*tableau*]$_2$	→	[*tableau*]$_{1-2}$
z	'*symb*'	→	'*z−symb*'
'*symb*'	*z*	→	'*symb−z*'
'$symb_1$'	'$symb_2$'	→	'$symb_1 - symb_2$'
#n_1	n_2	→	#n_3
n_1	#n_2	→	#n_3
#n_1	#n_2	→	#n_3
x_1 _*unité*$_1$	*y_unité*$_2$	→	($x_2 - y$)_*unité*$_2$
'*symb*'	*x_unité*	→	'*symb−x_unité*'
x_unité	'*symb*'	→	'*x_unité−symb*'

Accès clavier : ⊖

Indicateurs : Résultats numériques (−3)

Remarques : La différence entre un nombre réel a et un nombre complexe (x, y) est $(x-a, y)$ ou $(a-x, -y)$. La différence de deux nombres complexes (x_1, y_1) et (x_2, y_2) est (x_1-x_2, y_1-y_2).

La différence entre un tableau réel et un tableau complexe est un tableau complexe, où chaque élément x du tableau réel est traité comme élément complexe $(x, 0)$. Les deux tableaux fournis en arguments doivent avoir les mêmes dimensions.

La différence entre un entier binaire et un nombre réel est un entier binaire représentant la somme du nombre de niveau 2 et du complément à deux du nombre de niveau 1. (Le nombre réel est converti en entier binaire avant la soustraction.)

La différence de deux entiers binaires est un entier binaire représentant la somme du nombre de niveau 2 et du complément à deux du nombre de niveau 1.

La différence entre deux objets-unités est un objet-unité ayant les mêmes dimensions que l'objet de niveau 1. Les unités des deux arguments doivent être compatibles.

Il existe une ambiguïté avec certaines unités de température. Lorsque °C ou °F représente un niveau de température (valeur lue sur un thermomètre), cette dernière est une unité avec une constante additive : 0 °C = 273.15 K, et 0 °F = 459.67 °R. Mais lorsque °C ou °F représente une *différence* de température, celle-ci est une unité sans constante additive 1 °C = 1 K et 1 °F = 1 °R.

Le calculateur suppose que les unités de température simples `x_°C` et `x_°F` représentent des niveaux lorsqu'elles sont utilisées comme arguments des fonctions $<$, $>$, $\leq$, $\geq$, $==$ et $\neq$. Ainsi, pour effectuer l'opération, le calculateur convertit d'abord les valeurs Celsius en kelvins et les valeurs Fahrenheit en Rankines. (Pour d'autres fonctions ou pour des unités de température *composées*, comme `x_°C/min`, il suppose que les unités représentent des différences de température, de sorte qu'aucune constante additive n'est impliquée et, par conséquent, aucune conversion n'a lieu.)

Les opérateurs arithmétiques +, −, %CH et %T traitent les températures comme des différences, sans constante additive, mais les deux arguments doivent être des valeurs absolues (K et °R), des °C ou des °F. Aucune autre combinaison n'est permise.

Exemple : `25_ft 8_in` − renvoie `292_in`.

`[[ 5 1 ][ 3 3 ]] [[ 2 1 ][ 0 1 ]]` − renvoie `[[ 3 0 ][ 3 2 ]]`.

`'TOTAL' 'PARTIE'` − renvoie `'TOTAL-PARTIE'`.

Commandes associées : +, *, /, =

*

Fonction analytique de multiplication : Renvoie le produit des arguments.

{}

Niveau 2	Niveau 1	→	Niveau 1
z_1	z_2	→	$z_1\, z_2$
[[*matrice*]]	[*tableau*]	→	[[*matrice* x *tableau*]]
z	[*tableau*]	→	[*z* x *tableau*]
[*tableau*]	*z*	→	[*tableau* x *z*]
z	'symb'	→	'z * symb'
'symb'	*z*	→	'symb * z'
$'symb_1'$	$'symb_2'$	→	$'symb_1 * symb_2'$
$\#n_1$	n_2	→	$\#n_3$
n_1	$\#n_2$	→	$\#n_3$
$\#n_1$	$\#n_2$	→	$\#n_3$
x_unité	*y_unité*	→	$xy\_unité_x$ x $unité_y$
x	*y_unité*	→	*xy_unité*
x_unité	*y*	→	*xy_unité*
'symb'	*x_unité*	→	'symb * x_unité'
x_unité	'symb'	→	'x_unité * symb'

Accès clavier : ⊠

Indicateurs : Résultats numériques (−3), Taille de mot entier binaire (−5 à −10)

Remarques : Le produit d'un nombre réel a et d'un nombre complexe $(x,\ y)$ est le nombre complexe $(xa,\ ya)$.

Le produit de deux nombres complexes $(x_1,\ y_1)$ and $(x_2,\ y_2)$ est le nombre complexe $(x_1 x_2 - y_1 y_2,\ x_1 y_2 + x_2 y_1)$.

Le produit d'un tableau réel et d'un tableau ou d'un nombre complexe est un tableau complexe. Chaque élément x du tableau réel est traité comme un élément complexe $(x,\ 0)$.

La multiplication d'une matrice (niveau 2) par un tableau (niveau 1) renvoie une matrice. La matrice doit avoir un nombre de colonnes identique au nombre de lignes du tableau de niveau 1 (ou au nombre d'éléments s'il s'agit d'un vecteur).

Bien que les vecteurs soient saisis et affichés sous la forme d'une *ligne* de nombres, le HP 48 les traite comme des matrices $n \times 1$ lors de la multiplication de matrices ou du calcul des normes de matrice.

La multiplication d'un entier binaire par un nombre réel renvoie un entier binaire représentant le produit des deux arguments, tronqué selon la taille de mot en cours. (Le nombre réel est converti en entier binaire avant la multiplication.)

Le produit de deux entiers binaires est tronqué en fonction de la taille de mot entier binaire en vigueur.

Lors de la multiplication de deux objets-unités, les parties scalaires et les parties unités sont multipliées séparément.

Commandes associées : +, −, /, =

/

Fonction analytique de division : Renvoie le quotient des arguments : l'objet de niveau 2 est divisé par l'objet de niveau 1.

{ }

Level 2	Level 1	→	Level 1
z_1	z_2	→	z_1 / z_2
[*tableau*]	[[*matrice*]]	→	[[*matrice*$^{-1}$ x *tableau*]]
[*tableau*]	*z*	→	[*tableau/z*]
z	'*symb*'	→	'*z/symb*'
'*symb*'	*z*	→	'*symb/z*'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ / $symb_2$'
#n_1	n_2	→	#n_3
n_1	#n_2	→	#n_3
#n_1	#n_2	→	#n_3
$x\_unité_1$	$y\_unité_2$	→	$(x/y)\_unité_1$ / $unité_2$
x	*y_unité*	→	*(x/y)_1/unité*
x_unité	*y*	→	*(x/y)_unité*
'*symb*'	*x_unité*	→	'*symb/x_unité*'
x_unité	'*symb*'	→	'*x_unité/symb*'

Accès clavier :

◀ SOLVE SYS /

÷

Indicateurs : Résultats numériques (−3)

Remarques : Un nombre réel a divisé par un nombre complexe (x, y) renvoie $\left(\frac{ax}{x^2+y^2}, -\frac{ay}{x^2+y^2}\right)$. Un nombre complexe ($x$, y) divisé par un nombre réel a renvoie le nombre complexe (x/a , y/a).

Un nombre complexe (x_1 , y_1) divisé par un autre nombre complexe (x_2 , y_2) renvoie le quotient complexe suivant :

$$\left(\frac{x_1x_2+y_1y_2}{{x_2}^2+{y_2}^2},\frac{y_1x_2-x_1y_2}{{x_2}^2+{y_2}^2}\right)$$

Un tableau **B** divisé par une matrice **A** résout le système d'équations **AX=B** pour **X** ; à savoir, $\mathbf{X}=\mathbf{A}^{-1}\mathbf{B}$. Cette operation utilise une précision interne à 15 chiffres, ce qui permet d'obtenir un résultat plus précis que le calcul `INV(A)*B`. La matrice doit être carrée et comporter un nombre de colonnes identique au nombre de lignes du tableau (ou au nombre d'éléments s'il s'agit d'un vecteur).

Un entier binaire divisé par un nombre réel ou binaire renvoie un entier binaire représentant la partie entière du quotient. (Le nombre réel est converti est entier binaire avant la division.) Un diviseur égal à zéro renvoie `# 0`.

Lors de la division de deux objets-unités, les parties scalaires et les parties unités sont divisées séparément.

Commandes associées : +, −, *, =

---

^

Fonction analytique d'élévation à la puissance : Renvoie la valeur de l'objet de niveau 2 élevé à la puissance spécifiée par l'objet de niveau 1.

{}

Niveau 2	Niveau 1	→	Niveau 1
w	*z*	→	w^z
z	*'symb'*	→	*'z^(symb)'*
'symb'	*z*	→	*'(symb)^z'*
$'symb_1'$	$'symb_2'$	→	$'symb_1\ \hat{}\ (symb_2)'$
x_unité	*y*	→	x^y*_unité*^{y}
x_unité	*'symb'*	→	*'(x_unité)^(symb)'*

Accès clavier : y^x

Indicateurs : Solution principale (−1), Résultats numériques (−3)

Remarques : Si l'un des arguments est complexe, le résultat est complexe.

Les coupures et les relations inverses pour w^z sont déterminés par la relation suivante :

$$w^z = \exp(z(\ln\ w))$$

Commandes associées : EXP, ISOL, LN, XROOT

<

Fonction Inférieur à : Vérifie si un objet est inférieur à un autre.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	0/1
#n_1	#n_2	→	0/1
"*chaîne*$_1$"	"*chaîne*$_2$"	→	0/1
x	'*symb*'	→	'*x*<*symb*'
'*symb*'	*x*	→	'*symb*<*x*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$<*symb*$_2$'
x_unité$_1$	*y_unité*$_2$	→	0/1
x_unité	'*symb*'	→	'*x_unité*<*symb*'
'*symb*'	*x_unité*	→	'*symb*<*x_unité*'

Accès clavier : PRG TEST <

Indicateurs : Résultats numériques (−3)

Remarques : La fonction < renvoie un résultat de test vrai (1) si l'argument de niveau 2 est inférieur à celui de niveau 1 ; sinon elle renvoie un résultat de test faux (0).

Si un objet est symbolique (nom ou expression algébrique) et si l'autre est un nombre, un objet symbolique ou un objet-unité, la fonction <

renvoie une expression de comparaison symbolique qu'il est possible d'évaluer pour obtenir un résultat de test.

En ce qui concerne des nombres réels et des entiers binaires, "inférieur à" signifie "plus petit numériquement" (1 est inférieur à 2). En outre, pour des nombres réels, la notion de valeur plus négative intervient (-2 est inférieur à -1).

Pour des chaînes, l'infériorité fait intervenir le classement alphabétique : "ABC" est inférieur à "DEF" ; "AAA" est inférieur à "AAB" ; "A" est inférieur à "AA". En général, les caractères sont classés selon leur codes. Ainsi, "B" est inférieur à "a", car "B" a le code de caractère 66, et "a" le code 97.

Pour des objets-unités, les deux objets doivent avoir des dimensions compatibles, et ils sont convertis dans des unités communes pour que la comparaison soit possible. Si vous utilisez des unités de température simples, le calculateur suppose que les valeurs représentent des niveaux et non des différences de température. En revanche, il interprète les unités de température composées comme des différences. Pour de plus amples informations sur l'emploi d'unités de température dans des fonctions arithmétiques, reportez-vous à la fonction +.

Commandes associées : $\leq$, $>$, $\geq$, $==$, $\neq$

≤

Fonction Inférieur ou égal à : Vérifie si un objet donné est inférieur ou égal à un autre.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	0/1
#n_1	#n_2	→	0/1
"*chaîne*$_1$"	"*chaîne*$_2$"	→	0/1
x	'*symb*'	→	'*x* ≤ *symb*'
'*symb*'	*x*	→	'*symb* ≤ *x*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$ ≤ *symb*$_2$'
x_unité$_1$	*y_unité*$_2$	→	0/1
x_unité	'*symb*'	→	'*x_unité* ≤ *symb*'
'*symb*'	*x_unité*	→	'*symb* ≤ *x_unité*'

Accès clavier : PRG TEST ≤

Indicateurs : Résultats numériques (−3)

Remarques : La fonction ≤ renvoie un résultat de test vrai (1) si l'argument de niveau 2 est inférieur ou égal à celui du niveau 1, et un résultat faux (0) dans le cas contraire.

Si un objet est symbolique (nom ou expression algébrique) et si l'autre est un nombre, un objet symbolique ou un objet-unité, ≤ renvoie une expression de comparaison symbolique qu'il est possible d'évaluer pour obtenir un résultat de test.

En ce qui concerne des nombres réels et des entiers binaires, "inférieur ou égal à" signifie "plus petit ou égal numériquement" (1 est inférieur à 2). En outre, pour des nombres réels, la notion de valeur plus négative s'y ajoute (−2 est inférieur à −1).

Pour des chaînes, cette fonction fait intervenir le classement alphabétique : "ABC" est inférieur ou égal à "DEF" ; "AAA" est inférieur ou égal à "AAB" ; "A" est inférieur ou égal à "AA". En général, les caractères sont classés selon leur codes. Ainsi, "B" est inférieur à "a", car "B" a le code de caractère 66, et "a" le code 97.

Pour des objets-unités, les deux objets doivent avoir des dimensions compatibles, et ils sont convertis dans des unités communes pour que la comparaison soit possible. Si vous utilisez des unités de température simples, le calculateur suppose que les valeurs représentent des niveaux et non des différences de température. En revanche, il interprète les

unités de température composées comme des différences. Pour de plus amples informations sur l'emploi d'unités de température dans des fonctions arithmétiques, reportez-vous à la fonction +.

Commandes associées : <, >, ≥, ==, ≠

>

Fonction Supérieur à : Vérifie si un objet est supérieur à un autre.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	0/1
#n_1	#n_2	→	0/1
"*chaîne*$_1$"	"*chaîne*$_2$"	→	0/1
x	'*symb*'	→	'*x*>*symb*'
'*symb*'	*x*	→	'*symb*>*x*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$>*symb*$_2$'
x_unité$_1$	*y_unité*$_2$	→	0/1
x_unité	'*symb*'	→	'*x_unité*>*symb*'
'*symb*'	*x_unité*	→	'*symb*>*x_unité*'

Accès clavier : PRG TEST >

Indicateurs : Résultats numériques (−3)

Remarques : La fonction > renvoie un résultat de test vrai (1) si l'argument de niveau 2 est supérieur à celui de niveau 1 ; sinon elle renvoie un résultat de test faux (0).

Si un objet est symbolique (nom ou expression algébrique) et si l'autre est un nombre, un objet symbolique ou un objet-unité, > renvoie une expression de comparaison symbolique qu'il est possible d'évaluer pour obtenir un résultat de test.

Pour des nombres réels et des entiers binaires, "supérieur à" signifie "plus grand numériquement" (2 est supérieur à 1). En outre, pour des

>

nombres réels, la notion de valeur moins négative intervient (−1 est supérieur à −2).

Pour des chaînes, cette fonction fait intervenir le classement alphabétique : "DEF" est supérieur "ABC" ; "AAB" est supérieur à "AAA" ; "AA" est supérieur à "A". En général, les caractères sont classés selon leur codes. Ainsi, "a" est supérieur à "B", car "B" a le code de caractère 66, et "a" le code 97.

Pour des objets-unités, les deux objets doivent avoir des dimensions compatibles, et ils sont convertis dans des unités communes pour que la comparaison soit possible. Si vous utilisez des unités de température simples, le calculateur suppose que les valeurs représentent des niveaux et non des différences de température. En revanche, il interprète les unités de température composées comme des différences. Pour de plus amples informations sur l'emploi d'unités de température dans des fonctions arithmétiques, reportez-vous à la fonction +.

Commandes associées : <, ≤, ≥, ==, ≠

≥

Fonction Supérieur ou égal à : Vérifie si un objet est supérieur ou égal à un autre.

{ }

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	0/1
#n_1	#n_2	→	0/1
"*chaîne*$_1$"	"*chaîne*$_2$"	→	0/1
x	'*symb*'	→	'*x* ≥ *symb*'
'*symb*'	*x*	→	'*symb* ≥ *x*'
'*symb*$_1$'	'*symb*$_2$'	→	'*symb*$_1$ ≥ *symb*$_2$'
x_unité$_1$	*y_unité*$_2$	→	0/1
x_unité	'*symb*'	→	'*x_unité* ≥ *symb*'
'*symb*'	*x_unité*	→	'*symb* ≥ *x_unité*'

Accès clavier : (PRG) TEST ≥

Indicateurs : Résultats numériques (−3)

Remarques : La fonction ≥ renvoie un résultat de test vrai (1) si l'argument de niveau 2 est supérieur ou égal à celui du niveau 1 ; sinon elle renvoie un résultat de test faux (0).

Si un objet est symbolique (nom ou expression algébrique) et si l'autre est un nombre, un objet symbolique ou un objet-unité, ≥ renvoie une expression de comparaison symbolique qu'il est possible d'évaluer pour obtenir un résultat de test.

Pour des nombres réels et des entiers binaires, "supérieur ou égal à" signifie "plus grand ou égal numériquement" (2 est supérieur ou égal à 1). En outre, pour des nombres réels, la notion de valeur moins négative ou identique intervient (−1 est supérieur ou égal à −2).

Pour des chaînes, cette fonction fait intervenir le classement alphabétique : "DEF" est supérieur ou égal à "ABC" ; "AAB" est supérieur ou égal à "AAA" ; "AA" est supérieur ou égal à "A". En général, les caractères sont classés selon leur codes. Ainsi, "a" est supérieur ou égal à "B", car "B" a le code de caractère 66, et "a" le code 97.

Pour des objets-unités, les deux objets doivent avoir des dimensions compatibles, et ils sont convertis dans des unités communes pour que la comparaison soit possible. Si vous utilisez des unités de température simples, le calculateur suppose que les valeurs représentent des niveaux et non des différences de température. En revanche, il interprète les unités de température composées comme des différences. Pour de plus amples informations sur l'emploi d'unités de température dans des fonctions arithmétiques, reportez-vous à la fonction +.

Commandes associées : <, ≤, >, ==, ≠

=

Fonction analytique d'égalité : Renvoie une équation formée de deux argments.

{ }

Niveau 2	Niveau 1	→	Niveau 1
z_1	z_2	→	'$z_1 = z_2$'
z	'symb'	→	'z=symb'
'symb'	z	→	'symb=z'
'$symb_1$'	'$symb_2$'	→	'$symb_1 = symb_2$'
y	x_unité	→	'y=x_unité'
y_unité	x	→	'y_unité=x'
y_unité	x_unité	→	'y_unité=x_unité'
'symb'	x_unité	→	'symb=x_unité'
x_unité	'symb'	→	'x_unité=symb'

Accès clavier : ↶ =

Indicateurs : Résultats numériques (−3)

Remarques : Le signe "égale" met deux expressions en correspondance de telle sorte que la différence entre les deux soit nulle.

En mode Résultats symboliques, le résultat est une équation algébrique. En mode Résultats numériques, il représente la différence entre les deux arguments car = agit comme −. Ceci permet d'utiliser indifféremment des expressions et des équations comme arguments pour des extracteurs de racines symboliques et numériques.

L'évaluation numérique d'une équation avec l'application Solver HP suppose la soustraction des termes. Pour de plus amples informations sur les effets de la soustraction, reportez-vous à la fonction "−".

Il existe une ambiguïté avec certaines unités de température. Lorsque °C ou °F représente un niveau de température (valeur lue sur un thermomètre), cette dernière est une unité avec une constante additive : 0 °C = 273.15 K, et 0 °F = 459.67 °R. Mais lorsque °C or °F

représente une *différence* de température, celle-ci est une unité sans constante additive 1 °C = 1 K et 1 °F = 1 °R.

Les opérateurs arithmétiques +, −, %CH et %T traitent les températures comme des différences, sans constante additive. Cependant, +, −, %CH et %T exigent que les deux arguments soient des valeurs absolues (K et °R), des °C ou des °F. Aucune autre combinaison n'est permise.

Commandes associées : DEFINE, EVAL, −

==

Fonction d'égalité logique : Vérifie si deux objets sont égaux.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj_1	obj_2	→	0/1
(x,0)	x	→	0/1
x	(x,0)	→	0/1
z	'symb'	→	'z==symb'
'symb'	z	→	'symb==z'
'$symb_1$'	'$symb_2$'	→	'$symb_1$ ==$symb_2$'

Accès clavier : PRG TEST ==

Indicateurs : Résultats numériques (−3)

Remarques : La fonction == renvoie un résultat vrai (1) si les deux objets sont du même type et de même valeur ; sinon elle renvoie un résultat de test faux (0). Les listes et les programmes sont considérés comme ayant la même valeur si les objets qu'ils contiennent sont identiques.

Si un objet est de type algébrique ou s'il s'agit d'un nom et si l'autre est un nombre (réel ou complexe) ou une expression algébrique, ==

==

renvoie une expression de comparaison symbolique qu'il est possible d'évaluer pour obtenir un résultat de test.

La fonction == est utilisée pour des comparaisons, alors que = sépare les deux membres d'une équation.

Si la partie imaginaire d'un nombre complexe est 0, elle est ignorée lorsque le nombre complexe est comparé à un nom réel : par exemple `6 (6,0) ==` renvoie 1.

Pour des objets-unités, les deux objets doivent avoir des dimensions compatibles, et ils sont convertis dans des unités communes pour que la comparaison soit possible. Si vous utilisez des unités de température simples, le calculateur suppose que les valeurs représentent des niveaux et non des différences de température. En revanche, il interprète les unités de température composées comme des différences. Pour de plus amples informations sur l'emploi d'unités de température dans des fonctions arithmétiques, reportez-vous à la fonction +.

Commandes associées : SAME, TYPE, <, ≤, >, ≥, ≠

≠

Fonction différent de : Vérifie si deux objets sont différents.

{ }

Niveau 2	Niveau 1	→	Niveau 1
obj_1	obj_2	→	0/1
(x,0)	x	→	0/1
x	(x,0)	→	0/1
z	'symb'	→	'z ≠ symb'
'symb'	z	→	'symb ≠ z'
$'symb_1'$	$'symb_2'$	→	$'symb_1 \neq symb_2'$

Accès clavier : PRG TEST ≠

Indicateurs : Résultats numériques (−3)

Remarques : La fonction ≠ renvoie un résultat vrai (1) si les deux objets ont des valeurs différentes ; sinon elle renvoie un résultat faux (0). Les listes et les programmes sont considérés comme ayant les mêmes valeurs si les objets qu'ils contiennent sont identiques.

Si un objet est de type algébrique ou s'il s'agit d'un nom et si l'autre est un nombre (réel ou complexe) ou une expression algébrique, ≠ renvoie une expression de comparaison symbolique qu'il est possible d'évaluer pour obtenir un résultat de test.

Si la partie imaginaire d'un nombre complexe est 0, elle est ignorée lorsque le nombre complexe est comparé à un nom réel : par exemple `6 (6,0) ≠` renvoie `0`.

Pour des objets-unités, les deux objets doivent avoir des dimensions compatibles, et ils sont convertis dans des unités communes pour que la comparaison soit possible. Si vous utilisez des unités de température simples, le calculateur suppose que les valeurs représentent des niveaux et non des différences de température. En revanche, il interprète les unités de température composées comme des différences. Pour de plus amples informations sur l'emploi d'unités de température dans des fonctions arithmétiques, reportez-vous à la fonction +.

Commandes associées : SAME, TYPE, <, ≤, >, ≥, ==

!

Fonction de factorielle (Gamma) : Renvoie la factorielle $n!$ d'un entier positif n, ou la fonction gamma $\Gamma(x+1)$ d'un argument non entier x.

{ }

Niveau 1	→	Niveau 1
n	→	*n*!
x	→	Γ(*x*+1)
'*symb*'	→	'(*symb*!)'

Accès clavier : MTH NXT PROB !

!

Indicateurs : Résultats numériques (−3), Exception de dépassement de capacité inférieure (−20), Exception de dépassement de capacité (−21)

Remarques : Pour $x \geq 253.1190554375$ ou $n < 0$, ! entraîne une exception de dépassement de capacité (si l'indicateur −21 est armé, l'exception est traitée comme une erreur). Pour un non-entier $x \leq -254.1082426465$, ! entraîne une exception de dépassement de capacité inférieure (si l'indicateur −20 est armé, l'exception est traitée comme une erreur).

En syntaxe algébrique, ! suit son argument. Ainsi, la syntaxe algébrique de la factorielle de 7 est `'7!'`.

Pour des arguments non entiers x, $x! = \Gamma(x + 1)$, définie pour $x > -1$ comme suit :

$$\Gamma(x+1) = \int_0^\infty e^{-t} t^x dt$$

et définie pour d'autres valeurs de x par prolongement analytique :

$$\Gamma(x+1) = n \cdot \Gamma(x)$$

Commandes associées : COMB, PERM

∫

Fonction d'intégration : Intègre une *fonction* de la *limite inférieure* à la *limite supérieure* par rapport à une variable spécifiée d'intégration.

{ }

Niveau 4	Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
limite inf.	*limite sup.*	*fonction*	*'nom'*	→	*'symb$_{intégrale}$'*

Accès clavier : ▶ ∫

Indicateurs : Résultats numériques (−3), Format d'affichage numérique (−45 à −50)

Remarques : La syntaxe algébrique pour ∫ suit la syntaxe de la pile :

∫(limite inf., limite sup., fonction, nom)

où *limite inf.*, *limite sup.* et *fonction* peuvent être des nombres réels ou complexes, des objets-unités, des noms ou des expressions algébriques.

L'évaluation de ∫ en mode de Résultats symboliques (indicateur −3 *désarmé*) renvoie un résultat symbolique au niveau 1. Le HP 48 effectue l'intégration symbolique par *mise en correspondance des configurations*. Le HP 48 peut intégrer :

- Toutes les fonctions intégrées dont les primitives peuvent être exprimées par d'autres fonctions intégrées. Par exemple, l'intégration de SIN est possible car sa primitive COS est une fonction intégrée. Les arguments de ces fonctions doivent être linéaires.
- Des sommes, des différences et des opposés de fonctions intégrées dont les primitives peuvent être exprimées par d'autres fonctions intégrées. Par exemple, `'SIN(X)-COS(X)'`.
- Les dérivées de toutes les fonctions intégrées. Par exemple, l'intégration de `'INV(1+X^2)'` est possible car elle est la dérivée de la fonction ATAN.
- Des polynômes dont le terme de base est linéaire. Par exemple, `'X^3+X^2-2*X+6'` peut être intégré puisque X est un terme linéaire. Mais `'(X^2-6)^3+(X^2-6)^2'` ne peut pas l'être car `X^2-6` n'est pas linéaire.
- Des configurations sélectionnées composées de fonctions dont les primitives peuvent être exprimées par d'autres fonctions intégrées. Par exemple, `'1/(COS(X)*SIN(X))'` renvoie `'LN(TAN(X))'`.

Si le résultat de l'intégration est une expression sans signe d'intégration dans le résultat, l'intégration symbolique a été réussie. Dans le cas contraire, réorganisez l'expression et procédez à une nouvelle évaluation, ou estimez la réponse par intégration numérique.

Un résultat correct d'intération symbolique se présente comme suit :

'résultat|(nom=limite sup.)−(résultat|(nom=limite inf.))'

∫

Reportez-vous à la fonction | (remplacement) pour plus d'informations sur ses fonctionnalités. Une seconde évaluation remplace les limites d'intégration dans la variable d'intégration, terminant ainsi la procédure.

L'évaluation de ∫ en mode Résultats numériques (indicateur −3 *armé*) renvoie une approximation numérique et stocke l'erreur d'intégration dans la variable *IERR*. ∫ consulte l'indicateur de format numérique pour déterminer la précision de calcul à adopter.

Exemples : En mode Résultats symboliques, (indicateur −3 désarmé), cette séquence de commandes :

```
1 2 '10*X' 'X' ∫
```

renvoie

```
'10*(X^2/2)|(X=2)-(10*(X^2/2)|(X=1))'
```

Une nouvelle évaluation remplace les limites d'intégration et renvoie 15.

En mode Résultats numériques (indicateur −3 armé), la séquence de commandes ci-dessus renvoie l'approximation numérique 15. En outre, la variable *IERR* est créée et contient l'erreur d'intégration .00000000015.

Commandes associées : TAYLR, ∂, Σ

∂

Fonction de dérivée : Prend la dérivée d'une expression, d'un nombre ou d'un objet-unité par rapport à une variable spécifiée de différenciation.

