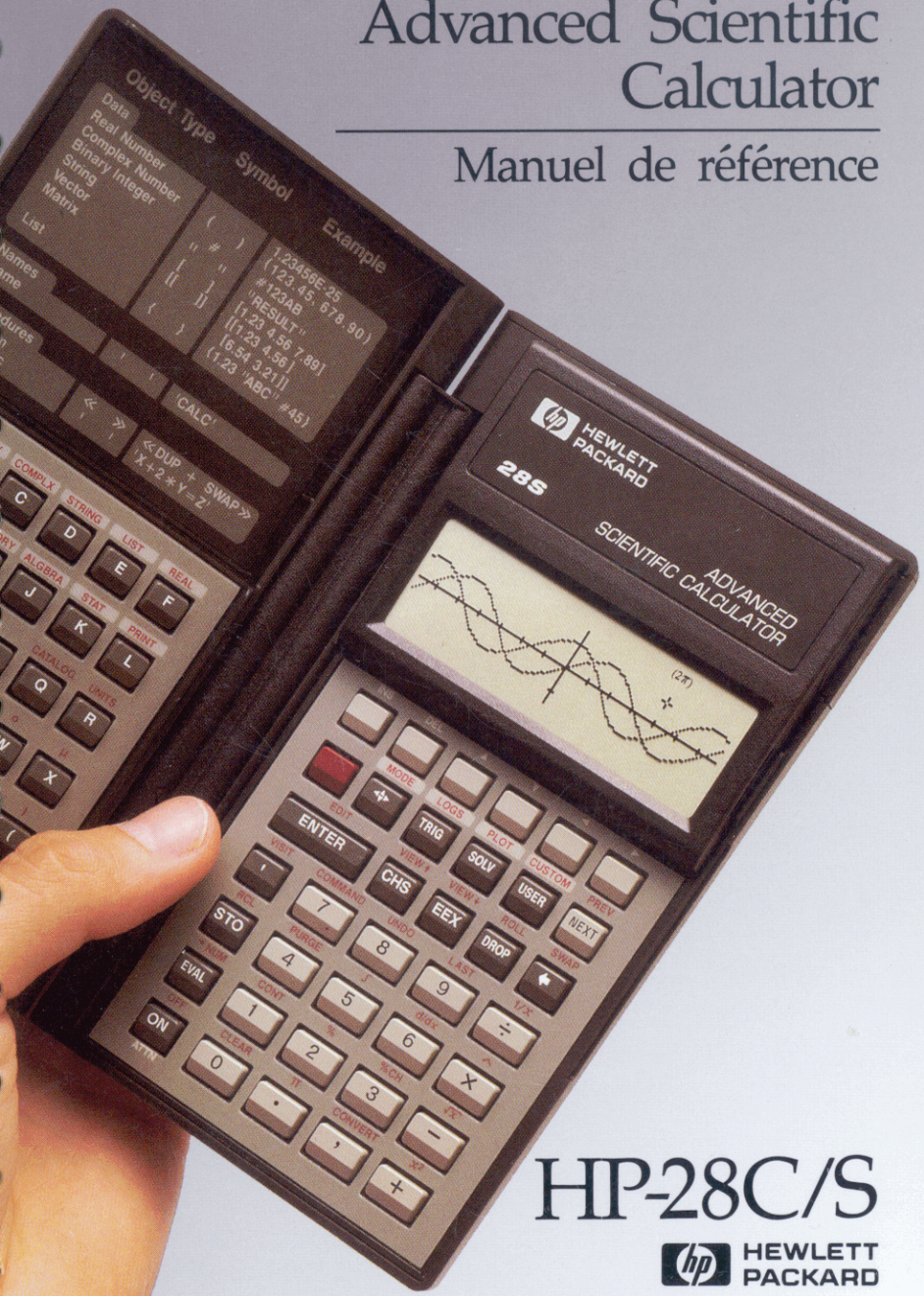
HEWLETT-PACKARD

Advanced Scientific Calculator

Manuel de référence



HP-28C/S

HEWLETT PACKARD

HP-28C/S

Manuel de référence

HEWLETT PACKARD

Edition 2 - février 1988
N° de référence : 00028-90023

Avertissement

1. Les informations contenues dans ce document peuvent faire l'objet de modifications sans préavis.
2. En raison de la complexité des techniques informatiques, ce document est remis au lecteur dans le seul but de faciliter sa compréhension du produit dont il traite. HPF décline en conséquence toute responsabilité pour tout dommage pouvant résulter des informations contenues dans ce document.
3. HPF ne garantit ni la fiabilité ni les conséquences de l'utilisation de ses produits logiciels lorsqu'ils sont utilisés sur des produits dont il n'a pas assuré la fourniture.
4. Les informations contenues dans ce document sont originales. Elles ont été conçues et mises au point par Hewlett-Packard. L'acheteur s'interdit en conséquence, sauf accord préalable et écrit de HPF :
   — de les divulguer ou d'en faciliter la divulgation ;
   — de les copier ou de les reproduire en tout ou en partie par n'importe quel moyen et sous n'importe quelle forme ;
   — de les traduire dans toute autre langue.

Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.

Historique de la publication

Edition 1	octobre 1986	Mfg. No. 00028-90024
Edition 2	février 1988	Mfg. No. 00028-90114

Présentation du HP-28C/S

Avec le HP-28C et le HP-28S, vous pouvez facilement résoudre des problèmes complexes, y compris ceux que vous ne pouviez pas résoudre à l'aide d'un calculateur jusqu'à présent. Le HP-28C/S ajoute en effet une nouvelle dimension à ses puissantes fonctions de calcul numérique : *le calcul symbolique*. Vous pouvez formuler un problème symboliquement, trouver une solution symbolique illustrant le comportement global du problème, puis obtenir des résultats numériques à partir de la solution symbolique.

Le HP-28C/S vous offre les possibilités suivantes :

- Manipulation algébrique. Développement, mise en facteur ou réorganisation des termes d'une expression et résolution symbolique d'une équation (obtention d'une variable définie symboliquement).
- Calcul intégral et différentiel. Calcul de dérivées et d'intégrales définies et indéfinies.
- Solutions numériques. En utilisant la fonction de résolution d'équations du HP-28C/S, calcul des zéros d'une expression ou des racines d'une équation. Résolution d'un système d'équations linéaires. Dans le cas de nombreux types de données différents, la manipulation des nombres complexes, vecteurs et matrices est aussi facile que celle des nombres réels.
- Tracés graphiques. Représentation graphique des expressions, équations, et données statistiques.
- Conversions d'unités. Conversions entre les 120 unités intégrées, ou définition de vos propres unités.
- Statistiques. Calculs de statistiques avec observations sur une ou deux variables, et calculs de probabilités.
- Bases de numération. Calculs sur des nombres binaires, octaux et hexadécimaux, et manipulations de bits.
- Saisie directe pour les formules algébriques, notation polonaise inverse pour les calculs interactifs.

Le *Manuel d'utilisation du HP-28C/S* vous guidera dans la découverte du calculateur par de nombreux exemples.

Le *Manuel de référence du HP-28C/S* (ce manuel) donne des informations précises sur les commandes et sur la façon dont le calculateur fonctionne. Les deux premiers chapitres exposent les principes généraux et les opérations de base. Le troisième chapitre est un dictionnaire des menus décrivant les concepts et les commandes de chaque menu.

Nous vous conseillons de suivre pas à pas les exemples du *Manuel d'utilisation du HP-28C/S* pour vous familiariser peu à peu avec votre calculateur. Lorsque vous désirerez approfondir vos connaissances sur l'une ou l'autre commande, cherchez-la dans le *Manuel de référence*. Lorsque vous aurez une bonne connaissance des commandes et voudrez comprendre le fonctionnement du calculateur, lisez les chapitres théoriques du *Manuel de référence*.

Ces manuels vous montrent l'utilisation du HP-28C/S en mathématiques, mais ils ne constituent pas pour autant un cours de mathématiques. Nous supposons que vous êtes déjà familier avec les principes mathématiques en jeu. Pour utiliser les fonctions de calcul intégral et différentiel, par exemple, il est nécessaire de connaître les bases de ces méthodes de calcul.

Mais d'autre part, il n'est pas nécessaire de connaître tous les calculs mathématiques dont est capable le HP-28C/S pour utiliser ceux qui vous intéressent particulièrement. Vous pouvez par exemple utiliser ses possibilités de calculs statistiques sans avoir la moindre notion de calcul intégral ou différentiel.

Table des matières

Annexes, glossaire, index

Comment utiliser ce manuel

Ce manuel contient une description du fonctionnement du HP-28C/S et des informations spécifiques de chacune des opérations du calculateur. La lecture de la table des matières vous donnera une idée générale de son contenu. Les autres types de renseignements peuvent être trouvés de la manière suivante.

Pour vous renseigner sur :	Faites référence à :
Une opération, une commande ou une fonction précises.	Index des opérations (page 381). Toutes les opérations, commandes et fonctions sont classées par ordre alphabétique. Chaque rubrique comprend une brève description, une référence à un menu ou à un des sujets traités dans le Dictionnaire ainsi que le numéro de page.
Un menu particulier.	Le chapitre 3, « Dictionnaire » (page 57). Tous les menus sont classés en ordre alphabétique.
Les concepts et principes de fonctionnement du HP-28C/S.	Le chapitre 1, « Principes généraux » (page 17).
Les opérations de base du HP-28C/S.	Le chapitre 2, « Opérations de base » (page 31).
La signification d'un message.	L'annexe A, « Messages » (page 349).
La signification d'un terme ou d'une expression.	Le glossaire (page 367).

L'organisation de ce manuel

Les chapitres 1 et 2 contiennent les informations générales sur le calculateur. « Principes généraux » est destiné au lecteur expérimenté, il décrit le fonctionnement du calculateur. Le chapitre suivant, « Opérations de base », décrit la saisie d'objets dans la ligne de commande, la création de variables et l'exécution d'opérations système.

Le chapitre 3, « Dictionnaire » constitue la partie la plus importante du manuel. Organisé en menus, il décrit en détail chaque opération, chaque commande et chaque fonction. Les effets de chaque commande et de chaque fonction sont représentés dans un diagramme de la pile opérationnelle (lisez « Les diagrammes de la pile opérationnelle », plus loin).

Le chapitre 3 traite de vastes sujets non attachés à un menu particulier. Vous y trouverez les sections Arithmétique, Calcul différentiel et intégral, CATALOG, Programmes et UNITS.

L'annexe A, « Messages », décrit les messages d'état et d'erreur que vous serez amené à rencontrer.

L'annexe B, « Si vous connaissez déjà la notation polonaise inverse » et l'annexe C, « Si vous ne connaissez pas la notation polonaise inverse » font la comparaison entre le HP-28C/S et d'autres types de calculateurs que vous connaissez peut-être.

Le glossaire définit les termes et expression utilisés dans ce manuel.

L'index des opérations est une liste alphabétique de toutes les opérations, commandes et fonctions du HP-28C/S. Chaque rubrique comprend une courte description, une référence au chapitre ou au titre de menu (ARRAY, BINARY, CATALOG etc.) où vous trouverez les informations et un numéro de page où vous trouverez l'information recherchée.

Les diagrammes de la pile opérationnelle

L'action d'une commande est définie par les valeurs et l'ordre de ses arguments et de ses résultats. Un *argument* est un objet, pris dans la pile, sur lequel s'exerce la commande. La commande renvoie alors un *résultat* vers la pile (quelques commandes agissent sur les modes, les variables, les indicateurs binaires ou l'affichage, plutôt que de renvoyer des objets).

La description de chaque commande comprend un *diagramme de la pile opérationelle,* qui donne un tableau des arguments et des résultats de la commande. Ces diagrammes ont l'aspect suivant :

XMPLE ***Exemple*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
objet$_1$	*objet*$_2$	➧	*objet*$_3$

Ce diagramme montre :

- Le nom de la commande (qui peut apparaître en ligne de commande) est « XMPLE ».
- La description de la commande (« Exemple », en toutes lettres).
- XMPLE est une fonction (autorisée dans les expressions algébriques).
- XMPLE nécessite deux arguments, *objet*$_1$ et *objet*$_2$, pris dans la pile opérationnelle, respectivement aux niveaux 2 et 1.
- XMPLE renvoie un résultat, *objet*$_3$, au niveau 1.

La flèche ➧ sépare les arguments (à gauche) des résultats (à droite). Elle signifie en fait « avec les arguments ci-joints, l'exécution d'XMPLE renvoie ce résultat en ligne de commande ».

Les arguments et résultats sont présentés sous diverses formes qui donnent autant d'informations que possible sur les objets. Les objets de types spéciaux sont affichés à l'intérieur de leurs délimiteurs. Les mots ou formules inclus avec les délimiteurs offrent une description supplémentaire des objets. Les diagrames de pile opérationnelle usent en général de la terminologie suivante :

Termes utilisés dans les diagrammes de la pile

Terme	Description
objet	N'importe quel objet.
x ou *y*	Nombre réel.
hms	Nombre réel en format heures-minutes-secondes.
n	Nombre réel entier positif.
indic. binaire	Nombre réel, nul (état faux) ou non nul (état vrai).
z	Nombre réel ou complexe.
(*x* , *y*)	Nombre complexe sous forme rectangulaire.
(*r* , *θ*)	Nombre complexe sous forme polaire.
# *n*	Entier binaire.
" *chaîne* "	Chaîne de caractères.
[*tableau*]	Matrice ou vecteur réel ou complexe.
[*vecteur*]	Vecteur réel ou complexe.
[*matrice*]	Matrice réelle ou complexe.
[*tableau R*]	Matrice ou vecteur réel.
[*tableau C*]	Matrice ou vecteur complexe.
{ *liste* }	Liste d'objets.
{ *index* }	Liste d'un ou deux nombre(s) réel(s) définissant l'élément d'un tableau.
{ *dim* }	Liste d'un ou deux nombre(s) réel(s) définissant la (les) dimension(s) d'un tableau.
' *nom* '	Nom ou nom local.
« *programme* »	Programme.
' *symbole* '	Equation, expression, ou nom traité comme un objet algébrique.

Le diagramme de pile peut contenir plus d'une ligne « argument » ➧ « résultat », selon les diverses combinaisons possibles d'arguments et de résultats. Quand c'est possible, les résultats sont écrits sous une forme qui montre la combinaison mathématique des arguments. Par exemple, le diagramme de pile de + comprend, entre autres, les saisies suivantes :

+ ***Addition*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	z_1+z_2
⊏ *tableau*$_1$ ⊐	⊏ *tableau*$_2$ ⊐	➧	⊏ *tableau*$_1$ + *tableau*$_2$ ⊐
z	' *symbole* '	➧	' z + (*symbole*) '

Ce diagramme montre que :

- Le fait d'ajouter deux nombres réels ou complexes z_1 et z_2 renvoie un troisième nombre réel ou complexe avec la valeur z_1+z_2.
- Le fait d'ajouter deux tableaux ⊏ tableau$_1$ ⊐ et ⊏ tableau$_2$ ⊐ renvoie un troisième tableau ⊏ tableau$_1$ + tableau$_2$ ⊐.
- Le fait d'ajouter un nombre complexe z et un objet symbolique ' *symbole* ' renvoie un objet symbolique ' z + (*symbole*) '.

1

Principes généraux

Le HP-28C/S fonctionne à partir de quelques principes de base. Ces principes sont un peu abstraits, mais leur généralité même est la clé de la puissance et de la souplesse du calculateur. Vous n'avez pas besoin d'une connaissance approfondie du calculateur pour l'utiliser ; mais comprendre sur quels principes il se base vous permettra d'en exploiter pleinement toutes les ressources.

Si vous n'avez pas encore utilisé le HP-28C/S, nous vous recommandons de commencer par étudier le *Manuel d'utilisation du HP-28C* ou du *28S*. Il vous guide pas à pas dans la résolution de problèmes types et illustre de manière pratique les principes généraux du calculateur. Une fois que vous aurez acquis de l'expérience, vous pourrez revenir à ce manuel pour comprendre comment le calculateur fonctionne d'une manière plus globale.

Ce chapitre commence par une description générale de la façon dont le calculateur fonctionne, puis il détaille les différents éléments de la description, et aborde enfin la pile opérationnelle, les modes et les erreurs. Les informations sur la saisie des différents « objets », les variables, et autres sujets similaires figurent au chapitre 2, intitulé « Opérations de base ». Le chapitre 3, « Dictionnaire », contient, regroupées par menus, toutes les informations requises sur chaque opération ou commande.

Principes de fonctionnement

Le fonctionnement du calculateur est basé sur l'évaluation des objets se trouvant dans la pile opérationnelle. Un objet peut être une donnée, un nom, ou une procédure. Evaluer un objet signifie effectuer l'action associée à cet objet. *Les données sont des objets qui ne sont associés à aucune action particulière (elles sont simplement des données) ; les noms sont des objets qui réfèrent à d'autres objets ; et les procédures sont des objets qui traitent les commandes et objets dont ils sont constitués.*

L'avantage de ce principe est son *uniformité*. Pour les opérations comme la saisie, la modification, la copie, le stockage et le rappel, vous traitez tous les objets de la même façon. Cette uniformité signifie pour vous un moins grand nombre de règles à vous rappeler.

Un de ses autres avantages est sa *souplesse*. Vous pouvez utiliser des objets et les combiner de multiples façons pour créer les outils nécessaires à la résolution d'un problème spécifique. Du fait que vous pouvez choisir si un objet symbolique doit être évalué, et quand il doit l'être, vous pouvez résoudre un problème à la fois symboliquement et numériquement.

Premier type d'objet : les données

Ces objets sont les données traitées en tant qu'unités logiques : données numériques, chaînes de caractères, et listes d'objets.

Données

Type	Objet	Description
Nombre réel	Nombre réel	Nombre décimal en virgule flottante appartenant à l'ensemble des réels.
Nombre complexe	Nombre complexe	Nombre décimal en virgule flottante appartenant à l'ensemble des complexes.
Entier binaire	Entier binaire	Nombre entier binaire codé sur 64 bits.
Chaîne	Chaîne	Chaîne de caractères.
Tableau réel	Vecteur réel Matrice réelle	Vecteur réel à n éléments. Matrice réelle de $n \times m$ éléments.
Tableau complexe	Vecteur complexe Matrice complexe	Vecteur complexe à n éléments. Matrice complexe de $n \times m$ éléments.
Liste	Liste	Liste d'objets.

Evaluer un objet qui est une donnée n'a aucun effet. Si vous placez une donnée dans la pile opérationnelle et appuyez sur [EVAL], la donnée reste simplement dans la pile, inchangée. Vous remarquerez que les objets contenus dans une liste ne sont pas évalués lorsque la liste est évaluée.

Deuxième type d'objet : les noms

Ces objets nomment d'autres objets stockés en mémoire utilisateur. Des *noms locaux* peuvent être créés par des procédures, puis être automatiquement supprimés lorsque l'évaluation de la procédure est terminée.

Noms

Type	Objet	Description
Nom	Nom	Réfère à un objet stocké en mémoire utilisateur.
	Nom local	Réfère à un objet temporairement stocké en mémoire locale.

Variables

Une variable est la combinaison d'un objet quelconque et d'un nom qui sont stockés ensemble. Le nom devient le *nom* de la variable ; l'autre objet est la *valeur* ou le *contenu* de la variable. Ils sont stockés ensemble en *mémoire utilisateur,* laquelle est distincte de la pile opérationnelle. Les variables du HP-28C/S remplacent les registres de données numérotés et les registres de programmation utilisés sur la plupart des calculateurs.

Il y a deux aspects dans l'évaluation d'un nom de variable : ce qui cause l'évaluation du nom, et le résultat de l'évaluation.

Quand un nom est-il évalué ?

- Un libellé du menu utilisateur (USER) est évalué lorsque vous appuyez sur la touche de fonction correspondante en mode exécution.
- Un nom sans guillemets dans la ligne de commande est évalué lorsque la ligne de commande est évaluée.
- Un nom sans guillemets dans une procédure est évalué lorsque la procédure est évaluée.
- Un nom contenu dans une variable est évalué lorsque le nom de la variable est évalué.
- Un nom au niveau 1 de la pile est évalué lorsque la commande EVAL est exécutée.

Que se passe-t-il lorsqu'un nom est évalué ?

- Evaluer un nom qui correspond à une variable place dans la pile opérationnelle l'objet contenu dans la variable, et l'évalue si cet objet est un nom ou un programme.
- Evaluer un nom qui ne correspond pas à une variable fait simplement revenir ce nom dans la pile opérationnelle.

Lorsque vous examinez quand et comment un nom est évalué, notez que :

- Vous pouvez rappeler les données contenues dans une variable simplement en évaluant le nom de cette variable.
- Un nom sans guillemets qui réfère à un programme a la même fonction qu'une commande destinée à évaluer le programme.
- Si un nom renvoie à un autre nom, qui renvoie lui-même à un autre nom, et ainsi de suite, l'évaluation du premier nom entraîne l'évaluation de tous les autres noms.

Remarque Ne créez pas une variable dont la valeur comprend le nom de la variable elle-même, par exemple en exécutant `'X' 'X' STO` ou `'X+Y' 'X' STO`. Une telle variable génèrerait en effet une boucle sans fin. Pour arrêter une boucle sans fin, vous devez exécuter un arrêt du système (ON ▲, décrit au chapitre « Opérations de base »), qui a également pour effet d'effacer la pile opérationnelle.

De même, ne créez pas de variables se faisant réciproquement référence de manière circulaire. L'évaluation d'une telle variable génèrerait également une boucle sans fin.

Variables locales

Les variables locales servent uniquement dans les structures de programme qui créent une variable. Par exemple, les fonctions définies par l'utilisateur et les structures FOR...NEXT utilisent des variables locales. Les noms identifiant les variables locales sont appelés *noms locaux* et sont décrits à la rubrique « Programmes ». Evaluer le nom d'une variable locale place simplement le contenu de la variable locale dans la pile opérationnelle.

Variables formelles

Dans les calculs symboliques, vous pouvez utiliser des noms comme des variables (au sens mathématique) avant d'attribuer des valeurs à ces variables. Si vous recherchez un résultat symbolique — la dérivée d'une expression, par exemple —, vous n'aurez probablement jamais besoin d'attribuer des valeurs aux variables.

Les noms utilisés comme variables mathématiques, mais qui ne sont pas associés à des objets stockés en mémoire, sont appelés *variables formelles*. Nous n'utiliserons ce terme que rarement, lorsqu'il est important de faire la distinction. Evaluer une variable formelle laisse simplement son nom dans la pile opérationnelle.

Troisième type d'objet : les procédures

Ces objets contiennent des *procédures*, c'est-à-dire des séquences d'objets et de commandes qui sont traitées lorsque la procédure est évaluée. Un *programme* est un objet qui peut contenir n'importe quelle séquence d'objets et de commandes, y compris celles affectant la pile opérationnelle, la mémoire utilisateur, ou les modes de fonctionnement du calculateur. Un *objet algébrique* contient un nombre limité de types d'objets et de commandes, et sa syntaxe est similaire à celle des équations et expressions mathématiques.

Procédures

Type	Objet	Description
Programme	Programme	Contient une séquence quelconque d'objets.
Obj. algébrique	Expression Equation	Contient une expression mathématique. Contient une équation mathématique liant deux expressions.

Programmes

Un programme est essentiellement une ligne de commande sous la forme d'un objet. Les objets et commandes que vous introduisez dans la ligne de commande constituent une procédure. Lorsque vous « encadrez » cette procédure avec des délimiteurs de programme, vous indiquez que vous voulez traiter cette procédure comme un objet qui sera évalué ultérieurement.

Quand un programme est-il évalué ?

- Vous pouvez évaluer un programme se trouvant au niveau 1 de la pile en exécutant la commande EVAL.
- Un programme stocké dans une variable est évalué lorsque le nom de la variable est évalué.
- Les commandes comme DRAW, $\int$, et ROOT évaluent de manière répétitive un programme qui est leur argument.
- Un programme qui est la partie procédure d'une structure à variable locale est évalué lorsque cette structure est évaluée.

Que se passe-t-il lorsqu'un programme est évalué ?

- Evaluer un programme place chaque objet dans la pile opérationnelle et l'évalue si cet objet est une commande ou un nom sans guillemets.

Lorsque vous examinez quand et comment un programme est évalué, notez que :

- Supposons qu'un programme contient un nom sans guillemets qui réfère à un autre programme, lequel contient à son tour un nom sans guillemets référant à un autre programme, et ainsi de suite. L'évaluation du premier programme entraîne l'évaluation de tous les programmes qui doivent être évalués (le dernier auquel il est fait référence est le premier à être complètement évalué).
- Si vous exécutez une commande qui évalue un programme, les commandes contenues dans ce programme peuvent remplacer (« écraser ») les arguments de la commande originale, qui sont stockés dans LAST.

Expressions

Une expression est une procédure représentant une expression mathématique qui est saisie et affichée avec une syntaxe correspondant aux formules mathématiques ordinaires.

Quand une expression est-elle évaluée ?

- Vous pouvez évaluer une expression en exécutant la commande EVAL (l'évaluation des expressions est l'utilisation la plus courante de EVAL).

- Les commandes comme DRAW, ∫, ROOT, TAYLR, et QUAD évaluent de manière répétitive une expression qui est leur argument.
- Une expression qui définit une fonction créée par l'utilisateur est évaluée lorsque la fonction est évaluée.

Que se passe-t-il lorsqu'une expression est évaluée ?

- L'évaluation d'une expression a pour résultat de placer chaque objet dans la pile opérationnelle et de l'évaluer. Ces objets sont évalués dans l'ordre dicté par la notation polonaise inverse (l'ordre du programme équivalent), et non pas dans l'ordre dans lequel ils apparaissent dans l'expression.

En ce qui concerne le moment où et la façon dont une expression est évaluée, notez que :

- Alors que l'évaluation d'un nom qui réfère à un programme a pour effet d'évaluer le programme, l'évaluation d'un nom qui réfère à une expression a pour effet de placer cette expression dans la pile opérationnelle.
- Si, dans une expression, un nom réfère à une seconde expression, l'évaluation de la première expression *n'a pas pour effet* d'évaluer la seconde expression, mais de remplacer le nom par cette seconde expression.

Equations

Les équations sont deux expressions reliées par le signe égale (« = »). L'évaluation d'une équation donne une nouvelle équation. La nouvelle expression gauche est le résultat de l'évaluation de l'expression gauche initiale, et la nouvelle expression droite est le résultat de l'évaluation de l'expression droite initiale.

Commandes

Les commandes sont des procédures intégrées au calculateur et que vous pouvez inclure dans des programmes. Vous pouvez considérer un nom de commande tel qu'il apparaît dans la ligne de commande (DROP ou SIN par exemple) comme le nom sans guillemets d'une procédure stockée de façon permanente dans le calculateur. Il s'agit donc de quelque chose de similaire aux noms et contenus de vos propres variables. En pratique, il est inutile de faire une distinction entre le nom sans guillemets et la procédure intégrée.

Les procédures intégrées sont classées en fonction de leur utilisation :

- Une *opération* est n'importe quelle procédure intégrée dans le calculateur, comme ENTER, CATALOG, ou TRACE.
- Une *commande* est une opération programmable, comme SWAP ou STO.
- Une *fonction* est une commande autorisée dans les expressions algébriques, par exemple IP ou MIN.
- Une *fonction analytique* est une fonction pour laquelle le HP-28C/S peut fournir une dérivée ou une inverse, par exemple SIN ou +.

Les procédures intégrées se caractérisent généralement par leur grande souplesse d'utilisation. Par exemple, SWAP est à la fois une commande et une opération, et IP est à la fois une fonction, une commande et une opération ; mais nous définissons SWAP comme une commande et IP comme une fonction.

La pile opérationnelle

La pile opérationnelle est une succession de *niveaux* numérotés, qui contiennent chacun un objet. Les objets sont introduits dans la pile au niveau 1, décalant ainsi d'un niveau « vers le haut » les autres objets se trouvant déjà dans la pile. De même, les objets quittent la pile à partir du niveau 1, décalant d'un niveau « vers le bas » tous les autres objets restant dans la pile. Tous les objets, de quelque type qu'ils soient, sont traités de la même façon dans la pile — simplement comme des objets.

Le HP-28C/S met à votre disposition des commandes pour dupliquer, supprimer et réoganiser les objets dans la pile. Plusieurs commandes sont accessibles directement sur le clavier : [DROP] (supprimer un objet), ■[SWAP] (permuter deux objets), ■[ROLL] (déplacer un objet vers le niveau 1), et ■[CLEAR] (effacer la pile) ; les autres se trouvent dans le menu STACK.

Pour la plupart des commandes, les objets « en entrée » (*arguments*) sont pris dans la pile et les objets « en sortie » (*résultats*) sont renvoyés dans la pile. Les arguments doivent se trouver dans la pile avant que la commande ne soit exécutée. La commande enlève alors ses arguments de la pile et les remplace par ses résultats. Par exemple, la fonction SIN prend une valeur (nombre réel ou complexe, ou expression algébrique) dans le niveau 1, en calcule le sinus, puis envoie le résutat au niveau 1. La fonction + prend deux valeurs de la pile et envoie leur somme dans la pile.

Ce type de logique, où la commande vient après les arguments, est appelée *logique suffixe* ou *notation polonaise inverse*, par référence au mathématicien polonais Jan Lukasiewicz (1878-1956).

(Notez que ces commandes à logique suffixe (ou inverse) comprennent des opérations qui peuvent utiliser une syntaxe de type préfixe sur d'autre calculateurs à notation polonaise inverse. Par exemple, pour choisir le mode d'affichage FIX avec deux chiffres après la virgule décimale, vous devez, sur le HP-28C/S, exécuter la séquence `2 FIX`. De même, pour stocker le nombre 12 dans une variable nommée VAR, vous devez exécuter la séquence `12 'VAR' STO`.)

Modes

Pour de nombreuses opérations, vous pouvez contrôler les résultats en choisissant un mode. Vous pouvez décider par exemple si les fonctions trigonométriques interprètent les nombres comme des degrés ou des radians en choisissant le mode DEGRES ou RADIANS.

Le tableau ci-après fait la liste des modes du HP-28C/S, regroupés par sujets. Pour chaque mode, le choix par défaut (activé chaque fois que vous réinitialisez la mémoire) est identifié par un astérisque.

La plupart des modes sont identifiés par un indicateur binaire, un témoin sur l'affichage, ou un menu lorsque le mode est choisi par une touche de menu (le libellé de menu correspondant à la touche activée apparaît en vidéo normale — caractères noirs sur fond clair — et non pas en vidéo inverse comme les autres).

Les modes du HP-28C/S

Mode	Choix	Identification
Angle	Degrés*/radians	menu MODE, indic. binaire 60 désarmé*/armé, témoin (2π)
Tonalité	Validée*/invalidée	Indic. binaire 51 désarmé*/armé
Valeur principale	Invalidée*/validée	Indic. binaire 34 désarmé*/armé

* Choix par défaut.

Les modes du HP-28C/S *(suite)*

Mode	Choix	Identification
Saisie et affichage généraux		
Mode de saisie	Exécution*/ algébrique/alpha	Curseur, témoin α
Caractères	Majusc.*/minusc.	Aucune
Affichage du niveau 1	Multi-lignes*/compact	Menu MODE, indic. binaire 45 armé*/désarmé
Saisie et affichage de nombres réels		
Séparateur décimal	Point*/virgule	menu MODE, indic. binaire 48 désarmé*/armé
Format des nbres réels	Standard*/fixe/ scientifique/ingénieur	menu MODE, indic. binaires 49 – 50
Nbre de chiffres décimaux	0* à 11	Indic. binaires 53 – 56
Saisie et affichage d'entiers binaires		
Système	Décimal*/hexadécimal/ octal/binaire	Menu BINARY, indic. binaires 43 – 44,
Taille du mot	1 à 64*	Indic. binaires 37 – 42
Récupération de données après erreur		
COMMAND	Validée*/invalidée	Menu MODE
UNDO	Validée*/invalidée	Menu MODE
LAST	Validée*/invalidée	Menu MODE, indic. binaire 31 armé*/désarmé
Evaluation		
De constantes symboliques	Symbolique*/ numérique	indic. binaire 35 armé*/désarmé
De fonctions	Symbolique*/ numérique	Indic. binaire 36 armé*/désarmé
Imprimante		
Impression automatique (TRACE)	Invalidée*/validée	Menu PRINT, indic. binaire 32 désarmé*/armé
CR automatique	Validé*/invalidé	Indic. binaire 33 désarmé*/armé
Impression plus rapide	Invalidée*/validée	Indic. binaire 52 désarmé*/armé

* Choix par défaut.

Témoins d'état

Les témoins en haut de l'affichage indiquent le mode angulaire, le mode de saisie, et autres informations sur l'état du calculateur.

Témoins d'état

Témoin	Signification
O	Un programme est suspendu.
⬏	La touche préfixe rouge a été actionnée.
α	Le mode de saisie alpha est activé.
((•))	Le HP-28C/S est occupé —il ne peut pas recevoir de données saisies au clavier.
▭	Les piles sont déchargées.
(2π)	Mode angulaire en cours : radians.
🖷	Le HP-28C/S envoie des données à l'imprimante.

Indicateurs binaires

Un *indicateur binaire* est une quantité qui représente une valeur *vraie* ou *fausse*. Il existe des *indicateurs binaires numériques et des indicateurs binaires d'utilisateur.*

Indicateurs binaires numériques. Dans la pile opérationnelle, un nombre réel non nul représente *l'état vrai*, et le nombre réel 0 représente *l'état faux*. Les indicateurs binaires numériques sont utilisés pour les branchements de programmes, comme IF...THEN...ELSE, et pour les tests logiques, comme XOR.

Indicateurs binaires d'utilisateur. Une partie distincte de la mémoire du calculateur contient des indicateurs binaires d'utilisateur, qui ont chacun deux états possibles : *armé* (vrai) ou *désarmé* (faux). Vous stockez la valeur *vrai* dans l'indicateur en l'armant, et la valeur *faux* en le désarmant. Vous pouvez tester la valeur d'un indicateur, qui renvoie alors l'indicateur numérique correspondant, 0 (faux) ou 1 (vrai), dans la pile opérationnelle.

Il y a 64 indicateurs binaires d'utilisateur, numérotés 1 à 64. Les indicateurs 1 à 30 sont à vocation générale et vous pouvez les utiliser à votre guise. Les indicateurs 31 à 64 ont des significations spéciales, définies ci-dessous — en les armant ou les désarmant, vous modifiez les modes qui leur sont associés.

Indicateurs binaires d'utilisateur à vocation spécifique

Nombre	Description	Valeur par défaut
31	Fonction LAST autorisée	Armé
32	Mode impression automatique	Désarmé
33	CR automatique	Désarmé
34	Valeur principale (intervalle de définition)	Désarmé
35	Evaluation symbolique de constantes	Armé
36	Evaluation symbolique de fonctions	Armé
37 - 42	Taille de mot pour les entiers binaires	Armé
43 - 44	Base de numération pour les entiers binaires	Désarmé
45	Affichage du niveau 1	Armé
46	Réservé	Désarmé
47	Réservé	Désarmé
48	Séparateur décimal	Désarmé
49 - 50	Format des nombres réels	Désarmé
51	Tonalité	Désarmé
52	Impression rapide	Désarmé
53 - 56	Nombre de chiffres décimaux	Désarmé
57	`Underflow` traité normalement	Désarmé
58	`Overflow` traité normalement	Désarmé
59	`Infinite Result` traité normalement	Armé
60	Angle	Désarmé
61	`Underflow-` traité comme une exception	Désarmé
62	`Underflow+` traité comme une exception	Désarmé
63	`Overflow` traité comme une exception	Désarmé
64	`Infinite Result` traité comme une exception	Désarmé

Erreurs et exceptions

Lorsqu'une erreur se produit, le calculateur émet une tonalité et affiche un message d'erreur sur la ligne supérieure de l'affichage. Les messages d'erreur sont décrits en annexe A, « Messages ».

Si l'erreur se produit pendant l'exécution d'une commande qui prend ses arguments dans la pile opérationnelle, ces arguments sont ramenés dans la pile si la fonction LAST est validée. Si la fonction LAST est invalidée, les arguments sont perdus.

Erreurs dans la ligne de commande

Une erreur peut se produire lorsque vous appuyez sur [ENTER], pendant que le calculateur traite le texte se trouvant dans la ligne de commande. Le calculateur émet alors une tonalité, affiche `Syntax Error` (erreur de syntaxe), rétablit la ligne de commande telle qu'elle était avant la pression sur [ENTER], et essaye d'indiquer quel est le problème. Si l'erreur provient d'une syntaxe incorrecte, le curseur se place à la fin du texte incorrect, qui est affiché en vidéo inverse. Si l'erreur provient d'une saisie incomplète, le curseur se place à la fin de la ligne.

Erreurs dans des programmes

S'il se produit une erreur dans un programme, le reste du programme (la partie non encore traitée) est abandonné. Si l'évaluation du programme avait commencé par un autre programme, le reste de ce programme est également abandonné.

Exceptions mathématiques

Certaines erreurs, qui peuvent se produire pendant des calculs ordinaires sur des nombres réels, sont considérées comme des *exceptions mathématiques*. Une exception peut fonctionner comme une erreur ordinaire et arrêter le calcul en cours, ou elle peut fournir un résultat par défaut, permettant ainsi au calcul de se poursuivre. Vous pouvez choisir comment les exceptions fonctionnent en armant ou désarmant les indicateurs binaires 57, 58 et 59. Le tableau ci-après décrit les exceptions mathématiques et les indicateurs binaires correspondants.

Exceptions mathémathiques

Exception	Description
`Infinite Result`	Cette exception se produit lorsqu'un calcul produit un résultat infini. Par exemple LN(0), TAN(90°), ou division par zéro. Si l'indicateur binaire 59 est armé*, les exceptions `Infinite Result` sont traitées comme des erreurs ordinaires. Si l'indicateur binaire 59 est désarmé, les exceptions `Infinite Result` donnent le résultat par défaut ±MAXR et arment l'indicateur binaire 64, qui est l'identificateur de `Infinite Result`.
`Overflow`	Cette exception se produit lorsqu'un calcul donne un résultat fini dont la valeur absolue est supérieure au plus grand nombre pouvant être représenté par le calculateur (MAXR). Par exemple 9E499 + 9E499, EXP(5000), ou FACT(2000). Si l'indicateur binaire 58 est armé, les exceptions `Overflow` sont traitées comme des erreurs ordinaires. Si l'indicateur binaire 58 est désarmé*, les exceptions `Overflow` donnent le résultat par défaut ±MAXR et arment l'indicateur binaire 63, qui est l'identificateur de `Overflow`.
`Underflow`	Cette exception se produit lorsqu'un calcul donne un résultat fini dont la valeur absolue est inférieure au plus petit nombre représentable par le calculateur (MINR). Par exemple 1E-499/2 et EXP(-5000). Si l'indicateur binaire 57 est armé, une exception `Underflow` est traitée comme une erreur ordinaire. Elle génère le message d'erreur `Negative Underflow` ou `Positive Underflow`, selon le signe du résultat du calcul. Si l'indicateur binaire 57 est désarmé*, une exception `Underflow` donne le résultat par défaut 0 et arme l'indicateur binaire 62, qui est l'identificateur de `Underflow+`, ou l'indicateur binaire 61, qui est l'identificateur de `Underflow-`, selon le signe du résultat du calcul.

* Choix par défaut.

2

Opérations de base

Ce chapitre montre comment introduire des « objets » en ligne de commande, comment créer, rappeler et effacer des variables, comment rappeler lignes de commande, piles et arguments précédents. Il indique comment traiter les situations où la quantité de mémoire est faible et comment effectuer des opérations sur le système.

Saisie des objets

Lorsque vous appuyez sur une touche pour commencer à saisir de nouveaux objets, le caractère correspondant à cette touche est introduit en *ligne de commande*. La ligne de ccommande peut contenir n'importe quel nombre d'objets, représentés sous forme de texte. Elle se trouve dans la partie inférieure de l'affichage, sous les libellés de menu, s'il y en a. La ligne de commande apparaît également lorsque les fonctions ■ EDIT ou ■ VISIT sont utilisées, ou pour visualiser ou modifier le contenu d'un objet.

Le contenu de la ligne de commande est traité par le calculateur lorsque vous appuyez sur ENTER, ou lorsque vous utilisez les touches ou les commandes qui exécutent automatiquement la fonction ENTER. Le contenu de la ligne de commande est évalué en tant que programme et la ligne de commande disparaît de l'écran.

Le nombre de caractères qui peuvent être introduits en ligne de commande n'a pas de limite. La ligne peut être partagée en plusieurs lignes en appuyant sur ■ NEWLINE, qui insère un caractère « nouvelle ligne » à l'endroit où se trouve le curseur. Ce caractère sépare les objets mais est ignoré lors de l'évaluation de la ligne de commande.

Si vous saisissez plus de 23 caractères en ligne de commande, des caractères disparaissent vers la gauche et trois points apparaissent à gauche de l'affichage pour indiquer leur existence. Si vous essayez alors de déplacer le curseur plus loin que l'extrémité gauche de l'affichage, les caractères disparus reviennent, se déplacent vers la droite et trois points de suspension apparaissent alors à la *droite* de l'affichage. Lorsque la ligne de commande comporte plusieurs lignes de texte, toutes ces lignes défilent ensemble vers la gauche ou vers la droite.

Saisie de nombres

Les nombres réels sont saisis grâce aux touches numériques, à la touche [CHS] et à la touche [EEX] pour former le nombre désiré.

Changer de signe : [CHS]. Le fait d'appuyer sur [CHS] change le signe du nombre figurant en ligne de commande. Si aucune ligne de commande n'est présente, le fait d'appuyer sur [CHS] exécute la commande NEG, qui change le signe de l'objet se trouvant au niveau 1. Cette commande est décrite sous « Arithmétique ».

Ce nombre peut être la mantisse ou l'exposant d'un nombre — la position du curseur détermine ce qui est changé. Si aucun signe n'est présent, un signe moins (−) est inséré au début du nombre. Si un signe plus (+) ou un signe moins sont présents, ils sont changés en leur opposé.

Si le curseur n'est pas placé sur un nombre valable, le fait d'appuyer sur [CHS] ajoute un signe moins à la ligne de commande.

Pour frapper un nombre négatif en tant que premier objet de la ligne de commande, il faut frapper un chiffre au moins avant d'appuyer sur [CHS]. Dans tous les autres cas, vous pouvez appuyer sur [CHS] avant, pendant, ou après avoir frappé le nombre.

Saisie d'exposant : [EEX]. Les nombres très grands ou très petits peuvent être saisis en notation scientifique. Un nombre peut être représenté par une *mantisse* et un *exposant*, la valeur du nombre étant alors le produit de la mantisse et de 10 élevé à la puissance de l'exposant.

Pour frapper un nombre en notation scientifique, frappez la mantisse, frappez [EEX] et ensuite frappez l'exposant. Le fait d'appuyer sur [EEX] ajoute le caractère `E` à la ligne de commande, séparant ainsi la mantisse de l'exposant.

Si le curseur n'est pas placé sur un nombre valable, ou s'il n'y a pas de ligne de commande, le fait d'appuyer sur la touche [EEX] ajoute le caractère `1E` à la ligne de commande. Si le curseur se trouve sur un nombre qui possède déjà un exposant, le fait d'appuyer sur [EEX] place le curseur sur le premier chiffre de l'exposant.

Espace arrière : [◀]

Le fait d'appuyer sur [◀] supprime le caractère à gauche du curseur, déplaçant d'un caractère vers la gauche le curseur et tous les caractères se trouvant à sa droite. Si vous maintenez l'appui sur [◀], l'action est répétée jusqu'à ce que vous relâchiez la touche. L'appui sur la touche [◀] n'a pas d'effet lorsque le curseur se trouve à l'extrémité gauche de la ligne.

Minuscules : [LC]

Il suffit d'appuyer sur [LC] (*Angl. "lower case"*) pour que les lettres du clavier s'inscrivent en minuscules sur la ligne de commande. La saisie des lettres continue en minuscules jusqu'à ce que vous appuyez une deuxième fois sur [LC], exécutiez ENTER ou appuyez sur [ON] pour effacer la ligne de commande.

Délimiteurs et séparateurs

Les différentes sortes d'objets sont saisis de la même manière qu'ils sont affichés. Plusieurs objets introduits consécutivement en ligne de commande doivent être séparés par l'un des signes suivants :

- Un délimiteur d'objet `(`, `)`, `[`, `]`, `{`, `}`, `#`, `"`, `'`, `«`, `»`.
- Un espace [SPACE] ou une nouvelle ligne [NEW LINE].
- Un point ou une virgule, selon ce qui n'est *pas* à ce moment le séparateur décimal. (Si l'indicateur binaire 48 est désarmé, les points sont les séparateurs décimaux et les virgules sont les séparateurs d'objets ; si l'indicateur binaire est armé, les virgules sont les séparateurs décimaux et les points les séparateurs d'objets.)

Pour les objets algébriques, les espaces ne sont pas pris en compte (sauf dans le cas des opérateurs AND, OR, XOR et NOT) et les arguments mis entre parenthèses (comme MOD(A,B)) doivent être séparés par le séparateur en cours, virgule ou point.

Lorsque vous effectuez la saisie d'un objet, la syntaxe doit être respectée. La plupart commencent et se terminent par des délimiteurs, comme dans « `"Le titre"` » ou dans `"m^2"`, et les vecteurs sont entourés de crochets, comme dans `[ 1 2 3 ]` ou dans `[ -5 6 7 -10,2 ]`. Le calculateur suit les mêmes règles de format lorsqu'il affiche un objet.

Le tableau ci-dessous est une version augmentée de celui qui figure au-dessus du clavier de gauche. Tous deux reprennent les délimiteurs appropriés aux différents objets.

Formats des objets

Objet	Format	Exemple
Nombre réel	*réel*	`-1,234E24`
Nombre complexe	(*réel* , *réel*)	`(1,23 , 4,56)`
Entier binaire	# *chiffres*	`# 123AF`
Chaîne	"*texte*"	`"HELLO"`
Vecteur réel	[*réel réel ...*]	`[1 2 3 4]`

Matrice réelle	[[*réel réel ...*]
	[*réel réel ...*]
	⋮
	[*réel réel ...*]]
		`[[1 2 3 4]`
		`[5 6 7 8]`
		`[9 0 1 2]`
		`[3 4 5 6]]`

Vecteur complexe	[(*réel* , *réel*) ...]	`[(1,2) (3,4)]`
Matrice complexe	[[(*réel* , *réel*) ...] [(*réel* , *réel*) ...] ⋮ [(*réel* , *réel*) ...]]	`[[(1,2) (3,4)]` `[(5,6) (7,8)]]`
Liste	{ *objet objet ...* }	`{1 "Titre" (1,2)}`

Formats des objets (*suite*)

Objet	Format	Exemple
Nom	'*nom*'	'MICHEL'
Nom local	'*nom*'	'MICHEL'
Programme	« *objet objet ...* »	« DUP 4 ROLL »
Expression	'*expression*'	'A+B'
Equation	'*expression=expression*'	'A+B=SIN(X)'

Tout délimiteur manquant en fin de ligne de commande est automatiquement ajouté lors de l'appui sur [ENTER].

Le curseur indique le mode

La forme du curseur indique le mode de saisie en cours ainsi que le choix du mode — insertion ou mode remplacement. (Vous trouverez une description des modes de saisie dans le paragraphe suivant, puis une description du menu du curseur, qui traite des modes insertion/remplacement). Le tableau ci-dessous indique les six combinaisons possibles des modes de saisie avec les modes insertion ou remplacement.

	Mode insertion	Mode remplacement
Mode exécution	▷	□
Mode algébrique	▷	▤
Mode alpha	▷	▩

Modes de saisie

Il y a trois modes de saisie des différents types d'objets. En général, c'est le *mode exécution* qui est utilisé pour saisir les données, le *mode algébrique* pour les noms et le *mode alpha* (ou mode *alphanumérique*) pour la saisie des programmes et des « chaînes » de mots. Le mode *alpha* peut être activé ou désactivé par simple appui sur [α] ; dans certains cas, le mode de saisie change automatiquement lorsque vous entamez la saisie d'un nouvel objet.

Le mode de saisie en cours affecte le fonctionnement des touches associées aux commandes — il décide si l'appui sur une touche provoque l'exécution de la commande ou si le *nom* de la commande est simplement ajouté en ligne de commande. Nous entendrons ici par le mot « touche » les frappes simples ou préfixées (« *shiftées* ») — consécutives à l'appui sur la touche préfixe rouge « *Shift* » - telles que [Shift] [π] - et les touches de menu, telles [SIN].

Les touches suivantes ne seront *pas* affectées par le mode de saisie en cours :

- Les touches représentant des opérations non programmables, telles [ENTER], [Shift] [CATALOG], ou [+ML]. Le fait d'appuyer sur une touche d'opération provoque toujours l'exécution de cette opération.
- Toutes les touches portant un seul caractère sur le clavier de gauche ajoutent simplement le caractère en ligne de commande. Toutefois, certaines touches, comme [Shift] [>], bien qu'elles correspondent à des noms de fonctions, se comportent comme des *touches de caractères.*
- Les touches de caractère [0] à [9], [.], [,] et [Shift] [π], sur le clavier de droite. Le fait d'appuyer sur une touche de caractère ajoute dans tous les cas ce caractère en ligne de commande.
- [CLUSR] dans le menu USER, [HALT] dans le menu PROGRAM CONTROL (CTRL) et n'importe quelle touche dans le menu PROGRAM BRANCH. L'appui sur l'une de ces touches ajoute toujours le nom de la commande en ligne de commande.

Outre les touches de commande, les touches de menu assignées aux noms de variables dans le menu USER sont affectées par le mode de saisie en cours. Les paragraphes suivants décrivent chaque mode de saisie et son action sur chaque type de touche. Les touches concernées se trouvent toutes sur le clavier de droite et sont principalement les touches de menu.

Mode exécution. C'est le mode de saisie par défaut — une nouvelle ligne de commande commence en général dans ce mode. Le curseur prend la forme □ ou ◊. En mode exécution :

- Le fait d'appuyer sur une *touche de commande* (telle que [STO]) provoque l'exécution de la commande.
- Le fait d'appuyer sur une *touche de fonction* (telle que [+]) provoque l'exécution de la fonction.
- Le fait d'appuyer sur une touche de variable dans le menu USER provoque l'évaluation du nom de la variable.

Pour éviter les frappes inutiles, la plupart des commandes exécutent ENTER avant d'exécuter la commande. Les exceptions sont STD, DEG et RAD dans le menu MODE et DEC, HEX, OCT et BIN dans le menu BINARY, qui provoquent l'exécution de leur commande sans provoquer l'exécution de ENTER — c'est-à-dire, sans affecter la ligne de commande.

Mode de saisie algébrique. L'appui sur ['] pour commencer un nom ou une expression algébrique active le mode *algébrique*. Le curseur prend la forme ▯ ou ⌖. L'appui sur ['] une seconde fois active à nouveau le mode exécution. En mode algébrique :

- L'appui sur une touche de commande provoque son exécution tout comme en mode exécution.
- L'appui sur une touche de fonction ajoute le nom de la fonction en ligne de commande. Si les arguments de la fonction sont entre parenthèses, comme dans SIN(X), l'ouverture de la parenthèse est également ajoutée.
- L'appui sur une touche de variable dans le menu USER ajoute le nom de la variable, sans guillemets, en ligne de commande.

Mode de saisie alpha. Le fait d'appuyer sur [«] ou sur ["] pour commencer un programme ou une chaîne de caractères alpha active le mode de saisie alpha, indiqué par le témoin **α** et la forme que prend le curseur (▒ ou ⌖). En mode exécution ou en mode algébrique, le fait d'appuyer sur [α] active le mode de saisie alpha ; un second appui sur [α] rétablit le mode précédent. A tout moment il est possible d'appuyer sur [α LOCK] pour « verrouiller » le mode de saisie alpha. Pour en sortir, appuyez sur [α].

En mode de saisie alpha :

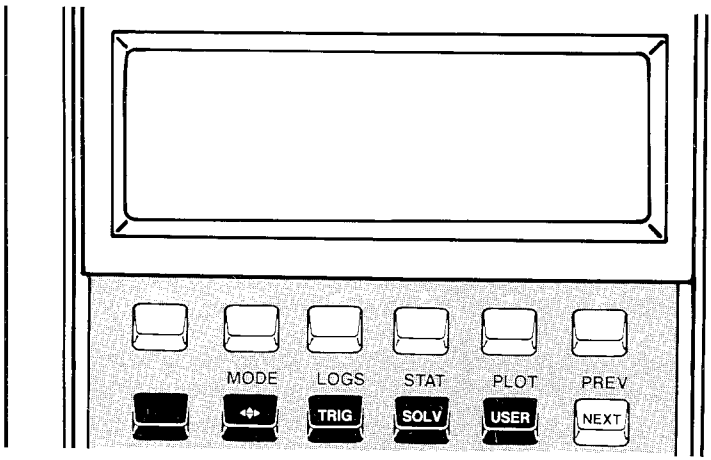
- L'appui sur une touche de commande ajoute le nom de la commande en ligne de commande.
- L'appui sur une touche de fonction ajoute le nom de cette fonction en ligne de commande.
- L'appui sur une touche de variable en menu USER ajoute le nom de la variable, sans guillemets, en ligne de commande.

Si le curseur est en mode insertion ou s'il est placé en fin de ligne de commande lorsque vous appuyez sur l'une de ces touches, un espace est ajouté au début et à la fin du texte ajouté afin de séparer les commandes.

Le menu du curseur : [◂✥▸]

Le fait d'appuyer sur [◂✥▸] active le menu de commande du curseur. Aucun libellé de menu n'apparaît et les touches de menu prennent les fonctions inscrites en blanc au-dessus d'elles. Le menu du curseur comporte des possibilités de modification plus étendues que l'espace arrière ([◂]). Le fait d'appuyer une seconde fois sur [◂✥▸] rétablit le menu précédent et ses libellés réapparaissent dans l'affichage.

Le menu du curseur contient des opérations primaires accessibles directement et des opérations secondaires accessibles par l'intermédiaire de la touche préfixe rouge (« Shift »). Les fonctions primaires sont indiquées en blanc au-dessus des touches de menu, comme illustré ci-dessous.



Le tableau suivant décrit les opérations primaires associées aux touches de menu du curseur. Si l'appui sur ces touches est maintenu, l'action de la touche est répétée jusqu'au moment où la touche est relâchée (sauf pour [INS]).

Opérations primaires du menu du curseur

Touche	Opération
[INS]	Passage du mode remplacement au mode insertion. En mode de remplacement, les nouveaux caractères remplacent les caractères existants ; le curseur prend la forme □, ▯ ou ■. En mode insertion, les nouveaux caractères sont insérés entre les caractères existants ; le curseur prend la forme ◇, ◈ ou ◆.
[DEL]	Suppression du caractère se trouvant sous le curseur.
[▲]	Passage du curseur à la ligne précédente.
[▼]	Passage du curseur à la ligne suivante.
[◀]	Déplacement du curseur d'un espace vers la gauche.
[▶]	Déplacement du curseur d'un espace vers la droite.

Le tableau suivant décrit les opérations secondaires associées avec les touches de menu du curseur. Sauf pour ■[INS], ces opérations sont équivalentes à des répétitions d'opérations primaires (frappes n'utilisant *pas* la touche préfixe rouge).

Opérations secondaires du menu du curseur

Touche	Opération
■[INS]	Suppression de tous les caractères à la gauche du curseur.
■[DEL]	Suppression du caractère se trouvant sous le curseur et de tous les caractères à sa droite.
■[▲]	Passage à la première ligne de la ligne de commande.
■[▼]	Passage à la dernière ligne de la ligne de commande.
■[◀]	Déplacement du curseur vers l'extrémité gauche de la ligne de commande.
■[▶]	Déplacement du curseur vers l'extrémité droite de la ligne de commande.

Saisie de la ligne de commande : ENTER

Le fait d'appuyer sur ENTER provoque l'évaluation de la ligne de commande. Si celle-ci est vide, le fait d'appuyer sur ENTER exécute la commande DUP qui reproduit le contenu du niveau 1. DUP est décrite sous la rubrique « STACK ».

Pour évaluer la ligne de commande, ENTER doit analyser la syntaxe du texte qui s'y trouve, pour créer des objets, combiner des objets en un programme, puis évaluer le programme. Voici ce qui se passe lorsque vous appuyez sur ENTER pour évaluer une ligne de commande :

1. Le témoin ((•)), annonçant que le calculateur est occupé, apparaît.
2. Si UNDO est actif, une copie de la pile opérationnelle en cours est sauvegardée.
3. La chaîne alpha figurant en ligne de commande est examinée et partagée en sections, en fonction de l'emplacement des délimiteurs et séparateurs.
4. La syntaxe de chaque section est vérifiée, le type d'objet est identifié et l'objet correspondant est placé dans la pile.
5. Si COMMAND est active, une copie de la ligne de commande est sauvegardée dans la pile des commandes.
6. Les objets de la pile sont combinés en un seul programme, qui est ensuite évalué.
7. Le témoin ((•)) disparaît.

Si la syntaxe d'une section de la chaîne n'est pas jugée correcte par la vérification de l'étape n° 4, les mots `Syntax Error` sont affichés. Les objets placés dans la pile par ENTER sont abandonnés et la ligne de commande est rétablie. La partie de texte incorrecte est mise en valeur en « vidéo inverse » et le curseur la suit immédiatement. Si c'est une syntaxe incomplète qui a causé l'erreur, le curseur est placé en fin de ligne.

Visualiser des objets : ■ VIEW↑, ■ VIEW↓

Si un objet est trop grand pour figurer sur la ligne d'affichage, vous pouvez visualiser les parties figurant en dehors de l'affichage en utilisant les touches ■ VIEW↑ et ■ VIEW↓. ■ VIEW↑ déplace la fenêtre d'affichage d'une ligne vers le haut et ■ VIEW↓ la déplace d'une ligne vers le bas. Ces deux touches, ■ VIEW↑ et ■ VIEW↓, peuvent être répétées et même utilisées durant la saisie ou la correction d'un objet.

Correction d'objets existants

Un objet peut être replacé en ligne de commande. Il peut alors être examiné à loisir et modifié en utilisant les opérations normales de correction en ligne de commande. EDIT provoque le renvoi en ligne commande de l'objet figurant au niveau 1. VISIT replace en ligne de commande un objet se trouvant à d'autres niveaux de la pile — ou en mémoire utilisateur.

Correction du niveau 1 : ■ EDIT

EDIT agit exactement à l'inverse de ENTER, puisqu'elle retire un objet du niveau 1 et le remet en ligne de commande. Plus précisément, elle agit de manière suivante :

1. Une copie de l'objet se trouvant au niveau 1 est mise sous forme de texte et placée en ligne de commande.
2. Le mode de saisie alpha est activé.
3. Le menu du curseur est activé pour permettre la correction.

L'objet de départ, en niveau 1, est mis en évidence pour rappeler qu'il s'agit bien d'une correction de cet objet et que l'original est toujours intact.

Lorsque la version « texte » de cet objet se trouve en ligne de commande, vous pouvez la modifier à loisir. Lorsque la correction est terminée, vous pouvez :

- Appuyer sur ON pour annuler cette correction, effacer la ligne de commande et laisser inchangée la version de l'objet se trouvant au niveau 1.
- Appuyer sur ENTER (ou sur une autre touche qui exécute ENTER) pour *remplacer* l'original du niveau 1. Plus exactement, l'objet du niveau 1 est rejeté et la ligne de commande est évaluée.

Si le menu du curseur est encore actif au moment où vous achevez la correction, le menu précédent est rétabli.

Correction d'une variable ou d'un niveau de la pile opérationnelle : ■ VISIT

VISIT est une version étendue de EDIT. Elle permet de visualiser ou de corriger un objet stocké dans une variable ou placé à un niveau supérieur au niveau n° 1.

Correction d'une variable. Pour corriger un objet stocké dans une variable, placez la variable en niveau 1 et appuyez sur ■ VISIT. L'objet stocké est copié en ligne de commande, le mode de saisie alpha est activé et le menu du curseur est activé.

Correction d'un niveau de la pile opérationnelle. Pour corriger l'objet figurant au niveau *n*, placez *n* au niveau 1 de la pile et appuyez sur ■ VISIT. L'objet est copié en ligne de commande, le mode de saisie alpha est activé et le menu du curseur est également activé. L'objet — sa version originale — est mis en « vidéo inverse » pour vous rappeler que vous effectuez une correction et que l'original est toujours sauvegardé.

On quitte VISIT de la même façon que l'on quitte EDIT :

- Appuyez sur ON pour annuler la correction, effacer la ligne de commande et laisser intacte la version originale.
- Appuyez sur ENTER (ou une touche qui exécute ENTER) pour remplacer l'objet original. Plus exactement, la ligne de commande est évaluée et l'objet résultant au niveau 1 remplace l'objet de départ.

Si le menu du curseur est toujours actif lorsque vous terminez la correction, le menu précédent est rétabli.

Evaluation d'objets

EVAL — ***Evaluation de l'objet*** — **Commande**

Niveau 1	
objet ➡	

EVAL évalue l'objet placé au niveau 1. Le résultat de cette évaluation, y compris les résultats renvoyés vers la pile, dépend de l'objet évalué. Le processus d'évaluation est décrit en détail dans le chapitre précédent, « Principes généraux ». L'évaluation de fonctions est déterminée par l'indicateur 36, qui définit le mode d'évaluation symbolique ou numérique. Voyez la partie intitulée « ALGEBRA ».

→NUM *Evaluation numérique* Commande

Niveau 1		Niveau 1
objet	➡	*z*

→NUM est identique à EVAL, sauf qu'il établit temporairement le mode d'évaluation numérique (décrit dans « ALGEBRA ») pour assurer que les fonctions renvoient des valeurs numériques. Le mode d'évaluation en cours est rétabli lorsque →NUM s'achève.

SYSEVAL *Evaluer objet système* Commande

Niveau 1		
# n	➡	

SYSEVAL est seulement destiné à la programmation d'applications Hewlett-Packard. L'utilisation généralisée de SYSEVAL peut corrompre la mémoire du calculateur ou provoquer des pertes de données en mémoire. *N'utilisez SYSEVAL que dans les limites spécifiées par les applications définies par Hewlett-Packard.*

SYSEVAL évalue l'objet système à l'adresse absolue # *n*. Vous pouvez afficher le numéro de version de votre HP-28C/S en exécutant `# 10 SYSEVAL` (en supposant une base DEC, qui est la *base par défaut*).

Noms

Les noms peuvent compter jusqu'à 127 caractères, bien qu'en pratique il soit préférable d'utiliser des noms ne dépassant pas cinq ou six caractères. Le premier caractère doit être une lettre. La distinction entre minuscules et majuscules existe de manière interne, mais les minuscules apparaissent sous forme de majuscules dans le menu USER. Les noms de commandes du HP-28C/S ne peuvent être utilisés comme noms de variables.

Les caractères autorisés sur le clavier sont les lettres, les chiffres et les caractères ?, Σ, π, →, μ et ". Les caractères suivants ne peuvent faire partie des noms de variables :

- Délimiteurs d'objets (#, ⌊, ⌋, ", ', ⟨, ⟩, (,), «, »).
- Symboles d'opérateurs algébriques (+, −, ×, /, ^, √, =, <, >, ≤, ≥, ≠, ∂, ∫).
- Séparateur en cours (. ou ,).

Noms réservés

Les noms suivants sont réservés à des utilisations spéciales :

- EQ fait référence à l'équation en cours utilisée par les commandes de SOLV et PLOT.
- ΣPAR fait référence à une liste de paramètres utilisés par des commandes statistiques.
- PPAR fait référence à une liste de paramètres utilisés par des commandes d'ajustement de courbes.
- ΣDAT fait référence au tableau statistique en cours.
- s1, s2 etc. sont créées par ISOL et QUAD pour représenter des signes arbitraires obtenus dans les solutions symboliques.
- n1, n2 etc. sont créées par ISOL et QUAD pour représenter des entiers arbitraires obtenus dans les solutions symboliques.

Vous pouvez utiliser n'importe lequel de ces noms dans vos programmes personnels, mais souvenez-vous que certaines commandes utilisent ces noms comme arguments implicites.

Noms entre guillemets/sans guillemets

Vous pouvez saisir un nom en ligne de commande avec ou sans guillemets, selon la manière dont vous désirez que le nom soit évalué.

Noms entre guillemets. Le fait de saisir un nom entre guillemets signifie « Placez ce nom dans la pile opérationnelle ». Un nom entre guillemets est placé dans la pile, mais n'est pas évalué lorsque la ligne de commande est évaluée.

Noms sans guillemets. Le fait de saisir le nom *sans* guillemets signifie « Evaluez l'objet qui porte ce nom ». Un nom sans guillemets est placé dans la pile opérationnelle, puis évalué lorsque la ligne de commande est évaluée.

Duplication de noms

Il est normalement impossible de créer deux variables portant le même nom. Toutefois certaines commandes (DRAW, ∫, QUAD, TAYLR) créent une variable-utilisateur temporaire dont le nom est le même que celui de l'argument spécifié pour la commande. Lorsque la commande achève son exécution, elle supprime cette variable temporaire. Si cependant la commande est abandonnée en cours d'exécution par une réinitialisation-système (ON ▲) ou par une erreur `Out of Memory`, la variable temporaire demeure en mémoire utilisateur.

Si elle reste en mémoire et si vous avez précédemmment créé une variable du même nom, deux variables du même nom se trouveront ensemble dans le menu-utilisateur. Utilisez PURGE pour effacer le nom en excédent ; PURGE effacera la variable créée le plus récemment, dans ce cas la variable temporaire.

Création, rappel et élimination de variables

Une variable est la combinaison d'un nom et d'un autre objet, sauvegardés ensemble en mémoire utilisateur. Le nom représente le nom de la variable ; l'autre objet est la valeur ou contenu de la variable.

Cette section vous indique comment créer un objet, comment rappeler le contenu d'une variable dans la pile opérationnelle sans évaluation et comment éliminer une variable.

STO ***Stocker*** **Commande**

Niveau 2	Niveau 1	
objet	*'nom'*	➡

Cette commande crée une variable dont le nom est *nom* et dont la valeur est *objet*. Les évaluations de *nom* mettent *objet* sur la pile et si *objet* est un nom ou un programme, l'évaluent.

RCL ***Rappeler*** **Commande**

Niveau 1		Niveau 1
'nom'	➡	*objet*

Cette commande recherche dans la mémoire-utilisateur le *nom* de la variable et renvoie son contenu, *objet*. L'objet n'est pas évalué.

Ceci est une différence importante entre RCL (*Angl. recall*) et EVAL (*evaluate*). RCL demande un nom de variable comme argument et renvoie le contenu de la variable. EVAL accepte n'importe quel objet comme argument et évalue l'objet en fonction des règles qui concernent cet objet. RCL et EVAL n'ont le même effet que lorsque l'argument est un nom qui réfère à des données, à une expression algébrique ou à une variable locale. Dans ces cas, tous deux renvoient l'objet stocké vers la pile opérationnelle.

Le tableau suivant est un résumé des conséquences de l'éxécution d'EVAL et de RCL (le nom `'ABC'` se trouvant en niveau 1) pour différentes valeurs de la variable associée, ABC.

Comparaison entre EVAL et RCL

Si 'ABC' contient :	'ABC' EVAL :	'ABC' RCL :
(non défini)	Renvoie 'ABC'.	Provoque l'erreur Undefined Name
Un nom 'DEF'.	Evalue 'DEF'.	Renvoie 'DEF'.
Un programme.	Evalue le programme.	Renvoie le programme.
Tout autre objet.	Renvoie l'objet.	Renvoie l'objet.

PURGE ***Eliminer*** **Commande**

Niveau 1	
'nom'	➡
{ nom$_1$ nom$_2$ ... }	➡

PURGE supprime une ou plusieurs variables de la mémoire-utilisateur. Si l'argument est un nom, PURGE supprime la variable correspondante. Si l'argument est une liste de noms, PURGE supprime chacune des variables nommées.

Récupération de données après erreur

Le HP-28C/S effectue automatiquement des copies de la ligne de commande, de la pile et des arguments. Ces copies vous permettent de « sortir » d'une situation d'erreur, c'est-à-dire de revenir à la situation antérieure à l'erreur : un calcul peut être recommencé sans avoir à repartir depuis le début. Les copies de la ligne de commande et des arguments sont également pratiques pour répéter certains calculs.

Ces copies utilisent une quantité substantielle de mémoire. Il vous est possible, pour cette raison, de choisir d'éviter la sauvegarde de la ligne de commande, de la pile des opérations ou des arguments, grâce au menu MODE. Dans ce paragraphe, nous considérons que tous les dispositifs de récupération de données après erreur sont activés, tout comme ils le sont après une réinitialisation.

Les opérations de récupération de données sont ■ [COMMAND], qui rappelle des copies de la ligne de commande, ■ [UNDO], qui rappelle des copies de la pile opérationnelle et ■ [LAST], qui rappelle les derniers arguments utilisés.

Récupération de données en ligne de commande : ■ [COMMAND]

Chaque exécution de ENTER effectue une copie de la ligne de commande. Un maximum de quatre lignes de commande sont stockées dans la *pile des commandes*. Le fait d'appuyer une seule fois sur ■ [COMMAND] rappelle la dernière ligne de commande — celle qui a été sauvegardée le plus récemment. Un second appui sur ■ [COMMAND] rappelle la ligne de commande précédente, et ainsi de suite. Si vous appuyez sur ■ [COMMAND] plus de quatre fois, la séquence reprend avec la ligne de commande la plus récente.

Récupération de la pile opérationnelle : ■ [UNDO]

Chaque appui sur la touche ENTER provoque une sauvegarde de la pile opérationnelle avant évaluation de la ligne de commande. L'appui sur ■ [UNDO] provoque l'effacement de la pile opérationnelle en cours et son remplacement par la version sauvegardée. UNDO rétablit la version qui existait avant d'appuyer sur [ENTER] (ou la touche qui provoque l'exécution de ENTER), mais elle n'affecte pas les changements qui s'étaient produits au niveau des indicateurs binaires définis par l'utilisateur ou de la mémoire utilisateur.

Lorsqu'un programme est suspendu, la fonction UNDO (y compris +UND, -UND et UNDO elle-même) est associée avec l'environnement suspendu. UNDO rétablira la pile opérationnelle existant avant le dernier ENTER, mais après que le programme ait été suspendu. Après la continuation d'un programme et son exécution complète, UNDO fournira une référence de la pile sauvegardée avant l'exécution du programme.

Récupération des derniers arguments

LAST ***Derniers arguments*** **Commande**

	Niveau 3	Niveau 2	Niveau 1
➧			$objet_1$
➧		$objet_1$	$objet_2$
➧	$objet_1$	$objet_2$	$objet_3$

Les commandes qui prennent des arguments dans la pile opérationnelle effectuent des copies de sauvegarde de ces objets. Le fait d'exécuter ■ LAST renvoie les objets les plus récemment sauvegardés par une commande. Les objets sont renvoyés aux niveaux de la pile opérationnelle qu'ils occupaient précédemment. Les commandes qui ne prennent pas d'arguments laissent les arguments sauvegardés inchangés.

Notez que lorsque LAST suit une commande qui évalue des procédures (telle que $\int$, ∂, ISOL, EVAL, ROOT etc.), les derniers arguments sauvegardés viennent de la procédure et non de la commande originale.

Si la mémoire s'épuise

Le HP-28C contient environ 2 048 octets de mémoire utilisateur, dont 400 sont réservés à l'usage du système, ce qui laisse environ 1 650 octets pour utilisation générale. Le HP-28S, lui, laisse à l'utilisateur environ 3 200 octets. Pratiquement, chaque opération disponible sur le HP-28C/S demande une certaine quantité de mémoire utilisateur, même l'interprétation de la ligne de commande. La quantité de mémoire utilisée par certaines commandes d'algèbre (COLCT, EXPAN, TAYLR) augmente rapidement au fur et à mesure que leurs arguments deviennent plus compliqués.

Pour utiliser le HP-28C/S de manière rationnelle, gardez présent à l'esprit qu'il est un *calculateur* conçu pour la résolution interactive de problèmes. Sa puissance réside dans ses opérations intégrées, non dans sa capacité à stocker des bases de données ou des bibliothèques de programmes. La sagesse recommande de laisser au moins quelques centaines d'octets libres pour utilisation par le système.

A cause du fait que le système d'exploitation du HP-28C/S partage la mémoire avec les objets définis par l'utilisateur, il est possible d'emplir la mémoire d'un nombre d'objets utilisateur si important que le fonctionnement normal du calculateur en soit compromis. Le HP-28C/S avertit l'utilisateur de l'épuisement des réserves d'espace-mémoire. Ces avertissements sont de sévérité croissante — au fur et à mesure de la baisse de la quantité de mémoire disponible :

1. `Insufficient Memory` *(mémoire insuffisante)*
2. `No Room for UNDO` *(pas assez de mémoire pour* UNDO*)*
3. `No Room to ENTER` *(pas assez de mémoire pour* ENTER*)*
4. `Low Memory!` *(plus beaucoup de mémoire)*
5. `No Room to Show Stack` *(pas assez de mémoire pour montrer la pile)*
6. `Out of Memory` *(plus de mémoire disponible)*

Insufficient Memory

S'il n'y a pas assez de mémoire pour permettre l'exécution d'une commande, la commande s'interrompt et la calculateur affiche : `Insufficient Memory`. Si LAST est activé, les arguments d'origine sont renvoyés à la pile. S'il est désactivé, les arguments sont perdus.

No Room for UNDO

Supposons que UNDO soit désactivé et que la pile opérationnelle comporte une matrice de 11 × 11. Même une simple opération comme NEG ne pourra être effectuée, parce qu'il n'y a pas en mémoire suffisamment d'espace pour la matrice d'origine et la matrice résultat. Dans de tels cas, une erreur `No Room for UNDO` se produit, ce qui désactive automatiquement UNDO. Vous pouvez alors ré-essayer l'opération qui a échoué, et ré-activer UNDO ultérieurement.

No Room to ENTER

S'il n'y a pas assez de mémoire disponible pour traiter la ligne de commande, le calculateur efface la ligne et affiche `No Room to ENTER`. Une copie de la ligne de commande est effectuée dans la pile des commandes si la pile des commandes est activée.

Si vous essayez de corriger un objet existant avec EDIT ou VISIT et si une copie de la ligne de commande qui n'a pu être traitée est sauvegardée dans la pile des commandes, éliminez la version originale de l'objet, appuyez sur ▮ [COMMAND] pour récupérer la ligne de commande contenant l'objet corrigé et appuyez sur [ENTER] pour introduire la version corrigée.

Low Memory!

S'il reste moins de 128 octets d'espace-mémoire, `Low Memory!` apparaît une seule fois sur la ligne supérieure de l'affichage. Le message apparaîtra lors de chaque frappe jusqu'à ce que la quantité d'espace-mémoire augmente. Effacez les objets non indispensables avant de continuer vos calculs.

No Room To Show Stack

Il est parfois possible que le HP-28C/S puisse terminer toutes les opérations en cours et n'aie cependant pas assez de mémoire pour permettre l'affichage normal de la pile opérationnelle. Dans ce cas, le calculateur affiche `No Room to Show Stack` sur la ligne supérieure de l'affichage. Les lignes de l'affichage qui, normalement, affichent les objets de la pile, n'indiquent plus ces objets que par leur type, par exemple `Real Number` pour un objet algébrique, `Algebraic` pour un nombre réel, etc.

La quantité de mémoire disponible pour afficher un objet de la pile varie selon le type d'objet. Les objets algébriques sont ceux qui, en général, nécessitent le plus de mémoire. Effacez l'un ou l'autre des objets en mémoire ou stockez un objet de la pile en tant que variable pour qu'il ne soit pas affiché.

Out of Memory

Le cas le plus grave est celui où le manque d'espace-mémoire est tellement aigu qu'il n'est plus possible au calculateur d'effectuer la moindre opération. La pile ne peut plus être affichée, les libellés de menu restent invisibles, il est devenu impossible de constituer une ligne de commande. Dans ce cas, il *faut* effacer une partie quelconque de la mémoire pour pouvoir continuer. Une processus `Out of Memory` est activé et il crée un affichage particulier :

Le calculateur vous demande successivement d'effacer :

1. La pile des commandes (s'il y a lieu).
2. La pile UNDO (s'il y a lieu).
3. Les arguments LAST (s'il y a lieu).
4. La pile opérationnelle.
5. Chaque variable utilisateur, désignée par son nom.

Pour chaque article que vous désirez effacer, appuyez sur la touche de menu marquée YES. Pour ceux que vous désirez conserver, appuyez sur celle qui est marquée NO. Après avoir appuyé sur YES au moins une fois, vous pouvez mettre fin au processus `Out of Memory` en appuyant sur ATTN. S'il reste assez de mémoire, le calculateur renvoie l'affichage normal. Sinon, le calculateur émet une tonalité et continue de proposer la succession d'effacements. Après avoir parcouru un cycle complet de choix, le processus `Out of Memory` tente de revenir au fonctionnement normal. Si l'espace-mémoire continue à manquer, le processus reprend la séquence des choix.

Opérations système

Certaines séquences de frappe interrompent le fonctionnement normal du HP-28C/S pour effectuer des opérations système. Elles comprennent l'ajustement du contraste de l'affichage, l'interruption de bouclages de programmes qu'il est impossible d'arrêter par la touche [ON], la réinitialisation ou l'exécution d'un test électronique du système.

Attention : [ON]

L'appui sur [ON] efface la ligne de commande et affiche la pile opérationnelle. Si une procédure est en cours d'exécution, l'appui sur [ON] l'arrête. Le résultat est similaire à l'exécution de ABORT (décrite dans « PROGRAM CONTROL »).

Le fait d'appuyer sur [■][OFF] éteint le HP-28C/S. Lorsque vous appuyez à nouveau sur [ON], le HP-28C/S reprend les opérations qu'il effectuait au moment où il a été éteint. Le HP-28C/S s'éteindra si vous ne vous en servez pas pendant plus de 10 minutes.

Contrôle du contraste : [ON][+], [ON][−]

Vous pouvez modifier le contraste de l'affichage du HP-28C/S de la manière suivante :

1. Appuyez sur la touche [ON] et maintenez l'appui.
2. Appuyez sur [+] pour augmenter le contraste ou sur [−] pour le diminuer. Tant que vous maintenez la pression sur la touche [ON], vous pouvez librement appuyer sur [+] ou sur [−] jusqu'à ce que vous obteniez le contraste qui vous convient.
3. Relâchez la touche [ON].

Arrêt du système : ON ▲

Pour exécuter un *arrêt système,* appuyez sur les touches ON et ▲, puis relâchez-les. Voici les conséquences d'un arrêt-système :

- Arrêt de toutes les exécutions de commandes ou de procédures.
- Effacement de toutes les variables locales.
- Effacement de la pile opérationnelle.
- Activation du menu du curseur.
- Rétablissement du fonctionnement normal du clavier.

L'utilisation la plus commune d'un arrêt système est l'arrêt d'une boucle lors de l'évaluation d'un nom. Souvenons-nous que l'évaluation d'un nom qui en référence un autre provoque l'évaluation du second nom. Comme le second nom renvoie au premier, la boucle sans fin est amorcée. Par exemple :

```
'Y' 'X' STO  'X' 'Y' STO X
```

provoque une boucle sans fin. L'évaluation d'un nom est capitale pour l'algèbre symbolique et le processus se déroule à très grande vitesse et ne peut être arrêté par la touche ON. Un arrêt système est nécessaire.

Réinitialisation : ON INS ▶

Pour effacer et remettre à zéro la totalité de la mémoire du HP-28C/S :

1. Appuyez sur ON et maintenez la pression.
2. Appuyez sur INS et sur ▶ et maintenez la pression.
3. Relâchez INS et ▶.
4. Relâchez ON.

Une réinitialisation a les effets suivants :

- Arrêt de toute exécution de commande ou de procédure.
- Effacement de toute variable locale.
- Suppression de toutes les variables-utilisateur.
- Effacement de la pile des résultats.
- Remise des indicateurs binaires à leurs valeurs par défaut.
- Activation du menu du curseur.
- Emission d'une tonailté et affichage du message `Memory Lost` en première ligne.
- Rétablissement du fonctionnement normal du clavier.

Annuler une réinitialisation : [ON] [DEL]

Si vous lancez un arrêt du système ([ON] [▲]) ou une réinitialisation ([ON] [INS] [▶]), l'arrêt système ou la réinitialisation ne prennent effet que lorsque vous relâchez la touche [ON]. Il est possible d'annuler le processus, avant de relâcher la touche [ON], de la façon suivante :

1. Relâchez toutes les touches, sauf [ON].
2. Appuyez sur [DEL], puis relâchez cette touche.
3. Relâchez la touche [ON].

Test système : [ON] [▼], [ON] [◀]

Le HP-28C/S comporte plusieurs tests du fonctionnement électronique pour vérifier sa fabrication et aider les ingénieurs de service après-vente à diagnostiquer les pannes. Les tests sont déclenchés en appuyant simultanément sur [ON] et sur [▼], puis en les relâchant. Chaque fois que le calculateur termine un test, comme il l'indique en renouvelant l'affichage, vous pouvez passer au suivant en appuyant sur n'importe quelle touche.

Au moment où l'affichage indique KEYBOARD TEST, appuyez sur les touches du clavier de gauche, de [A] à [F], puis de [G] à [L], puis de [M] à [R], etc. Lorsque vous aurez terminé le clavier de gauche, testez celui de droite en commençant par [INS].

Si le calculateur réussit tous ces tests, il affiche OK-28C ou OK-28S.

Vous pouvez lancer des tests répétitifs en appuyant simultanément sur [ON] et sur [◀]. Le calculateur répète la série complète de tests, jusqu'à ce que vous appuyez sur une touche. Il affiche alors FAIL, indiquant par là que le test a été interrompu.

Lors de leur exécution, les tests système effectuent aussi un arrêt système ([ON][▲]).

Dictionnaire

Toutes les opérations du HP-28C/S (sauf la commande SYSEVAL, qui est particulière) sont accessibles par le clavier. Les opérations fréquemment utilisées comme [ENTER], [+] et ■[√] sont toujours accessibles directement par une touche qui porte leur nom. Toutes les autres opérations sont accessibles par l'intermédiaire des *touches de menu* — qui sonnt les six touches sans libellé, en haut du clavier de droite (les variables que vous avez créées sont également accessibles par l'intermédiaire des touches de menu, comme décrit dans « SOLVE » et « USER ».)

Menus

La fonction que prend à un moment donné une touche de menu apparaît dans le *libellé de menu* correspondant, affiché en vidéo inverse directement au-dessus de la touche. Si aucun libellé de menu n'est affiché, c'est le menu du curseur qui est actif. Ce menu, qui sert aux opérations de déplacement du curseur, d'insertion et de suppression, est décrit au chapitre « Opérations de base ».

Les opérations sont regroupées par *menus* en fonction d'une application commune, la trigonométrie par exemple, ou en fonction d'un type d'argument commun, les tableaux par exemple. Vous choisissez un menu en appuyant sur une *touche de sélection de menu*, comme [TRIG] ou ■[ARRAY]. Chaque menu consiste en un ou plusieurs *jeux de libellés* (un jeu de libellés est l'ensemble des six commandes affectées aux six touches de menu à un moment donné). Une pression sur une touche de sélection de menu affecte le premier jeu de libellés aux touches de menu. Une pression sur [NEXT] leur affecte le second jeu de libellés, et ainsi de suite jusqu'à ce que vous reveniez au premier jeu de libellés. Une pression sur ■[PREV] vous fait passer d'un jeu de libellés à un autre dans le sens inverse de [NEXT].

Appuyer sur une touche de menu évalue l'opération qui lui est affectée ou en ajoute le nom dans la ligne de commande. Si vous appuyez sur une touche de menu à laquelle aucune fonction n'est affectée (le libellé de menu est alors vide), le calculateur émet une tonalité. Pour une description plus complète, consultez le paragraphe « Saisie des objets » au chapitre « Opérations de base ».

Le tableau suivant fait la liste des touches de sélection de menu et donne une description de tous les menus ainsi accessibles dans le HP-28C/S.

Touche de sélection de menu	Description
◀▶	Déplacement du curseur, opérations de correction.
■ ALGEBRA	Commandes opérant sur les objets algébriques.
■ ARRAY	Commandes opérant sur les vecteurs et matrices.
■ BINARY	Arithmétique sur les entiers, changements de base, manipulations de bits.
■ BRANCH	Branchements de programmes.
■ CMPLX	Commandes opérant sur les nombres complexes.
■ CTRL	Contrôle de programmes, arrêt, et exécution pas-à-pas.
■ LIST	Commandes opérant sur les listes.
■ LOGS	Fonctions logarithmiques, exponentielles et hyperboliques.
■ MODE	Format d'affichage, mode angulaire, récupération de données après erreur, séparateur décimal.
■ PLOT	Commandes de traçage.
■ PRINT	Commandes d'impression.
■ REAL	Commandes opérant sur les nombres réels.
SOLV	Commandes de résolution numérique et symbolique ; résolution d'équations et d'expressions algébriques.
■ STACK	Manipulation de la pile opérationnelle.
■ STAT	Commandes pour les statistiques et calculs de probabilités.
■ STORE	Arithmétique directe sur les variables et sur les matrices.
■ STRING	Opérations sur les chaînes de caractères.
■ TEST	Commandes d'indic. binaires et tests logiques.
TRIG	Fonctions trigonométriques, conversions polaire/rectangulaire et degré/radian, arithmétique sur les formats heures/minutes/secondes.
USER	Commandes sur variables et mémoire utilisateur.

ALGEBRA

COLCT	EXPAN	SIZE	FORM	OBSUB	EXSUB
TAYLR	ISOL	QUAD	SHOW	OBGET	EXGET

Objets algébriques

Un objet algébrique est une procédure qui est saisie et affichée sous forme mathématique. Il peut contenir des nombres, des noms de variables, des fonctions, et des opérateurs, tels qu'ils sont définis ci-dessous :

Nombre : Un nombre réel ou complexe.

Nom de variable : N'importe quel nom, qu'une variable lui soit associée ou non. Nous utiliserons le terme *variable formelle* pour référer à un nom auquel n'est associée aucune variable utilisateur. L'évaluation d'un tel nom ne produit aucun résultat ; comme une donnée, il renvoie simplement sa propre image.

Fonction : Une commande du HP-28C/S qui est autorisée dans une procédure algébrique. Les fonctions doivent donner un et exactement un résultat. Si un ou plusieurs arguments d'une fonction sont des objets algébriques, le résultat est algébrique. La plupart des fonctions apparaissent sous la forme d'un nom de fonction, suivi d'un ou plusieurs arguments entre parenthèses ; `'SIN(X)'` par exemple.

Opérateur : Une fonction qui ne nécessite généralement pas que ses arguments soient entre parenthèses. Les opérateurs NOT, $\sqrt{}$, et NEG (qui apparaît dans les objets algébriques sous la forme du signe $-$) sont des opérateurs *préfixes* : leurs noms apparaissent avant leurs arguments. Les opérateurs $+$, $-$, $*$, $/$, $\wedge$, $=$, $==$, $\neq$, $<$, $>$, $\leq$, $\geq$, AND, OR, et XOR sont des opérateurs *infixes* : leurs noms apparaissent entre leurs deux arguments.

...ALGEBRA

Ordre de priorité

L'*ordre de priorité* des opérateurs détermine l'ordre dans lequel s'effectue l'évaluation lorsque des expressions sont saisies sans parenthèses. Une opération prioritaire est exécutée la première. Lorsque des opérateurs ont le même ordre de priorité, l'évaluation s'effectue de la gauche vers la droite. Nous énumérons ci-dessous les fonctions algébriques du HP-28C/S par ordre de priorité, en commençant par celle qui a la plus grande priorité :

1. Les expressions entre parenthèses. Pour les expressions qui contiennent des termes imbriqués entre parenthèses les uns dans les autres, l'évaluation commence par le terme le plus « à l'intérieur » du système de parenthèses.
2. Les fonctions comme SIN, LOG, et FACT, dont les arguments doivent être entre parenthèses.
3. Les fonctions puissance (^) et racine carrée (√).
4. Le changement de signe (−), la multiplication (*), et la division (/).
5. L'addition (+) et la soustraction (−).
6. Les opérateurs relationnels (==, ≠, <, >, ≤, ≥).
7. AND et NOT.
8. OR et XOR.
9. =

Les objets algébriques et les programmes ont des structures internes identiques. Ces procédures sont toutes deux des séquences d'objets qui sont traitées séquentiellement lorsque les procédures sont évaluées. L'objet algébrique `'X+Y'` et le programme `« X Y + »` sont tous deux stockés sous la forme d'une séquence identique (la séquence en notation polonaise inverse). Les objets algébriques sont « marqués » comme algébriques de façon à être affichés sous la forme d'expressions mathématiques et à indiquer qu'ils obéissent aux règles de la syntaxe algébrique.

Syntaxe algébrique et sous-expressions

Une procédure obéit à la *syntaxe algébrique* si, lorsqu'elle est évaluée, elle ne prend aucun argument dans la pile opérationnelle et a pour résultat d'y envoyer un argument, et si elle peut être complètement subdivisée en une hiérarchie de *sous-expressions*. Une sous-expression peut être un nombre, un nom, ou une fonction et ses arguments. Par *hiérarchie* nous voulons dire que chaque sous-expression peut elle-même être l'argument d'une fonction. Prenons par exemple l'expression suivante :

```
'1-SIN(X+Y)'
```

Elle contient un nombre, `1`, et deux noms, `X` et `Y`, qui peuvent être chacun considéré comme une sous-expression simple. L'expression contient également trois fonctions, `+`, `-`, et `SIN`, qui constituent chacune, avec ses arguments, une sous-expression. Les arguments de `+` sont `X` et `Y` ; `X+Y` est l'argument de `SIN`, et `1` et `SIN(X+Y)` sont les arguments de `-`. La hiérarchie devient plus évidente si l'expression, avec ses opérateurs, est ré-écrite comme une fonction ordinaire (en *notation polonaise inverse*) :

$$-(1,\ \mathrm{SIN}\ (+(X,\ Y)))$$

Un objet ou une sous-expression à l'intérieur d'une expression est défini par sa *position* et son *rang*.

Pour définir la *position* d'un objet, on compte les sous-expressions de gauche à droite dans l'expression. Par exemple, dans l'expression `'1-SIN(X+Y)'`, `1` occupe la position 1, `-` la position 2, `SIN` la position 3, et ainsi de suite.

La position d'une sous-expression est la position « physique » de l'objet qui la définit. Dans le même exemple, `'SIN(X+Y)'` occupe la position 3, car elle est définie par `SIN` en position 3.

...ALGEBRA

Le *rang* d'un objet dans une expression algébrique est sa position « logique », c'est-à-dire le nombre de paires de parenthèses entourant cet objet lorsque l'expression est écrite sous une forme purement fonctionnelle. Par exemple, dans l'expression `'1-SIN(X+Y)'`, - correspond au rang 0, 1 et SIN au rang 1, + au rang 2, et X et Y au rang 3. Chaque expression algébrique a exactement un objet de rang 0.

Les fonctions définies par l'utilisateur constituent une exception manifeste à la règle de définition du rang d'une sous-expression. Dans `'F(A,B)'`, par exemple, où F est une fonction définie par l'utilisateur, F, A, et B sont toutes de rang 1 ; il n'y a pas de fonction explicite de rang 0. Ceci s'explique par le fait que F et ses arguments A et B sont tous les arguments d'une fonction spéciale « invisible » qui assure l'affichage et la logique d'évaluation des fonctions définies par l'utilisateur.

Si nous prenons l'expression ci-dessus et la ré-écrivons en enlevant les parenthèses et en positionnant les fonctions après leurs arguments, nous obtenons en notation polonaise inverse :

```
1 X Y + SIN -
```

Ceci définit un *programme* qui respecte la syntaxe algébrique et qui équivaut effectivement à l'objet algébrique correspondant. Toutefois, les programmes sont plus souples que les objets algébriques ; par exemple, nous pourrions insérer un DUP n'importe où dans le programme ci-dessus sans cesser pour autant d'avoir un programme tout à fait correct ; mais il ne respecterait plus la syntaxe algébrique. Comme DUP prend un argument dans la pile et en renvoie deux, il ne peut pas définir une sous-expression algébrique ou en faire partie.

Equations

Une *équation* algébrique est un objet algébrique contenant deux expressions liées par un signe égale (=). Mathématiquement, le signe égale implique l'égalité des deux sous-expressions situées de part et d'autre du signe. Dans le HP-28C/S, = est une fonction à deux arguments. Il est affiché comme un opérateur infixe, séparant les deux sous-expressions qui constituent ses arguments. Au plan interne du calculateur, une équation est une expression qui a le signe = comme objet de rang 0.

Lorsqu'une équation est évaluée numériquement, = est l'équivalent de -. Ceci permet aux expressions et équations d'être utilisées de façon interchangeable comme arguments dans les résolutions symboliques et numériques. Une équation équivaut à une expression dans laquelle = est remplacé par -, et une expression est l'équivalent du côté gauche d'une équation dans laquelle le côté droit est zéro.

Lorsqu'une équation est un argument d'une fonction, le résultat est également une équation, dans laquelle la fonction a été appliquée aux deux côtés. Ainsi

'X=Y' SIN donne 'SIN(X)=SIN(Y)'.

L'utilisation traditionnelle du signe = est ambiguë. Il est utilisé pour indiquer que deux expressions sont égales, comme dans $x + \sin y = 2z + t$. Ce type d'équation peut être résolu ; il faut pour cela ajuster une ou plusieurs variables pour parvenir à l'égalité des deux côtés de l'équation.

Le signe égale est aussi utilisé pour affecter une valeur à une variable, comme dans $x = 2y + z$. Cette équation signifie que le symbole x se substitue à l'expression plus longue $2y + z$; « résoudre » l'équation n'a ici aucun sens.

Cette ambiguïté du signe égale est éliminée par certains langages informatiques comme le BASIC, pour lequel « = » signifie « remplacer par » comme dans X = Y + Z. Cette notation n'implique aucunement qu'il s'agit d'une équation mathématique.

Dans le HP-28C/S, les signes égale signifient toujours l'égalité de deux expressions, de sorte que résoudre une équation équivaut à faire la différence entre les deux côtés de l'équation et zéro (l'affectation d'une valeur à une variable s'effectue avec STO, qui est strictement une commande suffixe nécessitant deux arguments).