{ }

Niveau 2	Niveau 1	→	Niveau 1
'*symb*$_1$'	'*nom*'	→	'*symb*$_2$'
z	'*nom*'	→	0
x_unité	'*nom*'	→	0

Accès clavier : → ∂

Indicateurs : Résultats numériques (−3)

Remarques : Exécutée avec la syntaxe de la pile, ∂ exécute une différenciation *complète* : l'expression '*symb*$_1$' est évaluée jusqu'à ce qu'elle ne contienne aucune dérivée. Durant ce processus, si la variable de différenciation *nom* a une valeur, la forme finale de l'expression remplace toutes les occurrences de la variable par cette valeur.

La syntaxe algébrique de ∂ est '∂*nom*(*symb*$_1$)'. En syntaxe algébrique, ∂ exécute une différenciation *progressive* de *symb*$_1$, selon la règle de différenciation par enchaînement : le résultat d'une évaluation de l'expression est la dérivée de l'argument-expression *symb*$_1$, multipliée par une nouvelle sous-expression qui représente la dérivée de l'argument de *symb*$_1$.

Si ∂ est appliquée à une fonction pour laquelle le HP 48 ne fournit pas de dérivée, ∂ renvoie une nouvelle fonction dont le nom est *der* suivi du nom de la fonction initiale.

Exemple : En mode radians, la séquence de commandes `'∂X(SIN(Y))' EVAL` renvoie `'COS(Y)*∂X(Y)'`.

Lorsque *Y* a la valeur `X^2`, la séquence de commandes `'SIN(Y)' 'X' ∂` renvoie `'COS(X^2)*(2*X)'`. La différenciation a été exécutée dans la syntaxe de la pile, de sorte que toutes les étapes ont été effectuées en une seule opération.

Commandes associées : TAYLR, ∫, Σ

%

Fonction de pourcentage : Renvoie *x* (niveau 2) pourcent de *y* (niveau 1).

{}

Niveau 2	Niveau 1	→	Niveau 1
x	*y*	→	*xy/100*
x	*'symb'*	→	*'%(x,symb)'*
'symb'	*x*	→	*'%(symb,x)'*
'$symb_1$'	*'$symb_2$'*	→	*'%($symb_1$, $symb_2$)'*
x	*y_unité*	→	*(xy/100)_unité*
x_unité	*y*	→	*(xy/100)_unité*
'symb'	*x_unité*	→	*'%(symb,x_unité)'*
x_unité	*'symb'*	→	*'%(x_unité,symb)'*

Accès clavier : (MTH) REAL %

Indicateurs : Résultats numériques (−3)

Remarques : Il existe une ambiguïté avec certaines unités de température. Lorsque °C ou °F représente un niveau de température (valeur lue sur un thermomètre), cette dernière est une unité avec une constante additive : 0 °C = 273.15 K, et 0 °F = 459.67 °R. Mais lorsque °C ou °F représente une *différence* de température, celle-ci est une unité sans constante additive 1 °C = 1 K et 1 °F = 1 °R.

Les opérateurs arithmétiques +, −, %CH et %T traitent les températures comme des différences, sans constante additive, mais les deux arguments doivent être des valeurs absolues (K et °R), des °C ou des °F. Aucune autre combinaison n'est permise.

Pour de plus amples informations sur l'emploi d'unités de température avec des fonctions arithmétiques, voir la fonction +.

Exemple : `23.7 995 %` renvoie `235.815`.

`15 176_kg %` renvoie `26.4_kg`.

`100_°C 50 %` renvoie `50_°C`.

Commandes associées : %CH, %T

π

Fonction π : Renvoie la constante symbolique 'π' ou sa représentation numérique 3.14159265359.

Niveau 1	→	Niveau 1
	→	'π'
	→	3.14159265359

Accès clavier : ⟲ π

Indicateurs : Constantes symboliques (−2), Résultats numériques (−3)

L'évaluation de π renvoie sa représentation numérique si l'indicateur −2 ou −3 est armé ; sinon, elle renvoie sa représentation symbolique.

Remarques : Le nombre renvoyé pour π est la plus proche approximation de la constante π avec une précision à 12 chiffres.

En mode radians et avec les indicateurs −2 et −3 désarmés (afin d'obtenir un résultat symbolique), les fonctions trigonométriques de π et de π/2 sont automatiquement simplifiées. Par exemple, l'évaluation de 'SIN(π)' renvoie zéro. Cependant, si l'indicateur −2 ou −3 est armé (pour obtenir un résultat numérique), l'évaluation de 'SIN(π)' renvoie l'approximation numérique -2.06761537357E-13.

Commandes associées : e, i, MAXR, MINR, →Qπ

Σ

Fonction de sommation : Calcule la valeur d'une série finie.

{ }

Niveau 4	Niveau 3	Niveau 2	Niveau 1	→	Niveau 1
'indx'	x_{init}	x_{final}	*smnd*	→	x_{somme}
'indx'	*'init'*	x_{final}	*smnd*	→	*'Σ(indx=init,*x_{final}*,smnd)'*
'indx'	x_{init}	*'final'*	*smnd*	→	*'Σ(indx=*x_{init}*,final,smnd)'*
'indx'	*'init'*	*'final'*	*smnd*	→	*'Σ(indx=init,final,smnd)'*

Accès clavier : ▶ Σ

Indicateurs : Constantes symboliques (−2), Résultats numériques (−3)

Remarques : L'argument utilisé comme opérande de la somme, *smnd*, peut être un nombre réel ou complexe, un objet-unité, un nom global ou local, ou un objet algébrique.

La syntaxe algébrique pour Σ diffère de la syntaxe de la pile. Elle est la suivante :

'Σ(index=initial,final,smnd)'

Exemples : En mode Résultats symboliques, (indicateurs −2 et −3 désarmés), la séquence de commandes `'N' 1 5 'A^N' Σ` renvoie `'A+A^2+A^3+A^4+A^5'`.

La séquence de commandes `'N' 1 'M' 'A^N' Σ` renvoie `'Σ(N=1,M,A^N)'`.

Commandes associées : TAYLR, ∫, ∂

Σ+

Commande Sigma Plus : Ajoute un ou plusieurs points de données à la matrice statistique en cours (variale réservée ΣDAT).

Niv m ... Niv 2	Niv 1	→	Niv 1
	x	→	
	$[\ x_1\ x_2 \cdots\ x_m\ ]$	→	
	$[[\ x_{11} \cdots\ x_{1m}\ ]\ [\ x_{n1} \cdots\ x_{nm}\ ]]$	→	
$x_1 \cdots\ x_{m-1}$	x_m	→	

Accès clavier : ◀ STAT DATA Σ+

Indicateurs : Aucun

Remarques : Pour une matrice statistique de m colonnes, la saisie des arguments de Σ+ diffère selon les cas :

- Pour saisir un point de données avec une seule valeur de coordonnées, l'argument de Σ+ doit être un nombre réel.
- Pour saisir un point de données avec plusieurs valeurs de coordonnées, l'argument de Σ+ doit être un vecteur de m valeurs réelles de coordonnées.
- Pour saisir plusieurs points de données, l'argument de Σ+ doit être une matrice de n lignes de m valeurs réelles de coordonnées.

Dans tous les cas, les valeurs de coordonnées des points de données sont ajoutées à la matrice statistique en cours (variable réservée ΣDAT) comme autant de lignes nouvelles. Si ΣDAT n'existe pas, Σ+ crée une matrice $n \times m$ et la stocke dans ΣDAT. Si ΣDAT existe, il se produit une erreur si elle ne contient pas une matrice réelle ou si le nombre des valeurs de coordonnées pour chaque point entré avec Σ+ ne correspond pas au nombre de colonnes de la matrice statistique en cours.

Une fois que ΣDAT existe, les points de données des coordonnées m peuvent être saisis comme m nombres réels distincts ou comme un vecteur de m éléments.

Σ+

Dans les deux cas, LASTARG renvoie un vecteur de m éléments.

Exemple : La séquence `CLΣ [ 2 3 4 ] Σ+ 3 1 7 Σ+` crée la matrice `[[ 2 3 4 ][ 3 1 7 ]]` dans ΣDAT.

Commandes associées : CLΣ, RCLΣ, STOΣ, Σ−

Σ−

Commande Sigma moins : Renvoie un vecteur de m nombres réels (ou un nombre x si $m = 1$) correspondant aux valeurs de coordonnées du dernier point entré par Σ+ dans la matrice statistique en cours (variable réservée ΣDAT).

	→	Niveau 1
	→	x
	→	$[\ x_1\ x_2\ \ldots\ x_m\ ]$

Accès clavier : ◀ STAT DATA Σ−

Indicateurs : Aucun

Remarques : La dernière ligne de la matrice statistique est supprimée.

Le vecteur renvoyé par Σ− peut être modifié ou remplacé, puis rétabli dans la matrice statistique par Σ+.

Commandes associées : CLΣ, RCLΣ, STOΣ, Σ+

√

Fonction analytique de racine carrée : Renvoie la racine carrée (positive) de l'argument.

{ }

Niveau 1	→	Niveau 1
z	→	$\sqrt{z}$
x_unité	→	$\sqrt{x}_{unit^1}/2$
'symb'	→	$'\sqrt{(symb)}'$

Accès clavier : (√x)

Indicateurs : Solution principale (−1), Résultats numériques (−3)

Remarques : Pour un nombre complexe (x_1, y_1), la racine carrée est le nombre complexe suivant :

$$(x_2, y_2) = (\sqrt{r}\, cos\frac{\theta}{2}, \sqrt{r}\, sin\frac{\theta}{2})$$

où r = ABS (x_1, y_1) et θ = ARG (x_1, y_1).

Si $(x_1, y_1) = (0, 0)$, la racine carrée est (0, 0).

L'inverse de SQ est une *relation*, non une fonction, puisque SQ envoie plusieurs arguments au même résultat. La relation inverse pour SQ est exprimée par ISOL comme la *solution générale* suivante :

```
'≤1*√Z'
```

La fonction √ est l'inverse d'une *partie* de SQ, partie définie par restriction du domaine de SQ de telle sorte que 1) chaque argument soit envoyé à un résultat distinct, et que 2) chaque résultat possible soit obtenu. Les points situés dans ce domaine restreint de SQ constituent les *valeurs principales* de la relation inverse. La fonction √ dans sa totalité est appelée la *détermination principale* de la relation inverse, et les points envoyés par √ à la frontière du domaine restreint de SQ forment les *coupures* de √.

La détermination principale utilisée par le HP 48 pour √ a été choisie car elle est analytique dans les régions où les arguments de la fonction

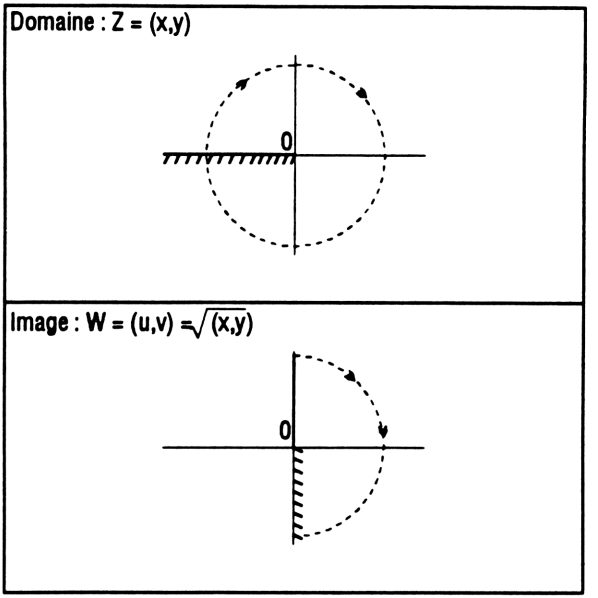
inverse à *valeurs réelles* sont définis. La coupure pour la fonction de racine carrée à valeur complexe apparaît dans la région où la fonction correspondante à valeurs réelles n'est pas définie. La détermination principale conserve également la plupart des symétries importantes.

Les graphiques ci-dessous montrent le domaine et l'image de √. Le graphique représentant le domaine indique l'emplacement de la coupure : le trait plein marque un bord de la coupure, les hachures marquent l'autre bord. Le graphique représentant l'image montre l'emplacement où chaque bord de la coupure est projeté par la fonction.

Ces graphiques montrent la relation inverse `'≤1*√Z'` dans le cas où *s*=1. Pour d'autres valeurs de *s1*, le demi-plan du graphique inférieur pivote. La réunion des demi-plans couvre la totalité du plan complexe, qui est le domaine de SQ.

Il suffit de regarder ces graphiques en inversant le domaine et l'image pour voir comment le domaine de SQ est restreint de façon à ce qu'une *fonction* inverse soit possible. Considérez le demi-plan du graphique inférieur comme le domaine restreint `Z = (x, y)`. SQ projette ce domaine sur la totalité du plan complexe dans l'image `W = (u, v) = SQ(x, y)` du graphique supérieur.

Commandes associées : SQ, ⍣, ISOL



_

Fonction d'association d'unité : Associe une expression-unité à un nombre réel. Cette fonction est automatiquement exécutée dans le catalogue des unités (→ UNITS).

Niveau 2	Niveau 1	→	Niveau 1
x	*'expression-unité'*	→	*x_unité*

Accès clavier : → _

Indicateurs : Aucun

Commande associée : →UNIT

| (Remplacement)

Fonction de remplacement : Remplace les noms d'une expression par des valeurs.

Niveau 2	Niveau 1	→	Niveau 1
'$symb_{ancien}$'	{ nom_1 '$symb_1$' nom_2 '$symb_2$' ... }	→	'$symb_{nouv}$'
x	{ nom_1 '$symb_1$' nom_2 '$symb_2$' ... }	→	x
(x, y)	{ nom_1 '$symb_1$' nom_2 '$symb_2$' ... }	→	(x, y)

Accès clavier : ◀ SYMBOLIC NXT |

Indicateurs : Résultats numériques (−3)

Remarques : | est principalement utilisée dans des objets algébriques, avec la syntaxe suivante :

'$symb_{ancien}$ | ‹ nom_1=$symb_1$, nom_2=$symb_2$ ... ›'

Elle permet à des objets algébriques de contenir, sous forme de variables, des informations de substitution pour les noms. Les fonctions symboliques qui diffèrent l'évaluation des noms (comme $\int$ et ∂) peuvent ensuite extraire ces informations des variables locales et les inclure dans l'expression. Cette procédure évite le problème qui surgirait inévitablement si les variables locales n'existaient plus au moment de l'évaluation des noms.

Commandes associées : APPLY, QUOTE

→

Commande de création de variables locales : Crée des variables locales.

Niveau n ... Niveau 1	→	Niveau 1
obj_1 ... obj_n	→	

Accès clavier : → →

Indicateurs : Aucun

Remarques : Les *structures de variables locales* spécifient une ou plusieurs variables locales et une procédure de définition.

Une structure de variable locale se compose de la commande → suivie d'un ou plusieurs noms, puis d'une procédure de définition (un programme ou une expression algébrique). La commande → stocke les objets de la pile dans des variables locales en utilisant les noms spécifiés. Les *variables locales* résultantes n'existent que pour la durée d'exécution de la procédure de définition. La syntaxe d'une structure de variable locale présente l'une des formes suivantes :

- → *nom*$_1$ *nom*$_2$ ... *nom*$_n$ « *programme* »
- → *nom*$_1$ *nom*$_2$ ... *nom*$_n$ '*expression algébrique*'

Exemple : Le programme

« → x y « x y * x y - + » »

prend un objet du niveau 2 et le stocke dans la variable locale x, prend un objet du niveau 1 et le stocke dans la variable locale y, et exécute des calculs avec x et y dans la procédure de définition (ici, un programme). Lorsque la procédure prend fin, les variables locales x et y sont supprimées.

Fonctions-utilisateur Une fonction-utilisateur est une variable contenant un programme composé uniquement d'une structure de variable locale.

Par exemple, la variable A, contenant le programme suivant :

→

```
« → x y z 'x*y/2+z' »
```

est une fonction-utilisateur. A l'instar d'une fonction intégrée, elle peut prendre ses arguments en syntaxe de la pile ou en syntaxe algébrique, et accepte aussi des arguments symboliques. En outre, une fonction-utilisateur est différenciable si sa procédure de définition est une expression algébrique contenant uniquement des fonctions différenciables.

Commandes associées : DEFINE, STO

4

Equations

La bibliothèque d'équations se compose de 15 rubriques et de plus de 100 titres, dotés chacun d'un numéro que vous pouvez utiliser avec la commande SOLVEQN pour spécifier le jeu d'équations. Ce numéro est indiqué entre parenthèses.

Veuillez vous reporter à la fin de cette section pour les références données pour chaque rubrique (Référence : x). Consultez ces références ou d'autres ouvrages pour déterminer les conditions préalables et les limitations de ces équations.

Les solutions données dans les exemples ont été arrondies à quatre décimales.

Colonnes et poutres (Columns and Beams: 1)

Noms de variable et description

ϵ	Décentrage (décalage) de la charge
σcr	Contrainte critique
σmax	Contrainte maximale
θ	Pente au point x
A	Section transversale
a	Distance par rapport au point d'application de la charge
c	Distance par rapport au bord (Colonnes excentriques), ou Distance par rapport au moment appliqué (poutres)
E	Coefficient d'élasticité
I	Moment d'inertie
K	Coefficient de longueur utile d'une colonne
L	Longueur d'une colonne ou d'une poutre
M	Moment appliqué
Mx	Moment de flexion interne au point x
P	Charge (Colonnes excentriques), ou Charge ponctuelle (poutres)
Pcr	Charge critique
r	Rayon de giration
V	Effort de cisaillement au point x
w	Charge répartie
x	Distance le long de la poutre
y	Flexion au point x

Pour des poutres simplement supportées et en porte-à-faux ("Flexion simple" à "Cisaillement par porte-à-faux"), les calculs varient selon la position de x par rapport aux charges.

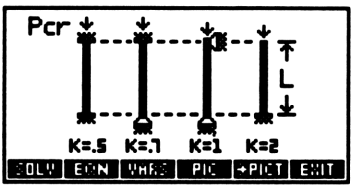
- Les charges appliquées sont positives vers le bas,
- le moment appliqué est positif dans le sens trigonométrique,
- la flexion est positive vers le haut,

- le pente est positive dans le sens trigonométrique,
- le moment de flexion interne est positif dans le sens trigonométrique à la gauche du point d'application de la charge
- et l'effort de cisaillement est positif vers le bas à la gauche du point d'application de la charge.

Référence : 2.

Déformation élastique (Elastic Buckling: 1, 1)

Ces équations s'appliquent à une colonne mince ($K \cdot L/r > 100$) avec un coefficient de longueur K.



Equations :

$$Pcr = \frac{\pi^2 \cdot E \cdot A}{\left(\frac{K \cdot L}{r}\right)^2} \qquad Pcr = \frac{\pi^2 \cdot E \cdot I}{\left(K \cdot L\right)^2} \qquad \sigma cr = \frac{Pcr}{A} \qquad r = \sqrt{\frac{I}{A}}$$

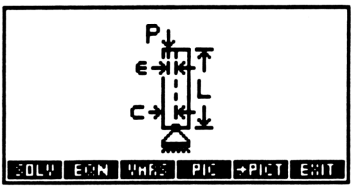
Exemple :

Soit : L=7.3152_m, r=4.1148_cm, E=199947961.502_kPa, A=53.0967_cm^2, K=0.7, I=8990598.7930_mm^4.

Solution : Pcr=676.6019_kN, σcr=127428.2444_kPa.

Colonnes excentriques (Eccentric Columns: 1, 2)

Voir "Déformation élastique".



Equations :

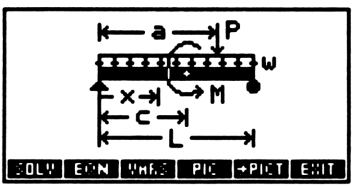
$$\sigma\max = \frac{P}{A} \cdot \left(1 + \frac{\epsilon \cdot c}{r^2} \cdot \left(\frac{1}{\cos\left(\frac{K \cdot L}{2 \cdot r} \cdot \sqrt{\frac{P}{E \cdot A}} \right)} \right) \right) \qquad r = \sqrt{\frac{I}{A}}$$

Exemple :

Soit : L=6.6542_m, A=187.9351_cm^2, r=8.4836_cm, E=206842718.795_kPa, I=135259652.16_mm^4, K=1, P=1908.2571_kN, c=15.24_cm, ϵ=1.1806_cm.

Solution : σmax=140853.0970_kPa.

Flexion simple (Simple Deflection: 1, 3)



Equation :

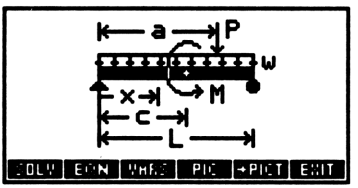
$$y = \frac{P \cdot \left(L - a\right) \cdot x}{6 \cdot L \cdot E \cdot I} \cdot \left(x^2 + \left(L - a\right)^2 - L^2\right) - \frac{M \cdot x}{E \cdot I} \cdot \left(c - \frac{x^2}{6 \cdot L} - \frac{L}{3} - \frac{c^2}{2 \cdot L}\right) - \frac{w \cdot x}{24 \cdot E \cdot I} \cdot \left(L^3 + x^2 \cdot \left(x - 2 \cdot L\right)\right)$$

Exemple :

Soit : L=20_ft, E=29000000_psi, I=40_in^4, a=10_ft, P=674.427_lbf, c=17_ft, M=3687.81_ft*lbf, w=102.783_lbf/ft, x=9_ft.

Solution : y=−.6005_in.

Pente simple (Simple Slope: 1, 4)



Equation :

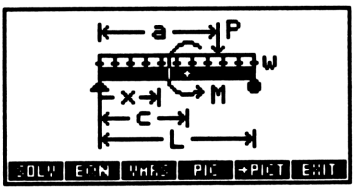
$$\Theta = \frac{P \cdot \left(L - a\right)}{6 \cdot L \cdot E \cdot I} \cdot \left(3 \cdot x^2 + \left(L - a\right)^2 - L^2\right) - \frac{M}{E \cdot I} \cdot \left(c - \frac{x^2}{2 \cdot L} - \frac{L}{3} - \frac{c^2}{2 \cdot L}\right) - \frac{w}{24 \cdot E \cdot I} \cdot \left(L^3 + x^2 \cdot \left(4 \cdot x - 6 \cdot L\right)\right)$$

Exemple :

Soit : L=20_ft, E=29000000_psi, I=40_in^4, a=10_ft, P=674.427_lbf, c=17_ft, M=3687.81_ft*lbf, w=102.783_lbf/ft, x=9_ft.

Solution : θ=−.0876_°.

Moment simple (Simple Moment: 1, 5)



Equation :

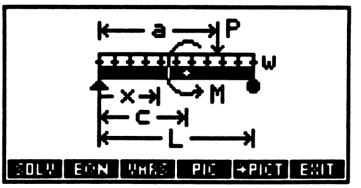
$$Mx = \frac{P \cdot \left(L - a\right) \cdot x}{L} + \frac{M \cdot x}{L} + \frac{w \cdot x}{2} \cdot \left(L - x\right)$$

Exemple :

Soit : L=20_ft, a=10_ft, P=674.427_lbf, c=17_ft, M=3687.81_ft*lbf, w=102.783_lbf/ft, x=9_ft.

Solution : Mx=9782.1945_ft*lbf.

Cisaillement simple (Simple Shear: 1, 6)



Equation :

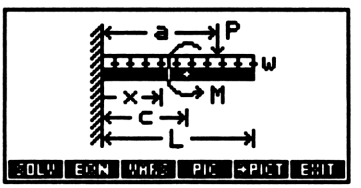
$$V = \frac{P \cdot \left(L - a\right)}{L} + \frac{M}{L} + \frac{w}{2} \cdot \left(L - 2 \cdot x\right)$$

Exemple :

Soit : L=20_ft, a=10_ft, P=674.427_lbf, M=3687.81_ft*lbf, w=102.783_lbf/ft, x=9_ft.

Solution : V=624.387_lbf.

Flexion d'un porte-à-faux (Cantilever Deflection: 1, 7)



Equation :

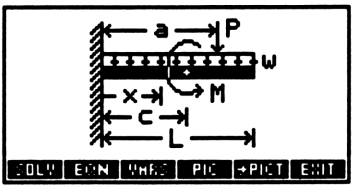
$$y = \frac{P \cdot x^2}{6 \cdot E \cdot I} \cdot \left(x - 3 \cdot a\right) + \frac{M \cdot x^2}{2 \cdot E \cdot I} - \frac{w \cdot x^2}{24 \cdot E \cdot I} \cdot \left(6 \cdot L^2 - 4 \cdot L \cdot x + x^2\right)$$

Exemple :

Soit : L=10_ft, E=29000000_psi, I=15_in^4, P=500_lbf, M=800_ft*lbf, a=3_ft, c=6_ft, w=100_lbf/ft, x=8_ft.

Solution : y=−.3316_in.

Pente d'un porte-à-faux (Cantilever Slope: 1, 8)



Equation :

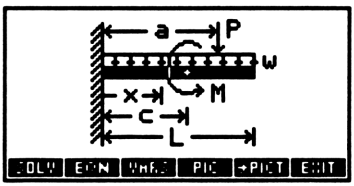
$$\Theta = \frac{P \cdot x}{2 \cdot E \cdot I} \cdot \left(x - 2 \cdot a\right) + \frac{M \cdot x}{E \cdot I} - \frac{w \cdot x}{6 \cdot E \cdot I} \cdot \left(3 \cdot L^2 - 3 \cdot L \cdot x + x^2\right)$$

Exemple :

Soit : L=10_ft, E=29000000_psi, I=15_in^4, P=500_lbf, M=800_ft*lbf, a=3_ft, c=6_ft, w=100_lbf/ft, x=8_ft.

Solution : Θ=−.2652_°.

Moment d'un porte-à-faux (Cantilever Moment: 1, 9)



Equation :

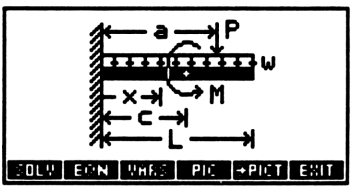
$$Mx = P \cdot \left(x - a\right) + M - \frac{W}{2} \cdot \left(L^2 - 2 \cdot L \cdot x + x^2\right)$$

Exemple :

Soit : L=10_ft, P=500_lbf, M=800_ft*lbf, a=3_ft, c=6_ft, w=100_lbf/ft, x=8_ft.

Solution : Mx=−200_ft*lbf.

Cisaillement d'un porte-à-faux (Cantilever Shear: 1, 10)



Equation :

$$V = P + w \cdot \left(L - x\right)$$

Exemple :

Soit : L=10_ft, P=500_lbf, a=3_ft, x=8_ft, w=100_lbf/ft.

Solution : V=200_lbf.

Electricité (Electricity: 2)

Noms de variable et description

ϵr	Permitivité relative
μr	Perméabilité relative
ω	Pulsation
$\omega 0$	Pulsation à la résonance
ϕ	Phase
$\phi p, \phi s$	Phases parallèle et série
ρ	Résistivité
ΔI	Variation du courant
Δt	Variation du temps
ΔV	Variation de tension
A	Section transversale d'un fil (Résistance d'un fil) ou D'un solénoïde (Inductance d'un solénoïde), ou Surface des armatures (Condensateur plan)
$C, C1, C2$	Capacité
Cp, Cs	Capacités parallèle et série
d	Espace entre les armatures
E	Energie
F	Force s'exerçant entre les charges
f	Fréquence
$f0$	Fréquence de résonance
I	Courant ou Courant total (Diviseur de courant)
$I1$	Courant dans R1
$Imax$	Courant maximal
L	Inductance, ou Longueur (Résistance d'un fil, Condensateur cylindrique)
$L1, L2$	Inductance
Lp, Ls	Inductances parallèle et série

Noms de variable et description (suite)

N	Nombre de spires
n	Nombre de spires par unité de longueur
P	Puissance
q	Charge
q1,*q2*	Charge ponctuelle
Qp,*Qs*	Facteurs de qualité parallèle et série
r	Distance entre charges
R,*R1*,*R2*	Résistance
ri,*ro*	Rayons intérieur et extérieur
Rp,*Rs*	Résistances parallèle et série
t	Temps
ti,*tf*	Instants initial et final
V	Tension, ou Tension totale (Diviseur de tension)
V1	Tension aux bornes de R1
Vi,*Vf*	Tensions initiale et finale
Vmax	Tension maximale
XC	Réactance d'un condensateur
XL	Réactance d'une bobine

Référence : 3.