...ALGEBRA

= ***Egale*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	'z_1=z_2'
z	'*symbole*'	➧	'z=*symbole*'
'*symbole*'	z	➧	'*symbole*=z'
'*symbole*$_1$'	'*symbole*$_2$'	➧	'*symbole*$_1$=*symbole*$_2$'

Cette fonction combine deux arguments, qui doivent être des noms, des expressions, des nombres réels ou des nombres complexes.

Si le HP-28C/S est en mode d'évaluation symbolique (indicateur binaire 36 armé), le résultat est une équation algébrique dont l'argument gauche se trouve au niveau 2 et l'argument droit au niveau 1 de la pile opérationnelle.

Si le HP-28C/S est en mode d'évaluation numérique (indicateur binaire désarmé), le résultat est la différence numérique entre les deux arguments. En effet, = joue le rôle de l'opérateur − en mode d'évaluation numérique.

Fonctions des arguments symboliques

Les modes d'évaluation

Mode d'évaluation symbolique (indicateur binaire 36 armé). En mode d'évaluation symbolique, les fonctions donnent des résultats symboliques si leurs arguments sont symboliques. C'est le mode d'évaluation par défaut. Par exemple :

```
     'X' SIN donne 'SIN(X)'
 'X^2+5' LN donne 'LN(X^2+5)'.
    3 'X' + donne '3+X'
2 'X' + SIN donne 'SIN(2+X)'.
'X' 1 2 IFTE donne 'IFTE(X,1,2)'.
```

Mode d'évaluation numérique (indic. binaire 36 désarmé). En mode d'évaluation numérique, chaque fonction essaye de convertir les arguments symboliques en données. Une fois les arguments convertis en nombres, la fonction est appliquée à ces arguments et donne un résultat numérique. Les arguments sont évalués de manière répétitive jusqu'à devenir des données ou des variables formelles. Si les arguments définitifs sont des variables formelles, le message d'erreur `Undefined Name` (un nom n'est pas défini) apparaît.

Simplification automatique

Lorsqu'elles sont évaluées, certaines fonctions remplacent certains arguments ou combinaisons d'arguments par des formes plus simples. Par exemple, lorsque `'1*X'` est évalué, la fonction `*` détecte qu'un de ses arguments est un `1`, et l'expression est alors remplacée par `'X'`. Il y a simplification automatique dans les cas suivants :

Expression originale	Expression simplifiée
Chgmnt de signe, inverse, racine carrée	
`-(-X)`	`X`
`INV(INV(X))`	`X`
`SQ(√X)`	`X`
`SQ(X^Y)`	`X^(Y*2)`
`SQ(i)`	`-1`
Addition et soustraction	
`0+X` ou `X+0`	`X`
`X-0`	`X`
`0-X`	`-X`
`X-X`	`0`
Multiplication	
`X*0` ou `0*X`	`0`
`X*1` ou `1*X`	`X`
`X*(-1)` ou `-1*X`	`-X`
`-X*(-1)` ou `-1*(-X)`	`X`
`i*i`	`-1`
`-X*INV(Y)`	`-(X/Y)`
`-X*Y`	`-(X*Y)`
`X*INV(Y)`	`X/Y`

...ALGEBRA

Expression originale	Expression simplifiée
Division	
X/1	X
0/X	0
1/INV(X)	X
1/X	INV(X)
Puissance	
1^X	1
X^0	1
X^1	X
(√X)^2	X
INV(X)^(-1)	X
X^(-1)	INV(X)
i^2	-1 ou (-1,0)*
i^(2,0)	(-1,0)
SIN, COS, TAN	
SIN(ASIN(X))	X
SIN(-X)	-SIN(X)
SIN(π)	0†
SIN(π/2)	1†
COS(ACOS(X))	X
COS(-X)	COS(X)
COS(π)	-1†
COS(π/2)	0†
TAN(ATAN(X))	X
TAN(-X)	-TAN(X)
TAN(π)	0†
ABS, MAX, MIN, MOD, SIGN	
ABS(ABS(X))	ABS(X)
ABS(-X)	ABS(X)
MAX(X,X)	X
MIN(X,X)	X
MOD(X,0)	X
MOD(0,X)	0
MOD(X,X)	0
MOD(MOD(X,Y),Y)	MOD(X,Y)
SIGN(SIGN(X))	SIGN(X)

* Selon que le mode d'évaluation est symbolique (indicateur binaire 36 armé) ou numérique (indicateur binaire 36 désarmé).

† Ne s'applique que lorsque le mode angulaire est RADIANS.

Expression originale	Expression simplifiée
ALOG, EXP, EXPM, SINH, COSH, TANH	
ALOG(LOG(X))	X
EXP(LN(X))	X
EXPM(LNP1(X))	X
SINH(ASINH(X))	X
COSH(ACOSH(X))	X
TANH(ATANH(X))	X
IM, RE, CONJ	
IM(IM(X))	0
IM(RE(X))	0
IM(CONJ(X))	-IM(X)
IM(i)	1
RE(RE(X))	RE(X)
RE(IM(X))	IM(X)
RE(CONJ(X))	RE(X)
RE(i)	0
CONJ(CONJ(X))	X
CONJ(RE(X))	RE(X)
CONJ(IM(X))	IM(X)
CONJ(i)	-i

Fonctions dans les équations

Appliquées aux équations en mode d'évaluation symbolique, les fonctions ont pour résultat des équations.

Si une fonction monadique (à un seul argument) est appliquée à une équation, le résultat est une équation obtenue par application de la fonction séparément aux côtés droit et gauche de l'équation. Par exemple :

'X+2=Y' SIN donne 'SIN(X+2)=SIN(Y)'.

...ALGEBRA

Si les deux arguments d'une fonction diadique (fonction à deux arguments) sont des équations, le résultat est une équation obtenue en reliant par le signe égale les expressions issues de l'exécution de la fonction séparément sur les deux côtés gauches de l'équation comme arguments, puis avec les deux côtés droits. Par exemple :

```
'X+Y=Z+T'  'SIN(Q)=5'  +  donne  'X+Y+SIN(Q)=Z+T+5'.
```

Si un argument d'une fonction diadique est un objet numérique ou une expression algébrique, et si l'autre argument est une équation, le premier est converti en une équation identité ayant l'objet initial des deux côtés du signe égale. On retrouve alors une fonction dont les deux arguments sont des équations. Par exemple :

```
'X=Y'  3  -  donne  'X-3=Y-3'.
```

Ces caractéristiques définissent le comportement des objets algébriques lorsqu'ils sont évalués (voir paragraphe ci-dessous) et elles vous permettent d'effectuer les calculs algébriques d'une manière interactive et selon la logique de la notation polonaise inverse, pratiquement de la même façon que vous effectuez des calculs numériques ordinaires.

Evaluation d'objets algébriques

L'*évaluation* d'objets algébriques est une fonction extrêmement puissante du HP-28C/S qui vous permet de consolider des expressions en effectuant des calculs numériques explicites, et de remplacer des nombres ou des expressions par des variables. Pour comprendre quel résultat vous pouvez attendre lorsque vous évaluez un objet algébrique, rappelez-vous qu'un objet algébrique équivaut à un programme, et qu'évaluer un programme signifie placer tous les objets du programme dans la pile opérationnelle et les évaluer lorsque ce sont des commandes ou des noms.

Pour expliquer ce que tout cela signifie, supposons que vous avez défini la variable X comme ayant la valeur 3 (en exécutant `3 'X' STO`), Y comme ayant la valeur 4, et Z la valeur `'X+T'`. Nous supposerons aussi que nous sommes en mode d'évaluation symbolique (indicateur binaire 36 armé), de manière que les fonctions acceptent des arguments symboliques.

Considérons d'abord l'expression 'X+Y'. Lorsque nous évaluons cette expression ('X+Y' EVAL), nous obtenons le résultat 7. Voici pourquoi : d'une manière interne, 'X+Y' est représenté sous la forme X Y +. Par conséquent, lorsque 'X+Y' est évaluée, X, Y, et + sont évalués en séquence :

1. Comme X est un nom, l'évaluer équivaut à évaluer l'objet stocké dans la variable X, c'est-à-dire le nombre 3. Evaluer X place donc 3 au niveau 1 de la pile opérationnelle.
2. De même, évaluer Y place 4 au niveau 1, ce qui pousse 3 au niveau 2.
3. Ensuite, + est évalué, avec 3 et 4 comme arguments numériques dans la pile opérationnelle. Ceci élimine 3 et 4 de la pile et y envoie le résultat 7.

Essayons maintenant d'évaluer 'X+T' :

1. L'évaluation de X place 3 au niveau 1.
2. T est un nom qui n'est pas associé à une variable, et l'évaluer a donc pour seul effet de l'envoyer lui-même au niveau 1, repoussant le 3 au niveau 2.
3. Cette fois, + a 3 et T comme arguments ; comme T est symbolique, + donne un résultat algébrique, '3+T'.

Evaluons enfin 'X+Y+Z'. D'une façon interne, cette expression est représentée sous la forme X Y + Z +. Selon la même logique que dans les exemples ci-dessus, l'évaluation donne le résultat '7+X+T'. Nous pouvons évaluer ce résultat une nouvelle fois et obtenir le nouveau résultat '10+T'. Une nouvelle évaluation n'apporte aucun changement, car aucune valeur n'est affectée à T.

Remarquez que l'évaluation de 'X+T+Y' (en gardant les mêmes valeurs que ci-dessus) donne '3+T+4', et non pas '7+T' ou 'T+7'. Les valeurs 3 et 4 obtenues en évaluant X et Y ne servent pas d'arguments au même opérateur + dans l'expression, et elles ne sont donc pas combinées. Si vous voulez combiner le 3 et le 4, vous pouvez utiliser soit la commande COLCT pour la mise en facteur automatique des termes, soit la commande FORM pour une réorganisation plus générale de l'expression.

...ALGEBRA

Constantes symboliques : e, π, i, MAXR, et MINR

Cinq objets algébriques, intégrés au calculateur, donnent une représentation numérique de certaines constantes. Ces objets ont pour particularité que leur évaluation est commandée par le mode d'évaluation des constantes symboliques (indicateur binaire 35) ET par le mode d'évaluation des fonctions (indicateur binaire 36). Lorsque l'indicateur binaire 36 est désarmé (mode d'évaluation numérique), ces objets sont évalués à leur valeur numérique (indépendamment de l'état de l'indicateur binaire 35, qui contrôle le mode d'évaluation des constantes). Lorsque l'indicateur binaire 36 est armé (mode d'évaluation symbolique) :

- Si l'indicateur binaire 35 est désarmé, ces objets sont évalués à leur valeur numérique. Par exemple :

  '2*i' EVAL donne (0,2).

- Si l'indicateur binaire 35 est armé, ces objets gardent leur valeur symbolique lorsqu'ils sont évalués. Par exemple :

  '2*i' EVAL donne '2*i'.

Le tableau suivant donne la liste des cinq objets et leur valeur numérique.

Constantes symboliques du HP-28C/S

Nom de l'objet	Valeur numérique
e	2,71828182846
π	3,14159265359
i	(0,00000000000 , 1,00000000000)
MAXR	9,99999999999E499
MINR	1,00000000000E-499

La valeur numérique de e et de π est l'approximation la plus proche des constantes ϵ et π qui puisse être exprimée avec une précision de 12 chiffres. La valeur numérique de i est la représentation exacte de la constante i. MAXR et MINR sont respectivement la plus grande et la plus petite valeur numérique non nulle pouvant être représentée par le HP-28C/S.

Pour une plus grande précision numérique, utilisez l'expression 'EXP(X)' plutôt que l'expression 'e^X'. La fonction EXP fait appel à un algorithme spécial pour calculer l'exponentielle avec une plus grande précision.

Lorsque vous avez choisi les radians comme mode angulaire, et que les indicateurs binaires 35 et 36 sont armés, les fonctions trigonométriques de π et de π/2 sont automatiquement simplifiées. Par exemple, évaluer 'SIN(π)' donne le résultat 0.

COLCT EXPAN SIZE FORM OBSUB EXSUB

Ces commandes modifient la forme des expressions algébriques, un peu comme vous le feriez si vous aviez à traiter ces expressions avec un crayon et un papier. COLCT, EXPAN, et FORM sont des opérations identité, c'est-à-dire qu'elles changent la forme d'une expression sans en changer la valeur. OBSUB et EXSUB vous permettent de modifier la valeur d'une expression en effectuant des remplacements par de nouveaux objets ou de nouvelles sous-expressions dans l'expression.

COLCT *Mise en facteur des termes* Commande

Niveau 1	Niveau 1
'*symbole*$_1$'	'*symbole*$_2$'

...ALGEBRA

COLCT (COLLECT) ré-écrit un objet algébrique de manière à le simplifier en en mettant en facteur les termes identiques. Plus précisément, COLCT :

- Evalue les sous-expressions numériques. Par exemple : `'1+2+LOG(10)'` est remplacé par 4.
- Met en facteur les termes numériques. Par exemple : `'1+X+2'` est remplacé par `'3+X'`.
- Organise les facteurs (les facteurs sont les opérandes de la multiplication), et combine les facteurs identiques. Par exemple : `'X^Z*Y*X^T*Y'` est remplacé par `'X^(T+Z)*Y^2'`.
- Organise les opérandes de l'addition, et combine ces opérandes lorsqu'ils sont identiques ou lorsqu'ils ne diffèrent que par leur coefficient numérique. Par exemple : `'X+X+Y+3*X'` est remplacé par `'5*X+Y'`.

COLCT opère séparément sur les deux côtés d'une équation, de façon que des termes identiques se trouvant dans des côtés opposés de l'équation ne soient pas combinés.

L'algorithme de réorganisation de COLCT (qui détermine si X précède Y) a été choisi pour sa vitesse d'exécution et non pour sa conformité à une forme standard ou évidente. Si l'ordre des termes dans une expression particulière obtenue après exécution de COLCT ne correspond pas à ce que vous souhaitiez, vous pouvez utiliser la commande FORM pour le modifier à nouveau.

EXPAN *Développer les produits* Commande

Niveau 1	Niveau 1
'*symbole*$_1$'	'*symbole*$_2$'

EXPAN (EXPAND) ré-écrit un objet algébrique en en développant les produits et les puissances. Plus précisément, EXPAN :

- Applique la distributivité de la multiplication et de la division par rapport à l'addition. Par exemple : `'A*(B+C)'` est développée en `'A*B+A*C'` ; `'(B+C)/A'` est développée en `'B/A+C/A'`.
- Développe les puissances dont les exposants sont des sommes. Par exemple : `'A^(B+C)'` est développée en `'A^B*A^C'`.
- Développe les puissances dont les exposants sont des entiers positifs. Par exemple : `'X^5'` est développée en `'X*X^4'`. Le carré d'une somme, `'(X+Y)^2'` ou `'SQ(X+Y)'`, est développé en `'X^2+2*X*Y+Y^2'`.

La commande EXPAN n'essaye pas d'effectuer tous les développements possibles d'une expression en une seule passe. Elle travaille au contraire rang après rang dans la hiérarchie des sous-expressions, en s'arrêtant chaque fois qu'elle trouve une sous-expression à développer. Elle analyse d'abord la sous-expression de rang 0 ; si elle peut être développée, elle l'est, et EXPAND s'arrête. Si elle ne peut pas l'être, EXPAN analyse chacune des sous-expressions de rang 1 et développe toutes celles qui peuvent l'être ; puis les sous-expressions de rang 2 sont analysées. Le processus se poursuit vers le bas de la hiérarchie jusqu'à ce qu'une sous-expression soit développée. Par exemple :

développez l'expression `'A^(B*(C^2+D))'`.

1. L'opérateur de rang 0 est le ^ de gauche. Comme il ne peut pas être développé, l'opérateur de rang 1, *, est analysé. L'un de ses arguments est une somme, et EXPAN applique donc la distributivité de la multiplication par rapport à l'addition, ce qui donne :

   `'A^(B*C^2+B*D)'`

2. L'opérateur de rang 0 est toujours le ^ de gauche, mais l'exposant est maintenant une somme, et la distributivité de la puissance par rapport à l'addition est appliquée lorsque vous exécutez à nouveau EXPAN :

   `'A^(B*C^2)*A^(B*D)'`

3. Il est possible de développer encore l'expression. L'opérateur de rang 0 est maintenant le * du milieu. Comme il ne peut pas être développé, les opérateurs de rang 1, c'est-à-dire les ^ externes aux parenthèses, sont analysés. Ils ne peuvent pas être développés non plus, et les opérateurs de rang 2, c'est-à-dire les * internes aux parenthèses, sont donc analysés. Comme ils ne peuvent pas non plus être développés, l'opérateur de rang 3, le dernier ^ interne aux parenthèses, est analysé. L'exposant est un entier positif, et la puissance est donc développée :

```
'A^(B*(C*C))*A^(B*D)'
```

SIZE *Taille* Commande

Niveau 1		Niveau 1
"*chaîne*"	➡	*n*
{ *liste* }	➡	*n*
[*tableau*]	➡	{ *liste* }
'*symbole*'	➡	*n*

La commande SIZE indique le nombre d'objets dont est constitué un objet algébrique.

Consultez « ARRAY », « LIST », et « STRING » pour l'utilisation de SIZE avec d'autres types d'objets.

FORM *Ré-organiser une expr. algébrique* Commande

Niveau 1		Niveau 3	Niveau 2	Niveau 1
'$symbole_1$'	➡			'$symbole_2$'
'$symbole_1$'	➡	'$symbole_2$'	*n*	'$symbole_3$'

FORM est un éditeur interactif d'expressions qui vous permet de réorganiser une expression ou une équation algébrique en fonction des règles standard des mathématiques. Son fonctionnement est décrit dans la section suivante, « ALGEBRA (FORM) ».

OBSUB — *Substituer objet* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
'*symbole*$_1$'	*n*	{ *objet* }	➡	'*symbole*$_2$'

OBSUB (OBJECT SUBSTITUTE) remplace un objet ayant une position spécifiée (*n*) dans un objet algébrique. L'objet de remplacement est le contenu d'une liste se trouvant au niveau 1, la position *n* est au niveau 2, et l'objet algébrique est au niveau 3 de la pile opérationnelle. Par exemple :

'A*B' 3 { C } OBSUB donne 'A*C'.

Vous pouvez substituer des fonctions aussi bien que des variables utilisateur. Par exemple :

'A*B' 2 { + } OBSUB donne 'A+B'.

EXSUB — *Substituer expression* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
'*symbole*$_1$'	*n*	'*symbole*$_2$'	➡	'*symbole*$_3$'

EXSUB (EXPRESSION SUBSTITUTE) remplace la sous-expression se trouvant en *n*ième position dans l'expression algébrique '*symbole*$_1$' par l'objet algébrique (ou le nom) '*symbole*$_2$' et donne l'expression résultat '*symbole*$_3$'. La *n*ième sous-expression est constituée du *n*ième objet de l'expression, plus ses arguments (s'il y en a).

'(A+B)*C' 2 'E^F' EXSUB donne 'E^F*C'.

...ALGEBRA

TAYLR ISOL QUAD SHOW OBGET EXGET

La commande TAYLR est décrite dans « Calcul différentiel et intégral », avec ∂ et $\int$. ISOL, QUAD, et SHOW sont décrites dans « SOLV ».

OBGET *Appeler objet* Commande

Niveau 2	Niveau 1		Niveau 1
'symbole'	*n*	➡	*« objet »*

OBGET (OBJECT GET) donne l'objet en *n*ième position dans l'objet algébrique *symbole* qui se trouve au niveau 2 de la pile opérationnelle. Cet objet est obtenu sous la forme d'un objet unique dans une liste. Par exemple :

'(A+B)*C' 2 OBGET donne « + ».

Si *n* excède le nombre des objets existants, OBGET donne l'objet de rang 0.

EXGET *Appeler expression* Commande

Niveau 2	Niveau 1		Niveau 1
'symbole$_1$*'*	*n*	➡	*'symbole*$_2$*'*

EXGET donne la sous-expression en *n*ième position dans l'objet algébrique *symbole*$_1$ qui se trouve au niveau 2 de la pile opérationnelle. La *n*ième sous-expression se compose du *n*ième objet et de ses arguments éventuels. Par exemple :

'(A+B)*C' 2 EXGET donne 'A+B'.

Si *n* excède le nombre d'objets existants, EXGET donne la sous-expression de rang 0.

ALGEBRA (FORM)

FORM ***Ré-organiser une expr. algébrique*** **Commande**

Niveau 1		Niveau 3	Niveau 2	Niveau 1
' $symbole_1$ '	➧			' $symbole_2$ '
' $symbole_1$ '	➧	' $symbole_2$ '	n	' $symbole_3$ '

La commande FORM est un éditeur interactif d'expressions qui vous permet de ré-organiser une expression ou équation algébrique conformément aux règles standard des mathématiques. Toutes les opérations mathématiques de FORM sont des identités ; c'est-à-dire que l'expression résultat *symbole*$_2$ aura la même valeur que l'expression argument initiale *symbole*$_1$, même si ces deux expressions se présentent sous des formes différentes. Par exemple, FORM vous permet de ré-organiser `'A+B'` en `'B+A'`, ce qui modifie la forme mais pas la valeur de l'expression.

Lorsque FORM est active, vous disposez d'une variante de la commande EXGET. Cette variante permet de dupliquer une sous-expression *symbole*$_3$ contenue dans *symbole*$_1$, et d'envoyer dans la pile opérationnelle l'expression *symbole*$_3$ et sa position *n*.

Lorsque FORM est exécutée, l'affichage normal de la pile opérationnelle est remplacé par un affichage spécial de l'objet algébrique, accompagné d'un menu des opérations disponibles dans FORM. Cet affichage spécial commence en ligne 2 (la seconde à partir du haut) et se continue en ligne 3 si l'objet est trop long pour être affiché sur une seule ligne. Si l'objet nécessite plus de deux lignes d'affichage, il vous faudra déplacer le curseur de la commande FORM pour voir le reste de l'objet.

...ALGEBRA (FORM)

Pour quitter FORM et continuer avec d'autres opérations du calculateur, appuyez sur [ON]. Vous pouvez aussi appuyer sur la touche de menu EXGET, ce qui a également pour effet de ramener dans la pile opérationnelle la sous-expression choisie $symbole_3$ et sa position n.

Le curseur de FORM met en valeur les objets individuellement dans l'affichage de l'expression (il ne met pas en valeur les caractères, comme le curseur de la ligne de commande). L'objet sur lequel se trouve le curseur apparaît en caractères blancs sur fond noir. Le curseur identifie à la fois l'*objet choisi*, qui apparaît en vidéo inverse, et la *sous-expression choisie*, qui est celle qui comprend l'objet choisi et ses éventuels arguments.

Vous pouvez déplacer le curseur vers la gauche ou la droite dans l'expression en appuyant sur les touches de menu [←] ou [→] ; lorsque le curseur se déplace, il passe d'un objet à l'autre, en sautant les parenthèses. Il se trouve toujours dans la ligne 2 de l'affichage. Si vous essayez de le déplacer au-delà de l'extrémité de cette ligne, l'expression défile d'une ligne vers le haut sur l'affichage et le curseur revient en début de la ligne 2. De même, si vous essayez de déplacer le curseur au-delà du début (extrémité gauche) de la ligne 2, l'expression défile d'une ligne vers le bas, et le curseur se place à la fin de la ligne 2.

L'affichage de l'expression diffère de l'affichage normal d'un objet algébrique dans la pile opérationnelle : des parenthèses supplémentaires sont insérées afin de rendre explicite l'ordre de priorité de tous les opérateurs. Vous pouvez ainsi identifier plus facilement la sous-expression choisie, c'est-à-dire celle qui contient l'objet choisi à l'aide du curseur. Cela est important car toutes les opérations du menu FORM opèrent sur la sous-expression choisie.

Pendant que FORM est active, vous disposez d'une série d'opérations spéciales par l'intermédiaire des touches de menu. Le menu initial contient six opérations, qui sont commune à toutes les sous-expressions. Vous pouvez accéder à d'autres menus d'opérations en appuyant sur les touches [NEXT] ou ■[PREV] ; le contenu de ces autres menus varie en fonction de l'objet choisi. Seules apparaissent les opérations qui s'appliquent à l'objet choisi.

Vous pouvez ré-afficher les libellés du menu initial à n'importe quel moment en appuyant sur [ENTER].

Les opérations de FORM

Dans les paragraphes suivants, nous décrivons toutes les opérations pouvant apparaître dans les menus FORM. Ces descriptions consistent essentiellement en exemples de structure des sous-expressions choisies « avant » et « après » l'exécution des opérations. Chaque opération possible est illustrée par un exemple du type :

←D Distributivité à gauche.

Avant	Après
((A+B)*C)	((A*C)+(B*C))

Par souci de simplicité, nous utiliserons des noms de variables comme A, B et C, mais il doit être entendu que ces variable peuvent représenter des objets ou sous-expressions plus généraux. L'exemple ci-dessus montre qu'exécuter ←D (appliquer la distributivité vers la gauche) sur '(A+B)*C' donne le résultat 'A*C+B*C'.

Les différentes opérations de FORM apparaissent dans le menu FORM lorsqu'elles s'appliquent à l'objet choisi. Par exemple, ←D apparaît dans le menu lorsque + est l'objet choisi, mais pas lorsque SIN est l'objet choisi. En outre, si une opération apparaît, vous ne pourrez l'exécuter que si elle s'applique à la sous-expression choisie. Ainsi, D→ apparaît lorsque * est l'objet choisi, car la distributivité est une propriété de la multiplication. Mais la touche de menu correspondante n'est opérante que si la sous-expression est de la forme '(A+B)*C' ou '(A-B)*C', à laquelle il est possible d'appliquer la distributivité ; sinon la touche est inopérante et le calculateur émet une tonalité lorsqu'elle est actionnée.

...ALGEBRA (FORM)

Le menu initial de FORM contient les opérations suivantes :

Opérations communes à toutes les sous-expressions

Opération	Description
COLCT	Met en facteur les termes identiques dans la sous-expression choisie. Cette opération fonctionne de la même façon que la commande COLCT, mais son action est limitée à la sous-expression choisie. Le curseur FORM est repositionné au début de l'expression sur l'affichage.
EXPAN	Développe les produits et les puissances dans la sous-expression choisie. Cette opération fonctionne de la même façon que la commande EXPAN, mais son action est limitée à la sous-expression choisie. Le curseur FORM est repositionné au début de l'expression sur l'affichage.
LEVEL	Affiche le rang de l'objet choisi ou de la sous-expression qui le contient. Le rang est affiché aussi longtemps que vous maintenez la touche LEVEL enfoncée.
EXGET	Quitte FORM, en laissant la version en cours de l'expression modifiée au niveau 3 de la pile opérationnelle, une copie de la sous-expression choisie au niveau 1, et l'indication de sa position au niveau 2.
[←]	Place le curseur de FORM sur l'objet précédent (vers la gauche) de l'expression.
[→]	Place le curseur de FORM sur l'objet suivant (vers la droite) de l'expression.

Commutativité, associativité, et distributivité

←→ Commutativité des arguments d'un opérateur.

Avant	Après
(A+B)	(B+A)
(-(A)+B)	(B-A)
(A-B)	(-(B)+A)
(A*B)	(B*A)
(INV(A)*B)	(B/A)
(A/B)	(INV(B)*A)

←A Associativité à gauche. La flèche indique la direction dans laquelle les parenthèses « se déplacent ».

Avant	Après
(A+(B+C))	((A+B)+C)
(A+(B-C))	((A+B)-C)
(A-(B+C))	((A-B)-C)
(A-(B-C))	((A-B)+C)
(A*(B*C))	((A*B)*C)
(A*(B/C))	((A*B)/C)
(A/(B*C))	((A/B)/C)
(A/(B/C))	((A/B)*C)
(A^(B*C))	((A^B)^C)

...ALGEBRA (FORM)

A→ **Associativité à droite.** La flèche indique la direction dans laquelle les parenthèses « se déplacent ».

Avant	Après
((A+B)+C)	(A+(B+C))
((A-B)+C)	(A-(B-C))
((A+B)-C)	(A+(B-C))
((A-B)-C)	(A-(B+C))
((A*B)*C)	(A*(B*C))
((A/B)*C)	(A/(B/C))
((A*B)/C)	(A*(B/C))
((A/B)/C)	(A/(B*C))
((A^B)^C)	(A^(B*C))

→() **Distributivité de l'opérateur préfixe.**

Avant	Après
-(A+B)	(-(A)-B)
-(A-B)	(-(A)+B)
-(A*B)	(-(A)*B)
-(A/B)	(-(A)/B)
-(LOG(A))	LOG(INV(A))
-(LN(A))	LN(INV(A))
INV(A*B)	(INV(A)/B)
INV(A/B)	(INV(A)*B)
INV(A^B)	(A^-(B))
INV(ALOG(A))	ALOG(-(A))
INV(EXP(A))	EXP(-(A))

...ALGEBRA (FORM)

Remarquez que, chaque fois qu'une expression est ré-écrite, la séquence * INV est convertie en /. De même, + − est remplacé par −.

←D Distributivité à gauche. La flèche pointe sur la sous-expression à laquelle s'applique la distributivité.

Avant	Après
((A+B)*C)	((A*C)+(B*C))
((A-B)*C)	((A*C)-(B*C))
((A+B)/C)	((A/C)+(B/C))
((A-B)/C)	((A/C)-(B/C))
((A*B)^C)	((A^C)*(B^C))
((A/B)^C)	((A^C)/(B^C))

D→ Distributivité à droite. La flèche pointe sur la sous-expression à laquelle s'applique la distributivité.

Avant	Après
(A*(B+C))	((A*B)+(A*C))
(A*(B-C))	((A*B)-(A*C))
(A/(B+C))	INV((INV(A)*B)+(INV(A)*C))
(A/(B-C))	INV((INV(A)*B)-(INV(A)*C))
(A^(B+C))	((A^B)*(A^C))
(A^(B-C))	((A^B)/(A^C))
LOG(A*B)	(LOG(A)+LOG(B))
LOG(A/B)	(LOG(A)-LOG(B))
ALOG(A+B)	(ALOG(A)*ALOG(B))
ALOG(A-B)	(ALOG(A)/ALOG(B))
LN(A*B)	(LN(A)+LN(B))

...ALGEBRA (FORM)

(suite)

Avant	Après
LN(A/B)	(LN(A)-LN(B))
EXP(A+B)	(EXP(A)*EXP(B))
EXP(A-B)	(EXP(A)/EXP(B))

←M Mise en facteur des arguments gauches. Cette opération met en facteur les arguments de +, −, *, et de / lorsque ces arguments ont un facteur commun ou une fonction monadique EXP, ALOG, LN, ou LOG commune. Dans le cas de facteurs communs, la flèche indique que les facteurs gauches sont communs.

Avant	Après
((A*B)+(A*C))	(A*(B+C))
((A*B)-(A*C))	(A*(B-C))
((A^B)*(A^C))	(A^(B+C))
((A^B)/(A^C))	(A^(B-C))
(LN(A)+LN(B))	LN(A*B)
(LN(A)-LN(B))	LN(A/B)
(LOG(A)+LOG(B))	LOG(A*B)
(LOG(A)-LOG(B))	LOG(A/B)
(EXP(A)*EXP(B))	EXP(A+B)
(EXP(A)/EXP(B))	EXP(A-B)
(ALOG(A)*ALOG(B))	ALOG(A+B)
(ALOG(A)/ALOG(B))	ALOG(A-B)

M→ Mettre en facteur les arguments droits. Cette opération met en facteur les arguments de +, −, *, et de / lorsque ces arguments ont un facteur commun. La flèche indique que les facteurs droits sont communs.

Avant	Après
((A*C)+(B*C))	((A+B)*C)
((A/C)+(B/C))	((A+B)/C)
((A*C)-(B*C))	((A-B)*C)
((A/C)-(B/C))	((A-B)/C)
((A^C)*(B^C))	((A*B)^C)
((A^C)/(B^C))	((A/B)^C)

Double changement de signe et double-inversion

DNEG Double changement de signe. Change deux fois de suite le signe d'une sous-expression.

Avant	Après
A	-(-(A))

...ALGEBRA (FORM)

-() **Double changement de signe et distributivité.** Cette opération équivaut à une double changement de signe **DNEG** suivi d'une application de la distributivité **→()** sur le résultat obtenu à l'intérieur des parenthèses.

Avant	Après
(A+B)	-(-(A)-B)
(A-B)	-(-(A)+B)
(-(A)-B)	-(A+B)
(A*B)	-(-(A)*B)
(-(A)*B)	-(A*B)
(-(A)/B)	-(A/B)
(A/B)	-(-(A)/B)
LOG(A)	-(LOG(INV(A)))
LOG(INV(A))	-(LOG(A))
LN(A)	-(LN(INV(A)))
LN(INV(A))	-(LN(A))

DINV **Double-inversion.** Inverse deux fois une sous-expression.

Avant	Après
A	INV(INV(A))

1/() **Double-inversion et distributivité.** Cette opération équivaut à une double inversion **DINV** suivie d'une application de la distributivité **→()** sur l'inverse obtenue à l'intérieur des parenthèses.

Avant	Après
(A*B)	INV(INV(A)/B)
(A/B)	INV(INV(A)*B)
(A^B)	INV(A^-(B))
(A^-(B))	INV(A^B)
ALOG(A)	INV(ALOG(-(A)))
ALOG(-(A))	INV(ALOG(A))
EXP(A)	INV(EXP(-(A)))
EXP(-(A))	INV(EXP(A))

Opérations neutres

***1** **Multiplication par 1.**

Avant	Après
A	A*1

/1 **Division par 1.**

Avant	Après
A	A / 1

...ALGEBRA (FORM)

^1 **Elévation à la puissance 1.**

Avant	Après
A	A ^ 1

+1-1 **Addition et soustraction successives de 1.**

Avant	Après
A	(A + 1) - 1

Ré-organisation des exponentielles

L* **Remplacer log. d'une puissance par multiplication par un log.**

Avant	Après
LOG(A^B) LN(A^B)	(LOG(A)*B) (LN(A)*B)

L() **Remplacer multiplication par un log. par log. d'une puissance.**

Avant	Après
(LOG(A)*B) (LN(A)*B)	LOG(A^B) LN(A^B)

...ALGEBRA (FORM)

E^ **Remplacer puissance-produit par puissance d'une puissance.**

Avant	Après
ALOG(A*B)	(ALOG(A)^B)
ALOG(A/B)	(ALOG(A)^INV(B))
EXP(A*B)	(EXP(A)^B)
EXP(A/B)	(EXP(A)^INV(B))

E() **Remplacer puissance d'une puissance par puissance-produit.**

Avant	Après
(ALOG(A)^B)	ALOG(A*B)
(ALOG(A)^INV(B))	ALOG(A/B)
(EXP(A)^B)	EXP(A*B)
(EXP(A)^INV(B))	EXP(A/B)

Addition de fractions

,AF **Combinaison par rapport à un dénominateur commun.**

Avant	Après
(A+(B/C))	(((A*C)+B)/C)
((A/B)+C)	((A+(B*C))/B)
((A/B)+(C/D))	(((A*D)+(B*C))/(B*D))
(A-(B/C))	(((A*C)-B)/C)
((A/B)-C)	((A-(B*C))/B)
((A/B)-(C/D))	(((A*D)-(B*C))/(B*D))

...ALGEBRA (FORM)

Remarque : si le dénominateur est déjà commun à deux fractions, vous obtiendrez un résultat plus simple en utilisant M→ ou ←M

Liste des opérations FORM, par fonctions

Les tableaux ci-après montrent quelles opérations apparaissent dans le menu FORM lorsque l'objet choisi est une fonction. Pour chaque opération, les tableaux donnent la sous-expression initiale et le résultat. Les opérations COLCT, EXPAN, LEVEL, DNEG, DINV, *1, /1, ^1, et +1−1 sont communes à toutes les fonctions et variables et ne figurent pas dans les tableaux. Si ces opérations communes sont les seules disponibles pour une fonction, il n'y a pas de tableau pour cette fonction (seules les opérations communes sont disponibles pour √ et SQ ; pour utiliser d'autres opérations, remplacez-les par ^0,5 and ^2).

Addition (+)

Opération	Avant	Après
←→	(A+B) (-(A)+B)	(B+A) (B-A)
←A	(A+(B+C)) (A+(B-C))	((A+B)+C) ((A+B)-C)
A→	((A+B)+C) ((A-B)+C)	(A+(B+C)) (A-(B-C))
←M	((A*B)+(A*C)) (LN(A)+LN(B)) (LOG(A)+LOG(B))	(A*(B+C)) LN(A*B) LOG(A*B)
M→	((A*C)+(B*C)) ((A/C)+(B/C))	((A+B)*C) ((A+B)/C)
-()	(A+B) -(A)+B	-(-(A)-B) -(A-B)
AF	(A+(B/C)) ((A/B)+(C/D)) ((A/B)+C)	(((A*C)+B)/C) (((A*D)+(B*C))/(B*D)) ((A+(B*C))/B)

Soustraction (−)

Opération	Avant	Après
←→	(A−B)	(−(B)+A)
←A	(A−(B+C)) (A−(B−C))	((A−B)−C) ((A−B)+C)
A→	((A+B)−C) ((A−B)−C)	(A+(B−C)) (A−(B+C))
←M	((A∗B)−(A∗C)) (LN(A)−LN(B)) (LOG(A)−LOG(B))	(A∗(B−C)) LN(A/B) LOG(A/B)
M→	((A∗C)−(B∗C)) ((A/C)−(B/C))	((A−B)∗C) ((A−B)/C)
−( )	(A−B) (−(A)−B)	−(−(A)+B) −(A+B)
AF	(A−(B/C)) ((A/B)−C) ((A/B)−(C/D))	(((A∗C)−B)/C) ((A−(B∗C))/B) (((A∗D)−(B∗C))/(B∗D))

Multiplication (∗)

Opération	Avant	Après
←→	(A∗B) (INV(A)∗B)	(B∗A) (B/A)
←A	(A∗(B∗C)) (A∗(B/C))	((A∗B)∗C) ((A∗B)/C)
A→	((A∗B)∗C) ((A/B)∗C)	(A∗(B∗C)) (A/(B/C))
←D	((A+B)∗C) ((A−B)∗C)	((A∗C)+(B∗C)) ((A∗C)−(B∗C))
D→	(A∗(B+C)) (A∗(B−C))	((A∗B)+(A∗C)) ((A∗B)−(A∗C))

...ALGEBRA (FORM)

(suite)

Opération	Avant	Après
←M	((A^B)*(A^C)) (ALOG(A)*ALOG(B)) (EXP(A)*EXP(B))	(A^(B+C)) ALOG(A+B) EXP(A+B)
M→	((A^C)*(B^C))	((A*B)^C)
-()	(A*B) (-(A)*B)	-(-(A)*B) -(A*B)
1/()	(A*B) (INV(A)*B)	INV(INV(A)/B) INV(A/B)
L()	(LOG(A)*B) (LN(A)*B)	LOG(A^B) LN(A^B)

Division (/)

Opération	Avant	Après
←→	(A/B)	(INV(B)*A)
←A	(A/(B*C)) (A/(B/C))	((A/B)/C) ((A/B)*C)
A→	((A*B)/C) ((A/B)/C)	(A*(B/C)) (A/(B*C))
←D	((A+B)/C) ((A-B)/C)	((A/C)+(B/C)) ((A/C)-(B/C))
D→	(A/(B+C)) (A/(B-C))	INV((INV(A)*B) +(INV(A)*C)) INV((INV(A)*B) -(INV(A)*C))
←M	((A^B)/(A^C)) (ALOG(A)/ALOG(B)) (EXP(A)/EXP(B))	(A^(B-C)) ALOG(A-B) EXP(A-B)

...ALGEBRA (FORM)

(suite)

Opération	Avant	Après
M→	((A^C)/(B^C))	((A/B)^C)
-()	(A/B) (-(A)/B)	-(-(A)/B) -(A/B)
L()	(LN(A)/B) (LOG(A)/B)	LN(A^INV(B)) LOG(A^INV(B))
1/()	(A/B)	INV(INV(A)*B)

Puissance (^)

Opération	Avant	Après
←A	(A^(B*C))	((A^B)^C)
A→	((A^B)^C)	(A^(B*C))
←D	((A*B)^C) ((A/B)^C)	((A^C)*(B^C)) ((A^C)/(B^C))
D→	(A^(B+C)) (A^(B-C))	((A^B)*(A^C)) ((A^B)/(A^C))
1/()	(A^B) (A^-(B))	INV(A^-(B)) INV(A^B)
E()	(ALOG(A)^B) (ALOG(A)^INV(B)) (EXP(A)^B) (EXP(A)^INV(B))	ALOG(A*B) ALOG(A/B) EXP(A*B) EXP(A/B)

...ALGEBRA (FORM)

Changement de signe (−)

Opération	Avant	Après
→()	-(A+B)	(-(A)-B)
	-(A-B)	(-(A)+B)
	-(LOG(A))	(-(A)*B)
	-(A/B)	(-(A)/B)
	-(A*B)	LOG(INV(A))
	-(LN(A))	LN(INV(A))

Inverse (INV)

Opération	Avant	Après
→()	INV(A*B)	(INV(A)/B)
	INV(A/B)	(INV(A)*B)
	INV(A^B)	(A^-(B))
	INV(ALOG(A))	ALOG(-(A))
	INV(EXP(A))	EXP(-(A))

Logarithme (LOG)

Opération	Avant	Après
D→	LOG(A*B)	(LOG(A)+LOG(B))
	LOG(A/B)	(LOG(A)-LOG(B))
-()	LOG(A)	-(LOG(INV(A)))
	LOG(INV(A))	-(LOG(A))
L*	LOG(A^B)	(LOG(A)*B)
	LOG(A^INV(B))	(LOG(A)/B)

...ALGEBRA (FORM)

Antilogarithme (ALOG)

Opération	Avant	Après
D→	ALOG(A+B) ALOG(A-B)	(ALOG(A)*ALOG(B)) (ALOG(A)/ALOG(B))
1/()	ALOG(A) ALOG(-(A))	INV(ALOG(-(A))) INV(ALOG(A))
E^	ALOG(A*B) ALOG(A/B)	(ALOG(A)^B) (ALOG(A)^INV(B))

Logarithme népérien (LN)

Opération	Avant	Après
D→	LN(A*B) LN(A/B)	(LN(A)+LN(B)) (LN(A)-LN(B))
-()	LN(A) LN(INV(A))	-(LN(INV(A))) -(LN(A))
L*	LN(A^INV(B))	(LN(A)*B)

Exponentielle (EXP)

Opération	Avant	Après
D→	EXP(A+B) EXP(A-B)	(EXP(A)*EXP(B)) (EXP(A)/EXP(B))
1/()	EXP(A) EXP(-(A))	INV(EXP(-(A))) INV(EXP(A))
E^	EXP(A*B) EXP(A/B)	(EXP(A)^B) (EXP(A)^INV(B))

Arithmétique

Cette section décrit les fonctions arithmétiques, +, −, ∗, /, ^, INV, √, SQ et NEG. Ces fonctions s'appliquent à plusieurs types d'objets. Elles sont décrites ici pour tous les types d'objets concernés ; elles sont également décrites dans d'autres sections, comme « ARRAY » et « COMPLEX », dans la mesure où elles s'appliquent à un objet en particulier.

+ ***Addition*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	z_1+z_2
[*tableau*$_1$]	[*tableau*$_2$]	➧	[*tableau*$_1$ *tableau*$_2$]
z	' *symbole* '	➧	' z (*symbole*) '
' *symbole* '	z	➧	' *symbole* z '
' *symbole*$_1$ '	' *symbole*$_2$ '	➧	' *symbole*$_1$ (*symbole*$_2$) '
{ *liste*$_1$ }	{ *liste*$_2$ }	➧	{ *liste*$_1$ *liste*$_2$ }
" *chaîne*$_1$ "	" *chaîne*$_2$ "	➧	" *chaîne*$_1$ *chaîne*$_2$ "
# n_1	n_2	➧	# n_1+n_2
n_1	# n_2	➧	# n_1+n_2
# n_1	# n_2	➧	# n_1+n_2

+ donne la somme de ses arguments, dans laquelle la nature de la somme est déterminée par le type d'arguments. Si les arguments sont :

Deux nombres réels. La somme est la somme ordinaire réelle des arguments.

Un nombre réel *u* et un nombre complexe (x, y). Le résultat est le nombre complexe $(x + u, y)$, obtenu en traitant le nombre réel comme un nombre complexe avec une partie imaginaire nulle.

Deux nombres complexes (x_1, y_1) et (x_2, y_2). Le résultat est la somme complexe $(x_1 + x_2, y_1 + y_2)$.

Un nombre et un objet algébrique. Le résultat est un objet algébrique représentant la somme symbolique.

Deux objets algébriques. Le résultat est un objet algébrique représentant la somme symbolique.

Deux listes. Le résultat est une liste obtenue par concaténation des objets de la liste du niveau 1 à la fin de la liste du niveau 2.

Deux chaînes. Le résultat est une chaîne obtenue par concaténation des caractères de la chaîne du niveau 1 à la fin de la chaîne du niveau 2.

Deux tableaux. Le résultat est la somme des tableaux, dans lesquels chaque élément est une somme réelle ou complexe des éléments correspondants des tableaux-arguments. Les deux tableaux doivent avoir les mêmes dimensions.

Un entier binaire et un nombre réel. Le résultat est un entier binaire qui est la somme des deux arguments, abrégée selon la taille de mot en vigueur. Le nombre réel est transformé en entier binaire avant l'addition.

Deux entiers binaires. Le résultat est un entier binaire qui est la somme des deux arguments, abrégée selon la taille de mot en vigueur.

...Arithmétique

— ***Soustraction*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	$z_1 - z_2$
⟦ *tableau*$_1$ ⟧	⟦ *tableau*$_2$ ⟧	➧	⟦ *tableau*$_1$ − *tableau*$_2$ ⟧
z	' *symbole* '	➧	' z − *symbole* '
' *symbole* '	z	➧	' *symbole* − z '
' *symbole*$_1$ '	' *symbole*$_2$ '	➧	' *symbole*$_1$ − *symbole*$_2$ '
# n_1	n_2	➧	# $n_1 - n_2$
n_1	# n_2	➧	# $n_1 - n_2$
# n_1	# n_2	➧	# $n_1 - n_2$

— donne la différence entre ses arguments ; la nature de la différence est déterminée par le type des arguments. L'objet du niveau 1 est soustrait de l'objet du niveau 2. Si les arguments sont :

Deux nombres réels. Le résultat est la différence réelle ordinaire des arguments.

Un nombre réel *u* et un nombre complexe (*x*, *y*). Le résultat est le nombre complexe $(x - u, y)$ ou $(u - x, -y)$, obtenus en traitant le nombre réel comme un nombre complexe avec une partie imaginaire nulle.

Deux nombres complexes (x_1, y_1) et (x_2, y_2). Le résultat est le nombre complexe $(x_1 - x_2, y_1 - y_2)$.

Un nombre et un objet algébrique. Le résultat est un objet algébrique représentant la différence symbolique.

Deux objets algébriques. Le résultat est un objet algébrique représentant la différence symbolique.