Loi de Coulomb (Coulomb's Law: 2, 1)

Cette équation décrit la force électrostatique s'appliquant entre deux particules électriquement chargées.

Equation :

$$F = \frac{1}{4 \cdot \pi \cdot \epsilon 0 \cdot \epsilon r} \cdot \left(\frac{q1 \cdot q2}{r^2} \right)$$

Exemple :

Soit : *q1*=1.6E−19_C, *q2*=1.6E−19_C, *r*=4.00E−13_cm, ϵr=1.00.

Solution : *F*=14.3801_N.

Loi d'Ohm et puissance (Ohm's Law and Power: 2, 2)

Equations :

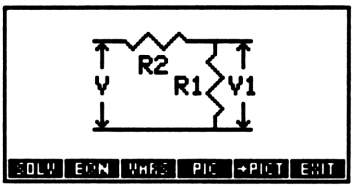
$$V = I \cdot R \qquad P = V \cdot I \qquad P = I^2 \cdot R \qquad P = \frac{V^2}{R}$$

Exemple :

Soit : V=24_V, I=16_A.

Solution : R=1.5_Ω, P=384_W.

Diviseur de tension (Voltage Divider: 2, 3)



Equation :

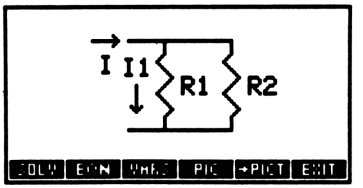
$$V1 = V \cdot \left(\frac{R1}{R1 + R2} \right)$$

Exemple :

Soit : $R1$=40_Ω, $R2$=10_Ω, V=100_V.

Solution : $V1$=80_V.

Diviseur de courant (Current Divider: 2, 4)



Equation :

$$I1 = I \cdot \left(\frac{R2}{R1 + R2} \right)$$

Exemple :

Soit : *R1*=10_Ω, *R2*=6_Ω, *I*=15_A.

Solution : *I1*=5.6250_A.

Résistance d'un fil (Wire Resistance: 2, 5)

Equation :

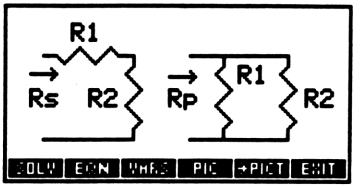
$$R = \frac{\rho \cdot L}{A}$$

Exemple :

Soit : ρ=.0035_Ω*cm, *L*=50_cm, *A*=1_cm^2.

Solution : *R*=0.175_Ω.

R série et parallèle (Series and Parallel R: 2, 6)



Equations :

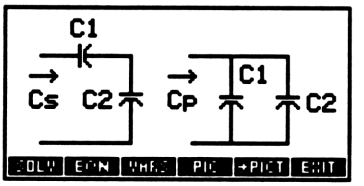
$$Rs = R1 + R2 \qquad \frac{1}{Rp} = \frac{1}{R1} + \frac{1}{R2}$$

Exemple :

Soit : $R1$=2_Ω, $R2$=3_Ω.

Solution : Rs=5_Ω, Rp=1.2000_Ω.

C série et parallèle (Series and Parallel C: 2, 7)



Equations :

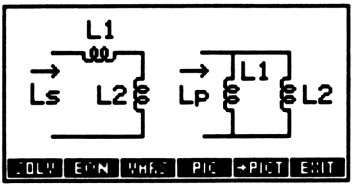
$$\frac{1}{Cs} = \frac{1}{C1} + \frac{1}{C2} \qquad Cp = C1 + C2$$

Exemple :

Soit : $C1$=2_μF, $C2$=3_μF.

Solution : Cs=1.2000_μF, Cp=5_μF.

L série et parallèle (Series and Parallel L: 2, 8)



Equations :

$$Ls = L1 + L2 \qquad \frac{1}{Lp} = \frac{1}{L1} + \frac{1}{L2}$$

Exemple :

Soit : $L1$=17_mH, $L2$=16.5_mH.

Solution : Ls=33.5000_mH, Lp=8.3731_mH.

Energie dans une capacité (Capacitive Energy: 2, 9)

Equation :

$$E = \frac{C \cdot V^2}{2}$$

Exemple :

Soit : E=.025_J, C=20_μF.

Solution : V=50_V.

Energie dans une inductance (Inductive Energy: 2, 10)

Equation :

$$E = \frac{L \cdot I^2}{2}$$

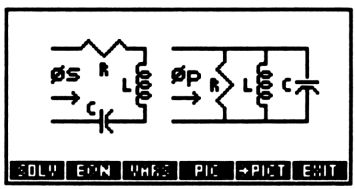
Exemple :

Soit : E=4_J, L=15_mH.

Solution : I=23.0940_A.

Retard du courant dans un circuit RLC (RLC Current Delay: 2, 11)

La phase (angle) est positive pour le courant en retard sur la tension.



Equations :

$$\text{TAN}\left(\phi s\right) = \frac{XL - XC}{R} \qquad \text{TAN}\left(\phi p\right) = \frac{\frac{1}{XC} - \frac{1}{XL}}{\frac{1}{R}}$$

$$XC = \frac{1}{\omega \cdot C} \qquad XL = \omega \cdot L \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : f=107_Hz, C=80_μF, L=20_mH, R=5_Ω.

Solution : ω=672.3008_r/s, ϕs=−45.8292_°, ϕp=−5.8772 _°, XC=18.5929_Ω, XL=13.4460_Ω.

Courant continu dans un condensateur (DC Capacitor Current: 2, 12)

Ces équations déterminent le courant continu nécessaire pour modifier la tension aux bornes d'un condensateur dans un certain laps de temps.

Equations :

$$I = C \cdot \left(\frac{\Delta V}{\Delta t} \right) \qquad \Delta V = Vf - Vi \qquad \Delta t = tf \cdot ti$$

Exemple :

Soit : C=15_μF, Vi=2.3_V, Vf=3.2_V, I=10_A, ti=0_s.

Solution : ΔV=.9000_V, Δt=1.3500_μs, tf=1.3500_μs.

Charge d'un condensateur (Capacitor Charge: 2, 13)

Equation :

$$q = C \cdot V$$

Exemple :

Soit : C=20_μF, V=100_V.

Solution : q=0.0020_C.

Tension continue dans une inductance (DC Inductor Voltage: 2, 14)

Ces équations déterminent la tension continue induite aux bornes d'une inductance par une variation de courant dans un certain laps de temps.

Equations :

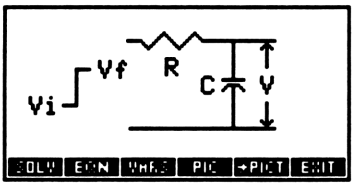
$$V = -L \cdot \left(\frac{\Delta I}{\Delta t}\right) \qquad \Delta I = If - Ii \qquad \Delta t = tf - ti$$

Exemple :

Soit : L=100_mH, V=52_V, Δt=32_μs, Ii=23_A, ti=0_s.

Solution : ΔI=−0.0166_A, If=22.9834_A, tf=32_μs.

Réponse à un transitoire d'un circuit RC (RC Transient: 2, 15)



Equation :

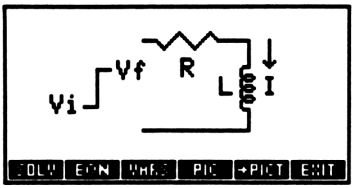
$$V = Vf - \left(Vf - Vi\right) \cdot e^{\frac{-t}{R \cdot C}}$$

Exemple :

Soit : Vi=0_V, C=50_μF, Vf=10_V, R=100_ω, t=2_ms.

Solution : V=3.2968_V.

Réponse à un transitoire RL (RL Transient: 2, 16)



Equation :

$$I = \frac{1}{R} \cdot \left(Vf - \left(Vf - Vi \right) \cdot e^{\frac{-t \cdot R}{L}} \right)$$

Exemple :

Soit : Vi=0_V, Vf=5_V, R=50_ω, L=50_mH, t=75_μs.

Solution : I=0.0072_A.

Fréquence de résonance (Resonant Frequency: 2, 17)

Equations :

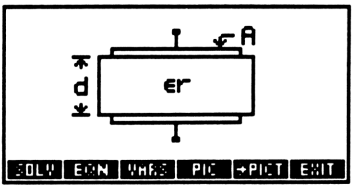
$$\omega 0 = \frac{1}{\sqrt{L \cdot C}} \qquad Qs = \frac{1}{R} \cdot \sqrt{\frac{L}{C}} \qquad Qp = R \cdot \sqrt{\frac{C}{L}} \qquad \omega 0 = 2 \cdot \pi \cdot f0$$

Exemple :

Soit : L=500_mH, C=8_μF, R=10_ω.

Solution : $\omega 0$=500_r/s, Qs=25.0000, Qp=0.0400, $f0$=79.5775_Hz.

Condensateur plan (Plate Capacitor: 2, 18)



Equation :

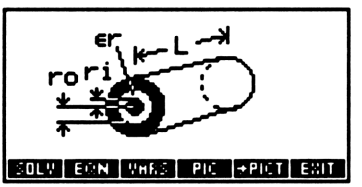
$$C = \frac{\epsilon 0 \cdot \epsilon r \cdot A}{d}$$

Exemple :

Soit : C=25_μF, ϵr=2.26, A=1_cm^2.

Solution : d=8.0042E−9_cm.

Condensateur cylindrique (Cylindrical Capacitor: 2, 19)



Equation :

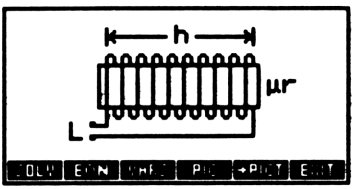
$$C = \frac{2 \cdot \pi \cdot \epsilon 0 \cdot \epsilon r \cdot L}{LN\left(\frac{ro}{ri}\right)}$$

Exemple :

Soit : ϵr=1, ro=1_cm, ri=.999_cm, L=10_cm.

Solution : C=0.0056_μF.

Inductance d'un solénoïde (Solenoid Inductance: 2, 20)



Equation :

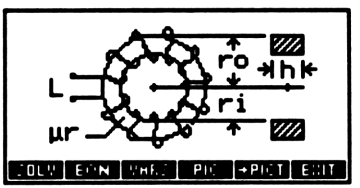
$$L = \mu 0 \cdot \mu r \cdot n^2 \cdot A \cdot h$$

Exemple :

Soit : μr=2.5, n=40_1/cm, A=.2_cm^2, h=3_cm.

Solution : L=0.0302_mH.

Inductance d'un tore (Toroid Inductance: 2, 21)



Equation :

$$L = \frac{\mu 0 \cdot \mu r \cdot N^2 \cdot h}{2 \cdot \pi} \cdot LN\left(\frac{ro}{ri}\right)$$

Exemple :

Soit : μr=1, N=5000, h=2_cm, ri=2_cm, ro=4_cm.

Solution : L=69.3147_mH.

Tension sinusoïdale (Sinusoidal Voltage: 2, 22)

Equations :

$$V = Vmax \cdot SIN\left(\omega \cdot t + \phi\right) \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : $Vmax$=110_V, t=30_μs, f=60_Hz, ϕ=15_°.

Solution : ω=376.9911_r/s, V=29.6699_V.

Courant sinusoïdal (Sinusoidal Current: 2, 23)

Equations :

$$I = Imax \cdot SIN\left(\omega \cdot t + \phi\right) \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : t=32_s, $Imax$=10_A, ω=636_r/s, ϕ=30_°.

Solution : I=9.5983_A, f=101.2225_Hz.

Mécanique des fluides (Fluids: 3)

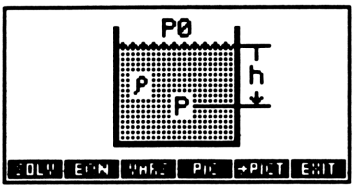
Noms de variable et description

ϵ	Rugosité
μ	Viscosité dynamique
ρ	Densité
ΔP	Variation de pression
Δy	Variation de hauteur
ΣK	Coefficients d'ajustement totaux
A	Section transversale
$A1, A2$	Sections transversales initiale et finale
D	Diamètre
$D1, D2$	Diamètres initial et final
h	Profondeur par rapport à la profondeur de référence $P0$
hL	Pertes de charge
L	Longueur
M	Débit massique
n	Viscosité cinématique
P	Pression à la profondeur h
$P0$	Pression de référence
$P1, P2$	Pressions initiale et finale
Q	Débit volumétrique
Re	Nombre de Reynolds
$v1, v2$	Vitesses initiale et finale
$vavg$	Vitesse moyenne
W	Puissance à l'entrée
$y1, y2$	Hauteurs initiale et finale

Références : 3, 6, 9.

Pression à une certaine profondeur (Pressure at Depth: 3, 1)

Cette équation exprime la pression hydrostatique d'un fluide incompressible. La profondeur h est positive vers le bas à partir de la référence.



Equation :

$$P = P0 + \rho \cdot g \cdot h$$

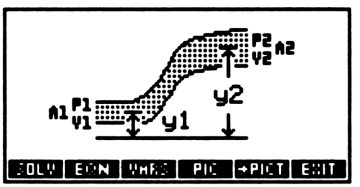
Exemple :

Soit : h=100_m, ρ=1025.1817_kg/m^3, $P0$=1_atm.

Solution : P=1106.6848_kPa.

Equations de Bernoulli (Bernoulli Equation: 3, 2)

Ces équations expriment l'écoulement continu d'un fluide incompressible.



Equations :

$$\frac{\Delta P}{\rho} + \frac{v2^2 - v1^2}{2} + g \cdot \Delta y = 0$$

$$\frac{\Delta P}{\rho} + \frac{v2^2 \cdot \left(1 - \left(\frac{A2}{A1}\right)^2\right)}{2} + g \cdot \Delta y = 0$$

$$\frac{\Delta P}{\rho} + \frac{v1^2 \cdot \left(\left(\frac{A1}{A2}\right)^2 - 1\right)}{2} + g \cdot \Delta y = 0$$

$$\Delta P = P2 - P1 \qquad \Delta y = y2 - y1 \qquad M = \rho \cdot Q$$

$$Q = A2 \cdot v2 \qquad Q = A1 \cdot v1$$

$$A1 = \frac{\pi \cdot D1^2}{4} \qquad A2 = \frac{\pi \cdot D2^2}{4}$$

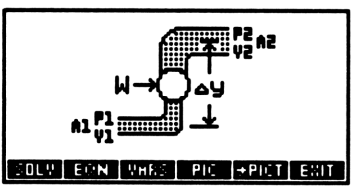
Exemple :

Soit : *P2*=25_psi, *P1*=75_psi, *y2*=35_ft, *y1*=0_ft, *D1*=18_in, *ρ*=64_lb/ft^3, *v1*=100_ft/s.

Solution : *Q*=10602.8752_ft^3/min, *M*=678584.0132_lb/min, *v2*=122.4213_ft/s, *A2*=207.8633_in^2, *D2*=16.2684_in, *A1*=254.4690_in^2, *ΔP*=−50_psi, *Δy*=35_ft.

Ecoulement avec pertes (Flow with Losses: 3, 3)

Ces équations étendent celles de Bernoulli pour tenir compte de la puissance d'entrée (ou de sortie) et de la perte de charge.



Equations :

$$M \cdot \left(\frac{\Delta P}{\rho} + \frac{v2^2 - v1^2}{2} + g \cdot \Delta y + hL \right) = W$$

$$M \cdot \left(\frac{\Delta P}{\rho} + \frac{v2^2 \cdot \left(1 - \left(\frac{A2}{A1} \right)^2 \right)}{2} + g \cdot \Delta y + hL \right) = W$$

$$M \cdot \left(\frac{\Delta P}{\rho} + \frac{v1^2 \cdot \left(\left(\frac{A1}{A2} \right)^2 - 1 \right)}{2} + g \cdot \Delta y + hL \right) = W$$

$$\Delta P = P2 - P1 \qquad \Delta y = y2 - y1 \qquad M = \rho \cdot Q$$

$$Q = A2 \cdot v2 \qquad Q = A1 \cdot v1$$

$$A1 = \frac{\pi \cdot D1^2}{4} \qquad A2 = \frac{\pi \cdot D2^2}{4}$$

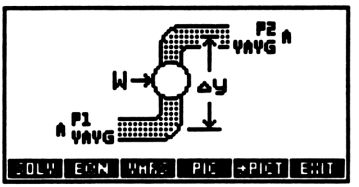
Exemple :

Soit : $P2$=30_psi, $P1$=65_psi, $y2$=100_ft, $y1$=0_ft, ρ=64_lb/ft^3, $D1$=24_in, hL=2.0_ft^2/s^2, W=25_hp, $v1$=100_ft/s.

Solution : Q=18849.5559_ft^3/min, M=1206371.5790_lb/min, ΔP=−35_psi, Δy=100_ft, $v2$=93.1269_ft/s, $A1$=452.3893_in^2, $A2$=485.7773_in^2, $D2$=24.8699_in.

Débit dans un tuyau plein (Flow in Full Pipes: 3, 4)

Ces équations adaptent celles de Bernoulli pour un tuyau plein de section ronde, et pour tenir compte de la puissance d'entrée (ou de sortie) et des pertes par friction. (Voir FANNING au chapitre 3).



Equations :

$$\rho \cdot \left(\frac{\pi \cdot D^2}{4} \right) \cdot vavg \cdot \left(\frac{\Delta P}{\rho} + g \cdot \Delta y + vavg^2 \cdot \left(2 \cdot f \cdot \left(\frac{L}{D} \right) + \frac{\Sigma K}{2} \right) \right) = W$$

$$\Delta P = P2 - P1 \qquad \Delta y = y2 - y1 \qquad M = \rho \cdot Q$$

$$Q = A \cdot vavg \qquad A = \frac{\pi \cdot D^2}{4} \qquad Re = \frac{D \cdot vavg \cdot \rho}{\mu} \qquad n = \frac{\mu}{\rho}$$

Exemple :

Soit : ρ=62.4_lb/ft^3, D=12_in, $vavg$=8_ft/s, $P2$=15_psi, $P1$=20_psi, $y2$=40_ft, $y1$=0_ft, μ=.00002_lbf*s/ft^2, ΣK=2.25, ϵ=.02_in, L=250_ft.

Solution : ΔP=−5_psi, Δy=40_ft, A=113.0973_in^2, n=1.0312_ft^2/s, Q=376.9911_ft^3/min, M=23524.2458_lb/min, W=25.8897_hp, Re=775780.5.

Forces et énergie (Forces and Energy: 4)

Noms de variable et description

α	Accélération angulaire
ω	Vitesse angulaire
ωi,ωf	Vitesses angulaires initiale et finale
ρ	Densité du fluide
τ	Couple
Θ	Déplacement angulaire
a	Accélération
A	Surface projetée par rapport au flux
ar	Accélération centripète à r
at	Accélération tangentielle à r
Cd	Coefficient de résistance
E	Energie
F	Force à r ou au point x, ou Force d'attraction (Loi de Hooke), ou Force d'attraction (Loi de la gravitation), ou Effort résistant (Effort résistant)
I	Moment d'inertie
k	Constante du ressort
Ki,Kf	Energies cinétiques initiale et finale
m,$m1$,$m2$	Mass
N	Vitesse de rotation
Ni,Nf	Vitesses de rotation initiale et finale
P	Puissance instantanée
$Pavg$	Puissance moyenne

Noms de variable et description (suite)

r	Rayon par rapport à l'axe de rotation, ou Distance entre les deux masses (Loi de la gravitation)
t	Temps
v	Vitesse
vf,*v1f*,*v2f*	Vitesse finale
vi,*v1i*	Vitesse initiale
W	Travail
x	Déplacement

Référence : 3.

Mécanique linéaire (Linear Mechanics: 4, 1)

Equations :

$$F = m \cdot a \qquad Ki = \frac{1}{2} \cdot m \cdot vi^2 \qquad Kf = \frac{1}{2} \cdot m \cdot vf^2 \qquad W = F \cdot x$$

$$W = Kf - Ki \qquad P = F \cdot v \qquad Pavg = \frac{W}{t} \qquad vf = vi + a \cdot t$$

Exemple :

Soit : *t*=10_s, *m*=50_lb, *a*=12.5_ft/s^2, *vi*=0_ft/s.

Solution : *vf*=125_ft/s, *x*=625_ft, *F*=19.4256_lbf, *Ki*=0_ft*lbf, *Kf*=12140.9961_ft*lbf, *W*=12140.9961_ft*lbf, *Pavg*=2.2075_hp.

Mécanique angulaire (Angular Mechanics: 4, 2)

Equations :

$$\tau = I \cdot \alpha \qquad Ki = \frac{1}{2} \cdot I \cdot \omega i^2 \qquad Kf = \frac{1}{2} \cdot I \cdot \omega f^2 \qquad W = \tau \cdot \Theta$$

$$W = Kf - Ki \qquad P = \tau \cdot \omega \qquad Pavg = \frac{W}{t} \qquad \omega f = \omega i + \alpha \cdot t$$

$$at = \alpha \cdot r \qquad \omega = 2 \cdot \pi \cdot N \qquad \omega i = 2 \cdot \pi \cdot Ni \qquad \omega f = 2 \cdot \pi \cdot Nf$$

Exemple :

Soit : I=1750_lb*in^2, Θ=360_°, r=3.5_in, α=10.5_r/min^2, ωi=0_r/s.

Solution : τ=1.1017E−3_ft*lbf, Ki=0_ft*lbf, W=6.9221E−3_ft*lbf, Kf=6.9221E−3_ft*lbf, at=8.5069E−4_ft/s^2, Ni=0_rpm, ωf=11.4868_r/min, t=1.0940_min, Nf=1.8282_rpm, $Pavg$=1.9174E−7_hp.

Force centripète (Centripetal Force: 4, 3)

Equations :

$$F = m \cdot \omega^2 \cdot r \qquad \omega = \frac{v}{r} \qquad ar = \frac{v^2}{r} \qquad \omega = 2 \cdot \pi \cdot N$$

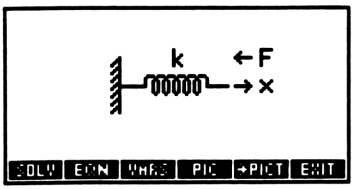
Exemple :

Soit : m=1_kg, r=5_cm, N=2000_Hz.

Solution : ω=12566.3706_r/s, ar=7895683.5209_m/s, F=7895683.5209_N, v=628.3185_m/s.

Loi de Hooke (Hooke's Law: 4, 4)

La force est celle exercée par le ressort.



Equations :

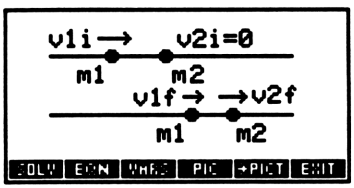
$$F = -k \cdot x \qquad W = \frac{-1}{2} \cdot k \cdot x^2$$

Exemple :

Soit : k=1725_lbf/in, x=1.25_in.

Solution : F=−2156.25_lbf, W=−112.3047_ft*lbf.

Collisions élastiques selon une dimension (1D Elastic Collisions: 4, 5)



Equations :

$$v1f = \frac{m1 - m2}{m1 + m2} \cdot v1i \qquad v2f = \frac{2 \cdot m1}{m1 + m2} \cdot v1i$$

Exemple :

Soit : $m1$=10_kg, $m2$=25_kg, $v1i$=100_m/s.

Solution : $v1f$=−42.8571_m/s, $v2f$=57.1429_m/s.

Effort résistant (Drag Force: 4, 6)

Equation :

$$F = Cd \cdot \left(\frac{\rho \cdot v^2}{2} \right) \cdot A$$

Exemple :

Soit : Cd=.05, ρ=1000_kg/m^3, A=7.5E6_cm^2, v=35_m/s.

Solution : F=22968750_N.

Loi de la gravitation (Law of Gravitation: 4, 7)

Equation :

$$F = G \cdot \left(\frac{m1 \cdot m2}{r^2} \right)$$

Exemple :

Soit : $m1$=2E15_kg, $m2$=2E18_kg, r=1000000_km.

Solution : F=266903.6_N.

Relation masse-énergie (Mass-Energy Relation: 4, 8)

Equation :

$$E = m \cdot c^2$$

Exemple :

Soit : m=9.1E−31_kg.

Solution : E=8.1787E−14_J.

Gaz (Gases: 5)

Noms de variable et description

λ	Trajectoire libre moyenne
ρ	Densité du flux
$\rho 0$	Densité de stagnation
A	Section du flux
At	Surface de l'embouchure
d	Diamètre moléculaire
k	Rapport de chaleur spécifique
M	Vitesse en Mach
m	Masse
MW	Poids moléculaire
n	Nombre de môles, ou Constante polytropique (Processus polytropiques)
P	Pression, ou Pression du flux (Flux isentropique)
$P0$	Pression de stagnation
Pc	Pression pseudocritique
Pi,Pf	Pressions initiale et finale
T	Température, ou Température du flux (Flux isentropique)

Noms de variable et description (suite)

T0	Température de stagnation
Tc	Température pseudocritique
Ti, *Tf*	Températures initiale et finale
V	Volume
Vi, *Vf*	Volumes initial et final
vrms	Vitesse efficace
W	Travail

Références : 1, 3.

Loi des gaz parfaits (Ideal Gas Law: 5, 1)

Equations :

$$P \cdot V = n \cdot R \cdot T \qquad m = n \cdot MW$$

Exemple :

Soit : T=16.85_°C, P=1_atm, V=25_l, MW=36_g/gmol.

Solution : n=1.0506_gmol, m=3.7820E−2_kg.

Changement d'état des gaz parfaits (Ideal Gas State Chg: 5, 2)

Equation :

$$\frac{Pf \cdot Vf}{Tf} = \frac{Pi \cdot Vi}{Ti}$$

Exemple :

Soit : Pi=1.5_kPa, Pf=1.5_kPa, Vi=2_l, Ti=100_°C, Tf=373.15_K.

Solution : Vf=2_l.

Dilatation isotherme (Isothermal Expansion: 5, 3)

Ces équations s'appliquent à un gaz parfait.

Equations :

$$W = n \cdot R \cdot T \cdot LN\left(\frac{Vf}{Vi}\right) \qquad m = n \cdot MW$$

Exemple :

Soit : Vi=2_l, Vf=125_l, T=300_°C, n=0.25_gmol, MW=64_g/gmol.

Solution : W=4926.4942_J, m=.016_kg.

Processus polytropiques (Polytropic Processes: 5, 4)

Ces équations expriment une variation réversible de la pression ou du volume d'un gaz parfait de telle sorte que $P * V^n$ soit constant. Elles tiennent compte des cas spéciaux des processus istothermes (n=1), des processus isentropiques (n=k, le rapport de chaleur spécifique), et des processus à pression constante (n=0).

Equations :

$$\frac{Pf}{Pi} = \left(\frac{Vf}{Vi}\right)^{-n} \qquad \frac{Tf}{Ti} = \left(\frac{Pf}{Pi}\right)^{\frac{n-1}{n}}$$

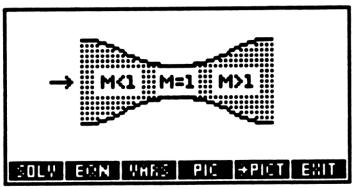
Exemple :

Soit : Pi=15_psi, Pf=35_psi, Vi=1_ft^3, Vf=0.50_ft^3, Ti=75_°F.

Solution : n=1.2224, Tf=164.1117_°F.

Flux isentropique (Isentropic Flow: 5, 5)

Le calcul est différent selon que la vitesse est inférieure ou supérieure à Mach 1. Cette vitesse correspond à celle du son dans le fluide compressible.



Equations :

$$\frac{T}{T0} = \frac{2}{2 + (k-1)\cdot M^2} \qquad \frac{P}{P0} = \left(\frac{T}{T0}\right)^{\frac{k}{k-1}}$$

$$\frac{\rho}{\rho 0} = \left(\frac{T}{T0}\right)^{\frac{1}{k-1}}$$

$$\frac{A}{At} = \frac{1}{M}\cdot\left(\frac{2}{k+1}\cdot\left(1 + \frac{k-1}{2}\cdot M^2\right)\right)^{\frac{k+1}{2\cdot(k-1)}}$$

Exemple :

Soit : k=2, M=.9, $T0$=26.85_°C, T=373.15_K, $\rho 0$=100_kg/m^3, $P0$=100_kPa, A=1_cm^2.

Solution : P=464.1152_kPa, At=0.9928_cm^2, ρ=215.4333_kg/m^3.

Loi des gaz réels (Real Gas Law: 5, 6)

Ces équations adaptent la loi des gaz parfaits afin de simuler le comportement des gaz réels (Voir ZFACTOR au chapitre 3).

Equations :

$$P \cdot V = n \cdot Z \cdot R \cdot T \qquad m = n \cdot MW$$

Exemple :

Soit : Pc=48_atm, Tc=298_K, P=5_kPa, V=10_l, MW=64_g/gmol, T=75_°C.

Solution : n=0.0173_gmol, m=1.1057E−3_kg.

Changement d'état des gaz réels (Real Gas State Change: 5, 7)

Cette équation adapte l'équation de changement d'état des gaz parfaits pour simuler le comportement des gaz réels (voir ZFACTOR au chapitre 3).

Equation :

$$\frac{Pf \cdot Vf}{Zf \cdot Tf} = \frac{Pi \cdot Vi}{Zi \cdot Ti}$$

Exemple :

Soit : Pc=48_atm, Pi=100_kPa, Pf=50_kPa, Ti=75_°C, Tc=298_K, Vi=10_l, Tf=250_°C.

(Pour mémoire : Zf et Zi sont calculés automatiquement par le HP 48 en utilisant ces variables.)

Solution : Vf=30.1703_l.

Théorie cinématique (Kinetic Theory: 5, 8)

Ces équations expriment les propriétés d'un gaz parfait.

Equations :

$$P = \frac{n \cdot MW \cdot vrms^2}{3 \cdot V} \qquad vrms = \sqrt{\frac{3 \cdot R \cdot T}{MW}}$$

$$\lambda = \frac{1}{\sqrt{2} \cdot \pi \cdot \left(\frac{n \cdot NA}{V}\right) \cdot d^2} \qquad m = n \cdot MW$$

Exemple :

Soit : P=100_kPa, V=2_l, T=26.85_°C, MW=18_g/gmol, d=2.5_nm.

Solution : $vrms$=644.7678_m/s, m=1.4433E−3_kg, n=.0802_gmol, λ=1.4916_nm.

Thermodynamique (Heat Transfer: 6)

Noms de variable et description

α	Coefficient de dilatation
δ	Allongement
$\lambda 1, \lambda 2$	Limites inférieure et supérieure de longueur d'onde
λmax	Longueur d'onde de puissance maximale d'émission
ΔT	Ecart de température
A	Surface
c	Chaleur spécifique

Noms de variable et description (suite)

eb12	Puissance d'émission sur la plage $\lambda 1$ à $\lambda 2$
eb	Puissance totale d'émission
f	Fraction de la puissance d'émission sur la plage $\lambda 1$ à $\lambda 2$
h,h1,h3	Coefficient de transfert thermique par convection
k,k1,k2,k3	Conductivité thermique
L,L1,L2,L3	Longueur
m	Masse
Q	Capacité thermique
q	Vitesse de transfert thermique
T	Température
Tc	Température de surface froide (Conduction), ou Température du fluide froid
Th	Température de surface chaude, ou Température du fluide chaud (Conduction + Convection)
Ti,Tf	Températures initiale et finale
U	Coefficient de transfert thermique global

Références : 7, 9.

Capacité thermique (Heat Capacity: 6, 1)

Equations :

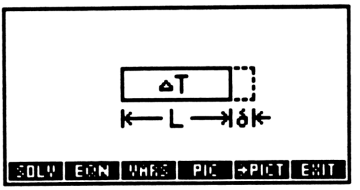
$$Q = m \cdot c \cdot \Delta T \qquad Q = m \cdot c \cdot \left(Tf - Ti \right)$$

Exemple :

Soit : ΔT=15_°C, Ti=0_°C, m=10_kg, Q=25_kJ.

Solution : Tf=15_°C, c=.1667_kJ/(kg*K).

Dilatation thermique (Thermal Expansion: 6, 2)



Equations :

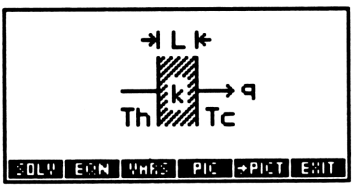
$$\delta = \alpha \cdot L \cdot \Delta T \qquad \delta = \alpha \cdot L \cdot \left(Tf - Ti \right)$$

Exemple :

Soit : ΔT=15_°C, L=10_m, Tf=25_°C, δ=1_cm.

Solution : Ti=10_°C, α=6.6667E−5_1/°C.

Conduction (Conduction: 6, 3)



Equations :

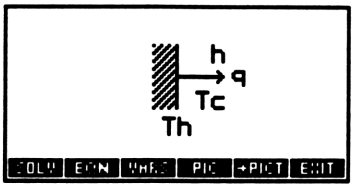
$$q = \frac{k \cdot A}{L} \cdot \Delta T \qquad q = \frac{k \cdot A}{L} \cdot \left(Th - Tc \right)$$

Exemple :

Soit : Tc=25_°C, Th=75_°C, A=12.5_m^2, L=1.5_cm, k=.12_W/(m*K).

Solution : q=5000_W, ΔT=50_°C.

Convection (Convection: 6, 4)



Equations :

$$q = h \cdot A \cdot \Delta T \qquad q = h \cdot A \cdot \left(Th - Tc \right)$$

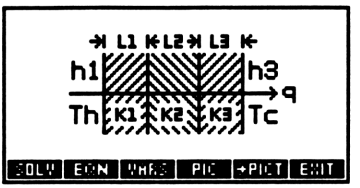
Exemple :

Soit : Tc=300_K, A=200_m^2, h=.005_W/(m^2*K), q=10_W.

Solution : ΔT=10_°C, Th=36.8500_°C.

Conduction + Convection (Conduction + Convection: 6, 5)

Si vous avez moins de trois milieux, donnez aux milieux excédentaires une épaisseur nulle et une conductivité quelconque différente de zéro. Les deux températures sont celles de fluides - si au lieu de cela vous avez à faire à une température de *surface*, donnez au coefficient de convection correspondant la valeur 10^{499}.



Equations :

$$q = \frac{A \cdot \Delta T}{\frac{1}{h1} + \frac{L1}{k1} + \frac{L2}{k2} + \frac{L3}{k3} + \frac{1}{h3}} \qquad q = \frac{A \cdot \left(Th - Tc\right)}{\frac{1}{h1} + \frac{L1}{k1} + \frac{L2}{k2} + \frac{L3}{k3} + \frac{1}{h3}}$$

$$U = \frac{q}{A \cdot \Delta T} \qquad U = \frac{q}{A \cdot \left(Th - Tc\right)}$$

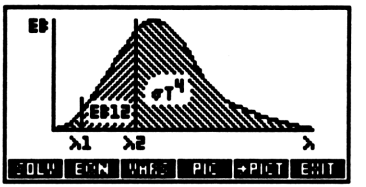
Exemple :

Soit : ΔT=35_°C, Th=55_°C, A=10_m^2, $h1$=.05_W/(m^2*K), $h3$=.05_W/(m^2*K), $L1$=3_cm, $L2$=5_cm, $L3$=3_cm, $k1$=.1_W/(m*K), $k2$=.5_W/(m*K), $k3$=.1_W/(m*K).

Solution : Tc=20_°C, U=0.0246_W/(m^2*K), q=8.5995_W.

Rayonnement du corps noir (Black Body Radiation: 6, 6)

Voir F0λ au chapitre 3.



Equations :

$$eb = \sigma \cdot T^4 \qquad f = F0\lambda\left(\lambda 2; T\right) - F0\lambda\left(\lambda 1; T\right)$$

$$eb12 = f \cdot eb \qquad \lambda max \cdot T = c3 \qquad q = eb \cdot A$$

Exemple :

Soit : T=1000_°C, $\lambda 1$=1000_nm, $\lambda 2$=600_nm, A=1_cm^2.

Solution : λmax=2276.0523_nm, eb=148984.2703_W/m^2, f=.0036, $eb12$=537.7264_W/m^2, q=14.8984_W.

Magnétisme (Magnetism: 7)

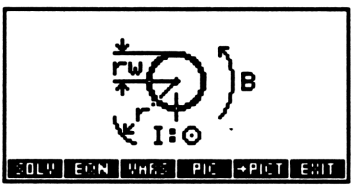
Noms de variable et description

μr	Perméabilité relative
B	Champ magnétique
d	Distance
Fba	Force
I,Ia,Ib	Courant
L	Longueur
N	Nombre total de spires
n	Nombre de spires par unité de longueur
r	Distance par rapport au centre du fil
ri,ro	Rayons intérieur et extérieur du tore
rw	Rayon du fil

Référence : 3.

Fil droit (Straight Wire: 7, 1)

Le calcul du champ magnétique est différent selon que le point où il s'applique est à l'intérieur ou à l'extérieur du fil.



Equation :

$$B = \frac{\mu 0 \cdot \mu r \cdot I}{2 \cdot \pi \cdot r}$$

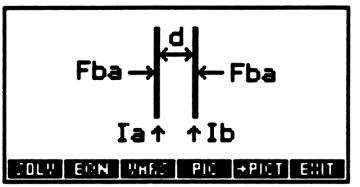
Exemple :

Soit : μr=1, rw=.25_cm, r=.2_cm, I=25_A.

Solution : B=.0016_T.

Force s'appliquant entre deux fils (Force between Wires: 7, 2)

La force s'appliquant entre les fils est positive pour une force d'attraction (pour des courants ayant le même signe).



Equation :

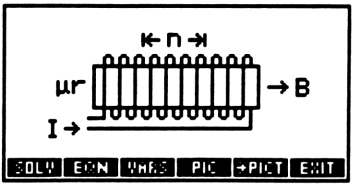
$$\mathrm{Fba} = \frac{\mu 0 \cdot \mu r \cdot \mathrm{L} \cdot \mathrm{Ib} \cdot \mathrm{Ia}}{2 \cdot \pi \cdot \mathrm{d}}$$

Exemple :

Soit : Ia=10_A, Ib=20_A, μr=1, L=50_cm, d=1_cm.

Solution : Fba=2.0000E−3_N.

Champ magnétique dans un solénoïde (B Field in Solenoid: 7, 3)



Equation :

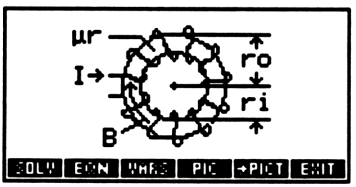
$$B = \mu 0 \cdot \mu r \cdot I \cdot n$$

Exemple :

Soit : μr=10, n=50, I=1.25_A.

Solution : B=0.0785_T.

Champ magnétique dans un tore (B Field in Toroid: 7, 4)



Equation :

$$B = \frac{\mu 0 \cdot \mu r \cdot I \cdot N}{2 \cdot \pi} \cdot \left(\frac{2}{ro + ri} \right)$$

Exemple :

Soit : μr=10, N=50, ri=5_cm, ro=7_cm, I=10_A.

Solution : B=1.6667E−2_T.

Mouvement (Motion: 8)

Noms de variable et description

α	Accélération angulaire
ω	Vitesse angulaire (Mouvement circulaire), ou Vitesse angulaire à l'instant t (Mouvement angulaire)
$\omega 0$	Vitesse angulaire initiale
ρ	Densité du fluide
θ	Position angulaire à l'instant t
$\theta 0$	Position angulaire initiale (Mouvement angulaire), ou Angle vertical initial (Jet de projectile)
a	Accéleration
A	Surface horizontale projetée
ar	Accélération centripète à r
Cd	Coefficient de résistance
m	Masse
M	Masse de la planète
N	Vitesse de rotation
R	Plage horizontale (Jet d'un projectile), ou Rayon de la planète (Vitesse de libération)
r	Rayon
t	Temps
v	Vitesse à l'instant t (Mouvement linéaire), ou Vitesse tangentielle à r (Mouvement circulaire), ou Vitesse terminale (Vitesse terminale), ou Vitesse de libération (Vitesse de libération)
$v0$	Vitesse initiale
vx	Composante horizontale de la vitesse à l'instant t
vy	Composante verticale de la vitesse à l'instant t
x	Position horizontale à l'instant t
$x0$	Position horizontale initiale
y	Position verticale à l'instant t
$y0$	Position verticale initiale

Référence : 3.

Mouvement linéaire (Linear Motion: 8, 1)

Equations :

$$x = x0 + v0 \cdot t + \frac{1}{2} \cdot a \cdot t^2 \qquad x = x0 + v \cdot t - \frac{1}{2} \cdot a \cdot t^2$$

$$x = x0 + \frac{1}{2} \cdot \left(v0 + v \right) \cdot t \qquad v = v0 + a \cdot t$$

Exemple :

Soit : $x0$=0_m, x=100_m, t=10_s, $v0$=1_m/s.

Solution : v=19_m/s, a=1.8_m/s^2.

Objet en chute libre (Object in Free Fall: 8, 2)

Equations :

$$y = y0 + v0 \cdot t - \frac{1}{2} \cdot g \cdot t^2 \qquad y = y0 + v \cdot t + \frac{1}{2} \cdot g \cdot t^2$$

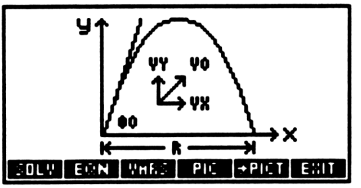
$$v^2 = v0^2 - 2 \cdot g \cdot \left(y - y0 \right) \qquad v = v0 - g \cdot t$$

Exemple :

Soit : $y0$=1000_ft, y=0_ft, $v0$=0_ft/s.

Solution : t=7.8843_s, v=−253.6991_ft/s.

Jet de projectile (Projectile Motion: 8, 3)



Equations :

$$x = x0 + v0 \cdot \mathrm{COS}\left(\theta 0\right) \cdot t \qquad y = y0 + v0 \cdot \mathrm{SIN}\left(\theta 0\right) \cdot t - \frac{1}{2} \cdot g \cdot t^2$$

$$vx = v0 \cdot \mathrm{COS}\left(\theta 0\right) \qquad vy = v0 \cdot \mathrm{SIN}\left(\theta 0\right) - g \cdot t$$

$$R = \frac{v0^2}{g} \cdot \mathrm{SIN}\left(2 \cdot \theta 0\right)$$

Exemple :

Soit : $x0$=0_ft, $y0$=0_ft, $\Theta 0$=45_°, $v0$=200_ft/s, t=10_s.

Solution : R=1243.2399_ft, vx=141.4214_ft/s, vy=−180.3186_ft/s, x=1414.2136_ft, y=−194.4864_ft.

Mouvement angulaire (Angular Motion: 8, 4)

Equations :

$$\theta = \theta 0 + \omega 0 \cdot t + \frac{1}{2} \cdot \alpha \cdot t^2 \qquad \theta = \theta 0 + \omega \cdot t - \frac{1}{2} \cdot \alpha \cdot t^2$$

$$\theta = \theta 0 + \frac{1}{2} \cdot \left(\omega 0 + \omega\right) \cdot t \qquad \omega = \omega 0 + \alpha \cdot t$$

Exemple :

Soit : $\theta 0$=0_°, $\omega 0$=0_r/min, α=1.5_r/min^2, t=30_s.

Solution : θ=10.7430_°, ω=.7500_r/min.

Mouvement circulaire (Circular Motion: 8, 5)

Equations :

$$\omega = \frac{v}{r} \qquad ar = \frac{v^2}{r} \qquad \omega = 2 \cdot \pi \cdot N$$

Exemple :

Soit : r=25_in, v=2500_ft/s.

Solution : ω=72000_r/min, ar=3000000_ft/s^2, N=11459.1559_rpm.

Vitesse terminale (Terminal Velocity: 8, 6)

Equation :

$$v = \sqrt{\frac{2 \cdot m \cdot g}{Cd \cdot \rho \cdot A}}$$

Exemple :

Soit : Cd=.15, ρ=.025lb/ft^3, A=100000_in^2, m=1250_lb.

Solution : v=1757.4709_ft/s.

Vitesse de libération (Escape Velocity: 8, 7)

Equation :

$$v = \sqrt{\frac{2 \cdot G \cdot M}{R}}$$

Exemple :

Soit : M=1.5E23_lb, R=5000_mi.

Solution : v=3485.1106_ft/s.

Optique (Optics: 9)

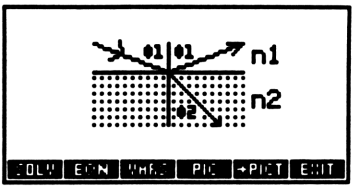
Noms de variable et description

$\theta 1$	Angle d'incidence
$\theta 2$	Angle de réfraction
θB	Angle de Brewster
θc	Angle critique
f	Distance focale
m	Grossissement
n,$n1$,$n2$	Indice de réfraction
r,$r1$,$r2$	Rayon de courbure
u	Distance à l'objet
v	Distance à l'image

Pour les problèmes de réflexion et de réfraction, la distance focale et le rayon de courbure sont positifs dans le sens de la lumière sortante (réfléchie ou réfractée). La distance à l'objet est positive à partir de la surface de la lentille. La distance à l'image est positive dans le sens de la lumière sortante (réfléchie ou réfractée). Le grossissement est positif pour une image droite.

Référence : 3.

Loi de la réfraction (Law of Refraction: 9, 1)



Equation :

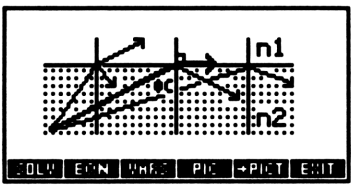
$$n1 \cdot \mathrm{SIN}\left(\theta 1\right) = n2 \cdot \mathrm{SIN}\left(\theta 2\right)$$

Exemple :

Soit : $n1=1$, $n2=1.333$, $\theta 1=45\_°$.

Solution : $\theta 2=32.0367\_°$.

Angle critique (Critical Angle: 9, 2)



Equation :

$$\mathrm{SIN}\left(\theta c\right) = \frac{n1}{n2}$$

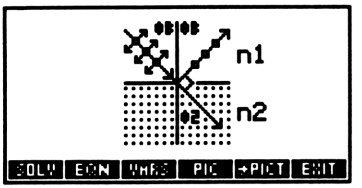
Exemple :

Soit : $n1=1$, $n2=1.5$.

Solution : $\theta c=41.8103\_°$.

Loi de Brewster (Brewster's Law: 9, 3)

L'angle de Brewster est l'angle de la lumière incidente pour lequel la lumière réfléchie est totalement polarisée.



Equations :

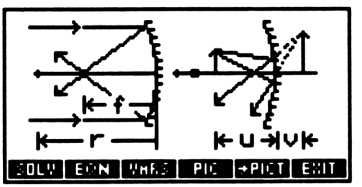
$$\text{TAN}\left(\theta\text{B}\right) = \frac{\text{n2}}{\text{n1}} \qquad \theta\text{B} + \theta 2 = 90$$

Exemple :

Soit : $n1=1$, $n2=1.5$.

Solution : $\theta B=56.3099\_°$, $\theta 2=33.6901\_°$.

Réflexion sphérique (Spherical Reflection: 9, 4)



Equations :

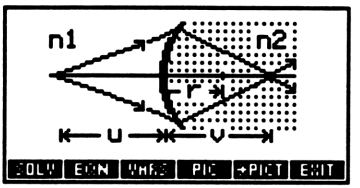
$$\frac{1}{u}+\frac{1}{v}=\frac{1}{f} \qquad f=\frac{1}{2}\cdot r \qquad m=\frac{-v}{u}$$

Exemple :

Soit : u=10_cm, v=300_cm, r=19.35_cm.

Solution : m=−30, f=9.6774_cm.

Réfraction sphérique (Spherical Refraction: 9, 5)



Equation :

$$\frac{n1}{u}+\frac{n2}{v}=\frac{n2-n1}{r}$$

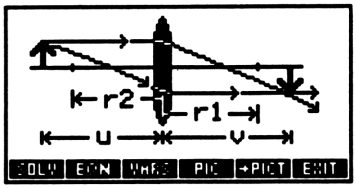
Exemple :

Soit : u=8_cm, v=12_cm, r=2_cm, $n1$=1.

Solution : $n2$=1.5000.

Lentille mince (Thin Lens: 9, 6)

r1 correspond au rayon de courbure de la surface avant de la lentille et *r2* à celui de sa surface arrière.



Equations :

$$\frac{1}{u}+\frac{1}{v}=\frac{1}{f} \qquad \frac{1}{f}=\left(n-1\right)\cdot\left(\frac{1}{r1}-\frac{1}{r2}\right) \qquad m=\frac{-v}{u}$$

Exemple :

Soit : *r1*=5_cm, *r2*=20_cm, *n*=1.5, *u*=50_cm.

Solution : *f*=13.3333_cm, *v*=18.1818_cm, *m*=−.3636.

Oscillations (Oscillations: 10)

Noms de variable et description

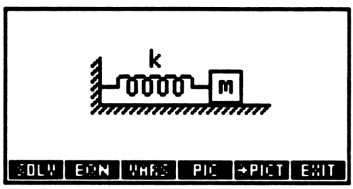
ω	Pulsation
ϕ	Phase
θ	Angle du cône
a	Accéleration à l'instant t
f	Fréquence
G	Module d'élasticité au cisaillement
h	Hauteur du cône

Noms de variable et description (suite)

I	Moment d'inertie
J	Moment polaire d'inertie
k	Constante du ressort
L	Longueur du pendule
m	Masse
t	Temps
T	Période
v	Vélocité à l'instant t
x	Déplacement à l'instant t
xm	Amplitude du déplacement

Référence : 3.

Système masse-ressort (Mass-Spring System: 10, 1)



Equations :

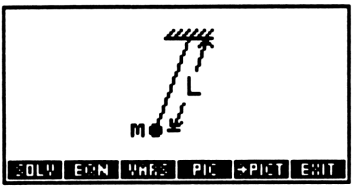
$$\omega = \sqrt{\frac{k}{m}} \qquad T = \frac{2 \cdot \pi}{\omega} \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : k=20_N/m, m=5_kg.

Solution : ω=2_r/s, T=3.1416_s, f=.3183_Hz.

Pendule simple (Simple Pendulum: 10, 2)



Equations :

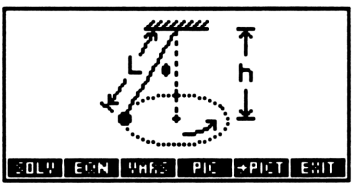
$$\omega = \sqrt{\frac{g}{L}} \qquad T = \frac{2 \cdot \pi}{\omega} \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : L=15_cm.

Solution : ω=8.0856_r/s, T=.7771_s, f=1.2869_Hz.

Pendule conique (Conical Pendulum: 10, 3)



Equations :

$$\omega = \sqrt{\frac{g}{h}} \qquad h = L \cdot \cos\left(\theta\right) \qquad T = \frac{2 \cdot \pi}{\omega}$$

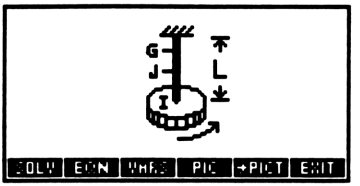
$$\omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : L=25_cm, h=20_cm.

Solution : θ=36.899_°, T=.8973_s, ω=7.0024_r/s, f=1.1145_Hz.

Pendule de torsion (Torsional Pendulum: 10, 4)



Equations :

$$\omega = \sqrt{\frac{G \cdot J}{L \cdot I}} \qquad T = \frac{2 \cdot \pi}{\omega} \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : G=1000_kPa, J=17_mm^4, L=26_cm, I=50_kg*m^2.

Solution : ω=1.1435E−3_r/s, f=1.8200E−4_Hz, T=5494.4862_s.

Harmonique simple (Simple Harmonic: 10, 5)

Equations :

$$x = xm \cdot COS\left(\omega \cdot t + \phi\right) \qquad v = -\omega \cdot xm \cdot SIN\left(\omega \cdot t + \phi\right)$$

$$a = -\omega^2 \cdot xm \cdot COS\left(\omega \cdot t + \phi\right) \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : xm=10_cm, ω=15_r/s, ϕ=25_°, t=25_μs.

Solution : x=9.0615_cm, v=−0.6344_m/s, a=−20.3884_m/s^2, f=2.3873_Hz.

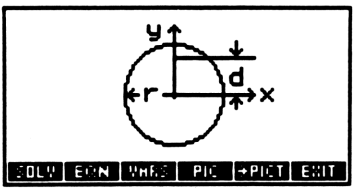
Géométrie plane (Plane Geometry: 11)

Noms de variable et description

β	Angle au centre d'un polygone
θ	Sommet d'un polygone
A	Surface
b	Longueur de la base (Rectangle, Triangle), ou Longueur du demi-axe dans la direction de x (Ellipse)
C	Circonférence
d	Distance à l'axe de rotation dans la direction y
h	Hauteur (Rectangle, Triangel), ou Longueur du demi-axe dans la direction de y (Ellipse)
I,Ix	Moment d'inertie par rapport à l'axe x
Id	Moment d'inertie dans la direction x à d
Iy	Moment d'inertie par rapport à l'axe y
J	Moment polaire d'inertie au centre de gravité
L	Longueur des côtés d'un polygone régulier
n	Nombre de côtés
P	Périmètre
r	Rayon
ri,ro	Rayons intérieur et extérieur
rs	Distance au côté du polygone
rv	Distance au sommet du polygone
v	Distance horizontale au sommet du polygone

Référence : 4.

Cercle (Circle: 11, 1)



Equations :

$$A = \pi \cdot r^2 \qquad C = 2 \cdot \pi \cdot r \qquad I = \frac{\pi \cdot r^4}{4}$$

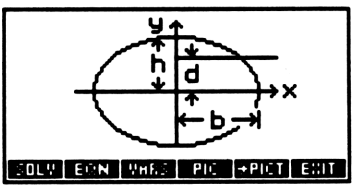
$$J = \frac{\pi \cdot r^4}{2} \qquad Id = I + A \cdot d^2$$

Exemple :

Soit : r=5_cm, d=1.5_cm.

Solution : C=31.4159_cm, A=78.5398_cm^2, I=4908738.5_mm^4, J=9817477.0_mm^4, Id=6675884.4_mm^4.

Ellipse (Ellipse: 11, 2)



Equations :

$$A = \pi \cdot b \cdot h \qquad C = 2 \cdot \pi \cdot \sqrt{\frac{b^2 + h^2}{2}} \qquad I = \frac{\pi \cdot b \cdot h^3}{4}$$

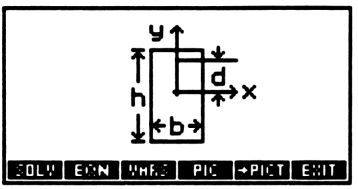
$$J = \frac{\pi \cdot b \cdot h}{4} \cdot \left(b^2 + h^2 \right) \qquad Id = I + A \cdot d^2$$

Exemple :

Soit : b=17.85_μm, h=78.9725_μin, d=.00000012_ft.

Solution : A=1.1249E−6_cm^2, C=7.9805E−3_cm,
I=1.1315E−10_mm^4, J=9.0733E−9_mm^4, Id=1.1330E−10_mm^4.

Rectangle (Rectangle: 11, 3)



Equations :

$$A = b \cdot h \qquad P = 2 \cdot b + 2 \cdot h \qquad I = \frac{b \cdot h^3}{12}$$

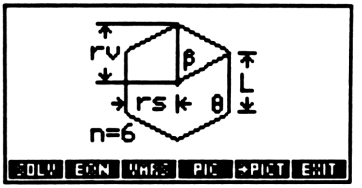
$$J = \frac{b \cdot h}{12} \cdot \left(b^2 + h^2 \right) \qquad Id = I + A \cdot d^2$$

Exemple :

Soit : b=4_chain, h=7_rd, d=39.26_in.
Set guesses for I, J, and Id in km^4.

Solution : A=28328108.2691_cm^2, P=23134.3662_cm,
I=2.9257E−7_km^4, J=1.8211E−6_km^4, Id=2.9539E−7_km^

Polygone régulier (Regular Polygon: 11, 4)



Equations :

$$A = \frac{\frac{1}{4} \cdot n \cdot L^2}{\text{TAN}\left(\frac{180}{n}\right)} \qquad P = n \cdot L \qquad rs = \frac{\frac{L}{2}}{\text{TAN}\left(\frac{180}{n}\right)}$$

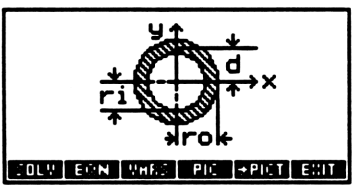
$$rv = \frac{\frac{L}{2}}{\text{SIN}\left(\frac{180}{n}\right)} \qquad \theta = \frac{n-2}{n} \cdot 180 \qquad \beta = \frac{360}{n}$$

Exemple :

Soit : n=8, L=.5_yd.

Solution : A=10092.9501_cm^2, P=365.7600_cm, rs=55.1889_cm, rv=59.7361_cm, θ=135_°, β=45_°.