Deux tableaux. Le résultat est un tableau dans lequel chaque élément est la différence réelle ou complexe entre les éléments correspondants des tableaux-arguments. Les deux tableaux doivent avoir les mêmes dimensions.

Un entier binaire et un nombre réel. Le résultat est un entier binaire qui est la somme du nombre au niveau 2 et du complément à deux du nombre au niveau 1. Le nombre réel est converti en un entier binaire avant la soustraction.

Deux entiers binaires. Le résultat est un entier binaire qui est la somme du nombre du niveau 2 et du complément à deux du nombre au niveau 1.

* ***Multiplication*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	$z_1 z_2$
⊏ *matrice* ⊐	⊏ *tableau* ⊐	➧	⊏ *matrice* × *tableau* ⊐
z	⊏ *tableau* ⊐	➧	⊏ z × *tableau* ⊐
⊏ *tableau* ⊐	z	➧	⊏ *tableau* × z ⊐
z	' *symbole* '	➧	' z ✱ (*symbole*) '
' *symbole* '	z	➧	' (*symbole*) ✱ z '
' $symbole_1$ '	' $symbole_2$ '	➧	' $symbole_1$ ✱ $symbole_2$ '
# n_1	n_2	➧	# $n_1 n_2$
n_1	# n_2	➧	# $n_1 n_2$
# n_1	# n_2	➧	# $n_1 n_2$

...Arithmétique

* donne le produit de ses arguments, dans lequel la nature du produit est déterminée par le type des arguments. Si les arguments sont :

Deux nombres réels. Le résultat est le produit réel ordinaire des arguments.

Un nombre réel *u* et un nombre complexe (x, y). Le résultat est le nombre complexe (xu, uz) obtenu en traitant le nombre réel comme un nombre complexe avec une partie imaginaire nulle.

Deux nombres complexes (x_1, y_1) et (x_2, y_2). Le résultat est le produit complexe $(x_1x_2 - y_1y_2, x_1y_2 + x_2y_1)$.

Un nombre et un objet algébrique. Le résultat est un objet algébrique représentant le produit symbolique.

Deux objets algébriques. Le résultat est un objet algébrique représentant le produit symbolique.

Un nombre et un tableau. Le résultat est le produit de la multiplication de chaque élément du tableau par le nombre.

Une matrice et un tableau. Le résultat est le produit matriciel des arguments. Le tableau du niveau 1 doit avoir le même nombre de lignes (d'éléments, si c'est un vecteur) que le nombre de colonnes de la matrice du niveau 2.

Un entier binaire et un nombre réel. Le résultat est un entier binaire qui est le produit des deux arguments, abrégé selon la taille du mot en cours. Le nombre réel est converti en un entier binaire avant la multiplication.

Deux entiers binaires. Le résultat est un entier binaire qui est le produit des deux arguments, abrégé selon la taille du mot en cours.

! ***Division*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	z_1/z_2
[*tableau*]	[*matrice*]	➧	[*tableau* × *matrice*$^{-1}$]
z	' *symbole* '	➧	' z / (*symbole*) '
' *symbole* '	z	➧	' (*symbole*) / z '
' *symbole*$_1$ '	' *symbole*$_2$ '	➧	' *symbole*$_1$ / *symbole*$_2$ '
# n_1	n_2	➧	# n_1/n_2
n_1	# n_2	➧	# n_1/n_2
# n_1	# n_2	➧	# n_1/n_2

/ ([÷]) donne le quotient (l'objet du niveau 2 divisé par l'objet du niveau 1) de ses arguments, dans lequel la nature du quotient est déterminée par le type des arguments. Si les arguments sont :

Deux nombres réels. Le résultat est le quotient ordinaire réel des arguments.

Un nombre réel *u* au niveau 2 et un nombre complexe (x, y) au niveau 1. Le résultat est le nombre complexe

$$(ux/(x^2 + y^2),\ -uy/(x^2 + y^2))$$

obtenu en traitant le nombre réel comme un nombre complexe avec une partie imaginaire nulle.

Un nombre complexe (x, y) au niveau 2 et un nombre réel *u* au niveau 1. Le résultat est le nombre complexe $(x/u,\ y/u)$ obtenu en traitant le nombre réel comme un nombre complexe avec une partie imaginaire nulle.

...Arithmétique

Un nombre complexe (x_1, y_1) au niveau 2 et un nombre complexe (x_2, y_2) au niveau 1. Le résultat est le quotient complexe

$$((x_1x_2 + y_1y_2)/(x_2^2 + y_2^2),\ (y_1x_2 - x_1y_2)/(x_2^2 + y_2^2)).$$

Un nombre et un objet algébrique. Le résultat est un objet algébrique représentant le quotient symbolique.

Deux objets algébriques. Le résultat est un objet algébrique représentant le quotient symbolique.

Un tableau et une matrice. Le résultat est le produit matriciel du tableau du niveau 2 avec l'inverse de la matrice du niveau 1. Le tableau du niveau 2 doit avoir le même nombre de lignes (d'éléments, si c'est un vecteur) que la matrice du niveau 1 a de colonnes.

Un entier binaire et un nombre réel. Le résultat est un entier binaire qui est la partie entière du quotient des deux arguments. Le nombre réel est converti en entier binaire avant la division. Un diviseur de valeur zéro renvoie # 0.

Deux entiers binaires. Le résultat est un entier binaire qui est la partie entière du quotient des deux arguments. Un diviseur de valeur zéro renvoie # 0.

^ ***Puissance*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
z_1	z_2	➧	$z_1^{z_2}$
z	'*symbole*'	➧	'z^(*symbole*)'
'*symbole*'	z	➧	'(*symbole*)^z'
'$symbole_1$'	'$symbole_2$'	➧	'$symbole_1$^($symbole_2$)'

^ donne la valeur de l'objet du niveau 2 élevé à la puissance donnée par un objet au niveau 1. Vous pouvez utiliser n'importe quelle combinaison de nombres réels, de nombres complexes et d'arguments algébriques. Si l'un des arguments est complexe, ^ renvoie un résultat complexe.

INV — *Inverse* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	*1/z*
⟦ *matrice* ⟧	➧	⟦ *matrice*$^{-1}$ ⟧
' *symbole* '	➧	' INV (*symbole*) '

INV (■ 1/x) renvoie l'inverse de l'argument.

Pour un argument complexe (x, y), l'inverse est le nombre complexe

$$(x/(x^2 + y^2),\ -y/(x^2 + y^2)).$$

Les tableaux-arguments doivent être des matrices carrées.

√ — *Racine carrée* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	$\sqrt{z}$
' *symbole* '	➧	' √ (*symbole*) '

...Arithmétique

$\sqrt{}$ donne la racine carrée (positive) de son argument. Dans le cas d'un nombre complexe (x_1, y_1), la racine carrée est le nombre complexe

$$(x_2, y_2) = (\sqrt{r} \cos \theta/2, \sqrt{r} \sin \theta/2)$$

avec

$$r = \text{abs } (x_1, y_1), \quad \theta = \arg (x_1, y_1).$$

Si $(x_1, y_1) = (0, 0)$, la racine carrée est $(0, 0)$.

Référez-vous à « Domaines principaux et solutions générales » dans la section « COMPLEX ».

SQ — *Carré* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	z^2
⟦ *matrice* ⟧	➧	⟦ *matrice* × *matrice* ⟧
' *symbole* '	➧	' SQ (*symbole*) '

SQ (SQUARE, ■ x²) donne le carré de son argument.

Pour un argument complexe (x, y), le carré est le nombre complexe

$$(x^2 - y^2, 2xy).$$

Les tableaux-arguments doivent être des matrices carrées.

NEG — *Changement de signe* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	$-z$
⟦ *tableau* ⟧	➧	⟦ −*tableau* ⟧
' *symbole* '	➧	' − (*symbole*) '

NEG (NEGATE) donne l'opposé de son argument.

Pour un tableau, l'opposé est un tableau composé de l'opposé de chaque élément du tableau. La touche [CHS] peut être utilisée pour exécuter NEG en cas d'absence d'une ligne de commande. S'il y a une ligne de commande, [CHS] agit sur la ligne de commande comme décrit dans « Opérations de base ».

Les touches de menu de NEG se trouvent dans les menus REAL et ARRAY.

ARRAY

→ARRY	ARRY→	PUT	GET	PUTI	GETI
SIZE	RDM	TRN	CON	IDN	RSD
CROSS	DOT	DET	ABS	RNRM	CNRM
R→C	C→R	RE	IM	CONJ	NEG

Les *tableaux* (« *ARRAYS* ») sont des ensembles ordonnés de nombres réels ou complexes qui satisfont certaines règles mathématiques. Dans le HP-28C/S, les tableaux uni-dimensionnels sont nommés *vecteurs* et les tableaux bidimensionnels sont nommés *matrices*. Nous utiliserons le terme « tableaux » pour référer à la fois aux vecteurs et aux matrices.

Bien que les vecteurs soient introduits dans la machine — et affichés — sous forme d'une *ligne* de nombres, le HP-28C/S les traite, pour la multiplication de matrices et les calculs de normes de matrices, comme des matrices $n \times 1$.

Un tableau peut contenir des nombres réels ou complexes. Nous utiliserons les termes *tableau réel*, ou *tableau R*, (*vecteur réel* ou *matrice réelle*) et *tableau complexe*, ou *tableau C*, lorsque nous décrirons les propriétés des tableaux qui sont spécifiques des nombres réels ou des nombres complexes.

Les tableaux sont introduits et affichés selon les formats suivants :

vecteur	[*nombre nombre ...*]
matrice	[[*nombre nombre ...*] [*nombre nombre ...*] ⋮ [*nombre nombre ...*]]

où *nombre* représente un nombre réel ou complexe.

...ARRAY

Lorsque vous introduisez un tableau dans le calculateur, vous pouvez mélanger nombres complexes et nombres réels. Si l'un des nombres d'un tableau est complexe, ce tableau sera complexe.

Vous pouvez inclure n'importe quel nombre de nouvelles lignes à n'importe quel endroit de la saisie, ou vous pouvez saisir tout le tableau sur une seule ligne de commande.

Lors de la saisie de matrices, le délimiteur qui termine chaque ligne,], peut être omis. Celui qui commence chaque ligne, [, est obligatoire. Si d'autres objets suivent le tableau en ligne de commande, vous devez terminer le tableau par]] avant d'introduire le nouvel objet.

Le terme *numéro d'ordre* fait référence à l'organisation séquentielle des éléments dans un tableau, en partant du premier élément (première ligne, première colonne), puis en progressant de gauche à droite sur chaque ligne, et de la ligne du haut vers la ligne du bas (pour les matrices).

Le menu STORE contient certaines commandes qui permettent d'effectuer des opérations sur des tableaux en utilisant le nom d'une variable qui contient un tableau, plutôt que de faire figurer le tableau lui-même dans la pile opérationnelle. Dans ce cas, le résultat d'une opération est stocké dans une variable, remplaçant son contenu d'origine. Cette méthode est plus économe de la mémoire que celle qui consiste à effectuer les opérations dans la pile opérationnelle, et elle permet ainsi d'utiliser des tableaux plus importants.

Les opérations, qui peuvent être fort longues dans le cas de tableaux, peuvent être interrompues par la touche [ON]. L'appui sur [ON] permet au HP-28C/S d'arrêter l'exécution de la commande opérant sur le tableau et d'effacer de la pile les arguments du tableau. Les arguments d'origine peuvent être récupérés par UNDO ou LAST.

En plus des fonctions présentes dans les menus ARRAY et STACK, les fonctions du clavier, décrites au paragraphe suivant, acceptent les tableaux en tant qu'arguments.

Fonctions disponibles au clavier

Vous trouverez des diagrammes complets de la pile opérationnelle pour ces fonctions sous « Arithmétique ».

+ ***Addition*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
⌈ *tableau*$_1$ ⌋	⌈ *tableau*$_2$ ⌋	➡	⌈ *tableau*$_1$ + *tableau*$_2$ ⌋

+ donne un tableau qui est la somme des deux tableaux-arguments. La somme d'un tableau réel et d'un tableau complexe est un tableau complexe, dans lequel chaque élément x du tableau réel est traité comme un élément complexe $(x, 0)$.

− ***Soustraction*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
⌈ *tableau*$_1$ ⌋	⌈ *tableau*$_2$ ⌋	➡	⌈ *tableau*$_1$ − *tableau*$_2$ ⌋

− donne un tableau qui est la différence des deux tableaux-arguments. Les deux tableaux doivent avoir les mêmes dimensions. La différence entre un tableau réel et un tableau complexe est un tableau complexe dont chaque élément x du tableau réel est traité comme un élément complexe $(x, 0)$.

* *Multiplication* Fonct. analyt.

Niveau 2	Niveau 1		Niveau 1
z	[*tableau*]	➧	[*z* × *tableau*]
[*tableau*]	*z*	➧	[*z* × *tableau*]
[*matrice*]	[*tableau*]	➧	[*matrice* × *tableau*]

* renvoie le produit de ses arguments, dont la nature est déterminée par le type des arguments. Si les arguments sont :

Un tableau et un nombre. Le résultat est le produit matriciel du nombre (réel ou complexe) et du tableau, obtenu en multipliant chaque élément du tableau par le scalaire.

Deux tableaux. Le résultat est le produit matriciel de deux tableaux. Le tableau du niveau 2 doit être une matrice (il ne peut être un vecteur) ; le niveau 1 peut contenir un vecteur ou une matrice. Le nombre de lignes dans le tableau du niveau 1 doit être égal au nombre de colonnes de la matrice du niveau 2.

Le produit d'un tableau réel et d'un tableau complexe est un tableau complexe. Chaque élément x du tableau réel est traité comme élément complexe $(x,0)$.

/ *Division* Fonct. analyt.

Niveau 2	Niveau 1		Niveau 1
[*matrice* **B**]	[*matrice* **A**]	➧	[*matrice* **X**]
[*vecteur* **B**]	[*matrice* **A**]	➧	[*vecteur* **X**]

...ARRAY

/ (÷) appliqué aux arguments d'un tableau, résout le système **AX** = **B** pour **X**. C'est-à-dire que / calcule $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$. / utilise une précision interne de 16 chiffres pour donner un résultat plus précis que celui obtenu en utilisant INV sur **A** et en multipliant le résultat par **B**.

A doit être une matrice carrée et **B** peut être une matrice ou un vecteur. Si **B** est une matrice, elle doit posséder le même nombre de lignes que **A**. Si **B** est un vecteur, il doit posséder autant d'éléments que **A** compte de colonnes.

Dans bien des cas, le HP-28C/S arrivera à une solution correcte même si le tableau-coefficient est singulier (**A** n'a pas d'inverse). Cette caractéristique permet de résoudre les systèmes d'équations sous-déterminés et sur-déterminés.

Pour un système sous-déterminé (contenant plus de variables qu'il n'y a d'équations), la tableau-coefficient aura moins de lignes que de colonnes. Pour trouver une solution :

1. Ajoutez suffisamment de lignes de zéros à la fin de votre tableau-coefficient pour en faire un carré.
2. Ajoutez un nombre de lignes correspondant de zéros au tableau constant.

Ces tableaux peuvent maintenant être utilisés avec / pour trouver une solution au système original.

Dans le cas d'un système sur-déterminé (contenant plus d'équations que de variables), le tableau-coefficient aura moins de colonnes que de lignes. Pour trouver une solution :

1. Ajoutez suffisamment de colonnes de zéros à la droite de votre tableau-coefficient pour qu'il prenne la forme d'un carré.
2. Ajoutez suffisamment de zéros à la suite de votre tableau constant pour assurer la conformité.

Vous pouvez maintenant utiliser ces tableaux avec / pour trouver une solution au système d'origine. Seuls les éléments du tableau-résultat qui correspondent aux variables d'origine seront significatifs.

Pour les systèmes sous-déterminés et sur-déterminés, le tableau-coefficient est sigulier et vous devez passer en revue les résultats donnés par / pour vérifier qu'ils sont en accord avec l'équation d'origine.

Amélioration de la précision des solutions des systèmes

Les nombres sont arrondis lors des calculs ; de ce fait, une solution **Z** calculée numériquement n'est pas en général la solution d'un système d'origine $\mathbf{AX} = \mathbf{B}$, mais celle du système modifié $(\mathbf{A} + \Delta\mathbf{A})\ \mathbf{Z} = \mathbf{B} + \Delta\mathbf{B}$. Les différences $\Delta\mathbf{A}$ et $\Delta\mathbf{B}$ satisfont $\|\Delta\mathbf{A}\| \leqslant \epsilon\|\mathbf{A}\|$ et $\|\Delta\mathbf{B}\| \leqslant \epsilon\|\mathbf{B}\|$, dans lesquelles ϵ est un petit nombre et $\|\mathbf{A}\|$ est la *norme* de **A**, une mesure de sa taille, analogue à la longueur d'un vecteur. Dans bien des cas, $\Delta\mathbf{A}$ et $\Delta\mathbf{B}$ se réduiront à une valeur inférieure au 12ème chiffre de chaque élément de **A** et **B**.

Dans le cas d'une solution calculée **Z**, le *tableau résiduel* est $\mathbf{R} = \mathbf{B} - \mathbf{AZ}$. Ensuite $\|\mathbf{R}\| \leqslant \epsilon\|\mathbf{A}\|\ \|\mathbf{Z}\|$; le tableau résiduel d'une solution calculée est petit. Cependant, l'*erreur* $\mathbf{Z} - \mathbf{X}$ peut ne pas être petite si **A** n'est pas conditionné de manière favorable, c'est-à-dire si $\|\mathbf{Z} - \mathbf{X}\| \leqslant \epsilon\|\mathbf{A}\|\ \|\mathbf{A}^{-1}\|\ \|\mathbf{Z}\|$.

Une règle simple pour estimer la précision de la solution calculée est :

(nombre de chiffres corrects)
$$\geqslant \text{(nombre de chiffres utilisés)} - \log\left(\|\mathbf{A}\|\ \|\mathbf{A}^{-1}\|\right) - \log 10n$$

dans laquelle n est la dimension de **A**. Pour le HP-28C/S, qui permet de représenter un nombre avec une précision de 12 chiffres,

$$\text{(nombre de chiffres corrects)} \geqslant 11 - \log\left(\|\mathbf{A}\|\ \|\mathbf{A}^{-1}\|\right) - \log n.$$

...ARRAY

Dans beaucoup d'applications, il se peut que la précision soit suffisante. Lorsqu'une précision plus grande est nécessaire, la solution calculée **Z** peut habituellement être améliorée par *itérations* (par corrections successives). Ce procédé comporte le calcul de la solution d'un système d'équations ; sa précision est ensuite améliorée en utilisant le tableau résiduel associé avec la solution pour améliorer cette solution.

Pour utiliser ces améliorations itératives, calculez d'abord une solution **Z** pour le système d'origine **AX** = **B**. Ensuite traitez **Z** comme une approximation de **X**, avec une erreur de type **E** = **X** − **Z**. **E** satisfait alors le système linéaire

$$\mathbf{AE} = \mathbf{AX} - \mathbf{AZ} = \mathbf{R},$$

dans lequel **R** est le tableau résiduel de **Z**. L'étape suivante consiste à calculer le tableau résiduel et à calculer **E** telle que **AE** = **R**. La solution obtenue par calcul, notée **F**, est traitée comme une estimation de **E** et est ajoutée à **Z** pour obtenir une nouvelle estimation de **X**.

Pour que **F** + **Z** soit une meilleure estimation de **X** que **Z**, le tableau résiduel **R** = **B** − **AZ** doit être calculé avec une précision étendue. C'est ce qu'effectue la fonction RSD (voir la description ci-après pour les détails de son utilisation).

Le procédé d'affinement peut être répété, mais l'amélioration se produit dans le premier affinement. La fonction / n'essaie pas d'effectuer un affinement à cause de la quantité de mémoire nécessaire pour conserver plusieurs copies des tableaux originaux. Voici un exemple de programme-utilisateur qui résout une équation matricielle, avec un affinement utilisant la fonction RSD :

```
« → B A « B A / B A 3 PICK RSD A / + » »
```

Ce programme prélève deux tableaux-arguments **B** et **A** de la pile opérationnelle, les mêmes que /, et donne le tableau-résultat **Z**, qui sera une meilleure approximation de la solution **X** que celle fournie par / elle-même.

INV — *Inverse* — Fonct. analyt.

Niveau 1	Niveau 1
[*matrice*] ➡	[*matrice*$^{-1}$]

INV (■ 1/x) donne la matrice inverse de son argument. L'argument doit être une matrice carrée réelle ou complexe.

SQ — *Carré* — Fonct. analyt.

Niveau 1	Niveau 1
[*matrice*$_1$] ➡	[*matrice*$_2$]

SQ (SQUARE, ■ x²) donne le produit matriciel d'une matrice carrée par elle-même.

NEG — *Changement de signe* — Fonct. analyt.

Niveau 1	Niveau 1
[*tableau*] ➡	[−*tableau*]

Appuyer sur CHS lorsqu'aucune ligne de commande n'est affichée exécute la fonction NEG (NEGATE), qui donne l'opposé de son argument. Pour un tableau, chaque élément du résultat est l'opposé de l'élément correspondant du tableau-argument.

Pour saisir la fonction NEG dans la ligne de commande, utilisez NEG (dans le quatrième jeu de touches de menu de ARRAY).

...ARRAY

→ARRY ARRY→ PUT GET PUTI GETI

Ce groupe de commandes permet de rappeler ou de modifier individuellement certains éléments d'un tableau.

→ARRY *Pile vers tableau* Commande

Niveau *nm*+1 ... Niveau 2 Niveau 1		Niveau 1
$x_1 \ \ldots \ x_n$	{ *n* } ➡	[*vecteur*]
$x_{11} \ \ldots \ x_{nm}$	{ *n m* } ➡	[*matrice*]

→ARRY donne un tableau composé d'éléments réels ou complexes prélevés un par un dans la pile, aux niveaux 2 et supérieurs. →ARRY prélève une liste représentant la taille du tableau-résultat dans le niveau 1 :

Vecteurs. Si la liste contient un entier simple n, n nombres sont retirés de la pile et un vecteur à n éléments est renvoyé.

Matrices. Si la liste contient deux entiers, n et m, les nombres nm sont retirés de la pile et renvoyés en tant qu'éléments d'une matrice $n \times m$.

Les éléments du tableau-résultat devraient être introduits dans la pile par numéro d'ordre, avec x_{11} (ou x_1) au niveau $nm + 1$ (ou $n + 1$) et x_{nm} (ou x_n) au niveau 2. Si l'un ou plus des éléments est un nombre complexe, le tableau résultat sera complexe.

ARRY→ *Tableau vers pile* Commande

Niveau 1		Niveau n+1 ... Niveau 2 Niveau 1
[*vecteur*]	➡	$x_1 \ \ldots \ x_n$ { *n* }
[*matrice*]	➡	$x_{11} \ \ldots \ x_{nm}$ { *n m* }

ARRY→ prend un tableau dans la pile opérationnelle et renvoie ses éléments vers la pile sous forme de nombres individuels réels ou complexes. ARRY→ renvoie aussi une liste représentant la taille du tableau vers le niveau 1. Les éléments sont placés dans la pile selon leur numéro d'ordre :

Vecteurs. Si l'argument est un vecteur à n éléments, le premier élément est renvoyé vers le niveau $n + 1$ et le $n$ième élément est renvoyé vers le niveau 2. Le niveau 1 contiendra la liste $\{ n \}$.

Matrices. Si l'argument est une matrice $n \times m$, l'élément x_{nm} est renvoyé vers le niveau 2 et l'élément x_{11} est renvoyé vers le niveau $(nm + 1)$.

PUT — *Placer élément* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
[*tableau*$_1$]	{ *index* }	*x*	➧	[*tableau*$_2$]
[*tableau*$_1$ *C*]	{ *index* }	*z*	➧	[*tableau*$_2$ *C*]
' *nom* '	{ *index* }	*z*	➧	
{ *liste*$_1$ }	*n*	*objet*	➧	{ *liste*$_2$ }
' *nom* '	*n*	*objet*	➧	

PUT stocke un élément dans un tableau ou une liste. Cette section du manuel décrit son utilisation avec un tableau ; vous lirez comment l'utiliser avec une liste dans « LIST ».

PUT prélève trois arguments dans la pile opérationnelle. Le niveau 1 doit contenir le nombre que vous désirez stocker dans le tableau. Si le nombre est complexe le tableau doit l'être aussi.

Le niveau 2 contient l'index de l'élément de tableau que vous désirez remplacer. Si le tableau est un vecteur, l'index comporte un nombre entier définissant le numéro de l'élément (identique, pour les vecteurs, au numéro d'ordre). Si le tableau est une matrice, l'index a la forme { *n° ligne n° colonne* }.

...ARRAY

Le niveau 3 peut contenir soit un tableau, soit un nombre. Si le niveau 3 contient un tableau, PUT renvoie ce tableau vers la pile après avoir remplacé un élément par le nombre pris au niveau 1.

Si le niveau 3 contient un nom, PUT stocke le nombre du niveau 1 comme un des éléments du tableau contenu dans le *nom* de la variable (*nom* ne peut être un nom local).

GET est l'opération inverse de PUT.

GET ***Appeler un élément*** **Commande**

Niveau 2	Niveau 1		Niveau 1
[*tableau*]	{ *index* }	➡	*z*
' *nom* '	{ *index* }	➡	*z*
{ *liste* }	*n*	➡	*objet*
' *nom* '	*n*	➡	*objet*

GET est un mécanisme conçu pour rappeler un élément d'un tableau ou d'une liste. Dans cette section du manuel, nous décrivons le rappel d'un élément d'un tableau. Le rappel d'un élément d'une liste est décrit sous « LIST ».

GET prend deux arguments de la pile opérationnelle. Le niveau 1 devrait contenir un index identifiant l'élément du tableau que vous désirez rappeler. Si le tableau est un vecteur, l'index doit contenir un entier spécifiant le numéro de l'élément. Si le tableau est une matrice, l'index doit avoir la forme { *n° ligne n° colonne* }.

Le niveau 2 peut contenir soit un tableau, soit un nom. Si le niveau 2 contient un tableau, GET renvoie dans la pile l'élément indexé de ce tableau. Si le niveau 2 contient un nom, GET renvoie dans la pile l'élément indexé du tableau contenu dans la variable *nom*.

PUT est l'opération inverse de GET.

PUTI ***Placer élément et incrémenter index*** **Commande**

Niveau 3	Niveau 2	Niveau 1		Niveau 2	Niveau 1
⊏ *tableau*$_1$ ⊐	{ *index*$_1$ }	*x*	➧	⊏ *tableau*$_2$ ⊐	{ *index*$_2$ }
⊏ *tableau*$_1$ *C* ⊐	{ *index*$_1$ }	*z*	➧	⊏ *tableau*$_2$ *C* ⊐	{ *index*$_2$ }
' *nom* '	{ *index*$_1$ }	*z*	➧	' *nom* '	{ *index*$_2$ }
{ *liste*$_1$ }	n_1	*objet*	➧	{ *liste*$_2$ }	n_2
' *nom* '	n_1	*objet*	➧	' *nom* '	n_2

PUTI est un mécanisme de stockage des éléments dans un tableau ou de stockage d'un objet dans une liste. Ce dernier est décrit en détail dans « LIST ».

PUTI stocke un nombre dans un tableau de la même manière que PUT, mais elle renvoie aussi le tableau (ou le nom de la variable) et l'index des éléments, incrémenté d'une unité. La pile opérationnelle est ainsi prête pour la saisie d'un nouveau nombre ; après quoi vous pouvez exécuter PUTI une fois encore pour stocker le nombre dans l'élément suivant du tableau. Lorsque l'index des éléments est égal à l'index maximal du tableau, PUTI renvoie l'index {1} (pour les vecteurs) ou {1 1} (pour les matrices), recommençant au début du tableau.

PUTI prend trois arguments dans la pile opérationnelle. Le niveau 1 doit contenir le nombre que vous désirez stocker dans le tableau. Si le nombre est complexe, le tableau doit lui aussi être complexe.

Le niveau 2 contient l'index identifiant le numéro de l'élément du tableau que vous désirez remplacer. Si le tableau est un vecteur, l'index doit contenir un nombre entier définissant le numéro de l'élément. Si le tableau est une matrice, l'index doit avoir la forme *{ n° ligne n° colonne }*.

Le niveau 3 peut contenir soit un tableau, soit un nom. S'il contient un tableau, PUTI renvoie le tableau au niveau 2, l'élément indexé étant remplacé par le nombre pris au niveau 1. L'index suivant est envoyé au niveau 1.

Si le niveau 3 contient un nom, PUTI stocke le nombre du niveau 1 en tant que *n*ième élément du tableau contenu dans la variable *nom*. Le nom est renvoyé au niveau 2 et l'index suivant au niveau 1.

GETI est l'opération inverse de PUTI.

GETI — *Appeler élément et incrémenter index* — Commande

Niveau 2	Niveau 1		Niveau 3	Niveau 2	Niveau 1
[*tableau*]	{ $index_1$ }	➡	[*tableau*]	{ $index_2$ }	*z*
' *nom* '	{ $index_1$ }	➡	' *nom* '	{ $index_2$ }	*z*
{ *liste* }	n_1	➡	{ *liste* }	n_2	*objet*
' *nom* '	n_1	➡	' *nom* '	n_2	*objet*

GETI est un mécanisme permettant le rappel d'un élément pris dans un tableau ou une liste. Le rappel d'éléments d'une liste est traité dans « LIST ».

GETI rappelle un élément d'un tableau de la même manière que GET, avec ceci de différent que GETI laisse le tableau sur la pile opérationnelle et fait passer l'index au numéro de l'élément suivant, pour faciliter les rappels successifs à partir d'un même tableau. Lorsque l'index des éléments est égal à l'index maximal du tableau, GETI renvoie l'index {1} (pour les vecteurs) ou {1 1} (pour les matrices), recommençant au début du tableau.

GETI prend deux arguments dans la pile opérationnelle. Le niveau 1 devrait contenir un index définissant l'élément du tableau que vous désirez rappeler, sous forme de liste d'un ou deux nombre entiers. Si le tableau est un vecteur, l'index doit contenir un entier qui définit le numéro de l'élément. Si le tableau est une matrice, l'index doit avoir la forme *{n° ligne n° colonne }*.

Le niveau 2 peut contenir soit un tableau, soit un nom. S'il contient un tableau, GETI renvoie la valeur indexée de ce tableau au niveau 1. GETI renvoie aussi le tableau au niveau 3 et l'index de l'élément suivant au niveau 2.

Si le niveau 2 contient un nom, GETI renvoie au niveau 1 l'élément indexé du tableau contenu dans la variable *nom*. GETI renvoie aussi le nom au niveau 3 et l'index de l'élément suivant au niveau 2.

PUTI représente l'opération inverse de GETI.

SIZE **RDM** **TRN** **CON** **IDN** **RSD**

SIZE ***Taille*** **Commande**

Niveau 1		**Niveau 1**
"chaîne"	➧	*n*
{ liste }	➧	*n*
[tableau]	➧	*{ liste }*
' symbole '	➧	*n*

...ARRAY

SIZE renvoie un objet représentant la taille — ou les dimensions — d'une liste, d'un tableau, d'une chaîne, ou d'un argument algébrique. Pour un tableau, SIZE donne une liste contenant un ou deux entiers :

- Si l'objet original est un vecteur, la liste contiendra un seul entier représentant le nombre d'éléments du vecteur.
- Si l'objet original est une matrice, la liste contiendra deux entiers représentant les dimensions de la matrice. Le premier entier est le nombre de lignes de la matrice et le second le nombre de colonnes.

Consultez les sections « STRING », « LIST » et « ALGEBRA » pour l'utilisation de SIZE avec d'autres types d'objets.

RDM ***Redimensionner*** **Commande**

Niveau 2	Niveau 1		Niveau 1
⌈ *tableau*$_1$ ⌋	{ *dim* }	➧	⌈ *tableau*$_2$ ⌋
' *nom* '	{ *dim* }	➧	

RDM (REDIMENSION) réorganise les éléments de *tableau*$_1$, pris au niveau 2 de la pile opérationnelle (ou contenu dans une variable *nom*) et donne *tableau*$_2$, qui a les dimensions spécifiées dans la liste d'un ou deux entiers prise au niveau 1 de la pile. Si le tableau se trouvant au niveau 2 est identifié par un nom, *tableau*$_2$ remplace *tableau*$_1$ en tant que contenu de la variable. Si la liste contient un seul entier n, *tableau*$_2$ est un vecteur à n éléments. Si la liste est de la forme $\{n\ m\}$, *tableau*$_2$ est une matrice $n \times m$.

Les éléments pris dans *tableau*$_1$ conservent les mêmes numéros d'ordre dans *tableau*$_2$. Si *tableau*$_2$ est dimensionné pour contenir moins d'éléments que *tableau*$_1$, les éléments de *tableau*$_1$ qui sont en excédent (et qui ont les derniers numéros d'ordre) sont supprimés. Si *tableau*$_2$ est dimensionné pour contenir plus d'éléments que *tableau*$_1$, les éléments excédentaires de *tableau*$_2$ (qui ont les derniers numéros d'ordre) sont remplis de zéros ((0, 0) si *tableau*$_1$ est complexe).

TRN — *Transposer* — Commande

Niveau 1		Niveau 1
⌈ *matrice*$_1$ ⌋	➧	⌈ *matrice*$_2$ ⌋
' *nom* '	➧	

TRN (TRANSPOSE) donne la transposée de son argument. C'est-à-dire qu'une matrice $n \times m$ **A** au niveau 1 (ou contenue dans *nom*) est remplacée par une matrice $m \times n$ $\mathbf{A}^t$, avec

$$\mathbf{A}^t{}_{ij} = \begin{cases} \mathbf{A}_{ji} \text{ pour les matrices réelles,} \\ \text{CONJ } (\mathbf{A}_{ji}) \text{ pour les matrices complexes.} \end{cases}$$

Si la matrice est identifiée par un nom, $\mathbf{A}^t$ remplace **A** dans *nom*.

CON — *Tableau constant* — Commande

Niveau 2	Niveau 1		Niveau 1
{ *dim* }	*z*	➧	⌈ *tableau* ⌋
⌈ *tableau*$_1$ ⌋	*x*	➧	⌈ *tableau*$_2$ ⌋
⌈ *tableau*$_1$ *C* ⌋	*z*	➧	⌈ *tableau*$_2$ *C* ⌋
' *nom* '	*z*	➧	

CON (CONSTANT ARRAY) donne un tableau *constant* — un tableau dont tous les éléments ont la même valeur. La valeur constante est le nombre réel ou complexe pris dans le niveau 1 de la pile opérationnelle. Le tableau résultat est soit un nouveau tableau, soit un tableau existant dont les éléments ont été remplacés par la valeur constante, selon l'objet qui se trouve au niveau de la pile.

Création d'un nouveau tableau. Si le niveau 2 contient une liste d'un ou deux entiers, un nouveau tableau est envoyé dans la pile. Si la liste contient un seul entier *n*, le résultat est un vecteur constant à *n* éléments. Si la liste est de la forme {*n m*}, le résultat est une matrice constante à *n* lignes et *m* colonnes.

Remplacement des éléments d'un tableau existant. Si le niveau 2 contient un nom, ce nom doit identifier une variable utilisateur contenant un tableau. Dans ce cas, les éléments du tableau sont remplacés par la constante prise dans le niveau 1 de la pile. Si la constante est un nombre complexe, le tableau initial doit être complexe.

Si le niveau 2 contient un tableau, le résultat obtenu est un tableau de mêmes dimensions dans lequel chaque élément est égal à la valeur constante. Si la constante est un nombre complexe, le tableau initial doit être complexe.

IDN — *Matrice unité (I)* — Commande

Niveau 1		Niveau 1
n	➧	⊏ *matrice unité réelle* ⊐
⊏ *matrice* ⊐	➧	⊏ *matrice unité* ⊐
' *nom* '	➧	

IDN (IDENTITY MATRIX) donne une matrice *unité* — une matrice carrée dont les éléments diagonaux sont égaux à 1 et les autres égaux à 0. La matrice résultat est soit une nouvelle matrice, soit une matrice carrée existante dont les éléments sont remplacés par ceux de la matrice unité, selon l'argument qui se trouve au niveau 1 de la pile.

Création d'une nouvelle matrice. Si l'argument est un nombre réel, le résultat envoyé dans la pile est une nouvelle matrice unité réelle, dont les nombres de lignes et de colonnes sont les mêmes que ceux de l'argument.

Remplacement des éléments d'une matrice existante. Si l'argument est un nom, ce nom doit représenter une variable utilisateur contenant un matrice carrée. Les éléments de la matrice sont alors remplacés par ceux de la matrice unité (complexe si la matrice initiale est complexe).

Si l'argument est une matrice carrée, le résultat obtenu dans la pile est une matrice unité de mêmes dimensions. Si la matrice initiale est complexe, la matrice unité obtenue est également complexe, avec des valeurs diagonales (1,0).

RSD — *Tableau résiduel* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
[*tableau* **B**]	[*matrice* **A**]	[*tableau* **Z**]	➡	[*tableau* **B**−**AZ**]

RSD (RESIDUAL) calcule le *tableau résiduel* **B** − **AZ** de trois tableaux **B**, **A** et **Z**. RSD est typiquement utilisée pour calculer une correction sur **Z**, lorsque **Z** a été obtenu comme approximation de la solution **X** du système d'équations **AX** = **B**. Consultez « Amélioration de la précision des solutions des systèmes », en page 111, pour une description de l'utilisation de RSD avec des systèmes d'équations.

A, **B** et **Z** sont soumis aux 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 être tous deux des vecteurs ou des matrices.
- **B** et **Z** doivent avoir le même nombre de colonnes, si ce sont des matrices.

...ARRAY

CROSS DOT DET ABS RNRM CNRM

CROSS *Produit vectoriel* Commande

Niveau 2	Niveau 1	Niveau 1
⌊ *vecteur* **A** ⌋	⌊ *vecteur* **B** ⌋ ➡	⌊ *vecteur* **A** × **B** ⌋

CROSS (CROSS PRODUCT) donne le produit vectoriel **A** × **B** de deux vecteurs **A** et **B** à trois éléments chacun, avec

$$(\mathbf{A} \times \mathbf{B})_1 = \mathbf{A}_2\mathbf{B}_3 - \mathbf{A}_3\mathbf{B}_2$$
$$(\mathbf{A} \times \mathbf{B})_2 = \mathbf{A}_3\mathbf{B}_1 - \mathbf{A}_1\mathbf{B}_3$$
$$(\mathbf{A} \times \mathbf{B})_3 = \mathbf{A}_1\mathbf{B}_2 - \mathbf{A}_2\mathbf{B}_1$$

DOT *Produit scalaire* Commande

Niveau 2	Niveau 1	Niveau 1
⌊ *tableau* **A** ⌋	⌊ *tableau* **B** ⌋ ➡	*x*

DOT (DOT PRODUCT) donne le produit scalaire **A**·**B** de deux tableaux **A** et **B**, calculé comme la somme des produits des éléments correspondants de deux tableaux. Par exemple :
`[1 2 3] [4 5 6] DOT` donne 1 × 4 + 2 × 5 + 3 × 6, soit 32.

Le produit scalaire de deux tableaux complexes est parfois défini comme la somme des produits des éléments conjugués d'un tableau et de leurs éléments correspondants dans l'autre tableau. Le HP-28C/S, lui, utilise les produits ordinaires, sans les conjugués. Cependant, si vous préférez l'autre définition du produit scalaire, vous pouvez exécuter CONJ sur un des tableaux, ou les deux, avant d'exécuter DOT.

DET — *Déterminant* — Commande

Niveau 1		Niveau 1
[*matrice*]	➡	*déterminant*

DET (DETERMINANT) donne le déterminant de son argument, qui doit être une matrice carrée.

ABS — *Valeur absolue* — Fonction

Niveau 1		Niveau 1
z	➡	|*z*|
[*tableau*]	➡	‖*tableau*‖
' *symbole* '	➡	' ABS (*symbole*) '

ABS (ABSOLUTE VALUE) donne la valeur absolue de son argument. Dans le cas d'un tableau, ABS donne la norme de Frobenius (ou euclidienne) du tableau, définie comme la racine carrée de la somme des carrés des valeurs absolues de tous les éléments.

Consultez les rubriques « REAL », « COMPLEX » et « ALGEBRA » pour l'utilisation de ABS avec d'autres types d'objets.

RNRM — *Norme de ligne* — Commande

Niveau 1		Niveau 1
[*tableau*]	➡	*norme de ligne*

RNRM (ROW NORM) donne la norme de ligne de son argument. La norme de ligne est la plus grande des sommes obtenues en additionnant, dans chaque ligne, les valeurs absolues de tous les éléments. Pour un vecteur, la norme de ligne est la plus grande valeur absolue de l'un des éléments.

...ARRAY

CNRM — *Norme de colonne* — Commande

Niveau 1	Niveau 1
[*tableau*] ➡	*norme de colonne*

CNRM (COLUMN NORM) donne la norme de colonne de son argument, c'est-à-dire la plus grande des sommes obtenues en additionnant, dans chaque colonne, les valeurs absolues de tous les éléments. Pour un vecteur, la norme de colonne est la somme des valeurs absolues de ses éléments.

R→C C→R RE IM CONJ NEG

R→C — *Transformer réels en complexe* — Commande

Niveau 2	Niveau 1		Niveau 1
x	*y*	➡	(*x* , *y*)
[*tableau*$_1$ *R*]	[*tableau*$_2$ *R*]	➡	[*tableau C*]

R→C combine deux nombres réels ou deux tableaux réels en un seul nombre ou tableau complexe, respectivement. L'objet se trouvant au niveau 2 est utilisé comme partie réelle du résultat, et l'objet au niveau 1 comme partie imaginaire.

Lorsque les arguments sont des tableaux, les éléments du tableau résultat complexe sont des nombres complexes, dont les parties réelles et imaginaires sont les éléments correspondants des tableaux-arguments se trouvant aux niveaux 2 et 1 respectivement. Les tableaux doivent avoir les mêmes dimensions.

C→R — *Transformer complexe en réels* — Commande

Niveau 1		Niveau 2	Niveau 1
(*x*, *y*)	➡	*x*	*y*
[*tableau C*]	➡	[*tableau*$_1$ *R*]	[*tableau*$_2$ *R*]

C→R renvoie respectivement au niveau 2 et au niveau 1 les parties réelle et imaginaire d'un nombre ou tableau complexe.

La partie réelle ou imaginaire d'un tableau complexe est un tableau réel de mêmes dimensions, dont les éléments sont les parties réelles ou imaginaires des éléments correspondants du tableau complexe.

RE — *Partie réelle* — Fonction

Niveau 1		Niveau 1
x	➡	*x*
(*x*, *y*)	➡	*x*
[*tableau R*]	➡	[*tableau R*]
[*tableau C*]	➡	[*tableau R*]
'*symbole*'	➡	'RE(*symbole*)'

RE (REAL PART) donne la partie réelle de son argument. Si l'argument est un tableau, RE donne un tableau réel dont les élément sont égaux aux parties réelles des éléments correspondants du tableau argument.

IM — *Partie imaginaire* — Fonction

Niveau 1		Niveau 1
x	➧	*0*
(*x*,*y*)	➧	*y*
[*tableau R*]	➧	[*tableau R nul*]
[*tableau C*]	➧	[*tableau R*]
'*symbole*'	➧	'IM(*symbole*)'

IM (IMAGINARY PART) donne la partie imaginaire de son argument. Si l'argument est un tableau, IM donne un tableau réel dont les éléments sont égaux aux parties imaginaires des éléments correspondants du tableau argument. Si le tableau argument est réel, tous les éléments du tableau résultat sont zéro.

CONJ — *Calcul du conjugué* — Fonct. analyt.

Niveau 1		Niveau 1
x	➧	*x*
(*x*,*y*)	➧	(*x*,−*y*)
[*tableau R*]	➧	[*tableau R*]
[*tableau*$_1$ *C*]	➧	[*tableau*$_2$ *C*]
'*symbole*'	➧	'CONJ(*symbole*)'

CONJ (CONJUGATE) donne le complexe conjugué d'un nombre ou tableau complexe. Le signe de la partie imaginaire d'un nombre complexe, ou de chaque élément d'un tableau complexe, est inversé. Pour les nombres ou tableaux réels, le conjugué est identique à l'argument initial.

NEG — *Changement de signe* — Fonct. analyt.

Niveau 1	Niveau 1
[*tableau*]	➧ [−*tableau*]

Dans le cas d'un tableau, chaque élément du tableau résultat est l'opposé de l'élément correspondant du tableau argument.

Lorsqu'aucune ligne de commande n'est présente, appuyer sur [CHS] entraîne l'exécution de la fonction NEG (NEGATE).

Vous trouverez dans la section « Arithmétique » un diagramme complet de la pile opérationnelle pour NEG (NEGATE).

BINARY

DEC	HEX	OCT	BIN	STWS	RCWS
RL	RR	RLB	RRB	R→B	B→R
SL	SR	SLB	SRB	ASR	
AND	OR	XOR	NOT		

Les *entiers binaires* sont des nombres entiers sans signe qui sont représentés de manière interne dans le HP-28C/S comme des nombres binaires de longueur allant de 1 à 64 octets. De tels nombres doivent être saisis et affichés comme une chaîne de chiffres précédée du délimiteur # .

La saisie et l'affichage d'entiers binaires est contrôlée par la *base* entière en cours qui peut être binaire (base 2), octale (base 8), décimale (base 10) ou hexadécimale (base 16). Si vous désirez changer la base en cours par l'une des touches de menu BIN , OCT , DEC ou HEX , la représentation interne d'un entier binaire dans la pile n'est pas changée mais les chiffres affichés changent et reflètent leur nouvelle base.

Les chiffres inclus dans une saisie d'entier binaire doivent respecter les conventions de la base en cours. Dans une base binaire, seuls les chiffres 0 et 1 sont tolérés ; dans une base octale, les chiffres de 0 à 7 ; dans une base décimale, de 0 à 9 ; enfin en base hexadécimale, les chiffres de 0 à 9 et les lettres de A à F. La base par défaut est la base 10.

Tous les entiers binaires saisis dans une même ligne de commande doivent l'être dans la même base (la base en cours). Les quatre touches de menu n'exécutent pas l'instruction ENTER, et vous pouvez donc changer la base même après avoir commencé à frapper en ligne de commande. La syntaxe des entiers binaires en ligne de commande est cependant vérifiée (en fonction de la base en cours) avant l'exécution de la ligne de commande ; vous ne pouvez donc pas saisir des nombres dans des bases différentes en incorporant les commandes de sélection des bases dans la ligne de commande.

L'affichage d'entiers binaires dans la pile est également affecté par la *longueur de mot* en cours, que vous pouvez régler de 1 à 64 octets par la commande STWS. Lorsqu'un entier binaire est affiché dans la pile, l'affichage ne montre que les bits de poids le plus faible à concurrence de la longueur de mot, même si le nombre n'a pas été tronqué. Si vous réduisez la longueur de mot, l'affichage se modifiera et montrera moins de bits, mais si vous augmentez ensuite la longueur de mot, les bits précédemment cachés seront affichés.

Le rôle principal de la longueur de mot est de contrôler les résultats que renvoient les commandes. Les commandes qui acceptent des arguments entiers binaires les tronquent jusqu'à ce qu'ils aient le nombre de bits (de poids le plus faible) spécifié par la longueur de mot en cours ; ils renvoient ensuite les résultats avec ce nombre de bits. La longueur de mot par défaut est 64 bits.

La base et la longueur de mot en cours sont encodés dans les indicateurs binaires-utilisateur 37 à 44. Les indicateurs 37 à 42 sont la représentaion binaire de la longueur de mot en cours moins 1 (l'indicateur 42 est le bit de poids le plus fort). Les indicateurs 43 et 44 déterminent la base en cours :

Indicateur 43	Indicateur 44	Base
0	0	Décimale
0	1	Binaire
1	0	Octale
1	1	Hexadécimale

En plus des commandes du menu BINARY décrites dans les sections suivantes, les fonctions arithmétiques +, −, * et / peuvent être utilisées avec des paires d'entiers binaires ou des combinaisons d'entiers réels et binaires, comme vous le lirez sous « Arithmétique ».

...BINARY

DEC HEX OCT BIN STWS RCWS

DEC — *Mode décimal* — Commande

DEC définit le mode décimal pour les opérations sur les entiers binaires. Ceux-ci peuvent comporter des chiffres allant de 0 à 9 ; ils seront affichés en base 10.

DEC désarme les indicateurs 43 et 44.

HEX — *Mode hexadécimal* — Commande

HEX définit le mode hexadécimal pour les opérations sur les entiers binaires. Ceux-ci peuvent comporter les chiffres de 0 à 9 et de A (dix) à F (quinze) ; ils seront affichés en base 16.

HEX arme les indicateurs-utilisateur 43 et 44.

OCT — *Mode octal* — Commande

OCT définit le mode octal pour les opérations sur les entiers binaires. Ceux-ci peuvent comporter les chiffres de 0 à 7 et seront affichés en base 8.

OCT arme l'indicateur-utilisateur 43 et désarme l'indicateur 44.

BIN — *Mode binaire* — Commande

➧

BIN définit le mode binaire pour les opérations sur les entiers binaires. Ceux-ci peuvent comporter les chiffres 0 et 1 et seront affichés en base 2.

BIN désarme l'indicateur-utilisateur 43 et arme l'indicateur 44.

STWS — *Stocker la longueur de mot* — Commande

Niveau 1	
n ➧	

STWS (STORE WORDSIZE) définit l'argument *n* comme longueur de mot en cours des entiers binaires, *n* étant un nombre réel entier compris entre 1 et 64. Si $n > 64$, la longueur de mot sera 64 ; si $n < 1$, la longueur de mot sera 1. Les indicateurs binaires d'utilisateur 37 à 42 sont la réprésentation binaire de $n - 1$ (l'indicateur 42 est le bit de poids fort).

RCWS — *Rappeler la longueur de mot* — Commande

	Niveau 1
➧ *n*	

RCWS (RECALL WORDSIZE) renvoie un entier binaire *n*, égal à la longuer de mot en cours de 1 à 64. Les indicateurs 37 à 42 sont la représentaion binaire de $n - 1$.

...BINARY

RL **RR** **RLB** **RRB** **R→B** **B→R**

Les commandes RL et RR provoquent une permutation circulaire des nombres entiers binaires (mis à la longueur de mot en cours) d'un bit vers la gauche ou vers la droite. Les commandes RLB et RRB sont équivalentes à RL ou à RR répétées huit fois. R→B et B→R convertissent les nombres réels en nombres binaires ou inversement.

RL ***Permuter d'un bit vers la gauche*** **Commande**

Niveau 1	Niveau 1
# n_1 ➧	# n_2

RL (ROTATE LEFT) exécute une permutation circulaire d'un bit vers la gauche sur un nombre entier binaire # n_1. Le bit situé à l'extrême gauche dans # n_1 devient le bit à l'extrême droite dans le résultat # n_2.

RR ***Permuter d'un bit vers la droite*** **Commande**

Niveau 1	Niveau 1
# n_1 ➧	# n_2

RR (ROTATE RIGHT) exécute une permutation circulaire d'un bit vers la droite sur un entier binaire # n_1. Le bit situé à l'extrême droite dans # n_1 devient le bit à l'extrême gauche dans le résultat # n_2.

RLB ***Permuter d'un octet vers la gauche*** **Commande**

Niveau 1	Niveau 1
# n_1 ➧	# n_2

RLB (ROTATE LEFT BYTE) exécute une permutation circulaire d'un octet vers la gauche sur un nombre entier binaire # n_1. L'octet le plus à gauche dans # n_1 devient celui à l'extrême droite dans le résultat # n_2.

RRB — *Permuter d'un octet vers la droite* — Commande

Niveau 1		Niveau 1
# n_1	➧	# n_2

RRB (ROTATE RIGHT BYTE) exécute une permutation circulaire d'un octet vers la droite sur un nombre entier # n_1. L'octet placé à l'extrême droite de # n_1 devient celui placé à l'extrême gauche du résultat, # n_2.

R→B — *Réel en binaire* — Commande

Niveau 1		Niveau 1
n	➧	# n

R→B convertit un nombre réel entier n, $0 \leqslant n \leqslant 1{,}84467440737E19$ en son équivalent entier binaire # n. Si $n < 0$, le résultat est # 0. Si $n > 1{,}84467440737E19$, le résultat est # FFFFFFFFFFFFFFFF (hex).

B→R — *Binaire en réel* — Commande

Niveau 1		Niveau 1
# n	➧	n

B→R convertit un entier binaire # n en son équivalent réel n. Si # $n >$ # 1000000000000 (décimal), seuls les 12 chiffres décimaux les plus significatifs sont conservés dans la mantisse du résultat.

...BINARY

SL SR SLB SRB ASR

Les commandes SL (SHIFT LEFT) et SR (SHIFT RIGHT) déplacent des entiers binaires (mis à la longueur de mot en cours) d'un bit vers la gauche ou vers la droite. Les bits ainsi placés en dehors de l'affichage disparaissent et sont perdus. Les commandes RLB et RRB (ci-dessus) équivalent à SL ou à SR répétées huit fois.

SL *Déplacer d'un bit vers la gauche* Commande

Niveau 1	Niveau 1
# n_1 ➧	# n_2

SL (SHIFT LEFT) exécute un déplacement d'un bit vers la gauche sur un entier binaire. Le bit le plus à gauche de n_1 est perdu. Le bit le plus à droite de n_2 est mis à zéro. SL est l'équivalent d'une multiplication binaire par deux (avec troncation à la longueur de mot en cours).

SR *Déplacer d'un bit vers la droite* Commande

Niveau 1	Niveau 1
# n_1 ➧	# n_2

SR (SHIFT RIGHT) exécute un déplacement d'un bit vers la droite sur un entier binaire. Le bit de droite de n_1 est perdu. Le bit de gauche de n_2 est mis à zéro. SR est équivalente à une division binaire par deux.

SLB *Déplacer d'un octet vers la gauche* Commande

Niveau 1	Niveau 1
# n_1	➧ # n_2

SLB (SHIFT LEFT BYTE) exécute un déplacement d'un octet vers la gauche sur un entier binaire. SLB est l'équivalent d'une multiplication par # 100 (hexadécimal), tronqué selon la longueur de mot en cours.

SRB *Déplacer d'un octet vers la droite* Commande

Niveau 1	Niveau 1
# n_1	➧ # n_2

SRB (SHIFT RIGHT BYTE) exécute un déplacement d'un octet vers la droite sur un entier binaire. SRB est équivalent à une division binaire par # 100 (hexadécimal).

ASR *Déplacement arithm. d'1 bit à droite* Commande

Niveau 1	Niveau 1
# n_1	➧ # n_2

ASR (ARITHMETIC SHIFT RIGHT) exécute un déplacement arithmétique d'un bit vers la droite sur un entier binaire. Dans un déplacement arithmétique, le bit de poids le plus fort conserve sa valeur et un déplacement vers la droite est exécuté sur la longueur de mot en cours moins 1 bit.

...BINARY

AND OR XOR NOT

Les commandes AND, OR, XOR et NOT peuvent être appliquées aux *indicateurs binaires* (nombres réels ou expressions algébriques) et aux entiers binaires. Dans le premier cas, les commandes agissent comme des opérateurs logiques qui combinent indicateurs vrais et faux. Pour les entiers binaires, les commandes exécutent des opérations logiques au niveau des bits de chaque argument.

La description suivante s'applique à l'utilisation des commandes avec des arguments entiers binaires. « PROGRAM TEST » décrit leur application aux indicateurs binaires.

AND *ET* **Fonction**

Niveau 2	Niveau 1		Niveau 1
# n_1	# n_2	➧	# n_3

AND renvoie le ET logique de deux arguments entiers binaires. Chaque bit du résultat est déterminé par les bits correspondants des deux arguments, comme indiqué dans ce tableau :

# n_1	# n_2	Résultat de AND : # n_3
0	0	0
0	1	0
1	0	0
1	1	1

OR — *OU* — Fonction

Niveau 2	Niveau 1		Niveau 1
# n_1	# n_2	➧	# n_3

OR renvoie le OU logique de deux arguments binaires entiers. Chaque bit du résultat est déterminé par les bits correspondants des deux arguments, en fonction du tableau ci-dessous :

# n_1	# n_2	Résultat de OR : # n_3
0	0	0
0	1	1
1	0	1
1	1	1

XOR — *OU exclusif* — Fonction

Niveau 2	Niveau 1		Niveau 1
# n_1	# n_2	➧	# n_3

XOR renvoie le OU exclusif logique de deux arguments entiers binaires. Chaque bit du résultat est déterminé par les bits correspondants des deux arguments, en fonction du tableau ci-dessous :

# n_1	# n_2	Résultat de XOR : # n_3
0	0	0
0	1	1
1	0	1
1	1	0

...BINARY

NOT ***NON*** **Fonction**

Niveau 1	Niveau 1
# n_1	➧ # n_2

NOT renvoie le complément de son argument. Chaque bit du résultat est le complément du bit correspondant de # n_1.

# n_1	Résultat de NOT : # n_2
0	1
1	0

Calcul différentiel et intégral

Le HP-28C/S peut exécuter des dérivations symboliques pour n'importe quelle expression algébrique (dans les limites de la mémoire disponible) et des intégrations numériques pour n'importe quelle procédure (pourvu qu'elle ait une syntaxe algébrique). Le calculateur peut en outre effectuer des intégrations symboliques d'expressions polynomiales. Pour les expressions plus générales, la commande ∫ permet de calculer une approximation de l'expression à l'aide d'une série de Taylor, puis d'intégrer symboliquement le polynôme résultat.

Dérivation

∂ **Dérivation** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
' $symbole_1$ '	' *nom* '	➧	' $symbole_2$ '

∂ (■ d/dx) calcule la dérivée d'une expression algébrique $symbole_1$ par rapport à une variable donnée *nom* (cette dernière ne peut pas être un nom local). La forme de l'expression résultat, $symbole_2$, varie selon que ∂ est exécutée comme une partie d'une expression algébrique, ou comme un objet « indépendant ».

Dérivation pas-à-pas dans les expressions algébriques

La fonction dérivée ∂ est représentée avec une syntaxe particulière dans les expressions algébriques :

' ∂*nom*‹*symbole*› ',

où *nom* est la variable de la dérivation et *symbole* l'expression sur laquelle s'effectue la dérivation.

...Calcul différentiel et intégral

Par exemple, `'∂X(SIN(Y))'` représente la dérivée de `SIN(Y)` par rapport à `X`. Lorsque l'expression globale est évaluée, la dérivation progresse d'un « pas » — le résultat est la dérivée de l'expression-argument, multipliée par une nouvelle sous-expression représentant la dérivée de son argument. Prenons un exemple pour clarifier les choses. Dérivons `SIN(Y)` par rapport à `X` en mode RADIANS, `Y` ayant la valeur `'X^2'` :

`'∂X(SIN(Y))' EVAL` donne `'COS(Y)*∂X(Y)'`.

Ceci est une stricte application de la *règle de la dérivation*. Cette description du comportement de ∂, ainsi que les propriétés générales de EVAL, suffisent pour la compréhension des résultats des évaluations suivantes de l'expression :

`EVAL` donne `'COS(X^2)*(∂X(X)*2*X^(2-1))'`,

`EVAL` donne `'COS(X^2)*(2*X)'`.

Evaluation complète de la dérivée

Lorsque ∂ est exécutée comme un objet individuel — c'est-à-dire comme une séquence

`'symbole' 'nom' ∂,`

et non comme une partie d'une expression algébrique, l'évaluation de l'expression se répète automatiquement jusqu'à ce que l'expression ne contienne plus de dérivée. Au cours de ce processus, si la variable *nom* de la dérivation a une valeur numérique, cette valeur remplacera le nom de la variable partout dans la forme finale de l'expression.

Pour comparer le comportement de ∂ avec celui de la dérivation pas-à-pas décrite au paragraphe précédent, prenons à nouveau comme exemple l'expression `'SIN(Y)'`, où `Y` a la valeur `'X^2'` :

`'SIN(Y)' 'X' ∂` donne `'COS(X^2)*(2*X)'`.

Toutes les étapes de la dérivation ont été effectuées directement, en une seule opération.

Ce qui détermine si la fonction ∂ effectue ou n'effectue pas l'évaluation automatique répétée, c'est la forme de l'argument qui se trouve au niveau 1 et qui définit la variable de la dérivation. Si cet argument est un nom, la dérivation complète directe est exécutée. Si cet argument est une expression algébrique contenant uniquement un nom, une étape seulement de la dérivation est effectuée. Normalement, les expressions algébriques contenant uniquement un nom sont automatiquement converties en noms. La syntaxe spéciale de ∂ permet toutefois d'utiliser cette exception comme le signal d'une dérivation complète ou pas-à-pas.

Dérivation des fonctions définies par l'utilisateur

Lorsque ∂ est appliquée à une fonction définie par l'utilisateur :

1. L'expression, composée du nom de la fonction et de son argument entre parenthèses, est remplacée par l'expression qui définit la fonction.
2. Les arguments de l'expression initiale sont remplacés par les noms locaux contenus dans la définition de la fonction.
3. La dérivée de la nouvelle expression est calculée.

Par exemple : définissez $F(a, b) = 2a + b$:

```
« → a b '2*a+b' » 'F' STO.
```

...Calcul différentiel et intégral

Dérivez ensuite 'F(X,X^2)' par rapport à X. La dérivation se déroule automatiquement comme suit :

1. 'F(X,X^2)' est remplacée par '2*a+b'.
2. X remplace a, et 'X^2' remplace b. L'expression est devenue '2*X+X^2'.
3. La dérivée de la nouvelle expression est calculée.
   - Si vous avez évalué 'àX(F(X,X^2))' le résultat est 'àX(2*X)+àX(X^2)'.
   - Si vous avez exécuté 'F(X,(X^2))' 'X' à, la dérivation se poursuit jusqu'au résultat final '2+2*X'.

Dérivées définies par l'utilisateur

Si ∂ est appliquée à une fonction du HP-28C/S qui n'a pas de dérivée intégrée, ∂ donne une dérivée formelle — c'est-à-dire une nouvelle fonction dont le nom est « der » suivi du nom de la fonction initiale. Par exemple, la définition du % contenue dans le HP-28C/S ne comporte pas de dérivée. Si vous effectuez la première étape de la dérivation de '%(X,Y)' par rapport à Z, vous obtenez

```
'der%(X,Y,àZ(X),àZ(Y))'
```

Chaque argument de la fonction % donne deux arguments pour la fonction der%. Dans cet exemple, l'argument X donne les arguments X et àZ(X), et l'argument Y donne les arguments Y et àZ(Y).

Vous pouvez continuer à dériver en créant une fonction représentant la dérivée. Voici une dérivée pour % :

```
« → x y dx dy '(x*dy+y*dx)/100' » 'der%' STO.
```

Avec cette définition, vous pouvez obtenir une dérivée correcte pour la fonction %. Par exemple :

```
'%(X,2*X)' 'X' à COLCT donne ',04*X'.
```

...Calcul différentiel et intégral

De même, si ∂ est appliquée à une fonction formelle définie par l'utilisateur (un nom suivi d'arguments entre parenthèses, pour lequel il n'existe pas de fonction définie par l'utilisateur en mémoire utilisateur), ∂ donne une dérivée formelle dont le nom est « der » suivi du nom de la fonction initiale définie par l'utilisateur. Par exemple, dériver une fonction formelle définie par l'utilisateur `'f(x1,x2,x3)'` par rapport à *x* donne

```
'derf(x1,x2,x3,∂x(x1),∂x(x2),∂x(x3))'
```

Intégration

∫ ***Intégrer*** **Commande**

Niveau 3	Niveau 2	Niveau 1		Niveau 2	Niveau 1
'symbole'	*'nom'*	*degré*	➧		*'intégrale'*
x	*{ nom a b }*	*précision*	➧	*intégrale*	*erreur*
'symbole'	*{ nom a b }*	*précision*	➧	*intégrale*	*erreur*
«programme»	*{ nom a b }*	*précision*	➧	*intégrale*	*erreur*
«programme»	*{ a b }*	*précision*	➧	*intégrale*	*erreur*

∫ donne soit une expression polynomiale représentant une intégrale symbolique indéfinie, soit deux nombres réels pour une intégrale numérique définie. La nature du résultat dépend des arguments. En général, ∫ nécessite trois arguments. Le niveau 3 contient l'objet à intégrer ; l'objet au niveau 2 détermine la forme de l'intégration ; l'objet au niveau 1 définit la précision de l'intégration.

...Calcul différentiel et intégral

Intégration symbolique

Dans certaines limites, ∫ permet une intégration symbolique. Elle peut donner l'intégrale exacte (indéfinie) d'une expression qui est un polynôme dans la variable d'intégration. Elle peut également donner une approximation de l'intégrale en utilisant une série de Taylor pour convertir l'expression à intégrer en un polynôme, puis en intégrant le polynôme.

Pour obtenir une intégrale symbolique, les arguments dans la pile doivent être :

3: *Expression à intégrer* (nom ou objet algébrique)
2: *Variable d'intégration* (nom)
1: *Degré du polynôme* (entier réel)

Le *degré du polynôme* définit l'ordre de l'approximation par la série de Taylor (ou l'ordre de *l'expression à intégrer*, s'il s'agit déjà d'un polynôme).

Intégration numérique

Pour obtenir une intégrale numérique, vous devez spécifier :

- L'expression à intégrer.
- La variable d'intégration.
- Les limites numériques de l'intégration.
- La précision de l'expression à intégrer, ou l'erreur considérée comme acceptable dans le résultat de l'intégration.

Utilisation d'une variable explicite d'intégration. Une intégration numérique, dans laquelle la variable est nommée à l'aide d'un nom apparaissant (généralement) dans la définition de l'objet utilisé comme expression à intégrer, est appelée *intégration avec variable explicite.* Nous décrivons au paragraphe suivant *l'intégration avec variable implicite,* dans laquelle la variable n'a pas à être nommée.

Pour l'intégration avec variable explicite, vous devez saisir les objets nécessaires comme suit :

3: *Expression à intégrer*
2: *Variable et limites d'intégration*
1: *Précision*

L'expression à intégrer est un objet qui peut être :

- Un nombre réel, représentant une expression constante à intégrer. Dans ce cas, la valeur de l'intégrale sera seulement :

  nombre (limite supérieure — limite inférieure).

- Une expression algébrique.
- Un programme. Le programme doit respecter la syntaxe algébrique — c'est-à-dire ne prendre aucun argument dans la pile, et y renvoyer un nombre réel.

La variable et les limites d'intégration doivent être au niveau 2, dans une liste de la forme :

« *nom limite-inf. limite-sup.* »,

où *nom* est un nom, et les limites des nombres réels.

La *précision* est un nombre réel qui spécifie la tolérance d'erreur de l'intégration ; cette tolérance est considérée comme l'erreur relative au cours de l'évaluation de l'expression à intégrer (la précision définit, dans le domaine de la variable d'intégration, l'intervalle sur lequel s'effectue la recherche de l'approximation de l'intégrale).

...Calcul différentiel et intégral

La précision est définie comme une erreur fractionnaire, c'est-à-dire

$$précision \geqslant \left| \frac{valeur\ exacte\ -\ valeur\ calculée}{valeur\ calculée} \right|$$

où *valeur* est la valeur de l'expression à intégrer en n'importe quel point de l'intervalle d'intégration. Même si l'expression à intégrer a une précision de 12 chiffres significatifs (ou proche de 12 chiffres), vous pouvez souhaiter réduire la durée de calcul en utilisant une moins grande précision (c'est-à-dire en augmentant la valeur de la différence entre valeurs exacte et calculée, car plus cette valeur est petite, plus l'approximation doit s'appliquer à un grand nombre de points sur l'intervalle).

La précision de l'expression à intégrer dépend principalement de trois considérations :

- La précision des constantes empiriques dans l'expression.
- La mesure dans laquelle l'expression peut décrire avec précision une situation physique.
- L'ampleur de l'erreur d'arrondi dans le processus interne d'évaluation de l'expression.

Les expressions comme $cos\ (x\ -\ sin\ x)$ sont des expressions purement mathématiques, qui ne contiennent aucune constante empirique. La seule limitation de la précision provient donc des erreurs d'arrondi qui peuvent s'accumuler en raison de la précision finie (12 chiffres) du processus d'évaluation numérique de l'expression. Pour l'intégration de telles expressions, vous pouvez naturellement spécifier une précision inférieure à la simple erreur d'arrondi (augmenter la différence entre valeurs exacte et calculée), afin de réduire le temps de calcul.

Lorsque l'expression à intégrer est liée à une situation physique réelle, d'autres considérations entrent en jeu. Vous devez alors vous demander si la précision que vous souhaitez pour l'intégrale calculée est justifiée par la précision de l'expression à intégrer elle-même. Par exemple, si l'expression à intégrer contient des constantes empiriques qui n'ont qu'une précision de 3 chiffres, rechercher une trop grande précision (une valeur inférieure à 1E-3, dans le cas présent) n'a probablement aucun sens.

...Calcul différentiel et intégral

En outre, pratiquement toutes les fonctions liées à une situation physique sont par essence imprécises, car elles ne représentent que le modèle mathématique d'un processus ou événement réel. Le modèle est typiquement une approximation qui néglige les effets de facteurs jugés sans importance en comparaison des facteurs pris en compte dans le modèle.

Pour illustrer l'intégration numérique, calculons

$$\int_1^2 exp\ x\ dx$$

avec une précision de 0,00001. La pile doit se présenter comme suit :

```
3: 'EXP(X)'
2: { X 1 2 }
1: ,00001
```

L'intégration numérique renvoie deux nombres dans la pile. La valeur de l'intégrale est donnée au niveau 2. L'erreur donnée au niveau 1 est la limite supérieure de l'erreur fractionnaire du calcul, pour laquelle normalement

$$erreur = précision \int |expr.\ à\ intégrer|$$

Si l'erreur est un nombre négatif, cela indique qu'il n'a pas été possible d'obtenir une convergence de l'approximation, et le résultat donné au niveau 2 est la dernière approximation calculée.

Pour l'intégrale de `'EXP(X)'` dans l'exemple ci-dessus, ∫ donne une valeur 4,67077 au niveau 2, et l'erreur 4,7E-5 au niveau 1.

...Calcul différentiel et intégral

Utilisation d'une variable implicite d'intégration. L'utilisation d'une variable explicite d'intégration vous permet de saisir l'expression à intégrer comme une expression algébrique ordinaire. Mais il est également possible de la saisir selon la logique polonaise inverse, ce qui peut réduire notablement le temps nécessaire au calcul en éliminant l'évaluation répétée du nom de la variable. On utilise pour cette méthode une variable *implicite* d'intégration. La pile doit se présenter comme suit :

3: *Expression à intégrer* (programme)
2: *Limites d'intégration* (liste)
1: *Précision* (nombre réel)

L'*expression à intégrer* doit être un programme qui prend un nombre réel dans la pile et y renvoie un autre nombre réel. ∫ évalue le programme pour chaque point d'approximation sur l'intervalle d'intégration. A chaque évaluation, ∫ place la valeur obtenue dans la pile. Le programme prend cette valeur et donne la valeur de l'expression à intégrer pour ce point.

Les *limites d'intégration* doivent être saisies comme une liste de deux nombres réels, sous la forme {*limite-inf. limite-sup.*}. La *précision* définit l'erreur fractionnaire du calcul, telle qu'elle est décrite dans la section précédente.

Pour évaluer par exemple l'intégrale :

$$\int_1^2 exp\ (x)\ dx$$

avec une précision de 0,00001, vous devez exécuter ∫ en ayant organisé la pile comme suit :

```
3: « EXP »
2: { 1 2 }
1: ,00001
```

Ceci donne la même valeur 4,67077 et la même précision 4,7E-5 que l'exemple de la section précédente, dans lequel nous avions utilisé une variable explicite d'intégration.

Série de Taylor

TAYLR ***Série de Taylor*** **Commande**

Niveau 3	Niveau 2	Niveau 1		Niveau 1
' *symbole*$_1$ '	' *nom* '	*n*	➡	' *symbole*$_2$ '

TAYLR (dans le menu ALGEBRA) calcule à l'aide d'une série de Taylor une approximation de l'objet algébrique *symbole*$_1$, à l'ordre *n* de la variable *nom*. L'approximation est évaluée au point *nom* = 0 (parfois appelée série de MacLaurin). L'approximation de Taylor de f(x) au point $x = 0$ est définie comme :

$$\sum_{i=1}^{n} \frac{x^i}{i!} \left(\frac{\partial^i}{\partial x^i} f(x) \right) \Bigg|_{x=0}$$

...Calcul différentiel et intégral

Translation du point d'évaluation

Si vous utilisez TAYLR simplement pour mettre un polynôme sous forme de puissances, le point d'évaluation n'a pas réellement d'importance car le résultat est exact. Mais si vous utilisez TAYLR pour obtenir une approximation d'une fonction mathématique, vous aurez peut-être besoin d'éloigner le point d'évaluation de zéro.

Par exemple, si vous êtes intéressé par le comportement d'une fonction dans une région particulière, l'approximation TAYLR sera plus utile si vous translatez le point d'évaluation vers cette région. En outre, si la fonction n'a pas de dérivée pour zéro, son approximation TAYLR n'aura aucun sens si vous ne translatez pas le point de façon à l'éloigner de zéro.

Remarque L'exécution de TAYLR peut donner un résultat qui n'a aucune signification si l'expression n'est pas différentiable pour zéro. Par exemple, si vous désarmez l'indicateur binaire 59 (pour empêcher l'apparition de messages d'erreur `Infinite Result` — résultat infini) et exécutez :

```
'∫X' 'X' 2 TAYLR
```

vous obtiendrez le résultat `'5,E499*X-1,25E499*X^2'`. Le coefficient de `X` est `∂X(X^,5)`, ce qui est égal à `,5 * X^ - ,5`, dont l'évaluation donne 5,E499 pour $x = 0$.

Bien que TAYLR évalue toujours la fonction et ses dérivées pour zéro, vous pouvez effectivement éloigner le point d'évaluation de zéro en changeant les variables dans l'expression. Supposons par exemple que la fonction est une expression en X, et que vous recherchez l'approximation TAYLR pour X = 2. Pour translater le point d'évaluation en modifiant les variables :

1. Stockez `'Y+2'` dans `'X'`.
2. Evaluez la fonction initiale, pour faire passer la variable de X à Y.

3. Trouvez l'approximation de Taylor pour Y = 0.
4. Supprimez X (s'il existe toujours en tant que variable).
5. Stockez `'X-2'` dans `'Y'`.
6. Evaluez la nouvelle fonction pour faire passer la variable de Y à X.
7. Supprimez Y.

Approximations de fonctions rationnelles

Une *fonction rationnelle* est le quotient de deux polynômes. Si le dénominateur divise exactement (sans reste) le numérateur, la fonction rationnelle équivaut à un polynôme. Par exemple :

$$\frac{x^3 + 2x^2 - 5x - 6}{x^2 - x - 2} = x + 3$$

Si votre expression est une fonction rationnelle de ce genre, vous pouvez la convertir en un polynôme équivalent à l'aide de TAYLR. Mais si le dénominateur ne divise pas exactement le numérateur (s'il y a un reste), la fonction rationnelle *n'est pas* un polynôme. Par exemple :

$$\frac{x^3 + 2x^2 - 5x - 2}{x^2 - x - 2} = x + 3 + \frac{4}{x^2 - x - 2}$$

Une telle fonction rationnelle ne peut pas être exprimée sous la forme d'un polynôme, mais vous pouvez utiliser TAYLR pour calculer un polynôme qui est exact pour une petite valeur de x (proche de zéro). Vous pouvez éloigner de $x = 0$ la région de plus grande précision et choisir la précision de l'approximation. Dans l'exemple ci-dessus, l'approximation TAYLR de premier degré au point $x = 0$ est $2x + 1$.

Division euclidienne de polynômes. Une autre approximation utile d'une fonction rationnelle est le polynôme quotient d'une division euclidienne. Considérons le membre droit de l'équation ci-dessus comme un polynôme plus un reste. Le polynôme est une bonne approximation de la fonction rationnelle lorsque le reste est petit — c'est-à-dire lorsque x est grand. Notez la différence entre le polynôme quotient ($x + 3$) et l'approximation TAYLR de même degré ($2x + 1$).

Les étapes ci-dessous vous indiquent comment effectuer une division euclidienne de polynômes sur le HP-28C/S. Le processus général est le même que pour une division euclidienne sur des nombres.

1. Créez des expressions pour le numérateur et le dénominateur, toutes deux sous forme polynomiale.
2. Stockez le dénominateur dans une variable nommée « D » (pour « diviseur »).
3. Stockez une valeur initiale pour zéro dans une variable nommée « Q » (pour « quotient »).

Le numérateur se trouvant dans la pile, procédez comme indiqué ci-dessous. Le numérateur est la valeur initiale du dividende. Chaque fois que vous répétez les étapes 4 à 8, vous ajoutez un terme à Q et réduisez le dividende.

4. Placez D dans la pile (au niveau 1).
5. Divisez le terme d'ordre le plus élevé du dividende (au niveau 2) par le terme d'ordre le plus élevé du diviseur (au niveau 1). Vous pouvez calculer le résultat mentalement puis le saisir, ou vous pouvez saisir une expression

   '*terme-dividende*/*terme-diviseur*'

   et la mettre sous forme poynomiale.

   Par exemple, si le dividende est $x^3 + 2x^2 - 5x - 2$ et le diviseur $x^2 - x - 2$, le résultat est x ; si le dividende est $3x^3 + x^2 - 7$ et le diviseur $2x^2 + 8x + 9$, le résultat est $1{,}5x$.

   Le résultat est un terme du polynôme quotient.

6. Faites une copie du terme quotient, et ajoutez cette copie à Q.
7. Multipliez le terme quotient et le diviseur.
8. Soustrayez le résultat du dividende. Le résultat est le nouveau dividende.

Si le nouveau dividende est d'un degré supérieur à celui du diviseur, répétez les étapes 4 à 8.

Lorsque le nouveau dividende est d'un degré inférieur à celui du diviseur, arrêtez-vous. Le polynôme quotient est stocké dans Q, et le reste est égal au dernier dividende divisé par le diviseur.

CATALOG

Le catalogue des commandes du HP-28C/S, activé par la touche ■ CATALOG, est une liste alphabétique de toutes les commandes reconnues par le HP-28C/S. Le catalogue peut être utilisé pour vérifier l'existence et l'orthographe d'une commande et pour apprendre quels types d'objets peuvent être utilisés comme arguments pour chaque commande. Les commandes qui commencent par un caractère non-alphabétique sont regroupées après XPON, la dernière commande commençant par une lettre.

Lorsque vous appuyez sur ■ CATALOG, l'affichage normal du HP-28C/S est remplacé par celui du catalogue :

La ligne supérieure indique un nom de commande. ABORT est la première commande du catalogue.

...CATALOG

Le menu du catalogue apparaît sur la dernière ligne de l'affichage. Les six touches de menu ont les rôles suivants :

Touche	Description
NEXT	Affiche la commande suivante du catalogue (touche à répétition).
PREV	Affiche la commande précédente du catalogue (touche à répétition).
SCAN	Défilement automatique des commandes du catalogue, du début à la fin, montrant brièvement chaque commande. Le libellé de menu SCAN se change en STOP ; le fait d'apuyer sur STOP arrête le défilement à la commande affichée à ce moment. Le défilement s'arrêtera automatiquement à la dernière commande du catalogue (→STR).
USE	Active l'affichage USAGE (voir ci-dessous) pour la commande en cours, pour indiquer les arguments utilisés par cette commande.
FETCH	Fait sortir du catalogue, et place l'abréviation de la commande en cours dans la ligne de commande, à l'emplacement du curseur ; si aucune ligne de commande n'est présente, FETCH en crée une.
QUIT	Fait sortir du catalogue en laissant toute ligne de commande existante inchangée.

En plus des opérations disponibles dans le menu de catalogue des commandes, vous pouvez :

- Appuyer sur une touche du clavier de gauche pour amener l'affichage du catalogue à la première commande qui commence par la lettre ou le caractère indiqués.
- Si aucune commande ne commence par cette lettre, le catalogue passera à la dernière commande qui commence par la lettre précédente de l'alphabet.

...CATALOG

- S'il n'y a pas de commande commençant par le caractère non alphabétique que vous avez choisi, le catalogue avancera vers +, la première commande commençant par un caractère non-alphabétique (la touche ■ [α LOCK] fait avancer le catalogue jusqu'à →STR, la dernière rubrique du catalogue).
- Appuyez sur [ON] pour sortir du catalogue et effacer toute ligne de commande en cours.

Toutes les touches qui ne sont pas actives pendant l'affichage du catalogue émettront une tonalité lorsque vous appuyez sur elles.

USAGE

L'appui sur la touche correspondant au libellé de menu USE active un deuxième niveau du catalogue, nommé l'affichage USAGE (utilisation). Pour la commande % par exemple, l'affichage initial de USAGE aura l'aspect suivant :

L'affichage indique que % peut prendre deux nombres réels comme arguments. Lorsque les touches de menu PREV et NEXT sont présentes, c'est l'indication que d'autres options sont disponibles. Dans le cas de %, si vous appuyez sur NEXT, l'affichage devient :

Ceci indique que % acceptera également un nombre réel au niveau 2 et une expression algébrique ou un nom au niveau 1. Il y a deux autres combinaisons pour %, que vous pouvez visualiser en appuyant deux fois sur NEXT. Vous pouvez également revenir en arrière en utilisant la touche PREV.

Pour sortir de l'affichage USAGE, vous pouvez :

- Appuyer sur QUIT pour revenir à l'affichage du catalogue pour la commande en cours. De là, vous pourrez vous déplacer dans le catalogue vers d'autres commandes ou sortir en appuyant sur QUIT une deuxième fois.
- Appuyer sur ON pour sortir du catalogue. ON efface également la ligne de commande en cours.

COMPLEX

R→C	C→R	RE	IM	CONJ	SIGN
R→P	P→R	ABS	NEG	ARG	

Le menu COMPLEX (■ CMPLX) contient les commandes spécifiques aux nombres complexes.

Les *nombres complexes* sont des couples ordonnés de nombres représentés par deux nombres réels entre parenthèses, séparés par le caractère que vous n'avez pas utilisé comme séparateur décimal : (1,234.5,678) par exemple, si vous avez choisi la virgule comme séparateur décimal (c'est alors le point qui sépare les deux nombres ; Cf. p. 196). Un nombre complexe (x, y) peut représenter :

- Un nombre complexe z en notation rectangulaire, dans lequel x est la partie réelle de z et y la partie imaginaire.
- Un nombre complexe z en notation polaire, dans lequel x est la valeur absolue de z et y l'angle polaire.
- Les coordonnées d'un point en deux dimensions, en notation rectangulaire, avec x comme abscisse, ou coordonnée horizontale, et y comme ordonnée, ou coordonnée verticale.
- Les coordonnées d'un point en deux dimensions, en notation polaire, avec x comme coordonnée radiale et y comme angle polaire.

Si l'analyse des nombres complexes ne vous est pas familière, vous préférerez peut-être considérer les *nombres complexes* comme des vecteurs à deux dimensions ou des coordonnées de points. La plupart des commandes de nombres complexes renvoient des résultats qui sont significatifs aussi bien en géométrie bidimensionnelle que pour les nombres complexes.

Sauf la commande P→R (« polaire en rectangulaire »), toutes les commandes du HP-28C/S qui traitent des valeurs de nombres complexes pré-supposent que leurs arguments sont exprimés en notation rectangulaire. De même, toutes les commandes qui renvoient des résultats sous forme de nombres complexes, sauf R→P (« rectangulaire en polaire »), expriment leurs résultats sous forme rectangulaire.

...COMPLEX

Outre les commandes décrites dans les sections suivantes, certaines commandes d'autres menus acceptent des nombres complexes comme arguments :

- Fonctions arithmétiques +, −, *, /, INV, √, SQ, ^.
- Fonctions trigonométriques SIN, ASIN, COS, ACOS, TAN, ATAN.
- Fonctions hyperboliques SINH, ASINH, COSH, ACOSH, TANH, ATANH.
- Fonctions logarithmiques EXP, LN, LOG, ALOG.

R→C C→R RE IM CONJ SIGN

Les commandes R→C, C→R, RE, IM et CONJ apparaissent aussi en quatrième ligne du menu ARRAY. Pour leur utilisation avec des arguments sous forme de tableaux, consultez la section « ARRAY ».

R→C *Réel en complexe* Commande

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	(*x*, *y*)
⟦ *tableau*$_1$ *R* ⟧	⟦ *tableau*$_2$ *R* ⟧	➧	⟦ *tableau C* ⟧

R→C combine deux nombres réels x et y en un nombre complexe. x est la partie réelle et y la partie imaginaire du résultat. x et y peuvent aussi être considérés respectivement comme les coordonnées horizontale et verticale du point (x, y) dans un espace bidimensionnel.

...COMPLEX

C→R — *Complexe en réel* — Commande

Niveau 1		Niveau 2	Niveau 1
(*x*, *y*)	➧	*x*	*y*
[*tableau C*]	➧	[*tableau*$_1$ *R*]	[*tableau*$_2$ *R*]

C→R sépare un nombre complexe (ou un couple de coordonnées) en ses composants, renvoyant la partie réelle (ou coordonnée horizontale) en niveau 2 et la partie imaginaire (ou coordonnée verticale) en niveau 1.

RE — *Partie réelle* — Fonction

Niveau 1		Niveau 1
(*x*, *y*)	➧	*x*
' *symbole* '	➧	' (*symbole*) RE '
[*tableau*$_1$]	➧	[*tableau*$_2$]

RE renvoie la partie réelle *x* de son argument complexe (x, y). *x* peut aussi être considéré comme abscisse, ou coordonnée horizontale, du point (*x*, *y*).

IM — *Partie imaginaire* — Fonction

Niveau 1		Niveau 1
(*x*, *y*)	➧	*y*
' *symbole* '	➧	' (*symbole*) IM '
[*tableau*$_1$]	➧	[*tableau*$_2$]

IM renvoie la partie imaginaire *y* de son argument complexe (*x*, *y*). *y* peut aussi être considéré comme l'ordonnée, ou coordonnée verticale, du point (*x*, *y*).

CONJ

Calcul du conjugué

Fonct. analyt.

Niveau 1		Niveau 1
x	➡	*x*
(*x*, *y*)	➡	(*x*, −*y*)
[*tableau R*]	➡	[*tableau R*]
[*tableau*$_1$ *C*]	➡	[*tableau*$_2$ *C*]
'*symbole*'	➡	'CONJ(*symbole*)'

CONJ renvoie le complexe conjugué d'un nombre complexe. La partie imaginaire d'un nombre complexe change de signe.

SIGN

Signe

Fonction

Niveau 1		Niveau 1
z_1	➡	z_2
'*symbole*'	➡	'SIGN(*symbole*)'

Pour un argument complexe (x_1, y_1), SIGN renvoie le vecteur unité dans la direction de (x_1, y_1) :

$$(x_2, y_2) = \left(x_1/\sqrt{x_1^2 + y_1^2},\ y_1/\sqrt{x_1^2 + y_1^2}\right)$$

...COMPLEX

R→P **P→R** **ABS** **NEG** **ARG**

R→P

Rectangulaire en polaire **Fonction**

Niveau 1		Niveau 1
x	➧	(*x*, 0)
(*x*, *y*)	➧	(*r*, *θ*)
'*symbole*'	➧	'R→P(*symbole*)'

R→P prend un nombre complexe (x, y) en notation rectangulaire et le convertit en notation polaire (r, θ), avec

$$r = \text{abs}\ (x, y), \quad \theta = \arg\ (x, y).$$

P→R

Polaire en rectangulaire **Fonction**

Niveau 1		Niveau 1
(*r*, *θ*)	➧	(*x*, *y*)
'*symbole*'	➧	'P→R(*symbole*)'

P→R prend un nombre complexe (r, θ) en notation polaire et le convertit en notation rectangulaire (x, y), avec

$$x = r \cos \theta, \quad y = r \sin \theta.$$

ABS — *Valeur absolue* — Fonction

Niveau 1		Niveau 1
z	➧	|z|
⟦ *tableau* ⟧	➧	‖ tableau ‖
' *symbole* '	➧	' ABS (*symbole*) '

ABS renvoie la valeur absolue de son argument. Pour un argument complexe (x, y), la valeur absolue est $\sqrt{(x^2 + y^2)}$.

NEG — *Changement de signe* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	$-z$
' *symbole* '	➧	' − (*symbole*) '
⟦ *tableau* ⟧	➧	⟦ −*tableau* ⟧

NEG (NEGATE) renvoie l'opposé de son argument. Lorsqu'aucune ligne de commande n'est présente, une pression sur [CHS] (*Change Sign*) exécute NEG (diagramme de pile complet dans « Arithmétique »).

ARG — *Argument* — Fonction

Niveau 1		Niveau 1
z	➧	θ
' *symbole* '	➧	' ARG (*symbole*) '

ARG donne l'angle polaire θ d'un nombre complexe (x, y) où

$$\theta = \begin{cases} \text{arc tan } y/x & \text{pour } x \geqslant 0, \\ \text{arc tan } y/x + \pi \text{ signe } y & \text{pour } x < 0, \text{ mode RADIANS,} \\ \text{arc tan } y/x + 180 \text{ signe } y & \text{pour } x < 0, \text{ mode DEGRES.} \end{cases}$$

Le mode angulaire en cours détermine si θ est exprimé en degrés ou en radians.

...COMPLEX

Domaines principaux et solutions générales

En général l'inverse d'une fonction est une *relation* - pour n'importe quel argument, l'inverse a plus d'une valeur. Considérons par exemple $\cos^{-1} z$; il y a pour chaque z une infinité de w tels que $\cos w = z$. Pour des relations du type $\cos^{-1}$, le HP-28C/S définit des fonctions comme ACOS. Ces fonctions donnent une valeur principale qui se trouve dans la partie de l'intervalle de définition appelée *domaine principal de définition*, dans lequel s'inscrit la *branche principale* de la fonction.

Les domaines principaux de définition utilisés dans le HP-28C/S sont analytiques dans les régions où leurs équivalents à valeur réelle sont définis - c'est-à-dire que les bornes d'un domaine se situent là où l'inverse à valeur réelle est indéfini. Les domaines principaux conservent aussi la plupart des symétries importantes, comme $\text{ASIN}(-z) = -\text{ASIN}(z)$.

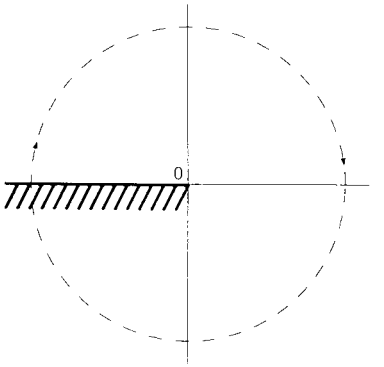
Les illustrations ci-après montrent les domaines principaux de √, LN, ASIN, ACOS, ATAN, ACOSH. Les représentations graphiques des intervalles de définition indiquent où se trouvent les bornes des intervalles : les traits continus bleus ou noirs se trouvent d'un côté de la borne, et les zones hachurées bleues ou noires de l'autre côté. Les représentations graphiques des domaines principaux de définition indiquent à quelle partie de la fonction correspond chaque côté de la borne. Dans les deux types de représentations graphiques, des pointillés vous aident à visualiser la fonction.

Les solutions générales, données par ISOL, sont également incluses (en supposant que l'indicateur binaire 34, valeur principale, est désarmé, et que le mode radians est sélectionné). Chaque solution générale est une expression qui représente les valeurs multiples des relations inverses.

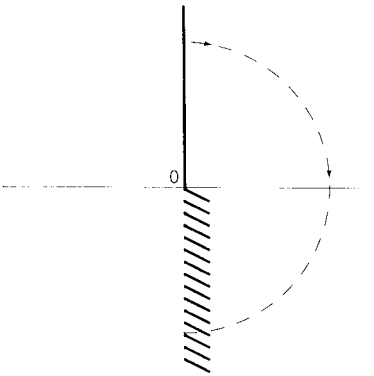
Les fonctions LOG, ^, ASINH et ATANH sont étroitement liées aux fonctions illustrées. Vous pouvez déduire des illustrations les valeurs principales de LOG, ^, ASINH et ATANH. Les solutions générales de ces fonctions sont également fournies.

Domaine principal de $\sqrt{Z}$

Intervalle de définition : `Z = (x,y)`



Valeur principale : `W = (u,v) = √(x,y)`

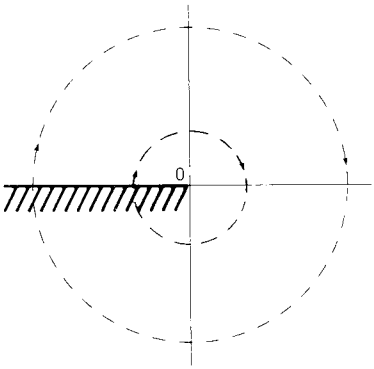


Solution générale : `'SQ(W)=Z' 'W' ISOL` donne `'±1*√Z'`.

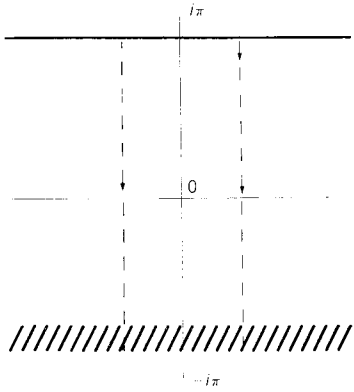
...COMPLEX

Domaine principal de LN(Z)

Intervalle de définition : Z = (x,y)



Valeur principale : W = (u,v) = LN(x,y)



Solution générale : 'EXP(W)=Z' 'W' ISOL donne 'LN(Z)+2*π*i*n1'.

Domaine principal de LOG(Z)

Vous pouvez déterminer le domaine principal de LOG à partir des illustrations de LN (en page précédente) et de la relation $\log(z) = \ln(z)/\ln(10)$.

Solution générale : `'ALOG(W)=Z' 'W' ISOL` donne
`'LOG(Z)+2*π*i*n1/2.30258509299'`

Domaine principal de U^Z

Vous pouvez déterminer le domaine principal des puissances complexes à partir des illustrations de LN (en page précédente) et de la relation $u^z = \exp(\ln(u)\, z)$.