Anneau circulaire (Circular Ring: 11, 5)



Equations :

$$A = \pi \cdot \left(ro^2 - ri^2 \right) \qquad I = \frac{\pi}{4} \cdot \left(ro^4 - ri^4 \right)$$

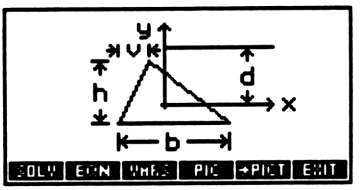
$$J = \frac{\pi}{2} \cdot \left(ro^4 - ri^4 \right) \qquad Id = I + A \cdot d^2$$

Exemple :

Soit : *ro*=4_μ, *ri*=25.0_k+, *d*=.1_mil.

Solution : *A*=3.0631E−7_cm^2, *I*=1.7038E−10_mm^4, *J*=3.4076E−10_mm^4, *Id*=3.0648E−10_mm^4.

Triangle (Triangle: 11, 6)



Equations :

$$A = \frac{b \cdot h}{2} \qquad P = b + \sqrt{v^2 + h^2} + \sqrt{(b - v)^2 + h^2}$$

$$Ix = \frac{b \cdot h^3}{36} \qquad Iy = \frac{b \cdot h}{36} \cdot \left(b^2 - b \cdot v + v^2 \right)$$

$$J = \frac{b \cdot h}{36} \cdot \left(h^2 + b^2 - b \cdot v + v^2 \right) \qquad Id = Ix + A \cdot d^2$$

Exemple :

Soit : h=4.33012781892_in, v=2.5_in, P=15_in, d=2_in.

Solution : b=5.0000_in, Ix=11.2764_in^4, Iy=11.2764_in^4, J=22.5527_in^4, A=10.8253_in^2, Id=54.5776_in^4.

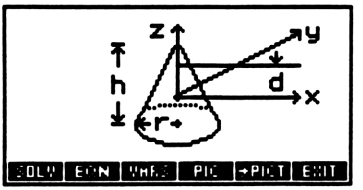
Géométrie dans l'espace (Solid Geometry: 12)

Noms de variable et description

A	Surface totale
b	Longueur de la base
d	Distance à l'axe de rotation dans la direction z
h	Hauteur dans la direction z (Cône, Cylindre), ou Hauteur dans la direction y (Parallélépipède)
I,Ixx	Moment d'inertie par rapport à l'axe x
Id	Moment d'inertie dans la direction de x à d
Izz	Moment d'inertie par rapport à l'axe z
m	Masse
r	Rayon
t	Epaisseur dans la direction z
V	Volume

Référence : 4.

Cône (Cone: 12, 1)



Equations :

$$V = \frac{\pi}{3} \cdot r^2 \cdot h \qquad A = \pi \cdot r^2 + \pi \cdot r \cdot \sqrt{r^2 + h^2}$$

$$Ixx = \frac{3}{20} \cdot m \cdot r^2 + \frac{3}{80} \cdot m \cdot h^2 \qquad Izz = \frac{3}{10} \cdot m \cdot r^2$$

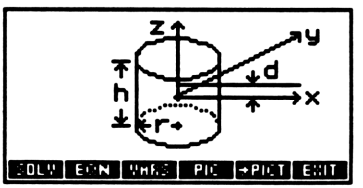
$$Id = Ixx + m \cdot d^2$$

Exemple :

Soit : r=7_cm, h=12.5_cm, m=12.25_kg, d=3.5_cm.

Solution : V=641.4085_cm^3, A=468.9953_cm^2, Ixx=0.0162_kg*m^2, Izz=0.0180_kg*m^2, Id=0.0312_kg*m^2.

Cylindre (Cylinder: 12, 2)



Equations :

$$V = \pi \cdot r^2 \cdot h \qquad A = 2 \cdot \pi \cdot r^2 + 2 \cdot \pi \cdot r \cdot h$$

$$Ixx = \frac{1}{4} \cdot m \cdot r^2 + \frac{1}{12} \cdot m \cdot h^2 \qquad Izz = \frac{1}{2} \cdot m \cdot r^2$$

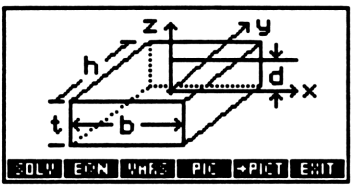
$$Id = Ixx + m \cdot d^2$$

Exemple :

Soit : r=8.5_in, h=65_in, m=12000_lbs, d=2.5_in.

Solution : V=14753.7045_in^3, A=3925.4200_in^2, Izz=4441750_lb*in^2, Izz=433500_lb*in^2, Id=4516750_lb*in^2.

Parallélépipède (Parallelepiped: 12, 3)



Equations :

$$V = b \cdot h \cdot t \qquad A = 2 \cdot \left(b \cdot h + b \cdot t + h \cdot t \right)$$

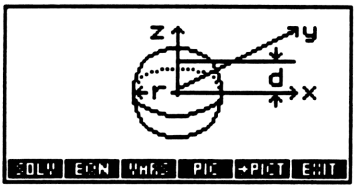
$$I = \frac{1}{12} \cdot m \cdot \left(h^2 + t^2 \right) \qquad Id = I + m \cdot d^2$$

Exemple :

Soit : b=36_in, h=12_in, t=72_in, m=83_lb, d=7_in.

Solution : V=31104_in^3, A=7776_in^2, I=36852_lb*in^2, Id=40919_lb*in^2.

Sphère (Sphere: 12, 4)



Equations :

$$V = \frac{4}{3}\cdot\pi\cdot r^3 \qquad A = 4\cdot\pi\cdot r^2 \qquad I = \frac{2}{5}\cdot m\cdot r^2 \qquad Id = I + m\cdot d^2$$

Exemple :

Soit : d=14_cm, m=3.75_kg, Id=486.5_lb*in^2.

Solution : r=21.4273_cm, V=41208.7268_cm^3, A=5769.5719_cm^2, I=0.0689_kg*m^2.

Composants à semi-conducteur (Solid State Devices: 13)

Noms de variable et description

αF	Gain en courant direct en base commune
αR	Gain en courant inverse en base commune
γ	Facteur de corps
λ	Paramètre de modulation
μn	Mobilité des électrons
ϕp	Potentiel de Fermi
ΔL	Ajustement de la longueur (Jonctions à transition PN), ou Empiètement du canal (Transistors NMOS)
ΔW	Ajustement de la largeur (Jonctions à transition PN), ou Contraction de la largeur (Transistors NMOS)
a	Epaisseur du canal
Aj	Surface réelle de la jonction
BV	Tension de claquage
Cj	Capacité de la jonction par unité de surface
Cox	Capacité du bioxide de silicium par unité de surface
E1	Facteur de champ de la tension de claquage
Emax	Champ électrique maximal
G0	Conductance du canal
gds	Conductance de sortie
gm	Transconductance
I	Courant dans la diode
IB	Courant de base total
IC	Courant de collecteur total
ICEO	Courant de collecteur (circuit collecteur-base ouvert)
ICO	Courant de collecteur (circuit émetteur-base ouvert)

Noms de variable et description (suite)

ICS	Courant de saturation collecteur-base
ID,IDS	Courant de drain
IE	Courant d'émetteur total
IS	Courant de saturation d'un transistor
J	Densité du courant
Js	Densité du courant de saturation
L	Longueur du masque dessiné (Jonctions à transition PN), ou Longueur de la porte dessinée (Transistors NMOS), ou Longueur du canal (JFET)
Le	Longueur réelle de la porte
NA	Dopage du côté P (Jonctions à transition PN), ou Dopage du substrat (Transistors NMOS)
ND	Dopage du côté N (Jonctions à transition PN), ou Dopage du canal N (JFET)
T	Température
tox	Epaisseur de la porte en bioxide de silicium
Va	Tension appliquée
VBC	Tension base-collecteur
VBE	Tension base-émetteur
Vbi	Tension interne
VBS	Tension de substrat
VCEsat	Tension de saturation collecteur-émetteur
VDS	Tension de drain appliquée
VDsat	Tension de saturation
VGS	Tension de porte appliquée
Vt	Tension de seuil
Vt0	Tension de seuil (pour une tension de substrat nulle)

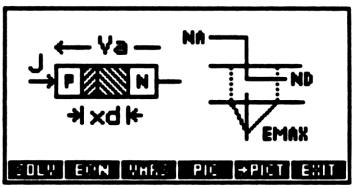
Noms de variable et description (suite)

W	Largeur du masque dessiné (Jonctions à transition PN), ou Largeur dessinée (Transistors NMOS), ou Largeur du canal (JFET)
We	Largeur réelle
xd	Largeur de la région d'appauvrissement
xdmax	Largeur de la couche d'appauvrissement
xj	Profondeur de la jonction

Références : 5, 8.

Jonctions à transition PN (PN Step Junctions: 13, 1)

Ces équations pour une diode au silicium à jonction PN utilisent un modèle de fonction à transition brutale entre deux régions - la densité de dopage change brutalement au niveau de la jonction. Les équations supposent que la densité du courant est déterminée par les porteurs minoritaires injectés à travers la région d'appauvrissement et que la jonction PN a une disposition rectangulaire. La température doit être comprise entre 77 et 500 K. (Voir SIDENS au chapitre 3).



Equations :

$$Vbi = \frac{k \cdot T}{q} \cdot LN\left(\frac{NA \cdot ND}{ni^2}\right)$$

$$xd = \sqrt{\frac{2 \cdot \epsilon si \cdot \epsilon 0}{q} \cdot (Vbi - Va) \cdot \left(\frac{1}{NA} + \frac{1}{ND}\right)}$$

$$Cj = \frac{\epsilon si \cdot \epsilon 0}{xd} \qquad Emax = \frac{2 \cdot \left(Vbi - Va\right)}{xd}$$

$$BV = \frac{\epsilon si \cdot \epsilon 0 \cdot E1^2}{2 \cdot q} \cdot \left(\frac{1}{NA} + \frac{1}{ND}\right) \qquad J = Js \cdot \left(e^{\frac{q \cdot Va}{k \cdot T}} - 1\right)$$

$$Aj = \left(W + 2 \cdot \Delta W\right) \cdot \left(L + 2 \cdot \Delta L\right) + \pi \cdot \left(W + L + 2 \cdot \Delta W + 2 \cdot \Delta L\right) \cdot xj + 2 \cdot \pi \cdot xj^2$$

$$I = J \cdot Aj$$

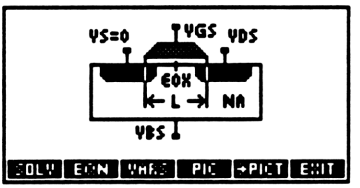
Exemple :

Soit : ND=1E22_cm^−3, NA=1E15_1/cm^3, T=26.85_°C, Js=1E−6_μA/cm^2, Va=−20_V, $E1$=3.3E5_V/cm, W=10_μ, ΔW=1_μ, L=10_μ, ΔL=1_μ, xj=2_μ.

Solution : Vbi=.9962_V, xd=5.2551_μ, Cj=2005.0141_pF/cm^2, $Emax$=79908.5240_V/cm, BV=358.0825_V, J=−1.0E−12_A/cm^2, Aj=3.1993E−6_cm^2, I=−3.1993E−15_mA.

Transistors NMOS (NMOS Transistors: 13, 2)

Ces équations pour un transistor NMOS au silicium utilisent un modèle de quadripôle. Elles tiennent compte des régions linéaire et non linéaire des caractéristiques et sont fondées sur une approximation de canal progressif (les champs électriques dans le sens du courant sont faibles par rapport à ceux perpendiculaires à ce courant). Les calculs du courant de drain et de transconductance sont différents selon que le transistor est dans la région linéaire, de saturation ou de blocage de ses caractéristiques. Les équations supposent que le composant est de forme rectangulaire et que les effets des paramètres de longueur du second ordre, de canal court, de mobilité élevée des porteurs et de saturation de la vitesse, ainsi que les courants de seuil sont négligeables. (Voir SIDENS au chapitre 3).



Equations :

$$We = W - 2 \cdot \Delta W \qquad Le = L - 2 \cdot \Delta L \qquad Cox = \frac{\epsilon ox \cdot \epsilon 0}{tox}$$

$$IDS = Cox \cdot \mu n \cdot \left(\frac{We}{Le} \right) \cdot \left((VGS - Vt) \cdot VDS - \frac{VDS^2}{2} \right) \cdot (1 + \lambda \cdot VDS)$$

$$\gamma = \frac{\sqrt{2 \cdot \epsilon si \cdot \epsilon 0 \cdot q \cdot NA}}{Cox}$$

$$Vt = Vt0 + \gamma \cdot \left(\sqrt{2 \cdot ABS(\phi p) + ABS(VBS)} - \sqrt{2 \cdot ABS(\phi p)} \right)$$

$$\phi p = \frac{-k \cdot T}{q} \cdot LN \left(\frac{NA}{ni} \right) \qquad gds = IDS \cdot \lambda$$

$$gm = \sqrt{Cox \cdot \mu n \cdot \left(\frac{We}{Le} \right) \cdot \left(1 + \lambda \cdot VDS \right) \cdot 2 \cdot IDS}$$

$$VDsat = VGS - Vt$$

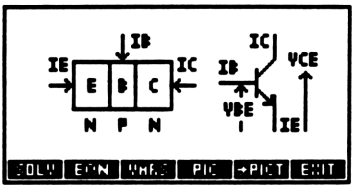
Exemple :

Soit : tox=700_+, NA=1E15_1/cm^3, μn=600_cm^2/(V*s), T=26.85_°C, $Vt0$=.75_V, VGS=5_V, VBS=0_V, VDS=5_V, W=25_μ, ΔW=1_μ, L=4_m, ΔL=.75_μ, λ=.05_1/V.

Solution : We=23_μ, Le=2.5_μ, Cox=49330.4750_pF/cm^2, γ=.3725_V^.5, ϕp=-.2898_V, Vt=.75_V, $VDsat$=4.25_V, IDS=3.0741_mA, gds=1.5370E-4_S, gm=1.4466_mA/V.

Transistors bipolaires (Bipolar Transistors: 13, 3)

Ces équations pour un transistor bipolaire NPN au silicium sont fondées sur les modèles développés pour les signaux importants par J.J. Ebers et J.L. Moll. Le calcul de la tension de décalage est différent selon que le transistor est saturé ou non. Les équations tiennent aussi compte des conditions spéciales où la jonction émetteur-base ou collecteur-base est ouverte, qui sont commodes pour mesurer les paramètres d'un transistor.



Equations :

$$IE = -IES \cdot \left(e^{\frac{q \cdot VBE}{k \cdot T}} - 1 \right) + \alpha R \cdot ICS \cdot \left(e^{\frac{q \cdot VBC}{k \cdot T}} - 1 \right)$$

$$IC = -ICS \cdot \left(e^{\frac{q \cdot VBC}{k \cdot T}} - 1 \right) + \alpha F \cdot IES \cdot \left(e^{\frac{q \cdot VBE}{k \cdot T}} - 1 \right)$$

$$IS = \alpha F \cdot IES \qquad IS = \alpha R \cdot ICS \qquad IB + IE + IC = 0$$

$$ICO = ICS \cdot \left(1 - \alpha F \cdot \alpha R \right) \qquad ICEO = \frac{ICO}{1 - \alpha F}$$

$$VCEsat = \frac{k \cdot T}{q} \cdot LN \left(\frac{1 + \frac{IC}{IB} \cdot \left(1 - \alpha R \right)}{\alpha R \cdot \left(1 - \frac{IC}{IB} \cdot \left(\frac{1 - \alpha F}{\alpha F} \right) \right)} \right)$$

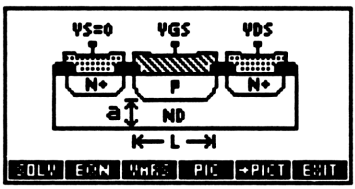
Exemple :

Soit : IES=1E−5_nA, ICS=2E−5_nA, T=26.85_°C, αF=.98, αR=.49, IC=1_mA, VBC=−10_V.

Solution : VBE=.6553_V, IS=0.0000098_nA, ICO=.000010396_nA, $ICEO$=.0005198_nA, IE=−1.0204_mA, IB=.0204_mA, $VCEsat$=0_V.

JFET (JFETs: 13, 4)

Ces équations pour un transistor à effet de champ au silicium de canal N (JFET) sont fondées sur l'approximation d'une jonction de transition brutale, qui suppose que les portes sont fortement dopées par rapport au canal. Le calcul du courant de drain est différent selon que l'épaisseur de la zone d'appauvrissement de la fonction de porte est inférieure ou supérieure à l'épaisseur du canal. Les équations supposent que le canal est dopé uniformément et que les effets des connexions (comme les résistances de contacts, de drain et de source) sont négligeables. (Voir SIDENS au chapitre 3).



Equations :

$$Vbi = \frac{k \cdot T}{q} \cdot LN\left(\frac{ND}{ni}\right)$$

$$xdmax = \sqrt{\frac{2 \cdot \epsilon si \cdot \epsilon 0}{q \cdot ND} \cdot \left(Vbi - VGS + VDS\right)}$$

$$G0 = q \cdot ND \cdot \mu n \cdot \left(\frac{a \cdot W}{L}\right)$$

$$ID = G0 \cdot \left(VDS - \frac{2}{3} \cdot \sqrt{\frac{2 \cdot \epsilon si \cdot \epsilon 0}{q \cdot ND \cdot a^2}} \cdot \left(\left(Vbi - VGS + VDS\right)^{\frac{3}{2}} - \left(Vbi - VGS\right)^{\frac{3}{2}}\right)\right)$$

$$VDsat = \frac{q \cdot ND \cdot a^2}{2 \cdot \epsilon si \cdot \epsilon 0} - (Vbi - VGS) \qquad Vt = Vbi - \frac{q \cdot ND \cdot a^2}{2 \cdot \epsilon si \cdot \epsilon 0}$$

$$gm = G0 \cdot \left(1 - \sqrt{\frac{2 \cdot \epsilon si \cdot \epsilon 0}{q \cdot ND \cdot a^2} \cdot (Vbi - VGS)}\right)$$

Exemple :

Soit : ND=1E16_1/cm^3, W=6_μ, a=1_μ, L=2_μ, μn=1248_cm^2/(V*s), VGS=−4_V, VDS=4_V, T=26.85_°C.

Solution : Vbi=.3493_V, $xdmax$=1.0479_μ, $G0$=5.9986E−4_S, ID=.2268_mA, $VDsat$=3.2537_V, Vt=−7.2537_V, gm=.1462_mA/V.

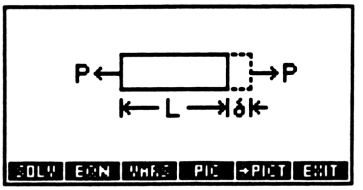
Résistance des matériaux (Stress Analysis:14)

Noms de variable et description

δ	Allongement
ϵ	Déformation normale
γ	Déformation par cisaillement
ϕ	Angle de torsion
σ	Contrainte normale
$\sigma 1$	Contrainte normale principale maximale
$\sigma 2$	Contrainte normale principale minimale
σavg	Contrainte normale au plan de contrainte maximale par cisaillement
σx	Contrainte normale dans la direction x
$\sigma x1$	Contrainte normale dans la direction x après rotation
σy	Contrainte normale dans la direction y
$\sigma y1$	Contrainte normale dans la direction y après rotation
τ	Contrainte par cisaillement
τmax	Contrainte par cisaillement maximale
$\tau x1y1$	Contrainte par cisaillement après rotation
τxy	Contrainte par cisaillement
θ	Angle de rotation
$\theta p1$	Angle par rapport au plan de contrainte normale principale maximale
$\theta p2$	Angle par rapport au plan de contrainte normale principale minimale
θs	Angle par rapport au plan contrainte par cisaillement maximale
A	Surface
E	Module d'élasticité
G	Module d'élasticité au cisaillement
J	Moment polaire d'inertie
L	Longueur
P	Charge
r	Rayon
T	Couple

Référence : 2.

Contrainte normale (Normal Stress: 14, 1)



Equations :

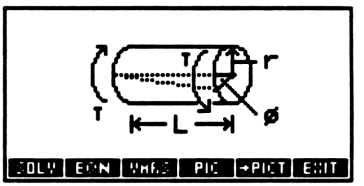
$$\sigma = E \cdot \epsilon \qquad \epsilon = \frac{\delta}{L} \qquad \sigma = \frac{P}{A}$$

Exemple :

Soit : P=40000_lbf, L=1_ft, A=3.14159265359_in^2, E=10E6_psi.

Solution : δ=0.0153_in, ϵ=0.0013, σ=12732.3954_psi.

Effort tranchant (Shear Stress: 14, 2)



Equations :

$$\tau = G \cdot \gamma \qquad \gamma = \frac{r \cdot \phi}{L} \qquad \tau = \frac{T \cdot r}{J}$$

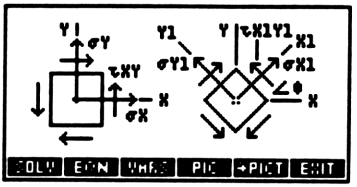
Exemple :

Soit : L=6_ft, r=2_in, J=10.4003897419_in^4, G=12000000_psi, τ=12000_psi.

Solution : T=5200.1949_ft*lbf, ϕ=2.0626_°, γ=5.7296E-2_°.

Contrainte exercée sur un élément (Stress on an Element: 14, 3)

Les contraintes et les déformations sont positives dans les directions indiquées.



Equations :

$$\sigma x1 = \frac{\sigma x + \sigma y}{2} + \frac{\sigma x - \sigma y}{2} \cdot \mathrm{COS}(2 \cdot \theta) + \tau xy \cdot \mathrm{SIN}(2 \cdot \theta)$$

$$\sigma x1 + \sigma y1 = \sigma x + \sigma y$$

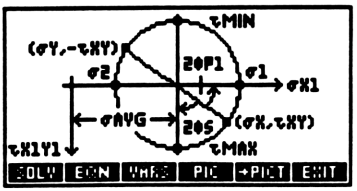
$$\tau x1y1 = -\left(\frac{\sigma x - \sigma y}{2}\right) \cdot \mathrm{SIN}(2 \cdot \theta) + \tau xy \cdot \mathrm{COS}(2 \cdot \theta)$$

Exemple :

Soit : σx=15000_kPa, σy=4755_kPa, τxy=7500_kPa, θ=30_°.

Solution : $\sigma x1$=18933.9405_kPa, $\sigma y1$=821.0595_kPa, $\tau x1y1$=−686.2151_kPa.

Cercle de Mohr (Mohr's Circle: 14, 4)



Equations :

$$\sigma 1 = \frac{\sigma x + \sigma y}{2} + \sqrt{\left(\frac{\sigma x - \sigma y}{2}\right)^2 + \tau xy^2}$$

$$\sigma 1 + \sigma 2 = \sigma x + \sigma y$$

$$\mathrm{SIN}(2 \cdot \theta p1) = \frac{\tau xy}{\sqrt{\left(\frac{\sigma x - \sigma y}{2}\right)^2 + \tau xy^2}}$$

$$\theta p2 = \theta p1 + 90 \qquad \tau max = \frac{\sigma 1 - \sigma 2}{2}$$

$$\theta s = \theta p1 - 45 \qquad \sigma avg = \frac{\sigma x + \sigma y}{2}$$

Exemple :

Soit : σx=−5600_psi, σy=−18400_psi, τxy=4800_psi.

Solution : $\sigma 1$=−4000_psi, $\sigma 2$=−20000_psi, t =18.4349_°, $\theta p2$=108.4349_°, τmax=8000_psi, θs=−26.5651_°, σavg=−12000_psi.

Ondes (Waves: 15)

Noms de variable et description

β	Niveau sonore
λ	Longueur d'onde
ω	Pulsation
ρ	Densité du milieu
B	Module d'élasticité dans la masse
f	Fréquence
I	Intensité acoustique
k	Indice angulaire de l'onde
s	Déplacement longitudinal au point x et à l'instant t
sm	Amplitude longitudinale
t	Temps
v	Vitesse du son dans le milieu (Ondes acoustiques), ou Vitesse de propagation de l'onde (Ondes transversales, Ondes longitudinales)
x	Position
y	Déplacement transversal au point x et à l'instant t
ym	Amplitude transversale

Référence : 3.

Ondes transversales (Transverse Waves: 15, 1)

Equations :

$$y = ym \cdot SIN\left(k \cdot x - \omega \cdot t\right) \qquad v = \lambda \cdot f \qquad k = \frac{2 \cdot \pi}{\lambda} \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : ym=6.37_cm, k=32.11_r/cm, x=.03_cm, ω=7000_r/s, t=1_s.

Solution : f=1114.0846_Hz, λ=.0020_cm, y=2.6655_cm, v=218.0006_cm/s.

Ondes longitudinales (Longitudinal Waves: 15, 2)

Equations :

$$s = sm \cdot COS\left(k \cdot x - \omega \cdot t\right) \qquad v = \lambda \cdot f \qquad k = \frac{2 \cdot \pi}{\lambda} \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : sm=6.37_cm, k=32.11_r/cm, x=.03_cm, ω=7000_r/s, t=1_s.

Solution : s=5.7855_cm, v=2.1800_m/s, λ=.1957_cm, f=1114.0846_Hz.

Ondes acoustiques (Sound Waves: 15, 3)

Equations :

$$v = \sqrt{\frac{B}{\rho}} \qquad I = \frac{1}{2} \cdot \rho \cdot v \cdot \omega^2 \cdot sm^2$$

$$\beta = 10 \cdot LOG\left(\frac{I}{I0}\right) \qquad \omega = 2 \cdot \pi \cdot f$$

Exemple :

Soit : sm=10_cm, ω=6000_r/s, B=12500_kPa, ρ=65_kg/m^3.

Solution : v=438.5290_m/s, I=5130789412.97_W/m^2, β=217.1018_dB, f=954.9297_Hz.

Références

1. Dranchuk, P.M., R.A. Purvis, and D.B. Robinson. "Computer Calculations of Natural Gas Compressibility Factors Using the Standing and Katz Correlation," In *Institute of Petroleum Technical Series*, no. IP 74-008. 1974.

2. Gere, James M., and Stephen P. Timoshenko. *Mechanics of Materials*, 2d ed. PWS Engineering, Boston, 1984.

3. Halliday, David, and Robert Resnick. *Fundamentals of Physics*, 3d ed. John Wiley & Sons, 1988.

4. Meriam, J.L., and L.G. Kraige. *Engineering Mechanics*, 2d ed. John Wiley & Sons, 1986.

5. Muller, Richard S., and Theodore I. Kamins. *Device Electronics for Integrated Cicuits*, 2d ed. John Wiley & Sons, 1986.

6. Serghides, T.K. "Estimate Friction Factor Accurately," In *Chemical Engineering*, Mar. 5, 1984.

7. Siegel, Robert, and John Howell. *Thermal Radiation Heat Transfer*, Vol. 1. National Aeronautics and Space Administration, 1968.