Domaine principal de ASINH(Z)

Vous pouvez déterminer le domaine principal de ASINH à partir des illustrations de ASIN (en page suivante) et de la relation $\operatorname{asinh} z = -i \operatorname{asin} iz$.

Solution générale : `'SINH(W)=Z' 'W' ISOL` donne
`'ASINH(Z)+2*π*i*n1'`

Domaine principal de ATANH(Z)

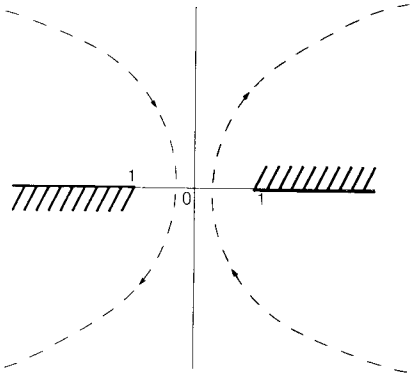
Vous pouvez déterminer le domaine principal de ATANH à partir des illustrations de ATAN (en page 172) et de la relation $\operatorname{atanh} z = -i \operatorname{atan} iz$.

Solution générale : `'TANH(W)=Z' 'W' ISOL` donne
`'ATANH(Z)+π*i*n1'`

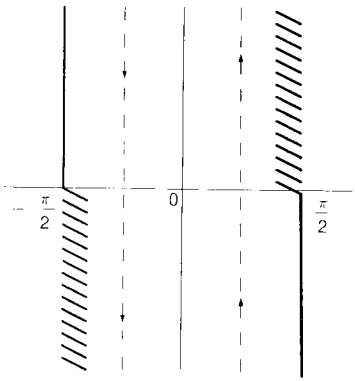
...COMPLEX

Domaine principal de ASIN(Z)

Intervalle de définition : Z = (x,y)



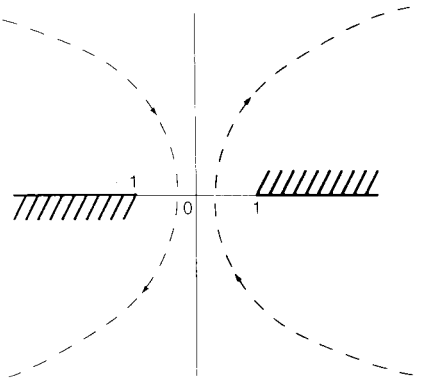
Valeur principale : W = (u,v) = ASIN(x,y)



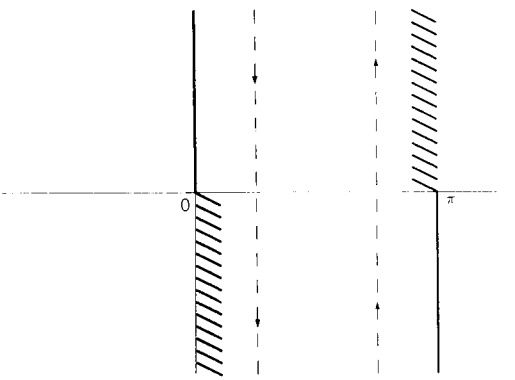
Solution générale : 'SIN(W)=Z' 'W' ISOL donne 'ASIN(Z)*(-1)^n1+π*n1'.

Domaine principal de ACOS(Z)

Intervalle de définition : Z = (x,y)



Valeur principale : W = (u,v) = ACOS(x,y)

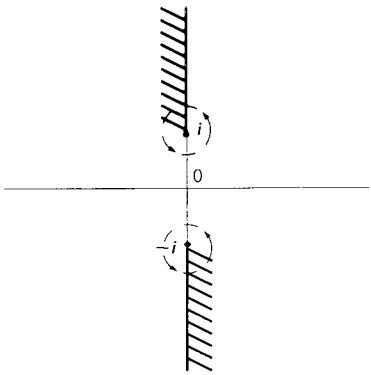


Solution générale : 'COS(W)=Z' 'W' ISOL donne
'≤1*ACOS(Z)+2*π*n1'

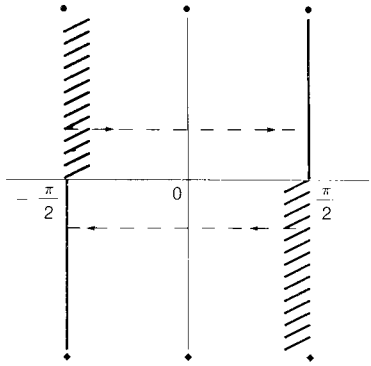
...COMPLEX

Domaine principal de ATAN(Z)

Intervalle de définition : Z = (x,y)



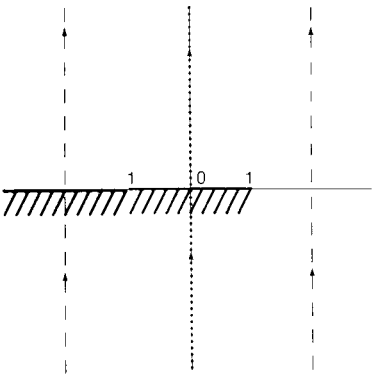
Valeur principale : W = (u,v) = ATAN(x,y)



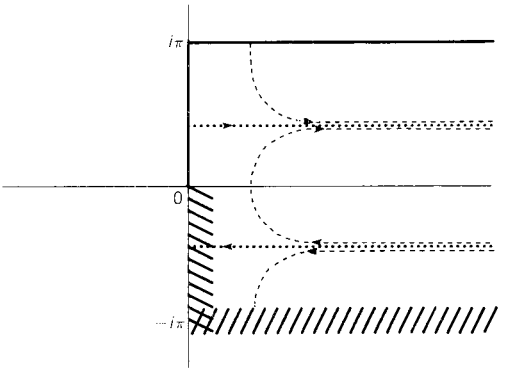
Solution générale : 'TAN(W)=Z' 'W' ISOL donne 'ATAN(Z)+π*n1'

Domaine principal de ACOSH(Z)

Intervalle de définition : Z = (x,y)



Valeur principale : W = (u,v) = ACOSH(x,y)



Solution générale : 'COSH(W)=Z' 'W' ISOL donne
's1*ACOSH(Z)+2*π*i*n1'

LIST

→LIST	LIST→	PUT	GET	PUTI	GETI
SUB	SIZE				

Une *liste* est un ensemble ordonné d'objets arbitraires ; elle est elle-même un objet et peut donc être introduite dans la pile ou stockée dans une variable. Les objets contenus dans la liste sont appelés *éléments*, et ils sont numérotés de gauche à droite en commençant par l'élément 1 à gauche. Les commandes du menu LIST vous permettent de créer et modifier les listes, et d'accéder aux objets contenus dans les listes.

Outre les commandes du menu LIST, vous pouvez aussi utiliser la fonction + du clavier pour combiner deux listes.

+ ***Addition*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
{ $liste_1$ }	{ $liste_2$ }	➡	{ $liste_1$ $liste_2$ }

+ réalise la concaténation de deux listes. C'est-à-dire qu'elle prend deux listes dans la pile et y renvoie une seule liste contenant les objets des deux listes initiales.

Vous trouverez un diagramme complet de la pile pour la fonction + dans la section « Arithmétique ».

→LIST LIST→ PUT GET PUTI GETI

→LIST ***Pile vers liste*** **Commande**

Niveau n+1 ... Niveau 2	Niveau 1		Niveau 1
$objet_1$ ... $objet_n$	n	➡	{ $objet_1$ ... $objet_n$ }

→LIST prend un nombre entier n dans le niveau 1, plus n objets supplémentaires dans les niveaux 2 à $n + 1$, et renvoie dans la pile une liste contenant les n objets.

→LIST est également disponible dans le menu STACK.

LIST→ *Liste vers pile* Commande

Niveau 1		Niveau n+1 ... Niveau 2	Niveau 1
{ $objet_1$ ... $objet_n$ }	➡	$objet_1$ ... $objet_n$	n

LIST→ prend une liste de n objets dans la pile et renvoie les objets constituant la liste dans les niveaux 2 à $n + 1$ de la pile. Le nombre n est renvoyé au niveau 1.

LIST→ est également disponible dans le menu STACK.

PUT *Placer élément* Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
[$tableau_1$]	{ *index* }	*x*	➡	[$tableau_2$]
' *nom* '	{ *index* }	*x*	➡	
[$tableau_1$ *C*]	{ *index* }	*z*	➡	[$tableau_2$ *C*]
' *nom* '	{ *index* }	*z*	➡	
{ $liste_1$ }	*n*	*objet*	➡	{ $liste_2$ }
' *nom* '	*n*	*objet*	➡	

...LIST

PUT est un mécanisme général de stockage d'éléments dans un tableau ou d'objets dans une liste. Nous ne décrirons ici que le stockage dans une liste. Le stockage dans un tableau est décrit à la section « ARRAY ».

PUT prend trois arguments dans la pile. Le niveau 1 doit contenir l'objet que vous voulez stocker dans la liste. Le niveau 2 contient un nombre réel spécifiant le numéro de l'élément que vous voulez remplacer dans la liste. Le niveau 3 peut contenir soit une liste soit un nom :

- Si le niveau 3 contient une liste, PUT renvoie cette liste dans la pile après en avoir remplacé le *n*ième élément par l'objet pris dans le niveau 1.
- Si le niveau 3 contient un nom, PUT remplace le *n*ième élément de la liste contenue dans la variable par l'objet se trouvant au niveau 1. La variable ne peut pas être une variable locale.

L'opération inverse de PUT est GET.

GET ***Appeler élément*** **Commande**

Niveau 2	Niveau 1		Niveau 1
[*tableau*]	{ *index* }	➧	*z*
' *nom* '	{ *index* }	➧	*z*
{ *liste* }	*n*	➧	*objet*
' *nom* '	*n*	➧	*objet*

GET est un mécanisme général de rappel d'un élément d'un tableau ou d'une liste. Nous décrirons le rappel de l'élément d'une liste. Le rappel de l'élément d'un tableau figure à la section « ARRAY ».

GET prend deux arguments dans la pile. Le niveau 1 doit contenir un nombre réel spécifiant le numéro de l'élément à rappeler. Le niveau 2 peut contenir soit une liste soit un nom :

- Si le niveau 2 contient une liste, GET renvoie le *n*ième élément de cette liste dans la pile.
- Si le niveau 2 contient un nom, GET renvoie dans la pile le *n*ième élément de la liste contenue dans la variable.

L'opération inverse de GET est PUT.

PUTI *Placer élément et incrémenter index* Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 2	Niveau 1
⟦ *tableau*$_1$ ⟧	⟨ *index*$_1$ ⟩	*x*	➧	⟦ *tableau*$_2$ ⟧	⟨ *index*$_2$ ⟩
' *nom* '	⟨ *index*$_1$ ⟩	*x*	➧	' *nom* '	⟨ *index*$_2$ ⟩
⟦ *tableau*$_1$ C ⟧	⟨ *index*$_1$ ⟩	*z*	➧	⟦ *tableau*$_2$ C ⟧	⟨ *index*$_2$ ⟩
' *nom* '	⟨ *index*$_1$ ⟩	*z*	➧	' *nom* '	⟨ *index*$_2$ ⟩
⟨ *liste*$_1$ ⟩	n_1	*objet*	➧	⟨ *liste*$_2$ ⟩	n_2
' *nom* '	n_1	*objet*	➧	' *nom* '	n_2

PUTI est un mécanisme général de stockage d'éléments dans un tableau ou d'objets dans une liste. Nous décrirons ici le stockage dans une liste. Le stockage dans un tableau figure à la section « ARRAY ».

...LIST

PUTI stocke un objet dans une liste de la même manière que PUT, mais renvoie en outre dans la pile la liste (ou un nom de variable) et le numéro de l'élément incrémenté d'une unité. La pile est alors prête à recevoir un nouvel objet, puis à une nouvelle exécution de PUTI pour stocker cet objet en position suivante dans la liste. Lorsque le numéro de l'élément est égal au nombre d'élément de la liste, PUTI renvoie le numéro d'élément 1, recommençant au début de la liste.

PUTI prend trois arguments dans la pile. Le niveau 1 doit contenir l'objet que vous voulez stocker dans la liste. Le niveau 2 contient un nombre réel spécifiant le numéro de l'élément à remplacer. Le niveau 3 peut contenir soit une liste soit un nom :

- Si le niveau 3 contient une liste, PUTI renvoie cette liste au niveau 2 après en avoir remplacé le *n*ième élément par l'objet pris dans le niveau 1. Le numéro de l'élément suivant ($n_1 + 1$, ou 1) est renvoyé au niveau 1.
- Si le niveau 3 contient un nom, PUTI stocke l'objet en tant que *n*ième élément de la liste contenue dans la variable. Cette dernière ne peut pas être une variable locale. Le nom est renvoyé au niveau 2, et le numéro de l'élément suivant ($n_1 + 1$, ou 1) est renvoyé au niveau 1.

L'opération inverse de PUTI est GETI.

GETI ***Appeler élément et incrémenter index*** **Commande**

Niveau 2	Niveau 1		Niveau 3	Niveau 2	Niveau 1
⊏ *tableau* ⊐	{ $index_1$ }	➧	⊏ *tableau* ⊐	{ $index_2$ }	*z*
' *nom* '	{ $index_1$ }	➧	' *nom* '	{ $index_2$ }	*z*
{ *liste* }	n_1	➧	{ *liste* }	n_2	*objet*
' *nom* '	n_1	➧	' *nom* '	n_2	*objet*

GETI est un mécanisme général de rappel d'un élément d'un tableau ou d'une liste. Le rappel d'un élément d'un tableau est décrit à la section « ARRAY ».

GETI rappelle un élément d'une liste de la même manière que GET, mais laisse en outre la liste dans la pile et incrémente d'une unité le numéro de l'élément, afin de faciliter les rappels successifs à partir de la même liste. Lorsque le numéro de l'élément est égal au nombre d'éléments de la liste, GETI renvoie 1 comme numéro d'élément, recommençant ainsi au début de la liste.

GETI prend deux arguments dans la pile. Le niveau 1 doit contenir un nombre réel, définissant le numéro de l'élément à rappeler. Le niveau peut contenir soit une liste soit un nom :

- Si le niveau 2 contient une liste, GETI renvoie au niveau 1 le *n*ième élément de cette liste. GETI renvoie également la liste au niveau 3, et le numéro de l'élément suivant ($n_1 + 1$, ou 1) au niveau 2.
- Si le niveau 2 contient un nom, GETI renvoie au niveau 1 le *n*ième élément de la liste contenue dans la variable. GETI renvoie aussi le nom au niveau 3 et le numéro de l'élément suivant ($n_1 + 1$ ou 1) au niveau 2.

L'opération inverse de GETI est PUTI.

SUB SIZE

SUB *Prendre un sous-ensemble* Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
"*chaîne*$_1$"	n_1	n_2	➧	"*chaîne*$_2$"
⟨ *liste*$_1$ ⟩	n_1	n_2	➧	⟨ *liste*$_2$ ⟩

SUB (SUBSET) prend une liste et deux nombres entiers n_1 et n_2 dans la pile, et y renvoie une nouvelle liste contenant les objets qui était les éléments n_1 à n_2 de la liste initiale. Si $n_2 < n_1$, SUB renvoie une liste vide.

SUB fonctionne d'une manière analogue pour les chaînes de caractères. Consultez « STRING ».

SIZE *Taille* Commande

Niveau 1		Niveau 1
"*chaîne*"	➧	*n*
⟨ *liste* ⟩	➧	*n*
⟦ *tableau* ⟧	➧	⟨ *liste* ⟩
' *symbole* '	➧	*n*

SIZE renvoie un objet représentant la taille, ou les dimensions, de l'argument. Ce dernier peut être une liste, un tableau, un objet algébrique ou une chaîne de caractères. Pour une liste, SIZE donne un nombre *n* qui est le nombre d'éléments de la liste.

Consultez « ALGEBRA », « ARRAY » et « STRING » respectivement pour les objets algébriques, les tableaux et les chaînes.

LOGS

LOG	ALOG	LN	EXP	LNP1	EXPM
SINH	ASINH	COSH	ACOSH	TANH	ATANH

Le menu LOGS contient les fonctions exponentielles, logarithmiques et hyperboliques. Toutes ces fonctions admettent des arguments réels et algébriques ; toutes, sauf LNP1 et EXPM admettent des arguments complexes.

LOG ALOG LN EXP LNP1 EXPM

LOG — *Logarithme en base 10* — Fonct. analyt.

Niveau 1		Niveau 1
z	➡	log z
'symbole'	➡	'LOG(symbole)'

LOG donne le logarithme en base 10 de son argument.

Si l'argument est 0 ou (0, 0), vous obtenez l'exception `Infinite Result` (résultat infini).

ALOG — *Antilogarithme en base 10* — Fonct. analyt.

Niveau 1		Niveau 1
z	➡	10^z
'symbole'	➡	'ALOG(symbole)'

ALOG donne l'antilogarithme en base 10 de son argument — c'est-à-dire 10 élevé à la puissance donnée par l'argument.

Pour les arguments complexes :

$$\text{alog}\,(x, y) = \exp cx \cos cy + i \exp cx \sin cy,$$

où $c = \ln 10$ (le calcul est effectué en mode RADIANS).

LN — *Logarithme népérien* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	ln z
'symbole'	➧	'LN(symbole)'

LN donne le logarithme népérien (en base e) de son argument.

Si l'argument est 0 ou (0, 0), vous obtenez l'exception `Infinite Result` (résultat infini).

EXP — *Exponentielle* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	exp z
'symbole'	➧	'EXP(symbole)'

EXP donne l'exponentielle, ou l'antilogarithme népérien (en base e), de son argument — c'est-à-dire e élevé à la puissance donnée par l'argument. EXP donne un résultat plus précis que `e^`, car elle utilise un algorithme spécial pour calculer l'exponentielle.

Pour des arguments complexes :

$$\exp\,(x, y) = \exp x \cos y + i \exp x \sin y$$

(le calcul est effectué en mode RADIANS).

LNP1 — *Log népérien de 1+x* — Fonct. analyt.

Niveau 1		Niveau 1
x	➡	ln (1+*x*)
'*symbole*'	➡	'LNP1(*symbole*)'

LNP1 donne $\ln(1 + x)$, où x est l'argument dont la valeur est un nombre réel. LNP1 sert essentiellement à déterminer le logarithme népérien de nombres proches de 1. LNP1 fournit un résultat plus précis que LN pour $\ln(1 + x)$ avec x proche de zéro.

Les arguments inférieurs à 1 causent une erreur `Undefined Result` (résultat indéterminé).

EXPM — *Exponentielle moins 1* — Fonct. analyt.

Niveau 1		Niveau 1
x	➡	exp (*x*)−1
'*symbole*'	➡	'EXPM(*symbole*)'

EXPM donne $e^x - 1$, où x est l'argument dont la valeur est un nombre réel. EXPM sert essentiellement à déterminer l'exponentielle des nombres proches de 0. EXPM fournit un résultat plus précis que EXP pour $e^x - 1$, lorsque x est proche de 0.

...LOGS

SINH ASINH COSH ACOSH TANH ATANH

Ce sont les fonctions hyperboliques et leurs inverses.

SINH *Sinus hyperbolique* Fonct. analyt.

Niveau 1		Niveau 1
z	➧	sinh *z*
'*symbole*'	➧	'SINH(*symbole*)'

SINH donne le sinus hyperbolique de son argument.

ASINH *Sinus hyperbolique inverse* Fonct. analyt.

Niveau 1		Niveau 1
z	➧	asinh *z*
'*symbole*'	➧	'ASINH(*symbole*)'

ASINH donne le sinus hyperbolique inverse de son argument. Avec des arguments réels $|x| > 1$, ASINH donne le résultat complexe correspondant à l'argument $(x, 0)$.

COSH — *Cosinus hyperbolique* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	cosh z
'*symbole*'	➧	'COSH(*symbole*)'

COSH donne le cosinus hyperbolique de son argument.

ACOSH — *Cosinus hyperbolique inverse* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	acosh z
'*symbole*'	➧	'ACOSH(*symbole*)'

ACOSH donne le cosinus hyperbolique inverse de son argument. Avec des arguments réels $|x| < 1$, ACOSH donne le résultat complexe correspondant à l'argument $(x, 0)$.

TANH — *Tangente hyperbolique* — Fonct. analyt.

Niveau 1		Niveau 1
z	➧	tanh z
'*symbole*'	➧	'TANH(*symbole*)'

TANH donne la tangente hyperbolique de son argument.

ATANH *Tangente hyperbolique inverse* Fonct. analyt.

Niveau 1		Niveau 1
z	➧	atanh *z*
'*symbole*'	➧	'ATANH(*symbole*)'

ATANH donne la tangente hyperbolique inverse de son argument. Pour des arguments réels $|x| > 1$, ATANH donne le résultat complexe correspondant à l'argument $(x, 0)$.

Pour un argument réel $x = \pm 1$, vous obtenez une exception `Infinite Result` (résultat infini). Si l'indicateur binaire 59 est désarmé, le signe du résultat (MAXR) est celui de l'argument.

MODE

STD	**FIX**	**SCI**	**ENG**	**DEG**	**RAD**
+CMD	**−CMD**	**+LAST**	**−LAST**	**+UND**	**−UND**
+ML	**−ML**	**RDX.**	**RDX,**	**PRMD**	

Le menu MODE contient les touches de menu qui commandent divers *modes* de fonctionnement du calculateur : mode d'affichage des nombres, mode angulaire, modes de récupération des données après erreur, mode de représentation du séparateur décimal, et mode d'affichage multi-lignes.

Les libellés de ce menu servent également de témoins pour vous indiquer quels sont les modes en cours. Pour chaque mode, il existe deux ou plusieurs touches pour valider l'option de fonctionnement choisie ; mais un seul libellé est affiché en vidéo normale (caractères noirs sur fond clair) ; il indique l'option en cours pour ce mode. Par exemple, vous choisissez le mode angulaire à l'aide des commandes DEG et RAD. Lorsque le mode angulaire est *degrés*, le libellé `⌈ DEG ⌋` est en vidéo normale et le libellé `RAD` en vidéo inverse (caractères blancs sur fond noir). Si vous appuyez sur `RAD`, ce libellé passe en vidéo normale (`⌈ RAD ⌋`) et le libellé `DEG` en vidéo inverse, pour indiquer que le mode angulaire est maintenant *radians*.

En mode exécution, toutes les commandes du menu MODE sauf FIX, SCI, et ENG (qui nécessitent des arguments) sont exécutées sans que vous ayez besoin d'utiliser ENTER, ce qui a l'avantage de laisser la ligne de commande inchangée.

STD FIX SCI ENG DEG RAD

Ces fonctions choisissent le mode d'affichage des nombres et le mode angulaire.

Les fonctions STD, FIX, SCI, et ENG commandent le format d'affichage des nombres en virgule flottante, tels qu'ils apparaissent dans les niveaux de la pile pour toutes sortes d'objets. Dans les objets algébriques, les nombres non-entiers en virgule flottante sont affichés dans le format en cours et les entiers sont toujours affichés en format STD (standard).

...MODE

Le mode d'affichage en cours est déterminé par les indicateurs binaires 49 et 50. L'exécution de l'une des fonctions commandant le format d'affichage modifie l'état de ces indicateurs binaires ; inversement, armer et désarmer ces indicateurs affecte le mode d'affichage. La correspondance s'établit comme suit :

Mode	Indic. binaire 49	Indic. binaire 50
Standard (STD)	0	0
Fixe (FIX)	1	0
Scientifique (SCI)	0	1
Ingénieur (ENG)	1	1

Les indicateurs binaires 53 à 56 déterminent le nombre de décimales, de 0 à 11. L'indicateur binaire 56 joue le rôle de bit de poids le plus fort.

STD ***Format standard*** **Commande**

STD (STANDARD) choisit le *format standard* comme mode d'affichage numérique. Le format standard (norme ANSI BASIC minimal, X3J2) donne les résultats suivants lors de l'affichage ou de l'impression d'un nombre :

- Les nombres pouvant être représentés exactement comme des entiers avec 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 qui ne sont pas des entiers, sont affichés avec un séparateur décimal, mais sans exposant. Les zéros de tête à gauche du séparateur décimal et les zéros de queue dans la partie fractionnaire sont omis.

- Tous les autres nombres sont affichés dans le format suivant :

  (signe) mantisse E (signe) exposant

  où la valeur de la mantisse est située dans l'intervalle $1 \leqslant x < 10$, et où l'exposant est un nombre de un à trois chiffres. Les zéros de queue dans la mantisse et les zéros de tête dans l'exposant sont omis.

Le tableau suivant donne des exemples de nombres affichés en format standard :

Nombre	Affiché :	Représentable avec 12 chiffres ?
10^{11}	100000000000	Oui (entier)
10^{12}	1,E12	Non
10^{-12}	,000000000001	Oui
$1{,}2 \times 10^{-11}$	,000000000012	Oui
$1{,}23 \times 10^{-11}$	1,23E-11	Non
12,345	12,345	Oui

FIX — *Format fixe* — Commande

Niveau 1	
n ➧	

FIX choisit le *format fixe* comme mode d'affichage numérique et utilise un nombre réel comme argument pour définir le nombre de décimales à afficher (entre 0 et 11). C'est la valeur arrondie de l'argument qui est utilisée. Si cette valeur est suppérieure à 11, c'est 11 qui est utilisé ; si elle est inférieure à 0, c'est 0 qui est utilisé.

...MODE

En format fixe, les nombres affichés ou imprimés apparaissent sous la forme

(signe) mantisse

La mantisse apparaît arrondie à *n* décimales à droite de la virgule, *n* étant le nombre de chiffres spécifié. Même lorsque le format fixe est validé, le HP-28C/S affiche automatiquement une valeur en format scientifique dans l'un des deux cas ci-dessous :

- Si le nombre de chiffres à afficher dépasse 12.
- Si une valeur non nulle arrondie à *n* décimales est affichée comme un zéro en format fixe.

SCI — *Format scientifique* — Commande

Niveau 1	
n ➧	

SCI (SCIENTIFIC) choisit le *format scientifique* comme mode d'affichage numérique et utilise un nombre réel comme argument pour définir le nombre de chiffres significatifs à afficher (entre 0 et 11). C'est la valeur arrondie de l'argument qui est utilisée. Si elle est supérieure à 11, c'est 11 qui est utilisé ; si elle est inférieure à 0, c'est 0 qui est utilisé.

En format scientifique, les nombres sont affichés ou imprimés en notation scientifique avec $n + 1$ chiffres significatifs, n étant le nombre de chiffres choisi (l'argument de SCI). Une valeur apparaît donc sous la forme

```
(signe) mantisse E (signe) exposant
```

où $1 \leqslant mantisse < 10$.

ENG — *Format ingénieur* — Commande

Niveau 1	
n	↵

ENG (ENGINEERING) choisit le *format ingénieur* comme mode d'affichage numérique et utilise un nombre réel comme argument pour définir le nombre de chiffres significatifs à afficher (entre 0 et 11). C'est la valeur arrondie de l'argument qui est utilisée. Si cette valeur est supérieure à 11, c'est 11 qui est utilisé ; si elle est inférieure à 0, c'est 0 qui est utilisé.

En format ingénieur, un nombre affiché ou imprimé apparaît sous la forme

```
(signe) mantisse E (signe) exposant
```

où $1 \leqslant$ *mantisse* < 1000, et où l'exposant est un multiple de 3. Le nombre de chiffres significatifs affichés est égal à la valeur de l'argument plus un. Si une valeur affichée a un exposant de -499, elle est affichée en format scientifique.

DEG — *Degrés* — Commande

↵

DEG (DEGREES) choisit les degrés comme mode angulaire. En mode DEGRES :

Lorsque les arguments sont des nombres réels. Les fonctions prenant comme arguments des angles exprimés en tant que nombres réels interprètent ces angles comme des degrés (les arguments complexes pour SIN, COS et TAN sont toujours supposés être en radians).

...MODE

Lorsque les résultats sont des nombres réels. Les fonctions donnant comme résultats des angles exprimés sous la forme de nombres réels donnent ces angles exprimés en degrés : ASIN, ACOS, ATAN, ARG, et R→P. (Les résultats complexes donnés par ASIN ou ACOS pour des arguments en dehors du domaine $x \leqslant 1$ sont toujours exprimés en radians.)

L'exécution de DEG fait disparaître le témoin (2π) et désarme l'indicateur binaire d'utilisateur 60.

RAD ***Radians*** **Commande**

RAD (RADIANS) choisit les radians comme mode angulaire. En mode RADIANS :

Lorsque les arguments sont des nombres réels. Les fonctions prenant comme arguments des angles exprimés en tant que nombres réels interprètent ces angles comme des radians (les arguments complexes pour SIN, COS et TAN sont toujours supposés être en radians).

Lorsque les résultats sont des nombres réels. Les fonctions donnant comme résultats des angles exprimés sous la forme de nombres réels donnent ces angles exprimés en radians : ASIN, ACOS, ATAN, ARG, et R→P. (Les résultats complexes donnés par ASIN ou ACOS pour des arguments en dehors du domaine $x \leqslant 1$ sont toujours exprimés en radians.)

L'exécution de RAD fait apparaître le témoin (2π) et arme l'indicateur binaire d'utilisateur 60.

+CMD −CMD +LAST −LAST +UND −UND

Ces six touches de menu commandent les trois mécanismes de récupération de données après erreur : COMMAND (récupération de la ligne de commande précédente), LAST (récupération des arguments précédents), et UNDO (récupération de la totalité de la pile opérationnelle précédente). Dans chaque libellé, « + » signifie « valider la fonction » et « − » signifie « invalider la fonction ».

Aucune de ces opérations ne peut être incluse dans un programme ; les touches de menu sont toujours des touches à exécution immédiate. Vous pouvez toutefois valider ou invalider LAST dans un programme en armant ou désarmant l'indicateur binaire 31.

Validation/invalidation de COMMAND

Opération	Description
+CMD	Valide COMMAND. Vous pouvez maintenant rappeler les lignes de commande précédentes en appuyant sur ■ COMMAND.
-CMD	Invalide COMMAND et récupère la mémoire dans laquelle était sauvegardées les lignes de commandes précédentes. En appuyant sur ■ COMMAND vous faitez maintenant apparaître le message d'erreur `COMMAND Stack Disabled` (la pile mémoire de COMMAND est invalidée).

Validation/invalidation de LAST

Opération	Description
+LAST	Valide LAST. Dans le cas des commandes nécessitant un à trois arguments, les arguments sont désormais sauvegardés et vous pouvez les rappeler à l'aide de LAST.
-LAST	Invalide LAST et récupère la mémoire qui servait à sauvegarder les arguments. En exécutant LAST, vous faites apparaître le message d'erreur `LAST Disabled` (LAST est invalidée).

...MODE

Validation/invalidation de UNDO

Opération	Description
+UND	Valide UNDO. Le contenu de la pile est maintenant sauvegardé à chaque exécution de ENTER, et une pression sur ■ UNDO remplace le contenu de la pile en cours par le contenu précédent.
-UND	Invalide UNDO et récupère la mémoire qui servait à sauvegarder le contenu de la pile. Une pression sur ■ UNDO fait apparaître le message d'erreur UNDO Disabled (UNDO invalidée).
	Les effets de +UND et de -UND sont « locaux », c'est-à-dire limités au programme qui est suspendu. Si un programme est suspendu, l'exécution de +UND et de -UND modifie les caractéristiques de sauvegarde de la pile uniquement pendant que ce programme est suspendu.

+ML −ML RDX. RDX, PRMD

Validation/invalidation du mode multi-lignes (ML)

Les objets contenus dans la pile sont affichés dans l'un des deux formats généraux d'affichage suivants : multi-lignes, ou compact. Les objets qui ne se trouvent pas au niveau 1 de la pile sont *toujours* affichés en format compact, c'est-à-dire qu'une seule ligne est utilisée pour afficher l'objet (si l'objet ne peut pas être complètement affiché sur une ligne, l'ellipse « ... » remplace le caractère le plus à droite, indiquant ainsi qu'il existe d'autres caractères vers la droite).

...MODE

Les touches de menu +ML et -ML vous permettent d'afficher les objets contenus dans le niveau 1 en format compact (-ML), ou en format multi-lignes (+ML). En format multi-lignes :

- Chaque ligne d'une matrice est affichée sur une ligne distincte, avec autant d'éléments affichés sur chaque ligne qu'il est possible compte tenu du mode d'affichage numérique choisi.
- Pour les procédures et les listes, de nouvelles lignes sont insérées dans le texte affiché de manière que les procédures et listes soient visibles dans leur intégralité. Les procédures, listes et matrices contenues dans leurs définition sont également affichées intégralement, sur plusieurs lignes si nécessaire.
- Les nombres, nombres complexes, vecteurs, noms et chaînes ne sont pas répartis sur les lignes d'affichage, et il peuvent donc se trouver tronqués sur l'affichage. Comme en format compact, les caractères « cachés » sont indiqués par l'ellipse « ... », à l'emplacement du caractère le plus à droite.

Si l'affichage de l'intégralité de l'objet nécessite plus de lignes d'affichage qu'il n'y en a de disponibles (quatre avec le menu du curseur, moins une s'il y a un autre menu, moins une ou plusieurs si une ligne de commande est active), vous pouvez voir les autres lignes en appuyant sur ■ VIEW↑ ou sur ■ VIEW↓ .

Le choix du format multi-lignes ou compact est lui aussi représenté par l'indicateur binaire 45. Lorsque cet indicateur est armé, le format est multi-lignes.

Opération	Description
+ML	Choisit le format multi-lignes pour les objets du niveau 1, et arme l'indicateur binaire 45. En format multi-lignes, une ou plusieurs lignes de l'affichage sont utilisées pour montrer l'objet se trouvant au niveau 1 de la pile. Les objets du niveau 1 imprimés sur l'imprimante HP-82240A en mode TRACE (impression automatique) seront également imprimés en format multi-lignes.
-ML	Choisit le format compact pour les objets du niveau 1, et désarme l'indicateur binaire 45. Les objets du niveau 1 imprimés sur l'imprimante HP-82240A en mode TRACE (impression automatique) sont également imprimés en format compact.

...MODE

Choix de la virgule ou du point comme séparateur décimal

RDX. et RDX, vous permettent de choisir le caractère qui servira à représenter le *séparateur décimal* (RADIX, abrégé en RDX), c'est-à-dire le symbole qui séparera la partie entière de la partie fractionnaire d'un nombre en virgule flottante. Vous pouvez choisir soit le point « . » (format américain), soit la virgule « , » (format européen) ; le caractère que vous ne choisirez pas comme séparateur décimal servira de séparateur entre les objets lorsque vous les saisirez en ligne de commande (sauf dans les objets algébriques, où il servira à séparer les arguments des fonctions à arguments multiples). Vous pouvez aussi utiliser l'espace (la touche [SPACE]) pour séparer les objets lors de leur saisie. Si vous choisissez le format européen (virgule comme séparateur décimal), vos nombres complexes seront affichés sous la forme $(x.y)$ et non pas (x,y).

Opération	Description
RDX.	Choisit le point comme séparateur décimal, et désarme l'indicateur binaire d'utilisateur 48. La virgule n'est alors pas le séparateur décimal et peut être utilisée comme séparateur d'objets dans la ligne de commande, en remplacement de l'espace (sauf dans les objets algébriques).
RDX,	Choisit la virgule comme séparateur décimal et arme l'indicateur binaire d'utilisateur 48. Le point n'est alors pas le séparateur décimal et peut être utilisé comme séparateur d'objets dans la ligne de commande, en remplacement de l'espace (sauf dans les objets algébriques).

PRMD ***Imprimer les modes en cours*** **Commande**

▶

PRMD (PRINT MODES) affiche et imprime une liste des modes en cours d'utilisation sur votre HP-28C/S. Cette liste indique l'état du mode d'affichage numérique, du mode multi-lignes, du mode angulaire, la base choisie pour les entiers binaires, le séparateur décimal choisi, et si les fonctions UNDO, COMMAND, et LAST sont validées ou invalidées. Une liste type ressemble à ceci :

PLOT

STEQ	**RCEQ**	**PMIN**	**PMAX**	**INDEP**	**DRAW**
PPAR	**RES**	**AXES**	**CENTR**	**∗W**	**∗H**
STOΣ	**RCLΣ**	**COLΣ**	**SCLΣ**	**DRWΣ**	
CLLCD	**DISP**	**PIXEL**	**DRAX**	**CLMF**	**PRLCD**

Les commandes du menu PLOT vous donnent la possibilité de créer des affichages spéciaux qui remplacent les affichages normaux de la pile et du menu. Vous pouvez représenter des fonctions mathématiques, créer des nuages de points à partir de données statistiques, afficher des données pendant l'exécution d'un programme et numériser des données à partir de tracés.

L'affichage

L'affichage à cristaux liquides (LCD en anglais) du HP-28C/S est constitué de 32 lignes de 137 *pixels* (ou points), organisées en quatre lignes de 23 espaces-caractères. Un espace-caractère a une largeur de six pixels sur huit, à l'exception de l'espace-caractère placé à l'extrême droite de chaque ligne, large de 5 pixels. Normalement les caractères affichés sont larges de 5 pixels, ce qui laisse une colonne de pixels vide entre les caractères.

L'affichage par défaut montre les libellés de menu, la ligne de commande et la pile opérationnelle. Vous pouvez momentanément modifier l'affichage par défaut comme suit :

- **Effacer l'affichage.** La commande CLLCD (CLEAR LCD) efface entièrement l'affichage (sauf les témoins).
- **Afficher des messages.** En utilisant DISP, vous pouvez afficher des objets dans l'une ou plusieurs des quatre lignes d'affichage.
- **Afficher des données sous forme graphique.** Les commandes PIXEL, DRAW et DRWΣ permettent de tracer respectivement des observations individuelles, des fonctions mathématiques et des points issus de la matrice statistique. DRAX vous permet de tracer des axes (DRAW AXES).

Quand vous exécutez l'une de ces commandes, un indicateur binaire système est automatiquement armé, évitant l'affichage normal de la pile et du menu, qui effacerait votre affichage spécial. L'indicateur est désarmé et l'affichage normal est rétabli lorsque vous frappez n'importe quelle touche après la fin de l'exécution de toutes les procédures en cours, c'est-à-dire, lorsque le témoin ((•)) disparaît. Vous pouvez utiliser la commande CLMF (CLEAR MESSAGE FLAG) dans un programme pour effacer l'indicateur binaire système.

Certaines touches produisent également des affichages spéciaux et redéfinissent temporairement le clavier :

- ■ CATALOG et ■ UNITS créent des affichages de catalogue et fournissent des touches de menu et des touches alphabétiques permettant de contrôler les catalogues.
  ■ CATALOG et ■ UNITS sont respectivement décrites sous « CATALOG : le catalogue des commandes » et « Conversion d'unités ».
- DRAW et DRWΣ exécutent respectivement DRAW et DRWΣ, pour produire des graphiques de fonctions mathématiques et des nuages de points à partir de données statistiques. De plus, si vous exécutez DRAW ou DRWE en appuyant sur leurs touches de menu pendant l'affichage des tracés, vous pouvez vous servir des touches de menu pour déplacer le curseur sur l'écran ou pour envoyer des coordonnées numérisées dans la pile.

Positions sur l'affichage

Pour permettre l'affichage de chaînes de caractères, l'écran est traité comme une succession de quatre lignes de caractères, numérotées de 1 à 4 à partir du haut. Chaque ligne doit être affichée comme une entité séparée ; il n'est pas possible d'afficher des caractères isolément, à des positions abitraires. DISP (DISPLAY) affiche un objet, qui est représenté comme une chaîne de caractères équivalant à l'affichage multi-lignes de l'objet dans la pile.

...PLOT

Pour les affichages graphiques de données, l'écran est traité comme une grille de 137 x 32 points, ou *pixels*. Chaque pixel est défini par ses *coordonnées*, un nombre complexe représentant un couple ordonné de coordonnées (x, y), dans lequel x est la coordonnée horizontale et y la coordonnée verticale (nous utiliserons les lettres x et y pour représenter les directions horizontale et verticale dans nos explications, mais tous les noms de variables sont admis pour les tracés sur le HP-28C/S).

Le positionnement des coordonnées sur l'affichage est déterminé par les coordonnées des points de cadrage P_{max} et P_{min}, définis à l'aide des commandes PMAX et PMIN. P_{max} est le pixel constituant l'angle supérieur droit de l'affichage graphique ; ses coordonnées sont (x_{max}, y_{max}). Quant à P_{min} (x_{min}, y_{min}), c'est le pixel constituant l'angle inférieur gauche. Les coordonnées par défaut sont $P_{max} = (6{,}8\ ,\ 1{,}6)$ et $P_{min} = (-6{,}8\ ,\ -1{,}5)$. En outre, les coordonnées du centre d'un pixel donné sont définies comme suit :

$$x = n_x\, w_x + x_{min}$$

$$y = n_y\, w_y + y_{min}$$

où n_x est le numéro horizontal du pixel et n_y son numéro vertical (pour P_{min}, $n_x = 0$ et $n_y = 0$; pour P_{max}, $n_x = 136$ et $n_y = 31$). w_x et w_y sont les largeurs horizontale et verticale des pixels :

$$w_x = (x_{max} - x_{min})/136.$$

$$w_y = (y_{max} - y_{min})/31.$$

Le pixel ayant $n_x = 68$ et $n_y = 15$ est défini comme le *pixel central*. Lorsque P_{max} et P_{min} ont leurs valeurs par défaut, les coordonnées du pixel central sont $(0, 0)$.

Tracés de fonctions mathématiques

Un *tracé de fonction* mathématique est un tracé des valeurs d'une procédure stockée dans la variable EQ (la même que celle utilisée par SOLVR), comme fonction d'une *variable indépendante* donnée. La procédure est évaluée complètement pour chacune des valeurs $137/r$ de la variable indépendante, de x_{min} à x_{max}, où r est la *résolution* du tracé. Un pixel (point) est ajouté au graphique pour chaque couple de coordonnées (*valeur de la variable indépendante, valeur de la procédure)*, tant que la valeur de la procédure est comprise dans l'intervalle de traçage défini par y_{min} et y_{max}. Le tracé inclut également des axes, gradués tous les 10 pixels.

Le tacé lui-même est généré par la commande DRAW. Si vous exécutez directement cette commande en appuyant sur la touche de menu DRAW, vous pouvez utiliser les touches du curseur pour numériser des données du tracé.

Un tracé de fonction produira une ou deux courbes, selon la définition de la procédure EQ :

- Si EQ contient une expression algébrique sans signe égale, DRAW tracera une seule courbe correspondant à la valeur de chaque expression pour chaque valeur de la variable indépendante comprise dans l'intervalle de traçage.
- Si EQ contient une équation algébrique, DRAW tracera deux courbes, une pour chaque côté de l'équation. Notez que les intersections des deux courbes se font aux valeurs de la variable indépendante qui sont les racines de l'équation, et qui peuvent être calculées à l'aide de la fonction de résolution d'équations SOLVR.
- Si EQ contient un programme, il sera traité comme une expression algébrique et tracé comme une seule courbe. Ceci suppose que le programme obéit à la syntaxe d'une expression algébrique : il ne peut prendre aucun argument dans la pile et doit renvoyer un et un seul objet dans la pile.

...PLOT

La procédure générale pour obtenir un tracé de fonction est résumée ci-après. Pour les détails, faites référence aux descriptions de chaque commande.

1. Stockez dans EQ la procédure devant être tracée, en utilisant STEQ.
2. Choisissez la variable indépendante avec INDEP.
3. Choisissez les limites de traçage, en utilisant PMIN, PMAX, CENTR, ∗H et ∗W.
4. Définissez l'intersection des axes, avec AXES.
5. Choisissez la résolution du tracé avec RES.
6. Exécutez DRAW.

N'importe laquelle des étapes 1 à 5 peut être omise, auquel cas les valeurs utilisées sont celles en cours.

Nuages de points statistiques

Un *nuage de points* statistiques est constitué de points individuels tirés d'un tableau statistique stocké dans la variable ΣDAT. Vous pouvez définir n'importe quelle colonne de valeurs du tableau comme correspondant à la coordonnée horizontale, et n'importe quelle autre colonne pour la coordonnée verticale. Un point est alors tracé pour chaque observation de la matrice.

La procédure générale pour obtenir un nuage de points est résumée ci-après. Pour les détails, reportez-vous aux descriptions de chaque commande.

1. Stockez dans ΣDAT les données statistiques devant être tracées, en utilisant STOΣ.
2. Choisissez la colonne des coordonnées horizontales et celle des coordonnées verticales avec COLΣ.
3. Choisissez les limites de traçage, en utilisant SCLΣ pour la mise à l'échelle automatique, ou PMIN, PMAX, CENTR, ∗H et ∗W.

4. Définissez l'intersection des axes, avec AXES.
5. Exécutez DRWΣ.

N'importe laquelle des étapes 1 à 4 peut être omise, auquel cas les valeurs utilisées sont celles en cours.

Traçage interactif

Si vous exécutez DRAW ou DRWΣ en appuyant sur la touche de menu correspondante, le HP-28C/S passe en mode de traçage interactif, qui vous permet de numériser des données du tracé tout en le visualisant. Lorsque vous commencez un traçage interactif :

1. L'affichage est effacé.
2. DRAW ou DRWΣ est exécutée, pour produire le tracé approprié (si vous appuyez sur [ON] avant que le tracé ne soit terminé, le tracé de points s'arrête et le mode interactif commence).
3. Un curseur en forme de croix apparaît au milieu de l'affichage (si les axes se croisent au milieu de l'écran, le curseur ne sera visible que lorsque vous aurez commencé à le déplacer car il se confond avec l'intersection des axes).
4. Les touches de menu deviennent des touches de déplacement du curseur et de numérisation :
   - Les deux touches à l'extrême gauche du menu envoient dans la pile les coordonnées du curseur sans mettre fin à l'affichage du tracé. [INS] envoie les coordonnées sous forme d'un nombre complexe *(x, y)*. [DEL] envoie les coordonnées sous forme de deux nombres réels, *x* au niveau 2 et *y* au niveau 1.
   - Les quatre touches de menu placées à droite agissent comme des touches classiques de commande du curseur, déplaçant le curseur vers le haut, le bas, la droite ou la gauche, ou à l'extrême de ces directions si vous les faites précéder d'une pression sur ■.

Le mode de traçage interactif continue jusqu'à ce que vous appuyez sur la touche [ON].

Vous pouvez numériser autant de points que vous le désirez pendant que ce mode est actif, par une utilisation répétée de [INS], de [DEL] et des touches du curseur.

...PLOT

Paramètres de traçage

Les facteurs de mise à l'échelle, nécessaires pour convertir un couple de coordonnées en une position sur l'affichage et inversement, sont stockés sous la forme d'une liste d'objets dans la variable PPAR. Nous les appelons ici *paramètres de traçage* :

Paramètre	Description
P_{min}	Un nombre complexe représentant les coordonnées du pixel constituant l'angle inférieur gauche de l'affichage graphique. Défini par PMIN, CENTR, ∗H, ∗W et SCLΣ.
P_{max}	Un nombre complexe représentant les coordonnées du pixel constituant l'angle supérieur droit de l'affichage graphique. Défini par PMAX, CENTR, ∗H, ∗W et SCLΣ.
Variable indépendante	Le nom de variable correspondant à l'axe horizontal dans un tracé de fonction mathématique. Défini par INDEP.
Résolution	Un nombre réel entier positif représentant l'espacement entre les points d'un tracé de fonction. Défini par RES.
P_{axes}	Un nombre complexe représentant les coordonnées de l'intersection des axes de traçage. Défini par AXES.