8. Sze, S. *Physics of Semiconductors*, 2d ed. John Wiley & Sons, 1981.

9. Welty, Wicks, and Wilson. *Fundamentals of Momentum, Heat and Mass Transfer*. John Wiley & Sons, 1969.

A

Messages d'erreur et d'état

Messages classés par ordre alphabétique

Message	Signification	# (hex)
Acknowledged	Alarme reçue	619
All Variables Known	Aucune inconnue pour laquelle l'équation peut être résolue	E405
Autoscaling	Echelle des axes x et/ou y automatique	610
Awaiting Server Cmd.	Indique que le mode serveur est actif	C0C
Bad Argument Type	Un ou plusieurs paramètres sur la pile étaient d'un type incorrect pour l'opération	202
Bad Argument Value	Valeur du paramètre hors des limites admissibles pour l'opération concernée	203
Bad Guess(es)	Estimation(s) fournie(s) à l'application HP Solve ou ROOT en dehors du domaine de la fonction	A01

Bad Packet Block check	Somme de contrôle calculée pour le bloc transmis ne correspond pas à la somme de contrôle dans le bloc	C01
Can't Edit Null Char.	Tentative d'éditer une chaîne contenant le caractère de code 0.	102
Circular Reference	Tentative de ranger un nom de variable dans lui-même	129
Connecting	Indique une vérification en cours de la connexion série ou infra-rouge	C0A
Constant?	L'application HP Solve ou ROOT retourne la même valeur en tout point d'échantillonnage de l'équation courante	A02
Copied to stack	La commande [→STACK] a copié l'équation sur la pile	623
Current equation:	Identifie l'équation courante	608
Deleting Column	L'application MatrixWriter supprime une colonne	504
Deleting Row	MatrixWriter est en train de supprimer une ligne	503
Directory Not Allowed	Un nom de répertoire existant est utilisé comme paramètre	12A
Directory Recursion	Tentative de rangement d'un répertoire dans lui-même	002
Empty catalog	Pas de donnée dans le catalogue courant (équation, stat, alarme)	60D

Empty stack	La pile opérationnelle ne contient aucune donnée	C15
Enter alarm, press SET	Invite à introduire une alarme	61A
Enter eqn, press NEW	Range une nouvelle équation dans *EQ*	60A
Enter value (zoom out if >1), press ENTER	Invite pour les opérations de zoom	622
EQ invalid for MINIT	*EQ* doit contenir au moins deux équations (ou program-gmes) et deux variables	E403
Extremum	Le résultat fourni par l'application HP Solve ou ROOT est un extrêmum plutôt qu'une racine	A06
HALT Not Allowed	Un programme contenant un HALT est exécuté alors qu'une application MatrixWriter, DRAW, ou HP Solve est active.	126
I/O setup menu	Identifie le menu d'initialisation des entrées-sorties	61C
Illegal During MROOT	Une commande du Multiple-Equation Solver a été tentée pendant l'exécution de MROOT	E406
Implicit ( ) off	Les parenthèses implicites sont désactivées	207
Implicit ( ) on	Les parenthèses implicites sont activées	208

Incomplete Subexpression	Une des touches ENTER ▶ ▼ a été actionnée avant que tous les paramètres d'une fonction aient été fournis	206
Inconsistent Units	Conversion tentée avec des unités incompatibles	B02
Infinite Result	Résultat infini provenant d'un calcul tel que 1 / 0	305
Inserting Column	L'application MatrixWriter est en train d'insérer une colonne	506
Inserting Row	L'application MatrixWriter est en train d'insérer une ligne	505
Insufficient Memory	Pas assez de mémoire libre pour effectuer l'opération	001
Insufficient Σ Data	Une commande statistique a été exécutée alors que Σ*DAT* ne contenait pas assez de données pour effectuer le calcul	603
Interrupted	L'application HP Solve ou ROOT a été interrompue par CANCEL	A03
Invalid Array Element	ENTER a fourni un objet de type incorrect pour la matrice courante	502
Invalid Card Data	La HP 48 ne reconnaît pas les données sur la carte enfichable	008
Invalid Date	La date introduite comme paramètre n'est pas un nombre réel au format correct	D01

Invalid Definition	Paramètre d'équation ayant une structure incorrecte pour DEFINE	12C
Invalid Dimension	Le tableau utilisé a des dimensions incorrectes	501
Invalid EQ	Tentative d'opération à partir du menu PICTURE FCN ou tentative d'exécution de DRAW en mode de tracé conique alors que *EQ* ne contient pas d'équation algébrique	607
Invalid IOPAR	La variable *IOPAR* ne contient pas une liste, on l'un des objets dans la liste est manquant ou non valable	C12
Invalid Mpar	La variable *Mpar* n'a pas été créée par MINIT	E401
Invalid Name	Nom de fichier reçu illégal, ou le serveur a demandé la transmission d'un fichier au nom illégal	C17
Invalid PPAR	La variable *PPAR* ne contient pas une liste, on l'un des objets dans la liste est manquant ou non valable	12E
Invalid PRTPAR	La variable *PRTPAR* ne contient pas une liste, on l'un des objets dans la liste est manquant ou non valable	C13

Invalid PTYPE	Type de tracé non valable pour l'équation courante	620
Invalid Repeat	L'intervalle de répétition d'alarme est hors limites	D03
Invalid Server Cmd.	Commande non valable reçue alors que la HP 48 se trouvait en mode serveur	C08
Invalid Syntax	La HP 48 n'a pas été capable d'exécuter la commande ENTER OBJ→ ou STR→ en raison d'une syntaxe non valable de l'objet	106
Invalid Time	L'heure introduite comme paramètre n'est pas un nombre réel au format correct	D02
Invalid Unit	Tentative d'opération sur les unités avec une unité utilisateur non valable ou non définie	B01
Invalid User Function	Type ou structure incorrects de l'objet exécuté en tant que fonction définie par l'utilisateur	103
Invalid Σ Data	Commande statistique exécutée avec un objet non valable rangé dans Σ*DAT*	601
Invalid Σ Data LN (Neg)	Tentative d'ajustement non linéaire, alors que la matrice Σ*DAT* contient un élément <0	605
Invalid Σ Data LN (0)	Idem alors que la matrice Σ*DAT* contient un élément nul	606

Invalid ΣPAR	La variable Σ*PAR* ne contient pas une liste, on l'un des objets dans la liste est manquant ou non valable	604
Keyword Conflict	Une carte enfichable entre en conflit avec la variable de la bibliothèque d'équations. Retirer la carte pour continuer	E303
LAST CMD Disabled	(→) (CMD) appuyé alors que cette procédure de récupération est désactivée	125
LAST STACK Disabled	(→) (UNDO) appuyé alors que cette procédure de récupération est désactivée	124
LASTARG Disabled	(→) (ARG) appuyé alors que cette procédure de récupération est désactivée	205
Low Battery	Piles système trop faibles pour imprimer ou établir une liaison d'entrée/sortie en toute sécurité	C14
Memory Clear	La mémoire de la HP 48 a été vidée de son contenu	005
Name Conflict	Tentative d'exécution de | ("où") pour attribuer une valeur numérique à une variable d'intégration ou à un indice de sommation	13C
Name the equation press ENTER	Donner un nom à l'équation et la ranger dans *EQ*	60B

Name the stat data, press ENTER	Donner un nom aux données statistiques et les ranger dans ΣDAT	621
Negative Underflow	Exception mathématique : le calcul a fourni un résultat $\neq 0$ négatif supérieur à –MINR	302
No Current Equation	[SOLVR], DRAW ou RCEQ ont été exécutés avec une variable *EQ* inexistante	104
No current equation.	Message de même type pour les applications de tracé et pour HP Solve	609
No Picture Available	Aucune image n'est disponible pour l'équation sélectionnée	E304
No Room in Port	Mémoire libre insuffisante dans le port RAM sélectionné	00B
No Room to Save Stack	Pas assez de mémoire libre pour sauvegarder une copie de la pile : LAST STACK est automatiquement désactivé	101
No Room to Show Stack	Le type seul des objets sur la pile est affiché en raison d'une trop faible capacité mémoire	131
No stat data to plot	Aucune donnée n'est rangée dans ΣDAT	60F
Non-Empty Directory	Tentative d'élimination d'un répertoire non vide	12B

Non-Real Result	L'exécution de l'application HP Solve, ROOT, DRAW ou ∫ a fourni un résultat autre qu'un nombre réel ou d'unité	12F
Nonexistent Alarm	La liste d'alarmes ne contenait pas celle que demande la commande d'alarme	D04
Nonexistent ∑DAT	Exécution d'une commande statistique en l'absence de liste *∑DAT*	602
Object Discarded	L'émetteur a envoyé un bloc EOF (Z) avec un "D" dans le champ de données	C0F
Object In Use	Tentative d'effectuer PURGE ou STO sur un objet de sauvegarde alors que l'objet qui y est rangé était en cours d'utilisation	009
Object Not in Port	Tentative d'accéder à un objet de sauvegarde ou à une bibliothèque inexistant(e)	00C
(OFF SCREEN)	La valeur de la fonction, la racine ou l'extrêmum n'était pas visible sur l'affichage courant	61F
Out of Memory	Un ou plusieurs objets doivent être éliminés pour pouvoir continuer les opérations	135
Overflow	Exception mathématique : le calcul a fourni un résultat plus grand que MAXR en valeur absolue	303

Packet #	Indique le numéro de bloc transmis pendant un envoi ou une réception	C10
Parity Error	Le bit de parité des octets reçus ne correspond pas à la configuration de parité courante	C05
Plot Type :	Etiquette présentant le type de tracé courant	61D
Port Closed	Défaillance matérielle éventuelle de la liaison infra-rouge ou série. Effectuer la procédure d'auto-test	C09
Port Not Available	Commande de port utilisée sur un port vide ou sur un port contenant une ROM au lieu d'une RAM Tentative d'exécuter une commande serveur qui utilise elle-même le port d'entrée/sortie	00A
Positive Underflow	Exception mathématique : le calcul a fourni un résultat $\neq 0$ positif inférieur à MINR	301
Power Lost	La calculatrice a été allumée après une perte de puissance d'alimentation. La mémoire peut avoir été perturbée	006
Processing Command	Indique le traitement en cours du bloc de commande hôte	C11

Protocol Error	Bloc reçu dont la longueur est inférieure à celle d'un bloc nul Le paramètre de longueur maximale de bloc reçu d'une autre machine est illégal	C07
Receive Buffer Overrun	Kermit : plus de 255 octets ou essais de transmission ont été envoyés avant que la HP 48 ne reçoive un autre bloc SRECV : les données entrantes ont créé un trop-plein de la mémoire tampon	C04
Receive Error	Saturation de l'UART ou erreur de forme	C03
Receiving	Identifie le nom de l'objet durant sa réception	C0E
Retry #	Indique le nombre d'essais durant une tentative d'échange de bloc	C0B
Select a model	Sélectionner un modèle d'ajustement statistique	614
Select plot type	Choisir un type de tracé	60C
Select repeat interval	Choisir un intervalle de répétition d'alarme	61B
Sending	Identifie le nom d'un objet durant son émission	C0D

Sign Reversal	L'applicaion HP Solve ou ROOT n'a pas été en mesure de trouver un point pour lequel l'équation courante s'annule, mais a trouvé deux points voisins entre lesquels l'expression change de signe	A05
Single Equation	Une seule équation a été fournie au Multiple-Equation Solver. Utiliser l'application HP Solve	E402
Timeout	Impression vers un port série : XOFF reçu et dépassement de temps en attendant le signal XON Kermit : dépassement de temps en attendant l'arrivée d'un bloc	C02
Too Few Arguments	La commande nécessitait plus de paramètres que le nombre d'objets disponibles sur la pile	201
Too Many Unknowns	Le Multiple-Equation Solver ne peut pas calculer une valeur à partir de celles connues. Fournir une valeur supplémentaire ou une nouvelle équation	E404
Transfer Failed	Dix tentatives consécutives pour recevoir un bloc correct n'ont donné aucun résultat	C06
Unable to find root	PROOT n'est pas en mesure de déterminer toutes les racines	C001

Unable to Isolate	ISOL a échoué en raison d'un nom manquant ou parce qu'un paramètre contenait une fonction non inversible	130
Undefined Local Name	Exécution ou rappel d'un nom local pour lequel la variable locale correspondante n'existait pas	003
Undefined Name	Exécution ou rappel d'un nom global pour lequel la variable correspondante n'existait pas	204
Undefined Result	Calcul tel que 0/0 ayant entraîné un résultat mathématique indéfini	304
Undefined XLIB Name	Exécution d'un nom XLIB alors que la bibliothèque spécifiée est absente	004
Warning :	Etiquette présentant le message d'état courant	007
Wrong Argument Count	Evaluation d'une fonction définie par l'utilisateur avec un nombre incorrect de paramètres entre parenthèses	128
x and y-axis zoom	Identifie une option zoom	627
x axis zoom	Identifie une option zoom	625
x axis zoom w/AUTO	Identifie une option zoom	624
y axis zoom	Identifie une option zoom	626

ZERO	Le résultat fourni par l'application HP Solve ou ROOT est une racine (point pour lequel l'équation courante est nulle)	A04
""	Indique une exécution non effectuée lorsque [EXECS] est actionné	61E

B

Table d'unités

Dans la version anglaise de ce manuel figure une table de conversion de diverses unités, essentiellement anglo-saxonnes, dans le système international (SI).

Nous avons pris la liberté de supprimer cette annexe.

Pour ceux qui sont intéressés, les principales unités figurent dans tous les bons manuels de physique.

C

Indicateurs système

Cette annexe dresse la liste des indicateurs système du HP 48. Vous pouvez armer, désarmer et tester tous les indicateurs. L'état par défaut des indicateurs est *désarmé*, sauf dans le cas des indicateurs de taille de mot d'entier binaire. (indicateurs −5 à −10).

Indicateurs système

Indic.	Description
−1	Solution principale. *Désarmé* : QUAD et ISOL renvoient un résultat représentant toutes les solutions possibles. *Armé* : QUAD et ISOL ne renvoient que la solution principale.
−2	Constantes symboliques. *Désarmé* : les constantes symboliques (e, i, π, MAXR et MINR) conservent leur forme symbolique lorsqu'elles sont évaluées, sauf si l'indicateur Résultats numériques −3 est armé. *Armé* : les constantes symboliques donnent des nombres, quel que soit l'état de l'indicateur Résultats numériques −3.
−3	Résultats numériques. *Désarmé* : les fonctions comportant des arguments symboliques, y compris des constantes symboliques, donnent des résultats symboliques. *Armé* : les fonctions comportant des arguments symboliques, y compris des constantes symboliques, donnent des nombres.
−4	Non utilisé.
−5 thru −10	Taille de mot entier binaire. Les états combinés des indicateurs −5 à −10 définissent la taille de mot de 1 à 64 bits.

Indicateurs système (suite)

Indic.	Description
−11 and −12	Base entier binaire. HEX: −11 *armé*, −12 *armé*. DEC: −11 *désarmé*, −12 *désarmé*. OCT: −11 *armé*, −12 *désarmé*. BIN: −11 *désarmé*, −12 *armé*.
−13	Non utilisé.
−14	Mode paiement financier. *Désarmé :* les calculs TVM s'effectuent en fonction de paiements en fin de période. *Armé :* les calculs TVM s'effectuent en fonction de paiements en début de période.
−15 and −16	Rectangulaires : −16 *désarmé*. Polaires/Cylindriques : −15 *désarmé*, −16 *armé*. Polaires/Sphériques : −15 *armé*, −16 *armé*.
−17 and −18	Degrés : −17 *désarmé*, −18 *désarmé*. Radians : −17 *armé*. Grades : −17 *désarmé*, −18 *armé*.
−19	*Désarmé* :→V2 crée un vecteur bidimensionnel à partir de 2 nombres réels. *Armé* :→V2 crée un nombre complexe à partir de 2 nombre réels.
−20	Exception de dépassement de capacité inférieure. *Désarmé* : l'exception de dépassement de capacité inférieure renvoie 0, et arme l'indicateur −23 ou −24. *Armé* : l'exception de dépassement de capacité inférieure est traitée comme une erreur.
−21	Exception de dépassement de capacité *Désarmé* : l'exception de dépassement de capacité renvoie ±9.99999999999E499 et arme l'indicateur −25. *Armé* : l'exception est traitée comme une erreur.
−22	Exception de résultat infini. *Désarmé* : l'exception de résultat infini est traitée comme une erreur. *Armé* : l'exception de résultat infini renvoie ±9.99999999999E499 et arme l'indicateur −26.

Indicateurs système (suite)

Indic.	Description
−23	Dépassement de capacité inférieure négative.
−24	Dépassement de capacité inférieure positive.
−25	Dépassement de capacité.
−26	Résultat infini. Lorsqu'une exception se produit, l'indicateur correspondant (−23 à −26) n'est armé que si l'exception n'est *pas* traitée comme une erreur.
−27	Décompilation de nombres symboliques-complexes. *Désarmé* : décompile les nombres symboliques-complexes. Par exemple, `'x+y*i'` renvoie `'(x,y)'`. *Armé* : les nombres symboliques-complexes ne sont pas décompilés.
−28	Traçage simultané d'équations multiples. *Désarmé* : les équations multiples sont tracées successivement. *Armé* : les équations multiples sont tracées simultanément.
−29	Traçage d'axes. *Désarmé* : les axes sont dessinés pour des tracés 2D et statistiques. *Armé* : les axes ne sont pas dessinés pour des tracés 2D et statistiques.
−30	Non utilisé.
−31	Remplissage de courbe. *Désarmé* : remplissage de courbe activé. *Armé* : remplissage de courbe supprimé.
−32	Curseur graphique. *Désarmé* : curseur graphique toujours foncé. *Armé* : curseur graphique foncé sur fond clair et clair sur fond sombre.
−33	Unité d'E-S. *Désarmé* : E/S acheminée vers le port série. *Armé* : E/S acheminée vers le port IR (infrarouge).

Indicateurs système (suite)

Indic.	Description
−34	Unité d'impression. *Désarmé* : sortie d'imprimante acheminée vers le port IR. *Armé* : sortie d'imprimante dirigée vers le port série si l'indicateur −33 est désarmé.
−35	Format de données des E/S. *Désarmé* : objets transmis sous forme ASCII. *Armé* : objets transmis en binaire (image mémoire).
−36	Ecrasement de RECV. *Désarmé* : si le nom de fichier reçu par le HP 48 correspond à un nom de variable HP 48 existant, un nouveau nom de variable avec une extension numérique est créé pour empêcher l'écrasement. *Armé* : si le nom de fichier reçu par le HP 48 correspond à un nom de variable HP 48 existant, la variable existante est écrasée.
−37	Impression en double interligne. *Désarmé* : impression en interligne simple. *Armé* : impression en double interligne.
−38	Saut de ligne. *Désarmé* : saut de ligne ajouté à la fin de chaque ligne d'impression. *Armé* : pas de saut de ligne ajouté à la fin de chaque ligne d'impression.
−39	Messages d'E/S. *Désarmé* : messages d'E/S affichés. *Armé* : messages d'E/S supprimés.

Indicateurs système (suite)

Indic.	Description
−40	Affichage de l'horloge. *Désarmé* : horloge affichée uniquement lorsque le menu TIME est sélectionné. *Armé* : horloge affichée en permanence.
−41	Format d'horloge. *Désarmé* : 12 heures. *Armé* : 24 heures.
−42	Format de date. *Désarmé* : MM/JJ/AA (mois/jour/année). *Armé* : JJ.MM.AA (jour/mois/année).
−43	Alarmes répétitives non reprogrammées. *Désarmé* : les alarmes de rendez-vous répétitives, n'ayant pas reçu d'accusé de réception, sont automatiquement replanifiées. *Armé* : les alarmes de rendez-vous répétitives, n'ayant pas reçu d'accusé de réception, ne sont pas replanifiées.
−44	Alarmes ayant reçu un accusé de réception sauvegardées. *Désarmé* : les alarmes de rendez-vous, ayant reçu un accusé de réception, sont sauvegardées dans la liste des alarmes. *Armé* : les alarmes de rendez-vous, ayant reçu un accusé de réception, sont sauvegardées dans la liste des alarmes.
−45 thru −48	Nombre de décimales. Les états combinés des indicateurs −45 à −48 définissent le nombre de décimales dans les modes fixe, scientifique et ingénieur.
−49 and −50	Format d'affichage numérique. Standard : −49 *désarmé*, −50 *désarmé*. Fixe : −49 *armé*, −50 *désarmé*. Scientifique : −49 *désarmé*, −50 *armé*. Ingénieur : −49 *armé*, −50 *armé*.
−51	Séparateur décimal. *Désarmé* : le séparateur décimal est un . (point). *Armé* : le séparateur décimal est une , (virgule).

Indicateurs système (suite)

Indic.	Description
−52	Affichage sur une seule ligne. *Désarmé* : l'affichage donne priorité à l'objet du niveau 1, en utilisant jusqu'à quatre lignes de l'affichage de la pile. *Armé* : l'affichage de l'objet du niveau 1 est limité à une ligne.
−53	Priorité. *Désarmé* : suppression de certaines parenthèses dans des expressions algébriques pour une meilleure lisibilité. *Armé* : toutes les parenthèses des expressions algébriques sont affichées.
−54	Eléments "très petits". *Désarmé* : les valeurs calculées par RANK (et d'autres commandes fournissant le rang d'une matrice), qui sont inférieures à 1×10^{-14} fois la taille de l'élément le plus grand de leur colonne, sont remplacées par des zéros. L'arrondi automatique est activé pour DET. *Armé* : les valeurs "très petites" calculées ne sont pas remplacées par des zéros. L'arrondi est désactivé pour DET.
−55	Derniers arguments. *Désarmé* : arguments d'opération sauvegardés. *Armé* : arguments d'opération non sauvegardés.
−56	Signal sonore d'erreur. *Désarmé* : signal sonore d'erreur et de commande BEEP activés. *Armé* : signal sonore d'erreur et de commande BEEP supprimés.
−57	Signal sonore d'alarme. *Désarmé* : signal sonore d'alarme activé. *Armé* : signal sonore d'alarme supprimé.
−58	Messages complets (en clair). *Désarmé* : affichage automatique des messages système et des données. *Armé* : suppression de l'affichage automatique des messages système et des données.

Indicateurs système (suite)

Indic.	Description
−59	Affichage rapide du gestionnaire de variables. *Désarmé* : le gestionnaire de variables affiche les noms de variables et leur contenu. *Armé* : le gestionnaire de variables affiche uniquement les noms de variables.
−60	Verrouillage alpha. *Désarmé* : frappe alpha activée pour un caractère en appuyant une fois sur [α]. Verrouillage alpha activé en appuyant deux fois sur [α]. *Armé* : verrouillage alpha activé en appuyant une fois sur [α]. (Frappe alpha pour un caractère non disponible.)
−61	Verrouillage mode Utilisateur. *Désarmé* : mode USR1 activé en appuyant une fois sur [↰][USER]. Mode utilisateur (USER) activé en appuyant deux fois sur [↰][USER]. *Armé* : mode Utilisateur activé en appuyant une fois sur [↰][USER]. (Mode USR1 non disponible).
−62	Mode utilisateur (USER). *Désarmé* : mode Utilisateur non actif. *Armé* : mode Utilisateur actif.
−63	Rôle de [ENTER]. *Désarmé* : [ENTER] évalue la ligne de commande. *Armé* : [ENTER] activée en fonction-utilisateur.
−64	Indicateur de bouclage d'index. *Désarmé* : la dernière exécution de GETI ou PUTI n'a pas incrémenté l'index sur le premier élément. *Armé* : la dernière exécution de GETI ou PUTI a incrémenté l'index sur le premier élément.

D

Variables réservées

Le HP 48 emploie les *variables réservées* suivantes. Elles ont des objectifs particuliers, et leurs noms sont employés en tant qu'arguments implicites pour certaines commandes. Evitez de faire usage du nom de ces variables à d'autres fins, car cela risque d'interférer avec l'exécution des commandes employant ces variables.

Vous pouvez modifier certaines des valeurs de ces variables avec des commandes programmables, alors que pour d'autres, vous devrez stocker les nouvelles valeurs dans l'emplacement adéquat.

Variable réservée	Contenu	Utilisé par
ALRMDAT	Paramètres d'alarme.	Opérations TIME ALRM
CST	Liste définissant le menu CST (personnalisé).	MENU, CST
noms "*der*"	Dérivée-utilisateur.	∂
EQ	Equation en cours.	ROOT, DRAW
EXPR	Expression en cours.	SYMBOLIC
IOPAR	Paramètres d'E-S.	Commandes d'E-S
MHpar	Etat du jeu du Démineur.	MINEHUNT
Mpar	Equations du solver d'équations multiples.	EQ LIB
n1, n2, ...	Entiers arbitraires.	ISOL, QUAD
Nmines	Données du jeu du Démineur.	MINEHUNT

Variable réservée	Contenu	Utilisé par
PPAR	Paramètres de traçage.	DRAW
PRTPAR	Paramètres d'impression.	Commandes PRINT
s1, s2, ...	Signes arbitraires.	ISOL, QUAD
VPAR	Paramètres de visualisation.	DRAW
ZPAR	Facteurs de zoom sur le tracé.	DRAW
ΣDAT	Données statistiques.	Application statistique, DRAW
ΣPAR	Paramètres statistiques.	Application statistique, DRAW

Contenus des variables réservées

La plupart des variables réservées (à l'exception de *ALRMDAT*, *IOPAR* et *PRTPAR*) peuvent être stockées avec des contenus différents dans des répertoires différents. Cela vous permet, par exemple, de sauvegarder plusieurs ensembles de données statistiques dans des répertoires différents.

ALRMDAT

ALRMDAT ne réside pas dans un répertoire particulier. Vous ne pouvez pas accéder à la variable elle-même, mais à ses données à partir d'un répertoire quelconque en employant les commandes RCLALARM et STOALARM, ou par l'intermédiaire du catalogue des alarmes.

ALRMDAT contient une liste des paramètres d'alarme suivants :

Paramètre (Commande)	Description	Valeur par défaut
date (→DATE)	Nombre réel spécifiant la date de l'alarme. *MM.JJAAA* (ou *JJ.MMAAA* si l'indicateur −42 est armé). Si *AAA* n'est pas inclus, l'année en cours est utilisée.	Date en cours.
heure (→TIME)	Nombre réel spécifiant l'heure de l'alarme: *HH.MMSS*.	00.0000
action	Chaîne ou objet : ■ une chaîne crée une *alarme de rendez-vous*, qui émet un signal sonore et affiche la chaîne. ■ tout autre objet crée une *alarme de contrôle*, qui exécute cet objet.	Chaîne vide (alarme de rendez-vous).
répéter	Nombre réel spécifiant l'intervalle séparant deux occurrences automatiques de l'alarme, exprimé en tops (un top d'horloge correspond à $^1/_{8192}$ seconde).	0

Les paramètres sans commandes peuvent être modifiés avec un programme en stockant de nouvelles valeurs dans la liste contenue dans *ALRMDAT* (employez la commande PUT).

CST

CST contient une liste (ou un nom spécifiant une liste) des objets qui définissent le menu CST (*personnalisé*). Les objets du menu personnalisé se comportent comme les objets des menus intégrés. Par exemple :

- Les noms se comportent comme les touches du menu VAR. Ainsi, si *ABC* est le nom d'une variable, ABC provoque l'évaluation de *ABC*, → ABC rappelle son contenu et ← ABC stocke le nouveau contenu dans *ABC*.
- Le libellé de menu correspondant à un répertoire porte une barre sur le côté gauche. L'utilisation de cette touche de menu ouvre ce répertoire.

- Les objets-unités opèrent comme les unités du catalogue des UNITS (possibilités de conversion avec touche shiftée-gauche, par exemple).
- Les touches chaîne copient la chaîne.
- Vous pouvez inclure des objets-sauvegarde dans la liste définissant un menu personnalisé en y ajoutant le nom de cet objet identifié par son emplacement de port (de 0 à 33).

Vous pouvez spécifier des libellés de menu et des actions de touche de manière indépendante en remplaçant un objet unique dans la liste de menu personnalisée par une liste se présentant sous la forme *{ "libellé-objet" action-objet }*. (Pour de plus amples informations, reportez-vous à "Personnalisation des menus" et "Amélioration des menus personnalisés", au chapitre 30 du *Manuel d'utilisation du HP 48.*)

Pour associer des actions shiftées à des touches de menu personnalisé, *action-objet* peut être une liste contenant trois objets-action dans l'ordre suivant :

- L'action non shiftée (nécessaire si vous voulez spécifier des actions shiftées).
- L'action shiftée-gauche.
- L'action shiftée-droite.

Reportez-vous à "Amélioration de menus personnalisés", au chapitre 30 du *Manuel d'utililsation HP 48.*

Noms "der"

Si ∂ est appliqué à une fonction qui n'a pas de dérivée intégrée, & renvoie une nouvelle fonction dont le nom est "der", suivi du nom de la fonction d'origine. Ces noms de fonction "der" sont des noms de variables réservées.

Pour de plus ammples informations, reportez-vous à "Création de dérivées-utilisateur", au chapitre 20 du *Manuel d'utilisation du HP 48.*

EQ

EQ contient l'équation en cours ou le nom de la variable contenant l'équation en cours.

EQ fournit l'équation pour ROOT, ainsi que pour la commande de traçage DRAW lorsque le type de tracé est FUNCTION, CONIC, POLAR, PARAMETER, TRUTH ou DIFFEQ. (*ΣDAT* fournit les informations lorsque le type de tracé est HISTOGRAM, BAR ou SCATTER.) L'objet figurant dans *EQ* peut être un objet algébrique, un nombre, un nom ou un programme. La façon dont DRAW interprète *EQ* dépend du type de tracé.

Pour une utilisation graphique, *EQ* peut également être une liste d'équations ou d'autres objets. Si *EQ* contient une liste, DRAW traite chaque objet tour à tour en tant qu'équation en cours et les trace successivement. Toutefois, ROOT dans l'application HP Solve ***ne peut pas*** résoudre *EQ* contenant une liste.

Pour modifier le contenu d'*EQ*, utilisez la commande STEQ.

EXPR

EXPR contient l'expression algébrique en cours (ou le nom de la variable contenant l'expression en cours) utilisée par l'application SYMBOLIC et ses commandes associées. L'objet figurant dans *EQ* doit être une expression algébrique ou un nom.

IOPAR

IOPAR est une variable du répertoire *HOME* qui contient une liste des paramètres d'E-S nécessaires aux communications avec un ordinateur. Elle est créée la première fois que vous transférez des données ou que vous ouvrez le port série (OPENIO), et elle est automatiquement mise à jour chaque fois que vous modifiez les paramètres d'E-S. Tous les paramètres de *IOPAR* sont des entiers.

Paramètre (Commande)	Description	Valeur par défaut
baud (BAUD)	Débit en bauds : T1200, 2400, 4800 ou 9600.	9600
parité (PARITY)	Parité utilisée : 0=aucune, 1=impaire, 2=paire, 3=marque, 4=espace. La valeur peut être positive ou négative : une parité positive est utilisée à la fois lors de l'émission et de la réception, une parité négative n'est utilisée que lors de l'émission.	0
régulation de la réception	Une valeur réelle différente de zéro active la régulation, alors qu'un zéro la désactive. La régulation de la réception émet un signal XOFF lorsque le tampon de réception est presque plein, et émet un signal XON lorsqu'il peut à nouveau accepter des données. La régulation n'est pas utilisée pour les E-S Kermit, mais intervient dans les transferts d'E-S en série.	0 (pas de régul)
régulation de l'émission	Une valeur réelle différente de zéro active la régulation, alors qu'un zéro la désactive. La régulation arrête l'émission à réception de XOFF, et la reprend à réception de XON. La régulation n'est pas utilisée pour les E-S Kermit, mais intervient dans les transferts d'E-S en série.	0 (pas de régul)

Paramètre (Commande)	Description	Valeur par défaut
somme de contrôle (CKSM)	Schéma de détection d'erreurs est demandé lors de l'exécution de SEND: ■ 1=total de contrôle arithmétique à 1 chiffre ■ 2=total de contrôle arithmétique à 2 chiffres ■ 3=contrôle de redondance cyclique à 3 chiffres.	3
code de traduction (TRANSIO)	Détermination des caractères à traduire : ■ 0=aucun ■ 1=traduit le caractère 10 (saut de ligne uniquement) en caractères 10 et 13 et inversement (saut de ligne et retour de chariot) ■ 2=traduit les caractères des codes 128 à 159 (80–9F hex) ■ 3=traduit les caractères des codes 128 à 255.	1

Les paramètres sans commandes peuvent être modifiés avec un programme en stockant de nouvelles valeurs dans la liste contenue dans *IOPAR* (utilisez la commande PUT), ou en modifiant directement *IOPAR*.

MHpar

MHpar stocke l'état d'une partie de Démineur interrompue. *MHpar* est créée lorsque vous quittez le jeu en appuyant sur STO. Si *MHpar* existe toujours lorsque vous relancez le Démineur, la partie interrompue reprend et *MHpar* est supprimée.

Mpar

Mpar est créée lorsque vous utilisez le solver d'équations multiples de la bibliothèque d'équations.*Mpar* stocke l'ensemble d'équations que vous utilisez.

Lorsque la bibliothèque d'équations lance le solver d'équations multiples, elle stocke la liste de l'ensemble d'équations dans *EQ*, et l'ensemble d'équations, une liste de variables et des informations complémentaires dans *Mpar*. *Mpar* est ensuite employé pour configurer le menu Solver pour l'ensemble d'équations en cours.

Mpar est structuré en tant que données de bibliothèque réservées à l'application solver d'équations multiples. Ainsi, vous ne pouvez visualiser et éditer *Mpar* directement, mais vous pouvez l'éditer indirectement en faisant appel à des commandes qui le modifient.

Vous pouvez en outre recourir à la commande MINIT (◀ EQ LIB MES MINIT) pour créer *Mpar* à partir d'un ensemble d'équations figurant dans la pile. Reportez-vous à "Définition d'un ensemble d'équations", au chapitre 25 du *Manuel d'utilisation du HP 48*

n1, n2, ...

Les commandes ISOL et QUAD renvoient des solutions *générales* (et non des solutions *principales*) pour des opérations. Une solution générale contient des variables correspondant à des entiers ou à des signes arbitraires, ou aux deux.

La variable *n1* représente un entier arbitraire 0, ±1, ±2, etc. Les entierse arbitraires supplémentaires sont représentés par *n2*, *n3*, etc.

Si l'indicateur −1 est armé, ISOL et QUAD renvoient des solutions principales, et dans ce cas, l'entier arbitraire est toujours égal à zéro.

Nmines

Nmines est une variable que vous pouvez créer dans le répertoire en cours pour contrôler le nombre de mines utilisées dans le jeu du Démineur. *Nmines* contient un entier compris entre 1 et 64, si *Nmines* est négatif, les mines sont visibles au cours de la partie.

PPAR

PPAR est une variable figurant dans le répertoire en cours. Elle contient une liste des paramètres de traçage utilisés par la commande DRAW pour tous les tracés mathématiques et statistiques, par AUTO pour la mise à l'échelle automatique et par les opérations graphiques (non programmables) interactives.

Paramètre (Commande)	Description	Valeur par défaut
(x_{min}, y_{min}) (XRNG, YRNG)	Nombre complexe spécifiant l'angle inférieur gauche de *PICT* (l'angle inférieur gauche de la plage d'affichage).	(−6.50, −3.1)
(x_{max}, y_{max}) (XRNG, YRNG)	Nombre complexe spécifiant l'angle supérieur droit de *PICT* (l'angle supérieur droit de la plage d'affichage).	(6.5, 3.2)
indep (INDEP)	Un nom spécifiant la variable indépendante, ou une liste contenant ce nom et deux nombres qui spécifient les valeurs minimale et maximale de la variable indépendante (le domaine de traçage).	*X*

Paramètre (Commande)	Description	Valeur par défaut
res (RES)	Résolution. Nombre réel spécifiant l'intervalle séparant les valeurs de la variable indépendante. Pour le tracé d'équations, il détermine l'intervalle de traçage sur l'axe x. Un nombre binaire spécifie la résolution en *pixels* (nombre de colonnes de pixels séparant les points). Un entier spécifie la résolution en unités-*utilisateur* nombre d'unités-utilisateur séparant les points). La résolution associée aux tracés statistiques est différente (voir plus loin).	0
axes (AXES)	Nombre complexe spécifiant les coordonnées en unités-utilisateur de l'origine du tracé, ou une liste contenant ce qui suit : ■ le nombre complexe spécifiant l'origine, ■ un nombre réel, un entier binaire ou une liste contenant deux nombres réels ou deux entiers binaires spécifiant la graduation (voir ATICK), ■ deux chaînes spécifiant des libellés pour les axes horizontal et vertical.	(0, 0)
ptype (BAR, etc.)	Nom de commande spécifiant le type de tracé (BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, ou YSLICE).	FUNCTION

Paramètre (Commande)	Description	Valeur par défaut
depend (DEPND)	Nom spécifiant la variable dépendante, ou une liste contenant le nom et deux nombres qui spécifient le domaine de traçage vertical. Pour DIFFEQ, le deuxième élément de la liste peut également être un vecteur réel qui représente la valeur initiale.	*Y*

Les paramètres sans commandes peuvent être modifiés avec un programme en stockant de nouvelles valeurs dans la liste contenue dans *PPAR* (utilisez la commande PUT).

L'opération RESET (◀ PLOT PPAR RESET) réinitialise les paramètres *PPAR* (sauf *ptype*) à leurs valeurs par défaut, et efface *PICT*.

Notez que *res* se comporte différemment pour les types de tracé statistiques BAR, HISTOGRAM et SCATTER. Pour BAR, *res* spécifie la largeur de la barre, pour HISTOGRAM, *res* spécifie la largeur du bloc; *res* n'affecte pas SCATTER.

PRTPAR

PRTPAR est une variable du répertoire *HOME* qui contient une liste de paramètres d'impression. Elle est créée automatiquement la première fois que vous utilisez une commande d'impression.

Paramètre (Commande)	Description	Valeur par défaut
délai (DELAY)	Nombre réel, compris entre 0 et 6.9, spécifiant le nombre de secondes pendant lequel le HP 48 attend entre chaque émission de lignes. Ce délai doit couvrir au moins le temps d'impression de la ligne la plus longue. Si le délai est trop bref pour l'imprimante, des données sont perdues. Le paramétrage du délai affecte en outre l'impression en série si la régulation de l'émission (dans *IOPAR*) n'est pas utilisée.	1.8
mappage (OLDPRT stocke la chaîne de mappage des caractères pour l'imprimante infrarouge HP 82240A).	Chaîne définissant le mappage du jeu de caractères étendu pour l'impression. Cette chaîne peut contenir autant de caractères que vous le souhaitez, le premier étant le nouveau caractère 128, le deuxième le nouveau caractère 129, etc. (tout numéro de caractère dépassant la longueur de la chaîne n'est pas mappé).(Voir exemple plus loin.)	Chaîne vide.
longueur de la ligne	Nombre réel spécifiant le nombre de caractères de la ligne pour l'impression série. N'affecte *pas* l'impression infrarouge.	80
terminaison de la ligne	Une chaîne spécifiant la méthode de terminaison de la ligne pour l'impression en série. (N'affecte *pas* l'impression infrarouge.)	Caractères de contrôle 13 (retour chariot) et 10 (saut de ligne).

Les paramètres sans commandes peuvent être modifiés avec un programme en stockant de nouvelles valeurs dans la liste contenue dans *PRTPAR* (utilisez la commande PUT).

La modification d'un paramètre est immédiatement effective, *sauf* lors de l'impression de l'affichage réalisée avec la combinaison de touches [ON][I/O] (car elle ne fait pas appel à *PRTPAR*). Cette méthode d'impression n'est affectée que par le paramètre de délai. Une modification de ce dernier n'affecte [ON][I/O] qu'après l'exécution de la dernière commande d'impression. Pour utiliser immédiatement un nouveau délai avec [ON][I/O], servez-vous de la commande DELAY.

Exemple : Si la chaîne de mappage est "ABCDEFGH" et que le caractère à imprimer possède la valeur 131, le caractère effectivement imprimé sera "D", car 131−128=3 et "A" a la valeur zéro. Un code de caractère supérieur ou égal à 136 n'est pas mappé étant donné que 136−128=8, valeur qui dépasse la longueur de la chaîne.

s1, s2, ...

Les commandes ISOL et QUAD renvoient des solutions *générales* (et non des solutions *principales*) pour des opérations. Une solution générale contient des variables correspondant à des entiers ou à des signes arbitraires, ou aux deux.

La variable *s1* représente un signe + ou − arbitraire. Les signes arbitraires supplémentaires sont représentés par *s2*, *s3*, etc.

Si l'indicateur −1 est armé, ISOL et QUAD renvoient des solutions principales, et dans ce cas, le signe arbitraire est toujours +1.

VPAR

VPAR est une variable du répertoire en cours. Elle contient une liste de paramètres utilisés par les types de tracés 3 D. La principale structure de données stockée dans VPAR décrit la "vue-volume", la région tridimensionnelle abstraite dans laquelle la fonction est tracée.

Paramètre (Commande)	Description	Valeur par défaut
(x_{left}, x_{right}) (XVOL)	Nombres réels spécifiant la largeur de la vue volumique.	(−1, 1)
(y_{far}, y_{near}) (YVOL)	Nombres réels spécifiant la profondeur de la vue volumique.	(−1, 1)
(z_{low}, z_{high}) (ZVOL)	Nombres réels spécifiant la hauteur de la vue volumique.	(−1, 1)
(x_{eye}, y_{eye}, z_{eye}) (EYEPT)	Nombres réels spécifiant le point dans l'espace à partir duquel le tracé est visualisé.	(0, −3, 0)
(x_{step}, y_{step}) (NUMX,NUMY)	Nombres réels spécifiant les incréments séparant les coordonnées x des coordonnées y tracées. Les incréments sont égaux à la plage des axes divisée par le nombre de pas. Se substitue (ou s'associe à) *res*.	(10, 8)
(xx_{left}, xx_{right}) (XXRNG)	Nombres réels qui spécifiant la largeur de la zone (domaine) d'entrée. Utilisé par GRIDMAP et PARSURFACE.	(−1, 1)
(yy_{far}, yy_{near}) (YYRNG)	Des nombres réels qui spécifient la profondeur de la zone (domaine) d'entrée. Utilisé par GRIDMAP et PARSURFACE.	(−1, 1)

Les paramètres sans commandes peuvent être modifiés avec un programme en stockant de nouvelles valeurs dans *VPAR* (utilisez la commande PUT).

L'opération RESET (◀ PLOT NXT 3D VPAR NXT RESET) réinitialise les paramètres *VPAR* à leurs valeurs par défaut.

ZPAR

ZPAR est une variable du répertoire en cours. Elle contient une liste de paramètres de zoom utilisés par la commande DRAW pour tous les tracés mathématiques et statistiques 2D.

Paramètre (Commande)	Description	Valeur par défaut
facteur h	Nombre réel spécifiant le facteur de zoom horizontal.	4
fateur v	Nombre réel spécifiant le facteur de zoom vertical.	4
indicateur de recentrage	0 ou 1 selon que l'option de recentrage au niveau du réticule a été sélectionnée ou non dans le masque de saisie des facteurs de zoom.	0
{ *liste* }	Liste vide, ou copie de la dernière variable *PPAR*.	

Utilisez le masque de saisie de définition des facteurs de zoom (ZFACT) pour modifier *ZPAR*.

ΣDAT

ΣDAT est une variable du répertoire en cours qui contient la matrice statistique en cours ou le nom de la variable contenant cette matrice. Cette dernière contient les données utilisées par les applications statistiques.

Matrice statistique correspondant aux variables 1 à m

var_1	var_2	...	var_m
x_{11}	x_{21}	...	x_{m1}
x_{12}	x_{22}	...	x_{m2}
⋮	⋮	⋮	⋮
x_{1n}	x_{2n}	...	x_{mn}

Vous pouvez créer une nouvelle matrice statistique en cours en entrant de nouvelles donnés, en modifiant les données ou en sélectionnant une autre matrice.

La commande CLΣ efface la matrice statistique en cours.

ΣPAR

ΣPAR est une variable du répertoire en cours qui contient la liste des paramètres statistiques en cours ou le nom de la variable contenant cette liste.

Paramètre (Commande)	Description	Valeur par défaut
$colonne_{indep}$ (XCOL)	Nombre réel spécifiant le numéro de la colonne des variables indépendantes.	1
$colonne_{dep}$ (YCOL)	Nombre réel spécifiant le numéro de la colonne des variables dépendantes.	2
intersection (LR)	Nombre réel spécifiant le coefficient d'intersection tel que déterminé par la régression en cours.	0
pente (LR)	Nombre réel spécifiant le coefficient de pente tel que déterminé par la régression en cours.	0
modèle (LINFIT, etc.)	Commande spécifiant le modèle de régression (LINFIT, EXPFIT, PWRFIT ou LOGFIT).	LINFIT

E

Nouvelles commandes

Vous trouverez dans les tableaux suivants les nouvelles commandes qui n'étaient pas disponibles sur le calculateur standard de la série HP 48S. Elles sont classées par ordre alphabétique et accompagnées d'une brève description. Par ailleurs, elles sont décrites dans le chapitre 3.

Nouvelles commandes

Commande	Description
ADD	Ajoute des éléments de liste.
AMORT	Calcule l'amortissement d'un prêt ou d'un investissement en fonction des paramètres d'amortissement en cours.
ANIMATE	Affiche des objets graphiques en séquence.
ATICK	Définit l'intervalle de graduation des axes dans la variable réservée *PPAR*.
CHOOSE	Crée une liste déroulante personnalisée.
CLTEACH	Supprime le sous-répertoire EXAMPLES et son contenu du répertoire HOME.
COL+	Insère un tableau (vecteur ou matrice) dans une matrice.
COL−	Supprime une colonne d'une matrice.
COL→	Transforme une série de vecteurs-colonnes et un nombre de colonnes en une matrice, ou transforme une séquence de nombres et un nombre d'éléments en un vecteur.

Nouvelles commandes (suite)

Commande	Description
→COL	Transforme une matrice en une série de vecteurs-colonnes, ou décompose un vecteur en ses différents éléments.
COND	Renvoie le numéro de condition de la norme 1 (norme de colonne) d'une matrice carrée.
CONLIB	Ouvre le catalogue de la bibliothèque des constantes.
CONST	Renvoie la valeur d'une constante.
CSWP	Permute les colonnes d'une matrice.
CYLIN	Active le mode de coordonnées cylindriques.
DARCY	Calcule le facteur de friction de Darcy de certains écoulements de fluide.
DIAG→	Prend un tableau et une dimension spécifiée et renvoie une matrice dont les éléments diagonaux principaux sont les éléments du tableau.
→DIAG	Renvoie un vecteur qui contient les éléments diagonaux principaux d'une matrice.
DIFFEQ	Spécifie le type de tracé des équations différentielles.
DOLIST	Applique des commandes, des programmes ou des fonctions-utilisateur à des listes.
DOSUBS	Applique un programme ou une commande à des groupes d'éléments dans une liste.
EGV	Calcule les valeurs propres et les vecteurs propres de droite pour une matrice carrée.
EGVL	Calcule les valeurs propres d'une matrice carrée.
ENDSUB	Permet d'accéder au nombre total de sous-listes utilisées pendant l'exécution d'un programme ou d'une commande en employant DOSUBS.

Nouvelles commandes (suite)

Commande	Description
EQNLIB	Lance l'application Equation Library (bibiothèque d'équations).
EYEPT	Spécifie les coordonnées du point de vue dans un tracé en perspective.
F0λ	Renvoie la fraction du pouvoir émissif total d'un corps noir.
FANNING	Calcule le facteur de friction de Fanning de certains écoulements de fluide.
FFT	Calcule la transformée de Fourier discrète 1D ou 2D d'un tableau.
FREE1	Libère la RAM préalablement fusionnée dans le port 1.
GRIDMAP	Sélectionne le type de tracé Gridmap.
HEAD	Renvoie le premier élément d'une liste ou d'une chaîne.
IFFT	Calcule l'inverse 1D ou 2D de la transformée discrète d'un vecteur ou d'une matrice.
INFORM	Affiche une boîte de dialogue utilisateur (masque de saisie).
LIBEVAL	Evalue des objets-bibliothèque non nommés par leurs adresses en mémoire.
LININ	Teste si une expression algébrique est structurellement linéaire pour une variable donnée.
ΣLIST	Renvoie la somme des éléments d'une liste.
ΠLIST	Renvoie le produit des éléments d'une liste.
ΔLIST	Renvoie l'ensemble des différences du premier ordre.
LQ	Renvoie la factorisation LQ d'une matrice $n \times m$.

Nouvelles commandes (suite)

Commande	Description
LSQ	Renvoie la norme minimale selon la méthode des moindres carrés d'un système d'équations linéaires.
LU	Renvoie la décomposition LU de Crout d'une matrice carrée.
MCALC	Désigne une variable sous forme de valeur calculée (non définie par l'utilisateur).
MERGE1	Fusionne la mémoire de la carte RAM enfichable du port 1 avec la mémoire utilisateur principale.
MINEHUNT	Lance le jeu MINEHUNT (Démineur).
MINIT	Crée la variable réservée *Mpar*.
MITM	Modifie les titres et l'ordre du menu d'équations multiples.
MROOT	Résout pour une ou plusieurs variables.
MSGBOX	Crée une boîte de messages utilisateur.
MSOLVR	Extrait le menu de variables du solver d'équations multiples pour l'ensemble d'équations défini par *Mpar*.
MUSER	Désigne une variable comme étant définie par l'utilisateur.
NDIST	Renvoie la distribution de probabilité normale.
NOVAL	Valeur de garde pour les valeurs initiales et de réinitialisation dans les boîtes de dialogue utilisateur.
NSUB	Permet d'accéder au numéro de la sous-liste en cours pendant une itération d'un programme ou d'une commande appliquée en employant DOSUBS.
NUMX	Définit le nombre de pas x pour chaque pas y dans les tracés en perspective 3D.
NUMY	Définit le nombre de pas y dans la vue volumique de tracé en perspective 3D.

Nouvelles commandes (suite)

Commande	Description
PARSURFACE	Sélectionne le type de tracé Parsurface.
PCOEF	Renvoie les coefficients d'un polynôme unitaire.
PCOV	Calcule la covariance d'une population.
PCONTOUR	Spécifie le type de tracé pseudo-contour.
PEVAL	Evalue un polynôme de degré n en x.
PINIT	Initialise les ports de carte enfichable.
PROOT	Renvoie toutes les racines d'un polynôme de degré n possédant des coefficients réels ou complexes.
PSDEV	Calcule l'écart type d'une population.
PVAR	Calcule la variance d'une population.
QR	Renvoie la factorisation QR d'une matrice $n \times m$.
RANK	Renvoie le rang d'une matrice rectangulaire.
RANM	Renvoie une matrice d'entiers aléatoires.
RCI	Multiplie une ligne d'une matrice par une constante.
RCIJ	Multiplie une ligne d'une matrice par une constante, puis ajoute le produit à une autre ligne de la matrice.
RECT	Active le mode de coordonnées rectangulaires.
REVLIST	Inverse l'ordre des éléments d'une liste.
RKF	Calcule la solution d'un problème à valeur initiale pour une équation différentielle, en utilisant la méthode Runge-Kutta-Fehlberg.
RKFERR	Renvoie l'estimation de l'erreur absolue pour un pas donné lors de la résolution du problème à valeur initiale pour une équation différentielle (en utilisant la méthode RKF).

Nouvelles commandes (suite)

Commande	Description
RKFSTEP	Calcule le pas suivant de la solution d'un problème à valeur initiale pour une équation différentielle.
ROW+	Insère un tableau dans une matrice.
ROW−	Supprime une ligne d'une matrice.
RREF	Calcule la forme réduite échelonnée d'une matrice rectangulaire.
RRK	Calcule la solution d'un problème à valeur initiale pour une équation différentielle avec des dérivées partielles connues.
RRKSTEP	Calcule le pas suivant d'un problème à valeur initiale pour une équation différentielle, et affiche la méthode utilisée pour parvenir à ce résultat.
RSBERR	Renvoie une estimation d'erreur pour un pas donné lors de la résolution d'un problème à valeur initiale pour une équation différentielle (en employant la méthode Rosenbrock).
RSWP	Permute les lignes d'une matrice.
SCHUR	Renvoie la décomposition de Schur d'une matrice carrée.
SEQ	Renvoie une liste de résultats générés par l'exécution répétée d'un objet sur une plage d'éléments spécifiée.
SIDENS	Calcule la densité intrinsèque du silicium en fonction de la température.
SLOPEFIELD	Sélectionne le type de tracé Slopefield.
SNRM	Calcule la norme spectrale d'un tableau.
SOLVEQN	Lance le solver pour un ensemble d'équations spécifié.
SORT	Trie les éléments d'une liste par ordre croissant.

Nouvelles commandes (suite)

Commande	Description
SPHERE	Active le mode de coordonnées sphériques.
SRAD	Renvoie le rayon spectral d'une matrice carrée.
STREAM	Applique un objet à chaque élément d'une liste.
SVD	Renvoie la décomposition en valeurs singulières d'une matrice $n \times m$.
SVL	Renvoie les valeurs singulières d'une matrice $m \times n$.
TAIL	Renvoie tous les éléments d'une liste ou d'une chaîne, sauf le premier.
TDELTA	Calcule une modification de température.
TEACH	Crée un sous-répertoire EXAMPLES dans le répertoire HOME et y charge des exemples de programmation, de tracé de graphiques et de résolutions d'équations intégrés au
TINC	Calcule un incrément de température.

Nouvelles commandes (suite)

Commande	Description
TRACE	Renvoie la trace d'une matrice carrée.
TVM	Lance le solver TVM.
TVMBEG	Spécifie le mode de paiement en début de période de décomposition.
TVMEND	Spécifie le mode de paiement en fin de période de décomposition.
TVMROOT	Calcule la solution pour la variable TVM spécifiée en utilisant les valeurs stockées dans les variables TVM restantes.
VERSION	Renvoie la version du logiciel et le message de copyright.
WIREFRAME	Sélectionne le type de tracé Wireframe.
XRECV	Reçoit un objet par XModem.
XSEND	Envoie un objet par XModem.

Nouvelles commandes (suite)

Commande	Description
XVOL	Définit la largeur de la vue volumique dans la variable réservée *VPAR*.
XXRNG	Spécifie la plage x de la zone d'entrée (domaine) pour des tracés GRIDMAP et PARSURFACE.
YSLICE	Spécifie le type de tracé Y-Slice (sections).
YVOL	Définit la profondeur de la vue volumique dans la variable réservée *VPAR*.
YYRNG	Spécifie la plage y d'une zone d'entrée (domaine) pour des tracés GRIDMAP et PARSURFACE.
ZFACTOR	Calcule le facteur de correction de la compressibilité du gaz pour le comportement non idéal d'un hydrocarbure.
ZVOL	Définit la hauteur de la vue volumique dans la variable réservée *VPAR*.

F

Références techniques

Cette annexe contient les informations suivantes :

- La taille des objets ;
- Les régles de simplification mathématique employées par le HP 48 ;
- Les configurations de différenciation symbolique employés par le HP 48 ;
- Les règles de développement d'EquationWriter.
- Les ouvrages de référence employées pour les constantes et les équations utilisées dans le HP 48 (différentes de celles de la bibliothèque des équations).

Taille des objets

Le tableau suivant dresse la liste des types d'objet et des tailles correspondantes en octets (notez que les caractères des noms, des chaînes et des libellés utilisent 1 octet chacun).

Taille des objets

Objet	Taille (octets)
Expression algébrique	5 + taille des objets inclus
Objet-sauvegarde	12 + nombre de caractères du nom + taille de l'objet inclus
Entier binaire	13
Commande	2,5
Matrice complexe	15 + 16 × nombre d'éléments
Nombre complexe	18,5
Vecteur complexe	12,5 + 16 × nombre d'éléments
Répertoire	6,5 + taille des variables incluses
Objet graphique	10 + nombre de lignes × CEIL(colonnes/8)
Liste	5 + taille des objets inclus
Matrice	15 + 8 × nombre d'éléments
Programme	12,5 + taille des objets inclus
Nom global ou local entre apostrophes	8.5 + nombre de caractères
Nombre réel	10,5
Chaîne	5 + nombre de caractères
Objet identifié	3,5 + nombre de caractères de l'identificateur + taille de l'objet non identifié
Objet-unité	7,5 +
grandeur réelle	2,5 ou 10,5
chaque préfixe	6
chaque nom d'unité	5 + nombre de caractères
chaque ×, ^, ou /	2,5
chaque exposant	2,5 ou 10,5
nom global ou local sans apostrophes	3,5 + nombre de caractères
Vecteur	12,5 + 8 × nombre d'éléments
Nom XLIB	5,5

Règles de simplification automatique

Les tableaux suivants dressent la liste des règles de simplification automatique du HP 48.

Addition et soustraction

Objet	simplifié
$x-x$	0
$0+x$	x
$(0,0)+x$	x
$0-x$	NEG(x)
$(0,0)-x$	NEG(x)
$x+0$	x

Objet	simplifié
$x+(0,0)$	x
$x+-p$	$x-p$
$x-0$	x
$x-(0,0)$	x
$x--p$	$x+p$

Multiplication et division

Objet	simplifié
INV(i)	$-i$
$y\times$INV(x)	y/x
$y/$INV(x)	$y\times x$
$0\times x$	0
$(0,0)\times x$	$(0,0)$
$i\times i$	-1
$1\times x$	x
$(1,0)\times x$	x
$(-1)\times x$	NEG(x)
$(-1,0)\times x$	NEG(x)
$x\times 0$	0
$x\times(0,0)$	$(0,0)$
$x\times 1$	x

Objet	simplifié
$x\times(1,0)$	x
$x\times(-1)$	NEG(x)
$x\times(-1,0)$	NEG(x)
$x/1$	x
$x/(1,0)$	x
$x/(-1)$	NEG(x)
$x/(-1,0)$	NEG(x)
$0/x$	0
$(0,0)/x$	$(0,0)$
$1/x$	INV(x)
$(1,0)x$	INV(x)
$(-1)/x$	$-$INV(x)
$(-1,0)/x$	$-$INV(x)

Puissances

Objet	simplifié	Objet	simplifié
1^x	1	$x^{(1,0)}$	x
$(1,0)^x$	(1,0)	$x^{(-1)}$	INV(x)
SQ($\sqrt{(x)}$)	x	$x^{(-1,0)}$	INV(x)
SQ(y^x)	$y^{(2\times x)}$	$(\sqrt{x})^2$	x
SQ(i)	-1	$(\sqrt{x})^{(2,0)}$	x
x^0	1	i^2	-1
$x^{(0,0)}$	(1,0)	$i^{(2,0)}$	$(-1,0)$
x^1	x		

Parties

Objet	simplifié	Objet	simplifié
ABS(ABS(x))	ABS(x)	MIN(x,x)	x
ABS(NEG(x))	ABS(x)	MOD(0,x)	0
CONJ(CONJ(x))	x	MOD(x,x)	0
CONJ(IM(x))	IM(x)	MOD(x,0)	x
CONJ(RE(x))	RE(x)	x MOD y MOD y	x MOD y
CONJ(i)	$-i$	RE(CONJ(x))	RE(x)
IM(CONJ(x))	$-$IM(x)	RE(IM(x))	IM(x)
IM(IM(x))	0	RE(RE(x))	RE(x)
IM(RE(x))	0	RE(π)	π
IM(p)	0	RE(i)	0
IM(i)	1	SIGN(SIGN(x))	SIGN(x)
MAX(x,x)	x		

Configurations d'intégration symbolique

Les tableaux suivants dressent la liste des configurations d'intégration symbolique utilisées par le HP 48. Ce sont les fonctions que le HP 48 est capable d'intégrer symboliquement.