STEQ RCEQ PMIN PMAX INDEP DRAW

Cet ensemble de commandes permet de choisir une procédure pour un tracé de fonction, de définir les paramètres primaires de traçage et de tracer la procédure.

STEQ — *Stocker équation* — Commande

Niveau 1	
objet ➡	

STEQ (STORE EQUATION) prend un objet dans la pile et le stocke dans la variable EQ (EQuation). Elle équivaut à `'EQ' STO`.

EQ est utilisée pour stocker une procédure (l'équation en cours), qui est utilisée comme argument implicite par la fonction de résolution d'équation SOLVR et par DRAW ; l'argument de STEQ doit donc normalement être une procédure.

RCEQ — *Rappeler équation* — Commande

	Niveau 1
➡ *objet*	

RCEQ (RECALL EQUATION) donne le contenu de la variable EQ. Elle équivaut à `'EQ' RCL`.

PMIN — *Limite inf. de traçage* — Commande

Niveau 1	
(*x*, *y*) ➡	

PMIN (PLOT MINIMA) définit les coordonnées du pixel constituant l'angle inférieur gauche de l'affichage graphique comme étant le point (*x*, *y*). Le nombre complexe (*x*, *y*) est stocké comme le premier élément de la liste contenue dans la variable PPAR.

...PLOT

PMAX — *Limite sup. de traçage* — Commande

Niveau 1	
(x,y) ➧	

PMAX (PLOT MAXIMA) définit les coordonnées du pixel constituant l'angle supérieur droit de l'affichage graphique comme étant le point (x, y). Le nombre complexe (x, y) est stocké comme deuxième élément de la liste contenue dans la variable PPAR.

INDEP — *Variable indépendante* — Commande

Niveau 1	
'nom' ➧	

INDEP prend un nom dans la pile et le stocke en tant que nom de variable indépendante ; il devient le troisième élément de la liste contenue dans la variable PPAR. Pour les exécutions suivantes de DRAW, ce nom sera utilisé comme variable indépendante correspondant à l'axe horizontal (abscisse) du tracé.

DRAW — *Tracer* — Commande

➧

DRAW produit des tracés de fonctions mathématiques sur l'affichage du HP-28C/S. Si vous exécutez DRAW en appuyant sur la touche de menu DRAW, le traçage est interactif, comme décrit sous « Traçage interactif » en page 203.

DRAW exécute automatiquement DRAX pour tracer les axes, puis trace une ou deux courbes représentant la (les) valeur(s) de l'équation en cours pour chacune des valeurs $137/r$ de la variable indépendante. L'équation en cours est la procédure stockée dans la variable EQ.

Si EQ contient une équation algébrique, les deux côtés de l'équation sont tracés séparément, produisant deux courbes. Si l'équation en cours est une expression algébrique ou un programme, une seule courbe est tracée.

La résolution r détermine le nombre de points tracés. $r = 1$ signifie qu'un point est tracé pour chaque colonne de pixels ; $r = 2$, pour une colonne de pixels sur deux, et ainsi de suite. r est défini par la commande RES. La valeur par défaut de r est 1 ; vous pouvez utiliser des valeurs plus grandes de r pour réduire le temps de traçage.

DRAW vérifie l'équation en cours pour voir si elle contient au moins une référence, directe ou indirecte, à la variable indépendante. Si la variable indépendante n'a jamais été choisie, c'est la première variable de l'équation en cours qui est utilisée, et qui est stockée dans PPAR. Si la variable indépendante n'est pas référencée du tout dans l'équation en cours, le message

nom_1 Not In Equation
Using nom_2

(nom_1 ne se trouve pas dans l'équation - nom_2 est utilisé) est affiché brièvement avant l'effacement de l'affichage et avant le début du traçage proprement dit. Ici nom_1 est la variable indépendante en cours, définie dans PPAR, et nom_2 est la première variable trouvée dans l'équation en cours. Si l'équation en cours ne contient pas de variable, la deuxième ligne du message est remplacée par Constant Equation (équation constante). Le nom de variable indépendante dans PPAR sera alors constant.

PPAR RES AXES CENTR *W *H

Ces commandes offrent d'autres manières de définir des paramètres de traçage.

...PLOT

PPAR — *Rappeler paramètres de traçage* — Opération

	Niveau 1
	➧ { *paramètres de traçage* }

L'opération PPAR (RECALL PLOT PARAMETERS) est une manière pratique d'examiner les paramètres de traçage en cours.

PPAR est une variable contenant une liste des paramètres de traçage, sous la forme

{ (x_{min}, y_{min}) (x_{max}, y_{max}) *var.-indép. résolution* (*axe des x, axe des y*) }

Une pression sur PPAR envoie la liste dans la pile. Le contenu de la liste est décrit sous « Paramètres de traçage » en page 204.

RES — *Résolution* — Commande

Niveau 1	
n →	

RES définit la *résolution* des tracés de fonctions mathématiques (DRAW) comme étant la valeur *n*. *n* est stocké en tant que quatrième élément de la liste stockée dans la variable PPAR. *n* détermine le nombre de points tracés : $n = 1$ signifie qu'un point est tracé pour chaque colonne de pixels d'affichage ; $n = 2$, pour une colonne de pixels sur deux, et ainsi de suite. La valeur par défaut de *n* est 1 ; vous pouvez désirer utiliser des valeurs plus grandes de *n* pour réduire le temps de traçage.

AXES — *Axes* — Commande

Niveau 1	
(x,y) ➧	

AXES définit les coordonnées de l'intersection des axes (tracés par DRAX, DRAW ou DRWΣ) comme étant le point (x, y). Le nombre complexe (x, y) est stocké en tant que cinquième et dernier élément de la liste contenue dans la variable PPAR. Les coordonnées par défaut sont (0, 0).

CENTR ***Centrer*** **Commande**

Niveau 1	
(x,y) ➡	

CENTR ajuste les paramètres de traçage de façon que le point représenté par l'argument (x, y) corresponde au pixel central ($n_x = 68$, $n_y = 15$) de l'affichage. La hauteur et la largeur du tracé restent inchangées. P_{max} et P_{min} sont remplacés par P_{max}' et P_{min}', avec :

$$x_{max}' = x + \tfrac{1}{2}(x_{max} - x_{min}),\ y_{max}' = y + 16/31\,(y_{max} - y_{min})$$

$$x_{min}' = x - \tfrac{1}{2}(x_{max} - x_{min}),\ y_{min}' = y - 15/31\,(y_{max} - y_{min})$$

∗W ***Multiplier la largeur*** **Commande**

Niveau 1	
x ➡	

∗W ajuste les paramètre de traçage de manière que x_{min} et x_{max} soient multipliés par le nombre x.

$$x_{min}' = x \times x_{min}$$

$$x_{max}' = x \times x_{max}$$

...PLOT

*H — *Multiplier la hauteur* — Commande

Niveau 1	
x ➡	

*H ajuste les paramètres de traçage de manière que y_{min} et y_{max} soient multipliés par le nombre x.

$$y_{min}' = x \times y_{min}$$

$$y_{max}' = x \times y_{max}$$

STOΣ RCLΣ COLΣ SCLΣ DRWΣ

Ce groupe de commandes vous permet de créer des nuages de points statistiques. Consultez « STAT » pour une description des possibilités statistiques générales offertes par le HP-28C/S.

STOΣ — *Stocker sigma* — Commande

Niveau 1	
⟦ *tableau R* ⟧ ➡	

STOΣ prend un tableau réel dans la pile et le stocke dans la variable ΣDAT. L'exécution de STOΣ équivaut à l'exécution de `'ΣDAT' STO`. Le tableau stocké devient la matrice statistique en cours.

RCLΣ — *Rappeler sigma* — Commande

	Niveau 1
➡	*objet*

RCLΣ (RECALL SIGMA) renvoie dans la pile le contenu de la variable ΣDAT. RCLΣ équivaut à `'ΣDAT' RCL`.

COLΣ — *Colonnes sigma* — Commande

Niveau 2	Niveau 1	
n_1	n_2	▪

COLΣ prend deux entiers réels, n_1 et n_2, et les stocke comme les deux premiers éléments de la liste contenue dans la ΣPAR. Ces nombres représentent des numéros de colonnes dans la matrice statistique en cours ΣDAT, et ils sont utilisés par des commandes statistiques qui fonctionnent avec des couples de colonnes. Consultez « STAT » pour plus de détails sur ΣPAR.

n_1 désigne la colonne correspondant à la variable indépendante pour LR, ou la coordonnée horizontale pour DRWΣ ou SCLΣ. n_2 désigne la variable dépendante ou la coordonnée verticale. Pour CORR et COV, l'ordre des deux numéros de colonne est sans importance.

Si une commande portant sur deux colonnes est exécutée alors que ΣPAR n'existe pas encore, cette dernière est automatiquement créée avec des valeurs par défaut $n_1 = 1$ et $n_2 = 2$.

SCLΣ — *Mettre Sigma à l'échelle* — Commande

▪

SCLΣ (SCALE SIGMA) met automatiquement à l'échelle les paramètres de traçage dans PPAR de manière que les nuages de points suivants remplissent exactement l'affichage. C'est-à-dire que les coordonnées horizontales de P_{max} et de P_{min} sont définies comme étant les coordonnées de valeur maximale et minimale, respectivement, dans la colonne de données indépendantes de la matrice statistique en cours. De même, les coordonnées verticales de P_{max} et de P_{min} sont définies à partir de la colonne de données dépendantes. Les numéros des colonnes de données indépendantes et dépendantes sont ceux stockés dans la variable ΣPAR.

...PLOT

DRWΣ ***Tracer sigma*** **Commande**

DRWΣ (DRAW SIGMA) exécute automatiquement DRAX afin de tracer les axes, puis crée un nuage de points statistiques à partir des points représentés par des couples de coordonnées pris dans les colonnes de données indépendantes et dépendantes de la matrice statistique en cours ΣDAT. Si vous exécutez DRWΣ en appuyant sur la touche de menu DRWΣ, il se produit un traçage interactif, comme décrit dans « Traçage interactif » en page 203.

Les colonnes de données indépendantes et dépendantes sont spécifiées dans la variable ΣPAR (respectivement 1 et 2, par défaut). DRWΣ trace un point pour chaque observation (couple de données) de la matrice statistique. Pour chaque point, la coordonnée horizontale est la valeur de la coordonnée dans la colonne de données indépendantes, et la coordonnée verticale est la valeur de la coordonnée dans la colonne de données dépendantes.

CLLCD DISP PIXEL DRAX CLMF PRLCD

Ces commandes vous permettent de créer des affichages spéciaux, et d'imprimer une image de l'affichage sur l'imprimante HP-82440A.

CLLCD ***Effacer affichage*** **Commande**

CLLCD (CLEAR LCD) efface l'affichage du HP-28C/S (sauf les témoins) et arme l'indicateur binaire système.

DISP — *Afficher* — Commande

Niveau 2	Niveau 1	
objet	*n*	➧

DISP (DISPLAY) affiche *objet* à la *n*ième ligne de l'affichage, *n* étant un entier réel. $n = 1$ choisit la ligne supérieure de l'affichage ; $n = 4$ la ligne inférieure. DISP arme l'indicateur binaire système afin de supprimer l'affichage normal de la pile.

Un objet est affiché par DISP sous la même forme que si cet objet se trouvait au niveau 1, en mode multi-lignes, sauf pour les chaînes, qui sont affichées sans leurs délimiteurs " pour faciliter l'affichage des messages. Si l'affichage de l'objet nécessite plus d'une ligne, l'affichage commence en ligne *n* et se poursuit vers le bas, soit jusqu'à la fin de l'objet, soit jusqu'en bas de l'affichage.

PIXEL — *Pixel* — Commande

Niveau 1	
(*x*,*y*)	➧

PIXEL allume le pixel correspondant aux coordonnées définies par le nombre complexe (x, y) et arme l'indicateur binaire système.

DRAX — *Tracer axes* — Commande

➧

...PLOT

DRAX (DRAW AXES) trace deux axes sur l'affichage, et arme l'indicateur binaire système. Les axes se coupent au point P_{axes}, défini dans la variable PPAR. Des graduations sont placées sur les axes tous les 10 pixels.

CLMF ***Désarmer indic. binaire système*** **Commande**

CLMF (CLEAR MESSAGE FLAG) désarme l'indicateur binaire système, qui est armé par CLLCD, DISP, PIXEL, DRAX, DRAW, et DRWΣ. Inclure CLMF dans un programme, après l'une de ces commandes, restaure l'affichage normal de la pile lorsque l'exécution du programme est terminée.

PRLCD ***Imprimer affichage*** **Commande**

PRLCD (PRINT LCD) permet d'imprimer des copies de tracés de fonctions mathématiques et de nuages de points statistiques. Comme PRLCD ne peut imprimer une copie que de l'affichage en cours, vous devez inclure PRLCD et DRAW (ou DRWΣ) dans la même ligne de commande. Par exemple :

CLLCD DRAW PRLCD ENTER

efface l'affichage, trace l'équation en cours, puis imprime une copie de l'affichage.

PR1	**PRST**	**PRVAR**	**PRLCD**	**TRACE**	**NORM**
PRSTC	**PRUSR**	**PRMD**	**CR**		

Le HP-28C/S transmet texte et graphiques à l'imprimante HP 82240A par rayons infra-rouges. La diode d'émission des infra-rouges se trouve sur le bord supérieur de la moitié droite du calculateur. Avant d'imprimer, vérifiez que l'imprimante peut recevoir le rayon infra-rouge émis par le HP-28C/S. Consultez le manuel de l'imprimante pour plus de renseignements sur l'utilisation de l'imprimante.

Vous pouvez utiliser les commandes d'impression pour imprimer des objets, des variables, des niveaux de la pile opérationnelle, des graphiques, etc. Vous pouvez aussi choisir le mode TRACE pour garder automatiquement une trace imprimée et continue de vos calculs.

Le témoin  apparaît chaque fois que le HP-28C/S transmet des données par sa diode infra-rouge. Le calculateur ne peut pas déterminer si l'impression a effectivement lieu, car la transmission ne s'effectue que dans un sens. Vérifiez que le mode TRACE n'est pas actif lorsqu'il n'y a pas d'imprimante ou lorsqu'elle n'est pas sous tension, car des émissions infra-rouges fréquentes ralentissent les opérations effectuées au clavier.

Formats d'impression

Les objets multi-lignes peuvent être imprimés en format compact ou multi-lignes. Le format d'impression compact est identique au format d'affichage compact. Le format d'impression multi-lignes est identique au format d'affichage multi-lignes, à cette exception près que les objets suivants sont intégralement imprimés :

- Les chaînes et noms de plus de 23 caractères se continuent sur la ligne d'impression suivante.

- Les parties réelle et imaginaire des nombres complexes sont imprimées sur des lignes séparées si elles ne tiennent pas sur la même ligne.
- Les tableaux sont imprimés avec un index avant chaque élément. Par exemple, l'index `1,1:` précède le premier élément.

En mode TRACE, le format de l'impression varie selon que le format d'affichage multi-lignes est activé ou désactivé (indicateur binaire 45 armé ou désarmé). La commande PRSTC (PRINT STACK COMPACT) imprime en format compact. Toutes les autres commandes d'impression impriment en format multi-lignes.

Pour imprimer plus vite

Lorsque l'imprimante est alimentée par des piles, sa vitesse diminue à mesure que les piles se déchargent. Le HP-28C/S régule normalement la transmission des données de manière à ne pas dépasser la vitesse de l'imprimante lorsque ses piles sont presque complètement déchargées.

Lorsque l'imprimante est alimentée par un adaptateur secteur, elle peut fonctionner plus vite. En armant l'indicateur binaire 52, vous pouvez améliorer la vitesse de transmission du calculateur de façon qu'elle corresponde à la vitesse accrue de l'imprimante. Si vous devez par la suite utiliser l'imprimante sur piles, vous devez désarmer l'indicateur binaire 52 afin de revenir à une transmission moins rapide des données.

N'armez pas l'indicateur binaire 52 lorsque l'imprimante est alimentée par piles. Bien qu'une imprimante munie de piles neuves puisse imprimer plus vite, elle finira par ralentir suffisamment pour perdre des données envoyées par le HP-28C/S. Cette perte de données perturbe la sortie imprimée et peut entraîner une modification de la configuration de l'imprimante.

Configuration de l'imprimante

Vous pouvez choisir divers modes d'impression en envoyant des *séquences d'échappement* à l'imprimante. Une séquence d'échappement se compose du caractère d'échappement (le caractère de valeur décimale 27) suivi d'un caractère supplémentaire. Lorsque l'imprimante reçoit une séquence d'échappement, elle passe au mode correspondant à la séquence. La séquence d'échappement elle-même n'est pas imprimée. L'imprimante HP 82240A reconnaît les séquences d'échappement suivantes :

Mode de l'imprimante	Séq. d'échappement
Impression de colonnes graphiques	27 001...166
Pas de soulignement*	27 250
Soulignement	27 251
Impression en largeur simple*	27 252
Impression en largeur double	27 253
Auto-test	27 254
Réinitialisation	27 255

* Mode par défaut.

Vous pouvez utiliser CHR (CHARACTER — voir « STRING ») et + (voir « Arithmétique ») pour créer des séquences d'échappement et PR1 pour les envoyer à l'imprimante. Par exemple, vous pouvez imprimer Souligner de la manière suivante :

```
27 CHR 251 CHR + "Sou" + 27 CHR + 250 CHR +
 "ligner" + PR1
```

...PRINT

PR1 PRST PRVAR PRLCD TRACE NORM

PR1 ***Imprimer niveau 1*** **Commande**

Niveau 1	Niveau 1
objet	*objet*

PR1 (PRINT 1) imprime le contenu du niveau 1 de la pile en mode d'impression multi-lignes. Tous les objets, sauf les chaînes, sont imprimés avec leur délimiteurs. Les chaînes sont imprimées sans leurs délimiteurs ("). Si le niveau est vide, le message `[Empty Stack]` (pile vide) est imprimé.

Impression d'une chaîne de texte

Vous pouvez imprimer une séquence quelconque de caractères en créant une chaîne contenant les caractères, en plaçant cet objet au niveau 1, et en exécutant PR1. L'imprimante imprime les caractères et laisse la tête d'impression à l'extrémité droite de la ligne d'impression. Lorsque l'impression reprend, elle commence à la ligne suivante.

Impression d'une chaîne graphique

Vous pouvez imprimer des graphiques en imprimant une chaîne commençant par le caractère d'échappement (caractère 27) et un caractère dont la valeur décimale n est comprise entre 1 et 166. Ensemble, ces deux caractères donnent à l'imprimante l'instruction d'interpréter les n caractères suivants ($n \leqslant 166$) comme des codes graphiques, chaque caractère définissant une colonne graphique. Consultez le manuel de l'imprimante pour plus de détails.

L'imprimante imprime les colonnes graphiques et laisse la tête d'impression à l'extrémité droite de la ligne d'impression. Lorsque l'impression reprend ensuite, elle recommence à la ligne suivante. Lorsque vous mettez l'imprimante sous tension, il vous faut imprimer du texte ou exécuter CR avant d'imprimer des graphiques.

Accumulation de données dans la mémoire tampon de l'imprimante

Vous pouvez imprimer n'importe quelle combinaison de texte, de graphiques, et d'objets sur une ligne en accumulant des données dans l'imprimante. Cette dernière stocke les données dans une partie de sa mémoire appelée *mémoire tampon* (ou plus simplement *tampon*).

Normalement, chaque commande d'impression met fin à la transmission des données par l'envoi d'un CR (*carriage right*, littéralement chariot à droite, c.-à-d. tête d'impression à droite) à l'imprimante. Lorsque cette dernière reçoit le CR, elle imprime les données qui se trouvaient en mémoire tampon et laisse la tête d'impression à l'extrémité droite de la ligne d'impression.

Vous pouvez empêcher la transmission automatique du CR en armant l'indicateur binaire 33. Les commandes d'impression suivantes envoient les données à l'imprimante mais n'envoient pas de CR. Les données s'accumulent donc dans la mémoire tampon de l'imprimante et ne sont imprimées qu'à votre commande. Lorsque l'indicateur binaire 33 est armé, respectez les règles suivantes :

- Envoyez un CR (caractère 4) ou un caractère de nouvelle ligne (caractère 10), ou exécutez le commande CR, chaque fois que vous voulez que l'imprimante imprime les données qu'elle a reçues.
- N'envoyez pas plus de 200 caractères sans demander une impression, sinon la mémoire tampon est pleine et les caractères suivants sont perdus.
- Laissez à l'imprimante le temps d'imprimer une ligne avant d'envoyer d'autres données. L'imprimante a besoin d'environ 1,8 secondes par ligne.
- Désarmez l'indicateur binaire 33 lorsque vous avez fini, afin de revenir à un fonctionnement normal des commandes d'impression.

PRST ***Imprimer la pile*** **Commande**

... **Niveau 1**	... **Niveau 1**
... *objet*	➡ ... *objet*

...PRINT

PRST (PRINT STACK) imprime tous les objets de la pile opérationnelle, en commençant par l'objet du plus haut niveau. Les objets sont imprimés en format d'impression multi-lignes.

PRVAR ***Imprimer variable*** **Commande**

Niveau 1	
' *nom* ' ➡	

PRVAR (PRINT VARIABLE) imprime l'objet stocké dans la variable *nom*, en format d'impression multi-lignes.

PRLCD ***Imprimer affichage*** **Commande**

➡

PRLCD (PRINT LCD) imprime une image pixel par pixel de l'affichage du HP-28C/S (à l'exclusion des témoins).

La largeur de l'image imprimée d'un objet est plus petite lorsque vous utilisez PRLCD que lorsque vous utilisez une commande d'impression comme PR1. La différence provient de l'espacement des caractères. Sur l'affichage, une seule colonne vide sépare deux caractères successifs, et PRLCD imprime cette colonne vide. En revanche, des commandes comme PR1 impriment deux colonnes vides entre deux caractères successifs.

Mode TRACE : TRACE, NORM

Vous pouvez garder une trace imprimée continue de vos calculs en validant le mode TRACE. Chaque fois que vous exécutez ENTER, soit en appuyant sur [ENTER], soit en appuyant sur une touche à exécution immédiate, le calculateur imprime le contenu de la ligne de commande, la commande à exécution immédiate, et le contenu du niveau 1 qui en résulte.

Pour valider le mode TRACE, appuyez sur TRACE. Le libellé TRACE apparaît alors en caractères noirs sur fond clair, ce qui vous confirme que le mode TRACE est validé. Vous pouvez choisir le mode TRACE dans un programme en armant l'indicateur binaire 32.

Pour invalider le mode TRACE, appuyez sur NORM. Le libellé NORM apparaît alors en caractères noirs sur fond clair, ce qui indique que le mode TRACE est invalidé. Vous pouvez invalider le mode TRACE dans un programme en désarmant l'indicateur binaire 32.

Le format d'impression de l'objet du niveau 1 varie selon que le format d'affichage multi-lignes est activé ou désactivé (indicateur binaire 45 armé ou désarmé). Si le format d'affichage multi-lignes est activé (indicateur 45 armé), l'objet est imprimé en format d'impression multi-lignes. Si le format d'affichage compact est activé (indicateur 45 désarmé), l'objet est imprimé en format d'impression compact.

PRSTC PRUSR PRMD CR

PRSTC ***Imprimer la pile (compact)*** **Commande**

... Niveau 1	... Niveau 1
... objet	➡ *... objet*

PRSTC (PRINT STACK COMPACT) imprime tous les objets de la pile, en commençant par l'objet du niveau le plus élevé. Les objets sont imprimés en format d'impression compact.

...PRINT

PRUSR ***Imprimer variables utilisateur*** **Commande**

PRUSR (PRINT USER VARIABLES) imprime le nom des variables d'utilisateur en cours. Les noms sont imprimés dans l'ordre dans lequel ils apparaissent dans le menu USER. S'il n'y a pas de variables d'utilisateur, PRUSR imprime `No User Variables` (pas de variables utilisateur).

PRMD ***Imprimer modes*** **Commande**

PRMD (PRINT MODES) affiche et imprime les choix en cours pour : le mode d'affichage numérique, la base des entiers binaires, le mode angulaire, le séparateur décimal, et l'état (validé/invalidé) des commandes UNDO, COMMAND, et LAST et de l'affichage multilignes.

CR ***Tête d'impression à droite*** **Commande**

CR (CARRIAGE RIGHT) imprime le contenu éventuel de la mémoire tampon de l'imprimante.

Programmes

Un *programme* est une procédure délimitée par des caractères « » et contenant une série de commandes, d'objets et de *structures de programme* qui sont exécutés en séquence lorsque le programme est évalué. Certaines structures de programme, comme celles décrites dans « PROGRAM BRANCH » ou celles définissant des noms locaux, doivent obéir à des règles de syntaxe spécifiques, mais le contenu d'un programme est par ailleurs beaucoup plus souple que celui d'un objet algébrique (qui est l'autre type de procédure).

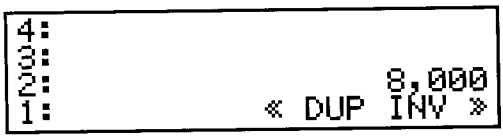
En termes simples, un programme est une ligne de commande dont l'évaluation est différée. Toute ligne de commande peut être transformée en un programme par l'insertion d'un « en début de ligne ; ensuite, lorsque vous appuyez sur [ENTER], la totalité de la ligne de commande est placée dans la pile comme un programme. Les objets individuels contenus dans le programme ne sont exécutés que lorsque le programme est évalué.

En transformant une ligne de commande en programme, vous pouvez non seulement en différer l'évaluation, mais aussi en répéter l'exécution autant de fois que vous le désirez. Vous pouvez faire un nombre quelconque de copies du programme dans la pile en utilisant les commandes ordinaires de manipulation de la pile ; ou bien vous pouvez stocker un programme dans une variable puis l'exécuter par son nom — ou en appuyant sur la touche de menu correspondante du menu USER. Une fois un programme stocké dans une variable, il ne se distingue pratiquement plus d'une commande (en fait, les commandes elles-mêmes ne sont que des programmes qui ont été introduits en mémoire morte — ROM — en usine, au lieu d'être introduites en RAM par l'utilisateur). Lorsque vous programmez le HP-28C/S, vous en augmentez le langage de programmation.

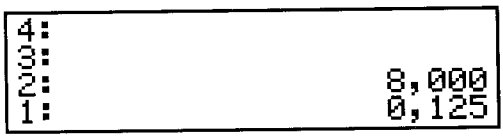
...Programmes

Evaluation des objets d'un programme

Evaluer un programme a pour effet de placer chaque objet du programme dans la pile et d'évaluer les objets qui sont des commandes ou des noms sans guillemets. Par exemple, avec la pile suivante :



appuyer sur [EVAL] donne :



DUP a été évaluée, ce qui a eu pour effet de copier 8,000 dans le niveau 2, puis INV a été évaluée, ce qui a remplacé le 8,000 du niveau par son inverse.

Programmes simples et complexes

Le type le plus simple de programme n'est qu'une séquence d'objets, qui sont exécutés séquentiellement sans interruption ni bouclage. Par exemple, le programme « 5 * 2 + » multiplie un nombre au niveau 1 par 5, puis ajoute 2 au résultat.

...Programmes

Si ceci est une opération que vous effectuez fréquemment, vous pouvez stocker le programme dans une variable, puis exécuter le programme autant de fois que vous le voulez en appuyant sur la touche du menu USER affectée à cette variable.

Vous pouvez accroître la complexité d'un programme de l'une des façons suivantes :

Branchements conditionnels. En utilisant les structures de branchement IF...THEN...END ou IF...THEN...ELSE...END (ou les commandes équivalentes IFT et IFTE), les programmes peuvent prendre des décisions sur la base de résultats calculés, et choisir des options d'exécution en fonction de ces résultats.

Boucles. Vous pouvez répéter l'exécution d'un programme ou d'une partie d'un programme un nombre défini ou indéfini de fois en utilisant les boucles FOR...NEXT, START...NEXT, DO...UNTIL...END, et WHILE...REPEAT...END.

Traitement des erreurs. En utilisant les branchements conditionnels IFERR...THEN...END ou IFERR...THEN...ELSE...END, vous pouvez permettre à un programme de traiter des erreurs prévues ou imprévues.

Suspensions. Avec la commande HALT, vous pouvez suspendre l'exécution d'un programme en des points prédéterminés pour permettre une saisie par l'utilisateur ou à d'autres fins, puis reprendre l'exécution par une pression sur ▮ CONT ou sur SST.

Programmes imbriqués. De la même façon que vous pouvez différer l'évaluation d'une ligne de commande en la plaçant entre des « », vous pouvez créer des programmes à l'intérieur d'autres programmes en mettant ces séquences de programme entre « ». Lorsqu'un tel programme « intérieur » est rencontré pendant l'exécution du programme « extérieur », il est placé dans la pile sans être évalué. Il peut être évalué par la suite avec EVAL ou toute autre commande acceptant un programme comme argument.

...Programmes

A mesure qu'un programme devient plus long et plus compliqué, il peut atteindre une taille qui le rend peu commode à lire sur l'affichage du HP-28C/S ou trop long à saisir. Pour cette raison, et pour encourager l'utilisateur à programmer dans les règles de l'art, nous vous recommandons de diviser les longs programmes en plusieurs programmes courts. Par exemple, le programme « A B C D » peut être ré-écrit sous la forme « AB CD », où AB est le programme « A B », et CD le programme « C D ».

La méthode qui consiste à écrire un long programme sous la forme d'une série de programmes courts facilite en outre la mise au point du programme. Chaque programme « secondaire » peut être testé indépendamment des autres, pour vérifier qu'il prend les nombres et types d'arguments corrects dans la pile, et qu'il y renvoie les résultats corrects. Il est ensuite simple de relier les programmes secondaires entre eux pour créer un programme principal composé des noms sans guillemets des programmes secondaires.

Variables locales et noms locaux

Une *variable locale* est la combinaison d'un objet et d'un *nom local*, qui sont stockés ensemble dans une partie de la mémoire temporairement réservée à cet effet pendant l'exécution d'une procédure. Lorsque l'exécution de la procédure est terminée, toutes les variables locales associées à cette procédure sont automatiquement supprimées.

Les *noms locaux* sont des objets servant à nommer les variables locales. La formation de ces noms est soumise aux mêmes restrictions que les noms ordinaires. Les variables locales, dans les limites des procédures qui les définissent, et les noms ordinaires sont presque interchangeables. Il existe toutefois quelques différences importantes :

- Lorsque des noms locaux sont évalués, ils renvoient dans la pile l'objet stocké dans les variables locales associées, non évalué. Ils n'évaluent pas automatiquement les noms ou programmes stockés dans leurs variables locales, comme le font les noms ordinaires.
- Vous ne pouvez pas utiliser un nom local avec guillemets comme argument de ▒ VISIT ou de l'une des commandes suivantes : CON, IDN, PRVAR, PURGE, PUT, PUTI, RDM, SCONJ, SINV, SNEG, STO+, STO-, STO*, STO/, TAYLR, ou TRN.
- Les variables locales n'apparaissent pas dans le menu des variables de la fonction de résolution d'équation (SOLVR).

Si une variable ordinaire porte le même nom qu'une variable locale, l'utilisation de ce nom dans la procédure de la variable locale réfèrera uniquement à la variable locale, et la variable ordinaire restera inchangée. De même, si une structure à variable locale se trouve imbriquée dans une autre, les noms locaux de la première structure (« extérieure ») peuvent être utilisés dans la seconde structure (« intérieure »).

Des noms locaux peuvent rester dans la pile ou dans des procédures et des listes même après que leurs variables associées aient été supprimées. Par exemple, `1 → x « 'x' »` ENTER laisse le nom local `'x'` dans la pile. Si vous essayez d'évaluer le nom local, ou si vous l'utilisez comme argument pour STO, RCL ou PURGE, vous obtenez l'erreur `Undefined Local Name` (nom local n'est pas défini).

Pour minimiser tout risque de confusion entre noms et noms locaux, nous vous recommandons d'adopter une convention spéciale pour le choix des noms locaux. Dans le présent manuel, par exemple, nous utilisons des lettres minuscules pour nommer les variables locales (qui ne peuvent jamais apparaître dans des libellés de menu), et des majuscules pour les variables ordinaires.

...Programmes

Création de variables locales

Les variables locales sont créées lors de l'utilisation de structures de programmes. La présente section décrit deux *structures à variables locales*, qui constituent le principal moyen de création de variables locales. Il existe également deux structures de branchement, FOR...NEXT et FOR...STEP, qui définissent des boucles définies dans lesquelles l'index de boucle est une variable locale.
Ces structures de branchement sont décrites dans « PROGRAM BRANCH ».

Les structures à variables locales sont de la forme :

→ *nom*$_1$ *nom*$_2$...« *programme* »

→ *nom*$_1$ *nom*$_2$...' *obj. algébrique* '

La commande → marque le début d'une structure à variable locale (le caractère → est obtenu par pression sur ■ U sur le clavier de gauche. Ici, → est une commande en soi, et elle est donc suivie d'un espace). Les *noms* définissent les noms locaux pour lesquels des variables locales sont créées. Le programme ou l'objet algébrique est appelé *procédure de définition* de la structure à variable locale. Son délimiteur initial, « ou ', met fin à la séquence des noms locaux.

Lorsque → est évaluée, elle prend un objet dans la pile pour chacun des noms locaux et stocke chaque objet dans une variable locale portant le nom correspondant. Les objets et leurs noms locaux sont organisés de façon que l'ordre des noms soit le même que celui dans lequel les objets ont été saisis dans la pile. Par exemple :

```
1 2 3 4 5 → a b c d e
```

affecte le nombre 1 à la variable locale A, 2 à b, 3 à c, 4 à d et 5 à e (comme il s'agit de variables locales, il n'y a pas de conflit avec la constante symbolique e).

Une fois que les variables locales ont été créées et leurs valeurs affectées, la procédure qui suit la liste de noms est évaluée. Dans cette procédure, vous pouvez utiliser des noms ordinaires identiques à ceux des variables locales (dans les limites énumérées plus haut). Lorsque l'exécution de la procédure est terminée, les variables locales sont automatiquement supprimées.

A titre d'exemple, supposons que vous voulez prendre 3 nombres dans la pile, multiplier le premier (niveau 3) par 4, le second (niveau 2) par 3, et le troisième (niveau 1) par 2, et ajouter les résultats. Un programmme simple pour réaliser ceci serait :

```
« 2 * SWAP 3 * + SWAP 4 * + »
```

Si vous utilisez des variables locales, le programme deviendrait :

```
« → a b c « a 4 * b 3 * + c 2 * + » »
```

L'utilisation de variables locales a éliminé les opérations SWAP. Dans ce cas simple, l'utilisation de variables locales est d'une valeur limitée, mais dans le cas d'un programme complexe, les variables locales peuvent vous aider à écrire le programme d'une manière plus simple, avec moins de risques d'erreur.

Le problème que nous avons pris comme exemple peut également être mis sous forme algébrique. Nous pouvons écrire notre programme comme suit :

```
« → a b c '4*a+3*b+2*c' »
```

et obtenir le même résultat.

...Programmes

Fonctions définies par l'utilisateur

Avec une syntaxe spéciale, la commande → peut servir à créer de nouvelles fonctions algébriques. Une fonction algébrique est une commande qui peut être utilisée dans la définition d'un objet algébrique. Dans cette définition, la fonction prend ses argument dans une séquence entre parenthèses se trouvant après le nom de la fonction. La commande SIN, par exemple, est une fonction algébrique typique nécessitant un argument. Dans une définition algébrique, elle est utilisée sous la forme `'SIN(X)'`, où `X` représente son argument.

Une *fonction définie par l'utilisateur*, avec *n* arguments, est définie par un programme ayant la syntaxe suivante :

« → nom_1 nom_2 ... nom_n '*expression*' »

où nom_1 nom_2 ... nom_n est une série de *n* noms de variables locales. *expression* est une expression algébrique contenant les noms des variables locales et représentant la définition mathématique de la fonction. Aucun objet ne peut précéder le → dans le programme, et aucun objet ne peut suivre '*expression*' .

A titre d'exemple, prenons la forme algébrique du programme défini dans la section précédente :

```
« → a b c '4*a+3*b+2*c' »
```

Il prend trois arguments, les multiplie respectivement par 4, 3 et 2 , et additionne les produits. Comme rien ne précède le → ni ne suit l'expression algébrique, ce programme est une fonction définie par l'utilisateur. Supposons que nous nommions cette fonction XYZ en stockant le programme dans la variable XYZ :

```
« → a b c '4*a+3*b+2*c' » 'XYZ' STO.
```

En notation polonaise inverse, nous pouvons exécuter `1 2 3 XYZ` pour obtenir le résultat 16 ($4 \times 1 + 3 \times 2 + 2 \times 3$). Mais nous pouvons aussi utiliser la notation algébrique :
`'XYZ(1,2,3)' EVAL` donne aussi 16 comme résultat. Vous n'êtes pas limité aux arguments numériques ; n'importe lequel des trois arguments de XYZ peut être une expression algébrique. XYZ elle-même peut être incluse dans une autre expression algébrique.

PROGRAM BRANCH

IF	IFERR	THEN	ELSE	END	
START	FOR	NEXT	STEP	IFT	IFTE
DO	UNTIL	END	WHILE	REPEAT	END

Le menu des branchements PROGRAM BRANCH (■ BRANCH) contient des commandes permettant de prendre des décisions et de réaliser des boucles dans un programme. Ces commandes ne peuvent apparaître que sous la forme d'un certain nombre de combinaisons appelées *structures de programmes*. Les structures de branchement peuvent être classées en quatre catégories : structures de décision, structures de traitement des erreurs, boucles définies, et boucles indéfinies.

Nous appellerons ci-dessous *clause* une séquence de programme quelconque;

1. Structures de décision.
   - IF *clause-test* THEN *clause-vraie* END. Si *clause-test* est vraie, exécuter *clause-vraie*. (IFT est une forme de structure à une seule commande.)
   - IF *clause-test* THEN *clause-vraie* ELSE *clause-sinon* END. Si *clause-test* est vraie, exécuter *clause-vraie* ; sinon exécuter *clause-sinon*. (IFTE est une forme de structure à une seule commande.)
2. Structures de traitement des erreurs.
   - IFERR *clause-piège* THEN *clause-erreur* END. Si une erreur se produit pendant l'exécution de *clause-piège*, exécuter *clause-erreur*.
   - IFERR *clause-piège* THEN *clause-erreur* ELSE *clause-normal* END. Si une erreur se produit pendant l'exécution de *clause-piège*, exécuter *clause-erreur* ; sinon, exécuter *clause-normale*.

3. Boucles définies.
   - *début fin* START *clause-boucle* NEXT. Exécuter *clause-boucle* une fois pour chaque valeur d'un compteur de boucle incrémenté de 1 de *début* à *fin*.
   - *début fin* START *clause-boucle pas* STEP. Exécuter *clause-boucle* une fois pour chaque valeur d'un compteur de boucle incrémenté de *pas* de *début* à *fin*.
   - *début fin* FOR *nom clause-boucle* NEXT. Exécuter *clause-boucle* une fois pour chaque valeur d'un *nom* de variable locale, utilisé comme compteur de boucle, incrémenté de 1 de *début* à *fin*.
   - *début fin* FOR *nom clause-boucle pas* STEP. Exécuter *clause-boucle* une fois pour chaque valeur d'un *nom* de variable locale, utilisé comme compteur de boucle, incrémenté de *pas* de *début* à *fin*.
4. Boucles indéfinies.
   - DO *clause-boucle* UNTIL *test-clause* END. Exécuter *clause-boucle* de manière répétée jusqu'à ce que *clause-test* soit vraie.
   - WHILE *test-clause* REPEAT *clause-boucle* END. Pendant que *clause-test* est vraie, exécuter *clause-boucle* de manière répétée.

Ces structures sont décrites plus loin dans la présente section.

...PROGRAM BRANCH

Tests et indicateurs binaires

Toutes les structures de programme (sauf les boucles définies) réalisent les décisions de branchement en fonction de l'évaluation d'une *clause test*. Une clause test est une séquence de programme qui renvoie un *indicateur binaire* lorsqu'elle est évaluée. Un indicateur binaire est un nombre réel ordinaire qui a nominalement la valeur 0 ou 1. S'il a la valeur 0, on dit qu'il est « faux » ou « désarmé » ; s'il a une autre valeur, on dit qu'il est « vrai » ou « armé ».

Toutes les décisions de branchement sont réalisées par test (vérification de l'état) d'un indicateur binaire pris dans la pile. Par exemple, dans une structure IF *clause-test* THEN *clause-vraie* END, si l'évaluation de *clause-test* donne un résultat (réel) non nul, la *clause-vraie* est évaluée. Si la *clause-test* donne un 0 au niveau 1, l'exécution saute à l'objet se trouvant immédiatement après END.

Une commande de *test* est une commande qui renvoie explicitement un indicateur binaire avec une valeur 0 ou 1. Par exemple, la commande < teste deux nombres réels (ou entiers binaires, ou chaînes) pour voir si le nombre se trouvant au niveau 2 est inférieur à celui se trouvant au niveau 1. Si c'est le cas, < renvoie l'indicateur binaire 1 ; sinon, il renvoie 0. Les autres commandes de test sont >, ≤, ≥, ==, ≠, FS?, FC?, FS?C, et FC?C ; toutes ces commandes sont décrites dans « PROGRAM TEST ».

Remplacement de GOTO

Les programmeurs habitués aux autres langages de programmation pour calculateurs, en particulier celui utilisant la notation polonaise inverse sur les autres calculateurs HP, ou le BASIC, auront peut-être noté l'absence de l'instruction GOTO dans le HP-28C/S. Cette instruction est couramment utilisée pour effectuer des branchements en fonction d'un test et pour minimiser la taille d'un programme en réutilisant des pas de programme. Nous allons examiner comment les instructions GOTO sont utilisées dans le HP-41 (en notation polonaise inverse) et en BASIC, et montrer comment obtenir des résultats équivalents sur le HP-28C/S.

...PROGRAM BRANCH

- Utilisation d'instructions GOTO pour réaliser des branchements en fonction d'un test. Par exemple, les programmes ci-dessous exécutent la séquence ABC DEF si le nombre dans le registre ou la variable X est positif ; sinon ils exécutent la séquence GHI JKL.

HP-41	BASIC
01 X>0? 02 GTO 01 03 GHI 04 JKL 05 GTO 02 06 LBL 01 07 ABC 08 DEF 09 LBL 02 ⋮	10 IF X>0 THEN GOTO 50 20 GHI 30 JKL 40 GOTO 70 50 ABC 60 DEF ⋮

Voici un équivalent sur le HP-28C/S :

```
IF 0 > THEN ABC DEF ELSE GHI JKL END
```

- Utilisation de GOTO pour réduire la taille d'un programme en réutilisant des pas de programme. Les deux exemples contiennent une séquence MNO PQR STU commune à deux branchements.

HP-41	BASIC
01 ABC 02 DEF 03 GTO 01 ⋮ 10 GHI 11 JKL 12 GTO 01 ⋮ 20 LBL 01 21 MNO 22 PQR 23 STU ⋮	10 ABC 20 DEF 30 GOTO 200 ⋮ 100 GHI 110 JKL 120 GOTO 200 ⋮ 200 MNO 210 PQR 220 STU ⋮

...PROGRAM BRANCH

Dans le HP-28C/S, la séquence commune MNO PQR STU... serait stockée comme un programme séparé :

```
« MNO PQR STU ... » 'COMMUN' STO
```

Puis chaque branchement du programme exécuterait COMMUN :

```
... ABC DEF COMMUN ... GHI JKL COMMUN ...
```

L'avantage de la programmation sur le HP-28C/S est qu'un programme a seulement une entrée et une sortie. Il devient donc simple d'écrire des programmes et de les tester individuellement. Lorsque vous combinez des sous-programmes en un programme principal, il ne vous reste plus qu'à tester si les programmes fonctionnent ensemble comme vous l'avez prévu.

IF IFERR THEN ELSE END

Ces commandes peuvent être combinées en un grand nombre de structures de décision et de traitement des erreurs.

IF *clause-test* THEN *clause-vraie* END. La commande THEN prend un indicateur binaire dans la pile. S'il est vrai (non nul), la *clause-vraie* est évaluée, puis l'exécution se poursuit après END. S'il est faux (0), l'exécution saute directement à l'objet se trouvant après END. (Notez que seul THEN utilise réellement l'indicateur — la position du IF est arbitraire tant qu'il précède THEN. *clause-test* IF THEN fonctionne de la même façon que IF *clause-test* THEN). Par exemple :

```
IF X 0 > THEN "Positif" END
```

envoie dans la pile la chaîne `"Positif"` si X contient un nombre réel positif.

IF *clause-test* THEN *clause-vraie* ELSE *clause-fausse* END. La commande THEN prend un indicateur binaire dans la pile. S'il est vrai (non nul), la *clause-vraie* est évaluée, puis l'exécution se poursuit après END. S'il est faux (0), la *clause-fausse* est évaluée, puis l'exécution se poursuit après END (notez que seul THEN utilise réellement l'indicateur — la position du IF est arbitraire tant qu'il précède THEN. *clause-test* IF THEN fonctionne de la même façon que IF *clause-test* THEN). Par exemple :

```
IF X 0 ≥ THEN "Positif" ELSE "Negatif" END
```

envoie dans la pile la chaîne `"Positif"` si X contient un nombre réel non-négatif, ou la chaîne `"Negatif"` si X contient un nombre réel négatif.

IFERR *clause-piège* THEN *clause-erreur* END. Cette structure évalue *clause-erreur* si une erreur se produit pendant l'exécution de *clause-piège*.

Lorsque *clause-piège* est évaluée, les éléments successifs de la clause sont normalement exécutés, sauf si une erreur se produit. Dans ce cas, l'exécution saute à *clause-erreur*. Le reste de *clause-piège* est supprimé. Par exemple :

```
IFERR WHILE 1 REPEAT + END THEN "OK" 1 DISP END
```

additionne tous les nombres de la pile. La fonction + est exécutée itérativement jusqu'à ce qu'une erreur se produise, ce qui indique alors que la pile est vide (ou qu'un type d'objet inadéquat a été trouvé). La *clause-erreur* affiche alors `OK`.

Lorsque vous écrivez des clauses d'erreur, gardez à l'esprit que l'état de la pile après une erreur peut dépendre du fait que LAST est validée ou non. Si LAST est validée, les commandes donnant une erreur renvoient leurs arguments dans la pile ; sinon les arguments sont supprimés.

...PROGRAM BRANCH

IFERR *clause-piège* THEN *clause-erreur* ELSE *clause-normale* END. Cette structure vous permet de spécifier qu'une *clause-erreur* doit être évaluée s'il se produit une erreur pendant l'exécution d'une *clause-piège*, et de spécifier qu'une *clause-normale* doit être exécutée s'il ne se produit aucune erreur.

Lorsque *clause-piège* est évaluée, les éléments successifs de la clause sont exécutés normalement, à moins qu'il se produise une erreur.

- S'il se produit une erreur, le reste de la *clause-piège* est supprimé et la *clause-erreur* est évaluée.
- S'il ne se produit pas d'erreur, l'évaluation de la *clause-piège* est suivie par l'évaluation de la *clause-normale*.