ϕ est une fonction linéaire de la variable d'intégration. Les primitives doivent être divisées par le coefficient de premier ordre dans ϕ afin de réduire l'expression à sa forme la plus simple. En outre, les configurations commençant par 1/ correspondent à INV ; par exemple, $1/\phi$ est identique à INV(ϕ).

Intégration symbolique

Configuration	Primitive
ACOS(ϕ)	ϕ×ACOS(ϕ)−$\sqrt{}$(1−ϕ^2)
ALOG(ϕ)	.434294481904×ALOG(ϕ)
ASIN(ϕ)	ϕ×ASIN(ϕ)+$\sqrt{}$(1−ϕ^2)
ATAN(ϕ)	ϕ×ATAN(ϕ−LN(1+ϕ^2)/2
COS(ϕ)	SIN(ϕ)
1/(COS(ϕ)×SIN(ϕ))	LN(TAN(ϕ))
COSH(ϕ)	SINH(ϕ)
1/(COSH(ϕ)×SINH(ϕ))	LN(TANH(ϕ))
1/(COSH(ϕ)^2)	TANH(ϕ)
EXP(ϕ)	EXP(ϕ)
EXPM(ϕ)	EXP(ϕ)−ϕ
LN(ϕ)	ϕ×LN(ϕ)−ϕ
LOG(ϕ)	.434294481904×ϕ×LN(ϕ)−ϕ
SIGN(ϕ)	ABS(ϕ)
SIN(ϕ)	−COS(ϕ)
1/(SIN(ϕ)×COS(ϕ))	LN(TAN(ϕ))
1/(SIN(ϕ)×TAN(ϕ))	−INV(SIN(ϕ))
1/(SIN(ϕ)×TAN(ϕ))	−INV(SIN(ϕ))
1/(SIN(ϕ)^2)	−INV(TAN(ϕ))
SINH(ϕ)	COSH(ϕ)
1/(SINH(ϕ)×^{2}	−INV(SIN(ϕ))

Intégration symbolique (suite)

Configuration	Primitive
$1/(\text{SINH}(\phi)\times\text{COSH}(\phi))$	$\text{LN}(\text{TANH}(\phi))$
$1/(\text{SINH}(\phi)\times\text{TANH}(\phi))$	$-\text{INV}(\text{SINH}(\phi))$
$\text{SQ}(\phi)$	$\phi^3/3$
$\text{TAN}(\phi)^2$	$\text{TAN}(\phi)-\phi$
$\text{TAN}(\phi)$	$-\text{LN}(\text{COS}(\phi))$
$\text{TAN}(\phi)/\text{COS}(\phi)$	$\text{INV}(\text{COS}(\phi))$
$1/\text{TAN}(\phi)$	$\text{LN}(\text{SIN}(\phi))$
$1/\text{TAN}(\phi)\times\text{SIN}(\phi))$	$-\text{INV}(\text{SIN}(\phi))$
$\text{TANH}(\phi)$	$\text{LN}(\text{COSH}(\phi))$
$\text{TANH}(\phi)/\text{COSH}(\phi)$	$\text{INV}(\text{COSH}(\phi))$
$1/\text{TANH}(\phi)$	$\text{LN}(\text{SINH}(\phi))$
$1/\text{TANH}(\phi)\times\text{SINH}(\phi))$	$-\text{INV}(\text{SINH}(\phi))$
$\sqrt{\phi}$	$2\times\phi^{1.5}/3$
$1/\sqrt{\phi}$	$2\times\sqrt{\phi}$
$1/(2\times\sqrt{(\phi)})$	$2\times\sqrt{(\phi)}\times.5$
ϕ^z (z symbolic)	$\text{IFTE}(z==-1,\text{LN}(\phi),\phi^{(z+1)}/(z+1))$
ϕ^z (z réel, $\neq 0,-1$)	$\phi^{(z+1)}/(z+1)$
ϕ^0	ϕ
ϕ^{-1}	$\text{LN}(\phi)$
$1/\phi$	$\text{LN}(\phi)$
$1/(1-\phi^2)$	$\text{ATANH}(\phi)$
$1/(1+\phi^2)$	$\text{ATAN}(\phi)$
$1/(\phi^2+1)$	$\text{ATAN}(\phi)$
$1/(\sqrt{(\phi-1)}\times\sqrt{(\phi+1)})$	$\text{ACOSH}(\phi)$
$1/\sqrt{(1-\phi^2)}$	$\text{ASIN}(\phi)$
$1/\sqrt{(1+\phi^2)}$	$\text{ASINH}(\phi)$
$1/\sqrt{(\phi^2+1)}$	$\text{ASINH}(\phi)$

Développements trigonométriques

Les tableaux suivants dressent la liste des développements de fonctions trigonométriques en mode radians lors de l'exécution des opérations →DEF, TRG* et →TRG. Ces opérations s'affichent dans le menu RULES de EquationWriter.

→DEF

Fonction	Développement
SIN(x)	$\frac{EXP(x \times i) - EXP(-(x \times i))}{2 \times i}$
COS(x)	$\frac{EXP(x \times i) + EXP(-(x \times i))}{2}$
TAN(x)	$\frac{EXP(x \times i \times 2) - 1}{(EXP(x \times i \times 2) + 1) \times i}$
SINH(x)	$-(SIN(x \times i) \times i)$
COSH(x)	$COS(x \times i)$
TANH(x)	$TAN(x \times i) \times -i$
ASIN(x)	$-i \times LN(\sqrt{1 - x^2} + i \times x)$
ACOS(x)	$\frac{\pi}{2} + i \times LN(\sqrt{1 - x^2} + i \times x)$
ATAN(x)	$-i \times LN(\frac{(1 + i \times x)}{\sqrt{1 + x^2}})$
ASINH(x)	$-LN(\sqrt{1 + x^2} - x)$
ACOSH(x)	$\sqrt{-(\frac{\pi}{2} + i \times LN(\sqrt{1 - x^2} + i \times x))^2}$
ATANH(x)	$-LN(\frac{(1 - x)}{\sqrt{1 - x^2}})$

TRG*

Fonction	Développement
SIN($x + y$)	$SIN(x) \times COS(y) + COS(x) \times SIN(x)$
COS($x + y$)	$COS(x) \times COS(y) + SIN(x) \times SIN(y)$
TAN($x + y$)	$\frac{TAN(x) + TAN(y)}{1 - TAN(x) \times TAN(y)}$
SINH($x + y$)	$SINH(x) \times COSH(y) + COSH(x) \times SINH(x)$
COSH($x + y$)	$COSH(x) \times COSH(y) + SINH(x) \times SINH(y)$
TANH($x + y$)	$\frac{TANH(x) + TANH(y)}{1 + TANH(x) \times TANH(y)}$

→TRG

Fonction	Développement
EXP(x)	$COS(\frac{x}{i}) + SIN(\frac{x}{i}) \times i$

Références

Les ouvrages de référence suivants ont été utilisés pour un grand nombre des constantes et équations employées dans le HP 48. (Reportez-vous à la partie "Références" du chapitre 4 ("Equations") pour connaître les ouvrages de référence utilisés pour la bibliothèque d'équations.)

1. E.A. Mechtly. *The International System of Units, Physical Constants and Conversion Factors*, Second Revision. National Aeronautics and Space Administration, Washington DC, 1973.

2. *The American Heritage Dictionary*. Houghton Mifflin Company, Boston, MA, 1979.

3. *American National Standard Metric Practice ANSI/IEEE Std 268-1982*. The Institute of Electrical and Electronics Engineers, Inc., New York, 1982.

4. *ASTM Standard Practice for Use of the International System of Units (SI) E380-89a*. American Society for Testing and Materials, Philadelphia, 1989.

5. *Handbook of Chemistry and Physics*, 64th Edition, 1983-1984. CRC Press, Inc, Boca Raton, FL, 1983.

6. *International Standard publication No. ISO 31/l-1978 (E)*.

7. *The International System of Units (SI)*, Fourth Edition. The National Bureau of Standards Special Publication 330, Washington D.C., 1981.

8. *National Aerospace Standard*. Aerospace Industries Association of America, Inc., Washington D.C., 1977.

9. *Physics Letters B*, vol 204, 14 April 1988 (ISSN 0370-2693).

G

Traitement en parallèle de listes

Le principe du traitement en parallèle est le suivant : si une commande peut être appliquée à un ou à plusieurs arguments, elle peut également être étendue de manière à s'appliquer à un ou à plusieurs *ensembles* d'arguments.

Exemples :

- `5 INV` renvoie `.2`, ainsi `{ 4 5 8 } INV` renvoie `{ .25 .2 .125 }`.
- `4 5 *` renvoie `20`, ainsi `{ 4 5 6 } { 5 6 7 } *` renvoie `{ 20 30 42 }` et `{ 4 5 6 } 5 *` renvoie `{ 20 25 30 }`.

Règles générales du traitement en parallèle

En résumé, une commande donnée peut utiliser le traitement en parallèle si toutes les conditions suivantes sont remplies :

- La commande vérifie s'il y a des types d'argument corrects. Les commandes s'appliquant à tous les types d'objets, telles que DUP, SWAP, ROT, etc. n'emploient pas le traitement en parallèle des listes.
- La commande prend exactement un, deux, trois, quatre ou cinq arguments, dont aucun ne peut constituer une liste en soi. Les commandes utilisant un nombre indéfini d'arguments (telles que →LIST) n'emploient pas le traitement en parallèle des listes.
- La commande ne constitue pas une commande de branchement dans un programme (IF, FOR, CASE, NEXT, etc).

La suite de cette annexe regroupe et décrit les nombreuses commandes du HP 48 par rapport au traitement en parallèle.

Groupe 1 : commandes incompatibles avec le traitement

en parallèle

Une commande doit prendre des arguments avant de pouvoir effectuer le traitement en parallèle, car une commande sans argument (telle que RAND, VARS ou REC) ne possède pas d'arguments avec lesquels constituer un groupe.

Groupe 2 : commandes nécessitant DOLIST pour le traitement en parallèle

Ce groupe de commandes ne peut pas utiliser directement le traitement en parallèle, mais peut y être amené grâce à la commande DOLIST (voir plus loin "Utilisation de DOLIST pour le traitement en parallèle"). Ce groupe comporte plusieurs sous-groupes :

- **Commande de manipulation de la pile.** Une commande de manipulation de la pile ne peut pas effectuer de traitement en parallèle, parce que la pile est manipulée en bloc et les objets-liste sont traitées de la même manière que tout autre objet. Certaines commandes de la pile (telles que DROPN) qui prennent des arguments de niveau 1 n'acceptent pas d'arguments liste de niveau 1.

- **Commandes opérant sur des listes considérées en bloc.** Certaines commandes acceptent des listes en tant qu'arguments mais ne les traitent pas différemment des autres objets de données. Elles exécutent leur fonction sur l'objet en bloc sans prendre en compte ses éléments. Par exemple, →STR convertit l'objet-liste entier en une chaîne, ne convertissant donc pas chaque élément individuellement, et la commande == compare l'objet de niveau 1 à l'objet de niveau 2 indépendamment de leur type.

- **Commandes de manipulation de listes.** Les commandes de manipulation de listes n'effectuent pas de traitement en parallèle étant donné qu'elles opèrent sur des arguments liste en tant que listes et non en tant qu'ensembles de données parallèles. Toutefois, une commande de manipulation de listes peut être forcée à effectuer le traitement en parallèle de listes avec la commande DOLIST. Par exemple, `{ { 1 2 3 } { 4 5 6 } } « ΠLIST » DOLIST` renvoie `{ 6 120 }`.

- **Autres commandes ayant des arguments liste.** Etant donné qu'une liste peut contenir un nombre quelconque d'objets de types divers, elle sert fréquemment à contenir un nombre variable de paramètres de différents types. Certaines commandes acceptent de telles listes,

et en raison de cette particularité, sont inaptes au traitement en parallèle, sauf avec DOLIST

- **Commandes opérant avec des index.** Nombre de commandes de manipulation de tableaux en déterminent la taille en lignes et en colonnes, ou en manipulent des éléments donnés par le biais de leurs indices de lignes et de colonnes. Avec ces commandes, les indices de lignes et de colonnes sont censés être des paires de nombres réels regroupées dans des listes. Par exemple, { 3 4 } RANM génère une matrice d'entiers aléatoires qui comporte 3 lignes et 4 colonnes. Etant donné que ces commandes utilisent en principe des listes comme arguments, elles sont incapables d'effectuer le traitement en parallèle, sauf avec DOLIST.
- **Commandes de contrôle de programmes.** Les structures et les commandes de contrôle de programmes n'effectuent pas de traitement en parallèle et ne peuvent pas y être forcées. En revanche, les programmes contenant ces structures peuvent effectuer le traitement en parallèle à condition d'utiliser la commande DOLIST. Par exemple, { 1 2 3 4 5 6 } 1 « IF DUP 3 ≤ THEN DROP END » DOLIST renvoie { 3 4 5 6 }.

Groupe 3 : Commandes effectuant parfois le traitement en parallèle

Les commandes graphiques qui prennent des coordonnées en pixels comme arguments supposent que ces dernières soient présentées sous forme de listes d'entiers binaires à deux éléments. Etant donné que ces commandes utilisent en principe des listes comme arguments, elles sont incapables d'effectuer le traitement en parallèlle, sauf avec DOLIST.

Dans le cas des commandes graphiques à deux arguments (BOX, LINE, TLINE), si l'un des arguments n'est pas une liste (un nombre complexe par exemple), les commandes effectuent le traitement en parallèle, en prenant l'argument liste comme coordonnées multiples du nombre complexe. Par exemple, (0,0) { (1,1) (3,2) } LINE trace deux lignes—entre (0,0) et (1,1) et entre (0,0) et (3,2).

Groupe 4 : ADD et +

Sur les calculateurs HP 48S et HP 48SX, la commande + a été employée pour ajouter des listes ou des éléments à des listes. Ainsi, { 1 2 3 } 4 + renvoie { 1 2 3 4 }. Avec l'avènement du traitement en

parallèle sur le HP 48 ; série G, la commande ADD a été créée en vue d'effectuer l'addition en parallèle à la place de +.

Cela a plusieurs implications :

- Pour additionner deux listes en parallèle, vous devez effectuer l'une des opérations suivantes :
  - Utiliser ADD à partir du menu (MTH) LIST .
  - Créer un menu personnalisé contenant la commande ADD.
  - Affecter la commande ADD à une touche-utilisateur.
- Les programmes-utilisateur doivent être écrits en utilisant ADD à la place de + si le programme doit être en mesure d'effectuer un traitement en parallèle direct, ou écrits avec +, appliqué à leurs arguments au moyen de DOLIST. Par exemple, des programmes tels que « →x 'x+2' » produisent une concaténation de listes lorsque x est une liste et non une addition en parallèle, sauf s'ils sont écrits sous la forme « →x 'x ADD 2' »
- Des expressions algébriques capables de faire des calculs avec des variables contenant des listes (notamment celles qui doivent devenir des fonctions-utilisateur) ne peuvent pas être créées en syntaxe RPN car l'utilisation de ADD pour l'ajout de deux arguments symboliques entraîne la concaténation des arguments avec + et non avec ADD. Par exemple, 'X' DUP 2 ^ SWAP 4 * ADD 'F(X)' SWAP = produit 'F(X)=X^2+4*X' et non 'F(X)=X^2 ADD 4*X'.

Groupe 5 : commandes activant des modes ou des états

Les commandes qui stockent des valeurs dans des emplacements spécifiques du système, de manière à contrôler certains modes et certains états machine peuvent en principe effectuer le traitement en parallèle. Le problème est que chaque nouveau paramètre de la liste annule la valeur définie par le paramètre précédent. Par exemple, { 1 2 3 4 5 } FIX est en fait identique à 5 FIX.

Groupe 6 : commandes à un argument et à résultat unique

Ces commandes sont les plus faciles à utiliser avec le traitement en parallèle. Il vous suffit de fournir à la commande une liste d'arguments au lieu de l'argument unique prévu. Exemples :

{ 1 -2 3 -4 } ABS renvoie { 1 2 3 4 }

DEG { 0 30 60 90 } SIN renvoie { 0 .5 .866025403784 1 }

{ 1 A 'SIN(Z)' } INV renvoie {1 'INV(A)' 'INV(SIN(Z))'

Groupe 7 : commandes à deux arguments et à résultat unique

Les commandes à deux arguments peuvent opérer en parallèle de trois manières différentes :

- { *liste* } { *liste* }
- { *liste* } *objet*
- *objet* { *liste* }

Sous la première forme, les éléments parallèles sont combinés par la commande :
{ 1 2 3} { 4 5 6 } % renvoie { .04 .1 .18 }.

Sous la deuxième forme, l'objet de niveau 1 est combiné successivement avec chaque élément de la liste de niveau 2 :
{ 1 2 3 } 30 %CH renvoie { 2900 1400 900 }.

Sous la troisième forme, l'objet de niveau 2 est combiné successivement avec chaque élément de la liste de niveau 1:
50 { 1 2 3 } %T renvoie { 2 4 6 }.

Groupe 8 : commandes à arguments multiples et à résultat unique

Les commandes qui prennent des arguments multiples (3, 4 ou 5) ne peuvent effectuer de traitement en parallèle que si tous les arguments sont des listes. Par exemple, {'SIN(X)' 'COS(X)' 'TAN(X)' } { X X X } { 0 0 0 } ROOT renvoie { 0 90 0 }. Notez que des listes doivent être utilisées même si les listes des niveau x1 et 2 contiennent des multiples du même élément.

Groupe 9 : commandes à résultats multiples

Une commande qui permet le traitement en parallèle mais produit des résultats multiples à partir de ses données d'entrée renvoie ses résultats sous la forme d'une liste unique. Par exemple, { 1 2 3 } { 4 5 6 } R→C C→R produit { 1 4 2 5 3 6 } et non le résultat plus attendu { 1 2 3 } { 4 5 6 }.

Le programme *UNMIX* suivant dissocie les données en fonction du nombre de listes de résultats prévu :

```
« OVER SIZE → l n s
 « 1 n
  FOR j j s
   FOR i l i GET n
   STEP s n / →LIST
  NEXT
 »
»
```

Partant de la liste { 1 4 2 5 3 6 } ci-dessus, comme résultat de C→R (une commande qui devrait renvoyer deux résultats), 2 UNMIX donne { 1 2 3 } { 4 5 6 }.

Groupe 10 : commandes spéciales

Certaines commandes se comportent de manière unique par rapport au traitement en parallèle :

- **DELALARM.** Cette commande peut prendre une liste d'arguments. Notez toutefois que des suppressions réalisées en amont dans la liste des alarmes modifient les indices des alarmes suivantes. Par conséquent, s'il n'y a que trois alarmes, { 1 3 } DELALARM provoque une erreur, alors que { 3 1 } DELALARM n'en provoque pas.
- **DOERR.** Cette commande force un état d'erreur qui provoque l'interruption de tous les programmes et de toutes les commandes en cours d'exécution. Ainsi, si vous fournissez une liste d'arguments à la commande, elle effectue le traitement en parallèle, mais le premier état d'erreur va provoquer son interruption prématurée, et aucun des arguments restants de la liste ne sera utilisé.
- **FREE**, **MERGE.** Seul le port 1 peut être libéré ou fusionné sur le HP 48GX. Par conséquent, même si une liste d'arguments est acceptable, une erreur se produit pour toute liste, sauf { 1 }.
- **RESTORE.** Cette commande effectue un démarrage à chaud du système après l'installation de l'objet-sauvegarde en mémoire. Toutes les fonctions prennent fin à ce moment-là. Ainsi, seul le premier objet-sauvegarde d'une la liste sera restauré.
- **_ (Association d'unités).** Cette commande crée des objets-unités en parallèle uniquement si le niveau 1 contient une liste. Par conséquent, 1 { ft in m } _ produit {1_ft 1_in 1_m }, alors que { 1 2 3 } 'm' _ produit une erreur.

- **STO+. Cette commande effectue une addition de listes en parallèle uniquement si les deux arguments sont des listes. Si l'un des arguments est une liste et pas l'autre,** STO+ ajoute l'argument non-liste à chaque élément de la liste.
- **STO−, STO*, STO/.** Ces commandes effectuent un traitement en parallèle si les deux arguments sont des listes, mais échouent autrement.

Utilisation de DOLIST pour le traitement en parallèle

Pratiquement toute commande ou tout programme utilisateur peut opérer en parallèle sur une ou plusieurs listes de données au moyen de DOLIST. Employez cette dernière comme suit :

- Le niveau 1 doit contenir une commande, un objet-programme ou le nom d'une variable qui contenant une commande ou un objet-programme.
- Le niveau 2 doit contenir un total d'arguments, sauf si l'objet de niveau 1 est une commande qui accepte le traitement en parallèle, un programme contenant une seule commande qui accepte le traitement en parallèle ou une fonction-utilisateur. Dans ces cas particuliers, le niveau 2 contient le premier des arguments de la liste.
- Si le niveau 2 est le total d'arguments, le niveau 3 est la première des listes d'arguments. Autrement, les niveaux compris entre 2 et *n* sont les listes d'arguments.

Par exemple, le programme suivant prend trois objets dans la pile, les identifie avec les noms a, b' 'et c, et les affiche à la suite sur la ligne 1 de l'affichage.

```
« → a b c
 « ( a b c ) DUP « EVAL » DOLIST
   SWAP « →TAG » DOLIST
   CLLCD 1 « 1 DISP 1 WAIT » DOLIST
 »
»
```

H

L'environnement HP 48

Les calculateurs HP 48G et HP 48GX bénéficient d'un environnement particulièrement riche, facilitant leur utilisation : périphériques "hard", livres, cartes d'applications, revues et clubs d'utilisateurs, assistances téléphonique et minitel, etc.
Voici les principales informations à ce sujet :

1 : Périphériques pour HP 48G et HP 48GX

- Câble de connexion PC + logiciel (**HP**)
- Câble de connexion Macintosh + logiciel (**HP**)
- Adaptateurs de câbles pour imprimantes et modems (**HP**)
- Cartes RAM (**HP**) pour HP 48GX (32 Ko, 128 Ko, 1Mo)
- Kit de rétroprojection pour HP 48GX (**HP**)

2 : Cartes d'application pour HP 48GX

- Pacmath (**D3I**)

Ces périphériques et cartes d'application sont en général disponibles dans le magasin où vous avez acheté votre calculatrice.

3 : Disquettes de programmes en français :

- **Perf2 :** la 2ème disquette du magazine Performance Calcul
- **Prog. Sxtant :** la disquette du magazine 48Sxtant

Disponibles auprès de ces magazines (voir page suivante)

4 : Livres pour HP 48G et HP 48GX

Editions D3I :
• *Les Secrets de la HP 48G/GX*, tome 1 (JM Ferrard)
• *Les Secrets de la HP 48G/GX*, tome 2 (JM Ferrard)
• *Mathez la HP 48G/GX* (JM Ferrard)
• *HP 48 en liberté* (R Pulluard)
• *Assembleur sur HP 48* (Kezirian)

Editions Dunod :
• *HP 48G et GX : Permis de conduire* (L Fieux)
• *HP 48G et GX pour le bac !* (H Lemberg)
• *HP 48G/GX/S/SX : Physique-Chimie en prépa + dsk* (Canon)
• *HP 48G/GX/S/SX en prépa + dsk* (Lesueur)
• *HP 48 : Faites vos jeux en assembleur + dsk*

Editions Angkor :
• *Voyage au centre de la HP 48G/GX* (P Courbis)
• *Initiation à l'assembleur pour HP 48* (Machba, Lecourt)

Ces livres sont en général disponibles dans le magasin où vous avez acheté votre calculatrice et dans les librairies scientifiques ou techniques.

5 : Revues et magazines en français

• **PERFORMANCE CALCUL** (ex *Haute Performance*)
POLE - BP 87 - 75622 Paris cedex 13
Voir bulletin d'abonnement à la fin du volume.

• **48 SXTANT**
Lijsterlaan 31, 2665 TH BLEISWIJK, Pays-Bas

6 : Minitel, assistance et après-vente :

• Minitel 3615 HPERF
Service minitel du magazine Performance Calcul : forum, questions-réponses, téléchargement de programmes, etc.
Abonnement 3614 possible (POLE - BP 87 - 75622 Paris cedex 13). Logiciel et câble de connexion PC disponibles auprès de POLE.

• HP Calculator Bulletin Board System.
Ce service permet l'échange de logiciels et d'informations entre utilisateurs de calculateurs HP, concepteurs de programmes et distributeurs. Il fonctionne à 300/1200/2400 bauds, en duplex intégral, sans parité, 8 bits, 1 bit d'arrêt. Le numéro de téléphone est aux USA : (503) 750 4448. Le service est gratuit, mais les frais de communication sont à votre charge.

• Support technique téléphonique France : (1) 69 18 20 64
Si vous êtes sans réponse à un problème après la consultation du manuel ... appelez Infopoint
• Support technique Belgique : (02) 778 31 11
• Support technique Canada francophone : (514) 697 42 32
• Support technique Suisse francophone : 156 83 73
(lundi - jeudi 9h à 12 h)

• Service après-vente, téléphone : (1) 69 18 20 22
Les calculateurs (*sauf pour ce qui concerne les piles ou les dommages causés par ces dernières*) sont garantis par Hewlett Packard contre tout vice de matière et de fabrication pour une durée d'un an à partir de la date de livraison, la facture faisant foi.
A l'issue de la période de garantie, les réparations sont effectuées moyennant un coût forfaitaire incluant pièces et main d'œuvre.

Index

E

M

N

O

S

T

U

X

Z

PERFORMANCE CALCUL

LE MAGAZINE DES UTILISATEURS DE CALCULATRICES HP

BULLETIN D'ABONNEMENT

à adresser à : POLE - B.P. 87 - 75622 - PARIS Cedex 13.

NOM : ..... **Prénom :** .....
Occupation : .....**Tél :** .....
Adresse : .....
.....

Je m'abonne à Performance Calcul !

❍ **Année 1995/1996 : 4 numéros + 2 Hors Série : 125 FF.**
Chaque abonné a droit à une petite annonce gratuite sur l'année d'abonnement (sans rétroactivité).

Ajoutez 25FF si vous n'habitez pas en France.

❍ Disquette de programmes **PERF2** pour HP48 S/SX/G/GX **: 48 FF.**

Numéros des années antérieures (Haute Performance)

❍ Numéros 1 à 4 : 48 FF.
❍ Numéros 5 à 8 : 58 FF.
❍ Nos 9 à 12 + HS "*Programmationsur HP 48G/GX*" : 75 FF.

Ajoutez 25FF par an si vous n'habitez pas en France.

Chaque année, Performance Calcul, c'est :

4 NUMÉROS PASSIONNANTS
\+ 2 HORS SÉRIE, PETITS GUIDES PRATIQUES INDISPENSABLES DE 64P.
\+ LE SERVICE MINITEL 3615 HPERF

LA BIBLE de la HP 48 G/GX

Manuel de référence avancé en français

Ce manuel, rédigé par les ingénieurs de Hewlett Packard, contient, sur près de 800 pages, toutes les informations (ou presque) dont peut avoir besoin un utilisateur de calculatrices du type HP 48G et HP 48 GX.

1 : Programmation
2 : Exemples de programmes
3 : Commandes
4 : Equations

Annexes :
A : Messages d'erreur
B : Tables d'unités
C : Indicateurs système
D : Variables réservées
E : Nouvelles commandes
F : Références techniques
G : Traitement en parallèle de listes
H : L'environnement HP 48

9 782909 737119

195FF

Diffusion FNAC