Dans les deux cas, l'exécution se poursuit ensuite au-delà de END.

START FOR NEXT STEP IFT IFTE

***début fin* START *clause-boucle* NEXT.** La commande START prend deux nombres réels, *début* et *fin*, dans la pile et les stocke comme valeurs de début et de fin d'un compteur de boucle. Ensuite, une séquence d'objets *clause-boucle* est évaluée. La commande NEXT incrémente le compteur de boucle de 1 ; si le compteur est inférieur ou égal à *fin*, *clause-boucle* est évaluée à nouveau. Ce processus se poursuit jusqu'à ce que le compteur dépasse le nombre *fin* ; l'exécution continue alors après NEXT. Par exemple :

```
1 10 START XYZ NEXT
```

évalue XYZ 10 fois.

***début fin* START *clause-boucle incrément* STEP.** Cette structure est similaire à celle de START...NEXT, sauf que STEP incrémente le compteur de boucle d'une quantité variable, alors NEXT l'incrémente toujours de 1.

START prend deux nombres réels, *début* et *fin*, dans la pile et les stocke comme première et dernière valeurs d'un compteur de boucle. Une séquence d'objets *clause-boucle* est ensuite évaluée. STEP incrémente le compteur de boucle du nombre réel *incrément* pris dans le niveau 1.

Si *pas* est positif et si le compteur de boucle est inférieur ou égal à *fin*, *clause-boucle* est évaluée à nouveau. Ceci continue jusqu'à ce que le compteur de boucle dépasse *fin*, à la suite de quoi l'exécution se poursuit après STEP.

Si *pas* est négatif et si le compteur de boucle est supérieur ou égal à *fin*, *clause-boucle* est évaluée à nouveau. Ceci se poursuit jusqu'à ce le compteur de boucle soit inférieur à *fin*, à la suite de quoi l'exécution continue après STEP. Par exemple :

```
10 1 START XYZ -2 STEP
```

évalue XYZ cinq fois.

***début fin* FOR *nom clause-boucle* NEXT.** Cette structure est une boucle définie dans le laquelle le compteur de boucle *nom* est une variable locale qui peut être évaluée dans la boucle (le nom suivant FOR doit être saisi sans guillemets). En séquence :

1. FOR prend deux nombres réels, *début* et *fin*, dans la pile. Elle crée une variable locale *nom* et stocke *début* comme la valeur initiale de *nom*.
2. La séquence d'objets *clause-boucle* est évaluée. Si *nom* est évalué dans la séquence, il donne la valeur en cours du compteur de boucle.
3. NEXT incrémente le compteur de boucle de 1. Si la valeur du compteur dépasse alors *fin*, l'exécution se poursuit avec l'objet suivant NEXT, et la variable locale *nom* est supprimée. Sinon les étapes 2 et 3 se répètent.

Par exemple :

```
1 5 FOR × × SQ NEXT
```

place les carrés des entiers 1 à 5 dans la pile.

***début fin* FOR *nom clause-boucle incrément* STEP.** Cette structure est une boucle définie dans le laquelle le compteur de boucle *nom* est une variable locale qui peut être évaluée dans la boucle (le nom suivant FOR doit être saisi sans guillemets). Elle est similaire à FOR...NEXT, sauf que le compteur de boucle est incrémenté d'une quantité variable. En séquence :

1. FOR prend deux nombres réels, *début* et *fin*, dans la pile. Elle crée une variable locale *nom* et stocke *début* comme la valeur initiale de *nom*.
2. La séquence d'objets *clause-boucle* est évaluée. Si *nom* est évalué dans la séquence, il donne la valeur en cours du compteur de boucle.
3. STEP prend le nombre réel *incrément* dans la pile et incrémente le compteur de boucle de la quantité représentée par *incrément*. Si le compteur de boucle est supérieur à *fin* (pour *incrément* > 0) ou inférieur à *fin* (pour *incrément* < 0), l'exécution se poursuit avec l'objet suivant immédiatement STEP, et la variable locale *nom* est supprimée. Sinon les étapes 2 et 3 sont répétées.

Par exemple :

```
1 11 FOR × × SQ 2 STEP
```

place les carrés des entiers 1, 3, 5, 7, 9, et 11 dans la pile.

IFT *If-Then* Commande

Niveau 2	Niveau 1	
indic. binaire	*objet*	➧

IFT est une forme de IF...THEN...END à une seule commande. IFT prend un indicateur binaire dans le niveau 2 et un objet arbitraire dans le niveau 1. Si l'indicateur binaire est vrai (non nul), l'objet est évalué ; si l'indicateur est faux (0), l'objet est supprimé. Par exemple :

```
X 0 > "Positif" IFT
```

envoie `"Positif"` dans le niveau 1 si X contient un nombre réel positif.

IFTE *If-Then-Else* Fonction

Niveau 3	Niveau 2	Niveau 1	
indic.binaire	*objet-vrai*	*objet-faux*	➧

IFTE est une forme de IF...THEN...ELSE...END à une seule commande. IFTE prend un indicateur binaire dans le niveau 3 et deux objets arbitraire dans les niveaux 1 et 2. Si l'indicateur est vrai (non nul), *objet-faux* est supprimé et *objet-vrai* est évalué. Si l'indicateur est faux (0), *objet-vrai* est supprimé et *objet-faux* est évalué. Par exemple :

```
X 0 ≥ "Positif" "Negatif" IFTE
```

envoie `"Positif"` dans la pile si X contient un nombre réel non négatif, ou envoie `"Negatif"` si X contient un nombre réel négatif.

IFTE est également utilisable dans les expressions algébriques, avec la syntaxe suivante :

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

...PROGRAM BRANCH

Lorsqu'une expression algébrique contenant IFTE est évaluée, son premier argument *expression-test* est évalué comme un indicateur binaire. S'il envoie un nombre réel non nul, *expression-vraie* est évaluée. S'il envoie un zéro, *expression-fausse* est évaluée. Par exemple :

```
'IFTE(X≠0,SIN(X)/X,1)'
```

est une expression qui renvoie la valeur de $\sin(x)/x$, même pour $x = 0$, ce qui causerait normalement une erreur `Infinite Result` (résultat infini).

DO UNTIL END WHILE REPEAT END

DO *clause-boucle* UNTIL *clause-test* END. Cette structure évalue de manière répétée une *clause-boucle* et une *clause-test*, jusqu'à ce que l'indicateur renvoyé par *clause-test* soit vrai (non nul). Par exemple :

```
DO X INCX X - UNTIL ,0001 < END.
```

Ici, INCX est un exemple de programme qui incrémente la variable X d'une petite quantité. La routine exécute itérativement INCX jusqu'à ce que la modification obtenue dans X soit inférieure à 0,0001.

WHILE *clause-test* REPEAT *clause-boucle* END. Cette structure évalue itérativement une *clause-test* et une *clause-boucle*, tant que l'indicateur binaire renvoyé par *clause-test* est vrai (non nul). Lorsque *clause-test* renvoie un indicateur binaire faux, la *clause-boucle* est sautée et l'exécution reprend après END. La *clause-test* renvoie un nombre réel, que REPEAT teste en tant qu'indicateur binaire. Par exemple :

```
WHILE CHAINE "P" POS REPEAT REMOVEP END.
```

Ici, REMOVEP est un programme qui enlève un caractère `P` d'une chaîne stockée dans la variable CHAINE. La séquence se répète jusqu'à ce qu'il ne reste plus de `P` dans la chaîne.

PROGRAM CONTROL

SST	**HALT**	**ABORT**	**KILL**	**WAIT**	**KEY**
BEEP	**CLLCD**	**DISP**	**CLMF**	**ERRN**	**ERRM**

Le menu PROGRAM CONTROL (■ CTRL) contient des commandes permettant de suspendre l'exécution des programmes et autorisant ainsi une interaction avec l'utilisateur.

Suspension de programmes

L'évaluation d'un programme entraîne normalement l'exécution des objets qu'il contient d'une manière continue jusqu'à la fin du programme. Les commandes du menu PROGRAM CONTROL permettent de prévoir des pauses dans des programmes ou d'en suspendre l'exécution autrement qu'à la fin :

Commande	Description
HALT	Suspend l'exécution d'un programme, qui reprendra ultérieurement.
ABORT	Arrête l'exécution d'un programme, qui ne pourra pas reprendre par la suite.
KILL	Arrête l'exécution d'un programme, et efface également tous les programmes suspendus.
WAIT	Réalise une pause dans l'exécution d'un programme, qui reprend automatiquement au bout d'une durée donnée.

Un programme *suspendu* est un programme dont l'exécution est provisoirement interrompue, de manière à *recommencer* à l'endroit où elle s'était arrêtée. Pendant qu'un programme est suspendu, vous pouvez effectuer n'importe quelle opération du HP-28C/S (sauf un arrêt du système, une réinitialisation de la mémoire, et la commande KILL) — saisir des données, visualiser des résultats, exécuter d'autres programmes, etc. — puis reprendre le programme là où il avait été arrêté.

...PROGRAM CONTROL

Le témoin **O** indique qu'un ou plusieurs programmes sont suspendus.

La commande HALT suspend l'exécution d'un programme à l'endroit où le HALT se trouve dans le programme. Pour relancer l'exécution du programme, vous pouvez :

- Appuyer sur ■ CONT (continuer) pour reprendre l'exécution continue à partir de l'objet suivant le HALT. Vous pouvez utiliser HALT en combinaison avec ■ CONT dans un programme lorsque vous voulez arrêter le programme pour la saisie d'une donnée par l'utilisateur, puis reprendre l'exécution.
- Appuyer sur SST (single-step, c.-à-d. exécution pas à pas — dans le menu PROGRAM CONTROL) pour exécuter l'objet suivant le HALT. L'utilisation répétée de SST continue l'exécution du programme, pas à pas. Ceci est un puissant outil de mise au point du programme, car vous pouvez voir la pile ou tout autre « état » du calculateur après chaque pas de programme.

Si vous ne choisissez pas l'une de ces options, le programme restera suspendu indéfiniment, sauf si vous exécutez KILL ou un arrêt système, ce qui efface tous les programmes suspendus.

Vous pouvez « imbriquer » les suspensions de programmes les unes dans les autres — c'est-à-dire que vous pouvez exécuter un programme contenant un HALT alors qu'un autre programme est déjà suspendu. Si vous continuez (■ CONT) le second programme, l'exécution s'arrêtera à nouveau à la fin du programme. Vous pourrez alors appuyer à nouveau sur ■ CONT pour reprendre l'exécution du premier programme.

Pendant qu'un programme est suspendu, les processus de sauvegarde et de récupération associés à UNDO sont « locaux » au programme. Consultez « MODE » pour des informations sur l'utilisation de UNDO avec des programmes suspendus.

SST HALT ABORT KILL WAIT KEY

Exécution pas à pas

SST (SINGLE STEP) exécute le pas suivant d'un programme suspendu. Dans ce contexte, le pas suivant signifie l'objet ou la commande qui suit, dans l'ordre d'exécution du programme, l'objet ou la commande le plus récemment évalué.

Lorsque vous appuyez sur SST, le pas de programme sur le point d'être exécuté est affiché brièvement en vidéo inverse, puis il est exécuté. Après chaque pas, la pile et les libellés de menu sont affichés en vidéo normale. Entre les pas, vous pouvez effectuer des opérations sans affecter les programmes suspendus. Naturellement, si vous modifiez la pile, vous devez vous assurer qu'elle contient les objets appropriés avant de reprendre l'exécution du programme.

Pour toutes les boucles de programme définies avec FOR...NEXT, START...NEXT, DO...UNTIL...END ou WHILE...REPEAT...END, la commande initiale (FOR, START, DO, ou WHILE) n'est affichée comme un pas que la première fois. Lors des itérations successives, chaque boucle par le premier objet ou la première commande immédiatement après la commande initiale de la boucle.

Si une erreur se produit pendant que vous évaluez un objet en exécution pas à pas, cette dernière ne passe pas au pas suivant. Ceci vous permet de corriger la source de l'erreur, puis d'exécuter à nouveau le pas qui contenait l'erreur.

Une pression sur SST lorsque le pas suivant est une IFERR entraîne l'exécution de la totalité de la structure IFERR...THEN...END ou IFERR...THEN...ELSE...END, en un seul pas. Si vous voulez progresser pas à pas dans un clause de la structure, incluez un HALT à l'intérieur de cette clause.

De même, une pression sur SST lorsque → est affiché entraîne l'exécution de la totalité de la structure → nom_1 nom_2 ... nom_n en un seul pas. Si les noms locaux sont suivis par un objet algébrique, ce dernier est immédiatement évalué, dans le même pas.

HALT ***Suspendre le programme*** **Commande**

```
→
```

HALT suspend l'exécution d'un programme à l'emplacement de la commande HALT dans le programme. HALT :

1. Fait apparaître le témoin **O**.
2. Si UNDO est validée, affecte de la mémoire à la sauvegarde temporaire de la pile.
3. Redonne au clavier ses fonctions de commande du calculateur, pour permettre des opérations normales.

Les programmes que vous relancerez par ▮ CONT ou par ▮ SST reprendront à partir de l'objet qui suit immédiatement la commande HALT dans le programme.

ABORT ***Abandonner le programme*** **Commande**

```
→
```

ABORT arrête l'exécution d'un programme à l'endroit où se trouve la commande ABORT dans le programme. L'exécution du programme ne peut pas reprendre.

KILL — *Abandonner programmes suspendus* — Commande

➡

KILL met fin à l'exécution du programme en cours et à celle de tous les autres programmes suspendus. Il est alors impossible de reprendre l'exécution de ces programmes à l'endroit où ils ont été arrêtés.

WAIT — *Attendre* — Commande

Niveau 1	
x ➡	

WAIT fait une pause de *x* secondes dans l'exécution d'un programme.

KEY — *Touche* — Commande

	Niveau 2	Niveau 1
➡		0
➡	"*chaîne*"	1

KEY renvoie dans la pile une chaîne représentant la touche la moins récemment actionnée parmi celles qui sont sauvegardées dans la mémoire tampon de stockage des touches actionnées ; dans le même temps, cette touche est supprimée de la mémoire tampon. Si la mémoire tampon est vide, KEY renvoie un indicateur binaire « faux » (0). Si elle contient une ou plusieurs touches, KEY supprime de la mémoire tampon la touche la moins récemment actionnée et renvoie un indicateur binaire « vrai » (1) au niveau 1 et une chaîne au niveau 2. Cette chaîne « nomme » la touche qui vient d'être éliminée de la mémoire tampon.

Cette mémoire tampon du HP-28C/S peut contenir jusqu'à 15 touches qui ont été actionnées mais pas encore traitées. Lorsque KEY supprime une touche de cette mémoire tampon, la touche est convertie en une chaîne compréhensible, qui représente le ou les caractère(s) se trouvant sur le dessus de la touche, à l'exception de :

...PROGRAM CONTROL

Touche	Chaîne
SPACE	" "
LC	"1"
INS	"INS"
DEL	"DEL"
▲	"UP"
▼	"DOWN"
◀	"LEFT"
▶	"RIGHT"
◀▶	"CURSOR"
◀	"BACK"

La touche ON conserve sa fonction ATTN et interrompt le programme en cours.

L'action de KEY peut être illustrée par le programme suivant :

```
« DO UNTIL KEY END "Y" SAME ».
```

Lorsque ce programme est exécuté, une pression sur Y renvoie 1 (vrai) au niveau 1, et une pression sur n'importe quelle autre touche renvoie 0 (faux).

BEEP CLLCD DISP CLMF ERRN ERRM

BEEP — *Emettre une tonalité* — Commande

Niveau 2	Niveau 1	
fréquence	*durée*	➧

BEEP émet une tonalité (bip) avec la *fréquence* et la *durée* spécifiées. La *fréquence* est exprimée en Hertz (arrondie à un entier) et la *durée* en secondes.

La fréquence de la tonalité est limitée par la résolution du générateur de tonalité du calculateur. La fréquence maximale est d'environ 4 400 Hz ; la durée maximale est de 1 048,575 secondes (# FFFFF msec). Les arguments supérieurs à ces valeurs maximales seront ramenés par défaut à ces valeurs.

Si vous armez l'indicateur binaire 51, la tonalité est invalidée, et l'exécution de BEEP ne produira donc aucun son.

CLLCD — *Effacer affichage* — Commande

➧

CLLCD (CLEAR LCD) efface l'affichage, à l'exception des témoins, et arme l'indicateur binaire système pour supprimer l'affichage normal de la pile et du menu.

...PROGRAM CONTROL

DISP *Afficher* Commande

Niveau 2	Niveau 1	
objet	*n*	➡

DISP (DISPLAY) affiche *objet* à la *n*ième ligne de l'affichage, *n* étant un entier réel. $n = 1$ spécifie la ligne supérieure de l'affichage, $n = 4$ la ligne inférieure. DSP arme l'indicateur binaire système pour supprimer l'affichage normal de la pile.

Un objet est affiché par DSIP sous la même forme que s'il se trouvait au niveau 1 en format d'affichage multi-lignes, sauf pour les chaînes, qui sont affichées ans leur délimiteurs " pour faciliter l'affichage des messages. Si l'affichage de l'objet nécessite plus d'une ligne, l'affichage commence en ligne *n* et se poursuit soit jusqu'à la fin de l'objet, soit jusqu'en bas de l'affichage.

CLMF *Effacer indic. binaire système* Commande

	➡

CLMF (CLEAR MESSAGE FLAG) désarme l'indicateur binaire système armé par CLLCD, DISP, PIXEL, DRAX, DRAW et DRWΣ. Inclure CLMF dans un programme après l'une de ces commandes rétablit l'affichage normal de la pile à la fin de l'exécution du programme.

ERRN ***Numéro d'erreur*** **Commande**

	Niveau 1
	➡ # *n*

ERRN (ERROR NUMBER) donne un entier binaire égal au numéro de l'erreur la plus récente du calculateur. Vous trouverez en annexe A un tableau des erreurs, messages d'erreur et numéros d'erreurs du HP-28C/S.

ERRM ***Message d'erreur*** **Commande**

	Niveau 1
	➡ " *message-d'erreur* "

ERRM (ERROR MESSAGE) donne une chaîne contenant le message d'erreur le plus récent du calculateur. Vous trouverez en annexe A un tableau des erreurs, messages d'erreur et numéros d'erreurs du HP-28C/S.

PROGRAM TEST

SF	**CF**	**FS?**	**FC?**	**FS?C**	**FC?C**
AND	**OR**	**XOR**	**NOT**	**SAME**	==
STOF	**RCLF**	**TYPE**			

Le menu PROGRAM TEST (■ TEST) contient des commandes permettant de modifier et de tester des indicateurs binaires, et d'exécuter des calculs logiques.

Les commandes de test renvoient un *indicateur binaire* à la suite d'une comparaison entre deux arguments, ou à la suite du test d'un indicateur binaire d'utilisateur. Les opérateurs de comparaison ≠, <, >, ≤, et ≥ sont présents en tant que caractères sur le clavier de gauche. Le reste des commandes de test (FS?, FC?, FS?C, FC?C, SAME, et ==) se trouve dans le menu TEST. Ce dernier contient en outre les opérations logiques AND, OR, XOR et NOT, qui vous permettent de combiner les valeurs des indicateurs binaires. Remarquez que la fonction = n'est pas un opérateur de comparaison ; elle définit une équation. == et SAME testent toutes deux l'égalité de deux objets.

Fonctions disponibles au clavier

≠ **Différent de** **Fonction**

Niveau 2	Niveau 1		Niveau 1
$objet_1$	$objet_2$	➡	*indicateur binaire*
z	' *symbole* '	➡	' *z≠symbole* '
' *symbole* '	*z*	➡	' *symbole≠z* '
' $symbole_1$ '	' $symbole_2$ '	➡	' $symbole_1 \neq symbole_2$ '

≠ prend deux objets dans les niveaux 1 et 2 et :

- Si l'un des objets n'est ni un objet algébrique ni un nom, cette fonction renvoie une valeur « faux » (0) lorsque les deux objets sont du même type et ont la même valeur, et une valeur « vrai » (1) dans les autres cas. Les listes et programmes sont considérés comme ayant la même valeur si les objets qu'il contiennent sont identiques.
- Si l'un des objets est un objet algébrique ou un nom et l'autre un nombre, un nom ou un objet algébrique, ≠ renvoie une expression représentant une comparaison symbolique de la forme '*symbole*$_1$≠*symbole*$_2$', où *symbole*$_1$ est l'objet pris au niveau 2 et *symbole*$_2$ celui pris au niveau 1. L'expression résultat peut être évaluée avec EVAL ou →NUM, et elle renvoie alors une valeur « vrai » ou « faux » dans la pile.

< ***Inférieur à*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic.binaire*
# n_1	# n_2	➧	*indic. binaire*
"*chaîne*$_1$"	"*chaîne*$_2$"	➧	*indic. binaire*
x	'*symbole*'	➧	'*x*<*symbole*'
'*symbole*'	*x*	➧	'*symbole*<*x*'
'*symbole*$_1$'	'*symbole*$_2$'	➧	'*symbole*$_1$<*symbole*$_2$'

> ***Supérieur à*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic. binaire*
# n_1	# n_2	➧	*indic. binaire*
"*chaîne*$_1$"	"*chaîne*$_2$"	➧	*indic. binaire*
x	'*symbole*'	➧	'*x*>*symbole*'
'*symbole*'	*x*	➧	'*symbole*>*x*'
'*symbole*$_1$'	'*symbole*$_2$'	➧	'*symbole*$_1$>*symbole*$_2$'

...PROGRAM TEST

≤ *Inférieur ou égal à* **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic. binaire*
# n_1	# n_2	➧	*indic. binaire*
"*chaîne*$_1$"	"*chaîne*$_2$"	➧	*indic. binaire*
x	'*symbole*'	➧	'*x*≤*symbole*'
'*symbole*'	*x*	➧	'*symbole*≤*x*'
'*symbole*$_1$'	'*symbole*$_2$'	➧	'*symbole*$_1$≤*symbole*$_2$'

≥ *Supérieur ou égal à* **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic. binaire*
# n_1	# n_2	➧	*indic. binaire*
"*chaîne*$_1$"	"*chaîne*$_2$"	➧	*indic. binaire*
x	'*symbole*'	➧	'*x*≥*symbole*'
'*symbole*'	*x*	➧	'*symbole*≥*x*'
'*symbole*$_1$'	'*symbole*$_2$'	➧	'*symbole*$_1$≥*symbole*$_2$'

La description ci-dessous réfère aux quatre diagrammes ci-dessus, qui représentent la pile opérationnelle.

Chacune des quatre commandes <, >, ≤ et ≥ prend deux objets dans la pile, applique la comparaison logique correspondant au nom de la commande, et renvoie une valeur « vrai » ou « faux » en fonction du résultat de la comparaison. L'ordre logique de la comparaison est *niveau 2 test niveau 1*, où *test* représente l'une des quatre comparaisons. Par exemple, si le niveau 2 contient un nombre réel *x*, et le niveau 1 un nombre réel *y*, la fonction < renvoie une valeur « vrai » (1) si *x* est inférieur à *y*, et une valeur « faux » (0) dans tous les autres cas.

Du fait qu'elle impliquent une séquence dans un ordre donné, $<$, $>$, $\leqslant$, et $\geqslant$, s'appliquent à un moins grand nombre de types d'objets que $\neq$, $==$, ou SAME :

- Pour les nombres réels et entiers binaires, « inférieur à » signifie numériquement plus petit (1 est inférieur à 2). Pour les nombres réels, « inférieur à » signifie aussi « plus négatif que » (-2 est inférieur à -1).
- Pour les chaînes, « inférieur à » signifie alphabétiquement antérieur à ("ABC" est inférieur à "DEF" ; "AAA" est inférieur à "AAB" ; "A" est inférieur à "AA"). En général, les caractères sont classés en fonction de leur valeur décimale. Cela signifie donc que "B" est inférieur à "a", car "B" a la valeur décimale 66, et "a" la valeur 97.

SF CF FS? FC? FS?C FC?C

Ce groupe de commandes arme, désarme, et teste les 64 indicateurs binaires d'utilisateur. Dans ce contexte, « armer » signifie « rendre vrai » ou « affecter la valeur 1 », et « désarmer » signifie « rendre faux » ou « affecter la valeur 0 ».

SF — *Armer indic. binaire* — Commande

Niveau 1	
n ➧	

SF (SET FLAG) arme l'indicateur binaire d'utilisateur spécifié par l'argument entier réel *n*, avec $1 \leqslant n \leqslant 64$.

CF — *Désarmer indic. binaire* — Commande

Niveau 1	
n ➧	

CF (CLEAR FLAG) désarme l'indicateur binaire d'utilisateur spécifié par l'argument entier réel *n*, avec $1 \leqslant n \leqslant 64$.

...PROGRAM TEST

FS? *Indic. binaire armé ?* Commande

Niveau 1	Niveau 1
n	➧ *indic. binaire*

FS? (FLAG SET?) teste (vérifie l'état de) l'indicateur binaire d'utilisateur spécifié par l'argument entier réel *n*, avec $1 \leqslant n \leqslant 64$. Si l'indicateur binaire d'utilisateur est armé, FS? renvoie une valeur « vrai » (1) ; sinon, elle renvoie une valeur « faux » (0).

FC? *Indic. binaire désarmé ?* Commande

Niveau 1	Niveau 1
n	➧ *indic. binaire*

FC? (FLAG CLEAR?) teste l'indicateur binaire d'utilisateur spécifié par l'argument entier réel *n*, avec $1 \leqslant n \leqslant 64$. Si l'indicateur binaire d'utilisateur est désarmé, FC? renvoie une valeur « vrai » (1) ; sinon, elle renvoie une valeur « faux » (0).

FS?C *Indic. binaire armé ? Désarmer* Commande

Niveau 1	Niveau 1
n	➧ *indic. binaire*

FS?C (FLAG SET? CLEAR) teste, puis désarme éventuellement, l'indicateur binaire d'utilisateur spécifié par l'argument entier réel *n*, avec $1 \leqslant n \leqslant 64$. Si l'indicateur binaire d'utilisateur est armé, FS?C renvoie une valeur « vrai » (1) ; sinon, elle renvoie une valeur « faux » (0).

FC?C *Indic. binaire désarmé ? Désarmer* Commande

Niveau 1	Niveau 1
n	➧ *indic. binaire*

FC?C (FLAG CLEAR? CLEAR) teste, puis désarme éventuellement, l'indicateur binaire d'utilisateur spécifié par l'argument entier réel *n*, avec $1 \leqslant n \leqslant 64$. Si cet indicateur binaire est désarmé, FC?C renvoie une valeur « vrai » (1) ; sinon, elle renvoie une valeur « faux » (0).

AND OR XOR NOT SAME ==

Les commandes AND, OR, XOR, et NOT peuvent être appliquées à des *indicateurs binaires* (nombres réels ou expressions algébriques) et à des entiers binaires. Dans le premier cas, les commandes fonctionnent comme des opérateurs logiques combinant les valeurs « vrai » et « faux » en des valeurs résultats « vrai » ou « faux ». Dans le cas des entiers binaires, les commandes réalisent des combinaisons logiques avec les bits individuels des arguments.

La description ci-dessous s'applique à l'utilisation des commandes avec des entiers réels comme arguments (indic. binaires). Vous trouverez dans la section « BINARY » leur application aux entiers binaires.

AND, OR, XOR, et NOT sont autorisées dans les objets algébriques. AND et NOT ont priorité sur OR ou XOR. AND, OR, et XOR sont affichées dans les objets algébriques comme des opérateurs *infixes* :

```
'X AND Y' '5+X XOR Z AND Y'
```

NOT apparaît comme un opérateur *préfixe* :

```
'NOT X' 'Z+NOT (A AND B)'
```

Si vous saisissez les commandes sous cette forme, n'oubliez pas d'utiliser des espaces pour les séparer des autres commandes ou objets. Vous pouvez également les saisir dans la ligne de commande sous la forme d'opérateurs préfixes :

```
'AND(X,Y)' 'AND(XOR(X,Z),Y)'
```

...PROGRAM TEST

AND *ET* Fonction

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic.binaire*
x	'*symbole*'	➧	'*x* AND *symbole*'
'*symbole*'	*x*	➧	'*symbole* AND *x*'
'$symbole_1$'	'$symbole_2$'	➧	'$symbole_1$ AND $symbole_2$'

AND donne une valeur logique (ou *état*) qui est le ET logique de deux autres valeurs logiques :

Premier argument x	Second argument y	Résultat de AND
vrai	vrai	vrai
vrai	faux	faux
faux	vrai	faux
faux	faux	faux

Si l'un ou les deux arguments sont des objets algébriques, le résultat est un objet algébrique de la forme '$symbole_1$ AND $symbole_2$', où $symbole_1$ et $symbole_2$ représentent les arguments.

OR *OU* Fonction

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic. binaire*
x	'*symbole*'	➧	'*x* OR *symbole*'
'*symbole*'	*x*	➧	'*symbole* OR *x*'
'$symbole_1$'	'$symbole_2$'	➧	'$symbole_1$ OR $symbole_2$'

OR renvoie une valeur logique (ou *état*) qui est le OU logique de deux valeurs logiques :

Premier argument x	Second argument y	Résultat de OR
vrai	vrai	vrai
vrai	faux	vrai
faux	vrai	vrai
faux	faux	faux

Si l'un ou les deux arguments sont des objets algébriques, le résultat est un objet algébrique de la forme '*symbole*$_1$ OR *symbole*$_2$', où *symbole*$_1$ et *symbole*$_2$ représentent les arguments.

XOR — *OU exclusif* — Fonction

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*indic. binaire*
x	'*symbole*'	➧	'*x* XOR *symbole*'
'*symbole*'	*x*	➧	'*symbole* XOR *x*'
'*symbole*$_1$'	'*symbole*$_2$'	➧	'*symbole*$_1$ XOR *symbole*$_2$'

XOR renvoie une valeur logique qui est le OU logique exclusif (XOR) de deux valeurs logiques :

Premier argument x	Second argument y	Résultat de XOR
vrai	vrai	faux
vrai	faux	vrai
faux	vrai	vrai
faux	faux	faux

Si l'un ou les deux arguments sont des objets algébriques, le résultat est un objet algébrique de la forme '*symbole*$_1$ XOR *symbole*$_2$', où *symbole*$_1$ et *symbole*$_2$ représentent les arguments.

NOT — *NON* — Fonction

Niveau 1		Niveau 1
x	➧	*indic. binaire*
'*symbole*'	➧	'NOT *symbole*'

NOT renvoie une valeur logique qui est l'inverse logique d'une valeur logique :

Argument x	Résultat de NOT
vrai	faux
faux	vrai

Si l'argument est un objet algébrique, le résultat est un objet algébrique de la forme 'NOT symbole', où *symbole* représente l'argument.

SAME — *Identique* — Commande

Niveau 2	Niveau 1		Niveau 1
objet$_1$	*objet*$_2$	➧	*indic. binaire*

SAME prend deux objets de même type dans les niveaux 1 et 2 et renvoie une valeur logique « vrai » (1) si les deux objets sont identiques, ou une valeur logique « faux » (0) s'ils ne sont pas identiques.

SAME a le même effet que == pour tous les types d'objets sauf les objets algébriques et les noms (== donne un indicateur binaire symbolique (algébrique) pour ces deux types d'objets).

SAME donne un indicateur binaire (nombre réel) pour tous les types d'objets, et n'est pas autorisée dans les expressions algébriques.

== ***Egal à*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
objet$_1$	*objet*$_2$	➡	*indic. binaire*
z	' *symbole* '	➡	' *z==symbole* '
' *symbole* '	*z*	➡	' *symbole==z* '
' *symbole*$_1$ '	' *symbole*$_2$ '	➡	' *symbole*$_1$*==symbole*$_2$ '

== prend deux objets dans les niveaux 1 et 2, et :

- Si l'un des objets n'est pas un objet algébrique (ou un nom), == renvoie une valeur logique « vrai » (1) lorsque les deux objets sont du même type et ont la même valeur, et une valeur logique « faux » (0) dans tous les autres cas. Les listes et programmes sont considérés comme ayant les mêmes valeurs si les objets qu'ils contiennent sont identiques.
- Si un objet est un objet algébrique (ou un nom), et l'autre un nombre ou un objet algébrique, == donne une expression représentant une comparaison symbolique de la forme ' *symbole*$_1$*==symbole*$_2$ ', où *symbole*$_1$ est l'objet pris au niveau 2 et *symbole*$_2$ l'objet pris au niveau 1. L'expression résultat peut être évaluée avec EVAL ou →NUM et renvoie alors un indicateur binaire (valeur logique « vrai » ou « faux ».

La fonction a été appelée == et non pas = de manière à faire la distinction entre la comparaison logique (==) et l'équation (=).

STOF RCLF TYPE

STOF *Stocker indic. binaires* Commande

Niveau 1	
# *n* ➧	

STOF (STORE FLAGS) fait correspondre l'état (vrai ou faux) des 64 indicateurs binaires d'utilisateur avec les bits d'un entier binaire # *n*. Un bit de valeur 1 arme l'indicateur binaire correspondant ; un bit de valeur 0 le désarme. Le premier bit (bit de poids le plus faible) de # *n* correspond à l'indicateur binaire 1 ; son 64ème bit (bit de poids le plus fort) correspond à l'indicateur binaire 64.

Si # *n* contient moins de 64 bits, les derniers bits inutilisés (jusqu'au bit de poids le plus fort) sont considérés comme ayant la valeur 0.

RCLF *Rappeler indic. binaires* Commande

	Niveau 1
➧ # *n*	

RCLF (RECALL FLAGS) renvoie un entier binaire # n de 64 bits représentant les états des 64 indicateurs binaires d'utilisateur. L'indicateur binaire 1 correspond au premier bit (bit de poids le plus faible) de l'entier binaire ; et l'indicateur binaire 64 est représenté par le 64ème bit (bit de poids le plus fort).

Vous pouvez ainsi sauvegarder les états de tous les indicateurs binaires d'utilisateur en utilisant RCLF, et rétablir ultérieurement ces états en utilisant STOF. N'oubliez pas que la taille de mot en cours doit être 64 bits (taille par défaut) si vous voulez pouvoir sauvegarder et rétablir la valeur de *tous* les indicateurs binaires d'utilisateur. Si la taille de mot en cours est 32, par exemple, RCLF renvoie un entier binaire de 32 bits ; l'exécution de STOF avec un entier binaire de 32 bits rétablit seulement la valeur des indicateurs binaires 1 à 32, et désarme les indicateurs binaires 33 à 64.

Après une réinitialisation de la mémoire, RCLF renvoie la valeur # 4001FFC40000000 (hexadécimal), qui correspond aux états logiques par défaut des 64 indicateurs binaires.

TYPE *Type* **Commande**

Niveau 1	Niveau 1
objet	*n*

La commande TYPE donne un nombre réel représentant le type de l'objet se trouvant au niveau 1. Les différents types d'objets et les nombres correspondants figurent dans le tableau ci-dessous :

Types d'objets et nombres correspondants

Objet	Nombre corresp.
Nombre réel	0
Nombre complexe	1
Chaîne	2
Réel (vecteur ou matrice)	3
Complexe (vecteur ou matrice)	4
Liste	5
Nom	6
Nom local	7
Programme	8
Objet algébrique	9
Entier binaire	10

REAL

NEG	**FACT**	**RAND**	**RDZ**	**MAXR**	**MINR**
ABS	**SIGN**	**MANT**	**XPON**		
IP	**FP**	**FLOOR**	**CEIL**	**RND**	
MAX	**MIN**	**MOD**	**%T**		

Sur le HP-28C/S, un *nombre réel* est un nombre décimal en virgule flottante consistant en une mantisse à 12 chiffres et en un exposant à 3 chiffres compris entre - 499 et + 499. Les nombres réels sont des objets qui sont saisis et affichés sous la forme d'une chaîne de caractères numériques, sans délimiteurs ni espaces intercalés. Les caractères numériques comprennent les chiffres 0 à 9, les signes + et −, un séparateur décimal (« . » ou « , » selon ce que vous avez choisi) et la lettre E pour indiquer le début de l'exposant. Le format général d'un nombre réel est le suivant :

(signe) mantisse E *(signe) exposant*

Lorsque vous saisissez un nombre réel, son format est le suivant :

- Le *signe* de la mantisse peut être un +, un −, ou être omis (c'est alors implicitement un +).
- La *mantisse* peut avoir un nombre quelconque de chiffres, avec un séparateur décimal à un endroit quelconque de la séquence. Si vous saisissez plus de 12 chiffres, la mantisse est arrondie à 12 chiffres (lorsqu'un chiffre est exactement à mi-chemin entre deux autres, il est arrondi au chiffre supérieur). Les zéros de tête sont ignorés dans la mantisse s'ils sont suivis de chiffres non nuls.
- L'exposant est optionnel ; si vous en incluez un, il doit être séparé de la mantisse par un « E ».
- Le *signe* de l'exposant peut être +, −, ou omis (+ implicite).
- L'*exposant* doit contenir trois chiffres ou moins, et être compris entre 0 et 499. Les zéros de tête avant l'exposant sont ignorés.

Les nombres réels sont affichés selon le mode d'affichage en cours pour les nombres réels. Il est possible que l'affichage ne montre pas tous les chiffres significatifs d'un nombre, mais dans tous les cas la précision totale (12 chiffres) d'un nombre est préservée dans la version interne utilisée par le calculateur.

Le menu REAL contient des fonctions opérant sur des arguments qui sont des nombres réels ou des expressions algébriques à valeur réelle, ou introduisant des nombres réels spéciaux dans la pile. En plus des fonctions du menu, vous disposez de % et de %CH sur le clavier.

Fonctions disponibles sur le clavier

% ***Pourcentage*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*xy*/100
x	'*symbole*'	➧	'%(*x*,*symbole*)'
'*symbole*'	*x*	➧	'%(*symbole*,*x*)'
'$symbole_1$'	'$symbole_2$'	➧	'%(*symbole*$_1$,*symbole*$_2$)'

% prend dans la pile deux arguments à valeur réelle, *x* et *y*, et y renvoie *x pourcent de y* — c'est-à-dire *xy*/100.

%CH ***Différence en pourcent*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	100(*y*−*x*)/*x*
x	'*symbole*'	➧	'%CH(*x*,*symbole*)'
'*symbole*'	*x*	➧	'%CH(*symbole*,*x*)'
'$symbole_1$'	'$symbole_2$'	➧	'%CH(*symbole*$_1$,*symbole*$_2$)'

%CH (%CHANGE) calcule la différence en pourcent entre les arguments à valeur réelle *u* et *x*, qui se trouvent respectivement aux niveaux 1 et 2. C'est-à-dire que %CH donne 100(*y* − *x*)/*x*.

...REAL

π π **Fonction**

	Niveau 1
➡	3,14159265359
➡	'π'

π renvoie dans la pile la constante symbolique 'π' ou la valeur numérique 3,14159265359, qui est l'approximation la plus proche de π pouvant être représentée par le calculateur. Pour des renseignements sur les constantes symboliques, voir page 70.

e e **Fonction**

	Niveau 1
➡	2,71828182846
➡	'e'

e donne la constante symbolique 'e' ou la valeur numérique 2,71828182846, qui est la plus proche approximation de *e* (la base des logarithmes népériens) pouvant être représentée par le calculateur. Pour des informations sur les constantes symboliques, voir page 70.

NEG FACT RAND RDZ MAXR MINR

NEG ***Changement de signe*** **Fonct. analyt.**

Niveau 1		Niveau 1
z	➡	−*z*
'*symbole*'	➡	'−*symbole*'

NEG (NEGATE) donne l'opposé de son argument. Lorsqu'aucune ligne de commande n'est présente, une pression sur [CHS] exécute NEG. Vous trouverez dans « Arithmétique » un diagramme de la pile opérationnelle pour NEG.

FACT — *Factorielle (gamma)* — Fonction

Niveau 1		Niveau 1
n	➡	*n*!
x	➡	Γ(*x*+1)
'*symbole*'	➡	'FACT(*symbole*)'

FACT donne la factorielle $n!$ d'un argument entier positif n. Pour les arguments non entiers x, FACT(x) = $\Gamma(x + 1)$, qui est définie pour $x > -1$ comme

$$\Gamma(x + 1) = \int_0^\infty e^{-t} t^x \, dt$$

et qui est définie pour les autres valeurs de x par continuation analytique. Si $x \geq 253{,}1190554375$ ou si x est un entier négatif, FACT génère une exception `Overflow` ; si $x \leq -254{,}1082426465$, FACT génère une exception `Underflow`.

RAND — *Nombre aléatoire* — Commande

	Niveau 1
➡	*x*

RAND (RANDOM) donne le nombre réel suivant dans une séquence de nombres pseudo-aléatoires, et met à jour la racine des nombres aléatoires.

Le HP-28C/S utilise une méthode linéaire et une valeur « racine » pour générer un nombre aléatoire x, qui est toujours compris entre 0 et 1 ($0 \leq x < 1$). Chaque exécution successive de RAND donne une valeur calculée à partir d'une « racine » basée sur la valeur précédente de RAND. Vous pouvez modifier la racine à l'aide de RDZ.

...REAL

RDZ — *Racine des nbs aléatoires* — Commande

Niveau 1	
x ➡	

RDZ (RANDOMIZE) prend un nombre réel comme racine pour la commande RAND. Si l'argument est 0, une valeur aléatoire basée sur l'horloge du système sert de racine. Après une réinitialisation de la mémoire, la valeur racine est 0,529199358633.

MAXR — *Réel maximal* — Fonction

	Niveau 1
➡	9,99999999999E499
➡	'MAXR'

MAXR donne la constante symbolique 'MAXR' ou la valeur numérique 9,99999999999E499, qui est le plus grand nombre pouvant être représenté par le calculateur. Pour des renseignements sur les constantes symboliques, voir page 70.

MINR — *Réel minimal* — Fonction

	Niveau 1
➡	1,00000000000E-499
➡	'MINR'

MINR donne la constante symbolique 'MINR' ou la valeur numérique 1E−499, qui est le plus petit nombre positif pouvant être représenté par le calculateur. Pour des renseignements sur les constantes symboliques, voir page 70.

ABS SIGN MANT XPON

ABS — *Valeur absolue* — Fonction

Niveau 1		Niveau 1
z	➧	$\|z\|$
[tableau]	➧	‖tableau‖
'*symbole*'	➧	'ABS(*symbole*)'

ABS donne la valeur absolue de son argument. Voir « COMPLEX » et « ARRAY » pour l'utilisation de ABS avec d'autres types d'objets (nombres complexes et tableaux). ABS peut être dérivée, mais pas inversée (résolue) par le HP-28C/S.

SIGN — *Signe* — Fonction

Niveau 1		Niveau 1
z_1	➧	z_2
'*symbole*'	➧	'SIGN(*symbole*)'

SIGN donne le signe de son argument, défini comme $+1$ pour les arguments positifs réels, comme -1 pour les arguments réels négatifs, et comme 0 pour l'argument 0. Voir COMPLEX pour les arguments complexes.

...REAL

MANT — *Mantisse* — Fonction

Niveau 1	Niveau 1
x	*y*
'*symbole*'	'MANT(*symbole*)'

MANT donne la mantisse de son argument. Par exemple,

`1,2E34 MANT` donne `1,2`.

XPON — *Exposant* — Fonction

Niveau 1	Niveau 1
x	*n*
'*symbole*'	'XPON(*symbole*)'

XPON donne l'exposant de son argument. Par exemple,

`1,2E34 XPON` donne `34`.

IP FP FLOOR CEIL RND

IP — *Partie entière* — Fonction

Niveau 1	Niveau 1
x	*n*
'*symbole*'	'IP(*symbole*)'

IP (INTEGER PART) donne la partie entière de son argument. Le résultat a le même signe que l'argument.

FP — *Partie fractionnaire* — Fonction

Niveau 1	Niveau 1
x	*y*
'*symbole*'	'FP(*symbole*)'

FP (FRACTIONAL PART) donne la partie fractionnaire de son argument. Le résultat a le même signe que l'argument.

FLOOR — *Plancher* — Fonction

Niveau 1	Niveau 1
x	*n*
'*symbole*'	'FLOOR(*symbole*)'

FLOOR donne le plus grand entier inférieur ou égal à son argument. Si l'argument est un entier, le résultat est identique à l'argument.

CEIL — *Plafond* — Fonction

Niveau 1	Niveau 1
x	*n*
'*symbole*'	'CEIL(*symbole*)'

CEIL (CEILING) donne le plus petit entier supérieur ou égal à son argument. Si l'argument est lui-même un entier, le résultat est identique à l'argument.

...REAL

RND — *Arrondi* — Fonction

Niveau 1		Niveau 1
x	➡	*y*
'*symbole*'	➡	'RND(*symbole*)'

RND (ROUND) arrondit son argument de manière que la représentation interne d'un nombre (avec sa pleine précision) corresponde au nombre affiché, en fonction du mode d'affichage en cours :

- En mode SCI ou ENG, la représentation interne du résultat est identique au nombre affiché.
- En mode FIX, la représentation interne du résultat est identique à sa valeur affichée, sauf si la valeur affichée a dépassé la limite inférieure ou supérieure de la notation SCI. Ainsi, en mode n FIX, le résultat est 0 si l'argument est inférieur à 10^{-n}. Il ne se produit aucun arrondi si l'argument est supérieur ou égal à 10^{11}.
- En mode STD, aucun arrondi n'est effectué.

Les nombres supérieurs ou égaux à 9,5E499 ne sont pas arrondis.

MAX MIN MOD %T

MAX — *Maximum* — Fonction

Niveau 2	Niveau 1		Niveau 1
x	*y*	➡	max(*x*,*y*)
x	'*symbole*'	➡	'MAX(*x*,*symbole*)'
'*symbole*'	*x*	➡	'MAX(*symbole*,*x*)'
'$symbole_1$'	'$symbole_2$'	➡	'MAX($symbole_1$,$symbole_2$)'

MAX donne le plus grand (le plus positif) de ses deux arguments.

MIN — *Minimum* — Fonction

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	min(*x*,*y*)
x	'*symbole*'	➧	'MIN(*x*,*symbole*)'
'*symbole*'	*x*	➧	'MIN(*symbole*,*x*)'
'$symbole_1$'	'$symbole_2$'	➧	'MIN($symbole_1$,$symbole_2$)'

MIN donne le plus petit (le plus négatif) de ses deux arguments.

MOD — *Modulo* — Fonction

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	*x* mod *y*
x	'*symbole*'	➧	'MOD(*x*,*symbole*)'
'*symbole*'	*x*	➧	'MOD(*symbole*,*x*)'
'$symbole_1$'	'$symbole_2$'	➧	'MOD($symbole_1$,$symbole_2$)'

Appliqué à des arguments à valeur réelle x et y, MOD donne un reste défini par

$$x \text{ mod } y = x - y \text{ plancher } (x/y)$$

Mod (x, y) est périodique en x avec une période y. Mod (x, y) se situe sur l'intervalle $[0, y)$ pour $y > 0$ et sur l'intervalle $(y, 0]$ pour $y < 0$.

...REAL

%T ***Pourcentage du total*** **Fonction**

Niveau 2	Niveau 1		Niveau 1
x	*y*	➧	100*y*/*x*
x	'*symbole*'	➧	'%T(*x*,*symbole*)'
'*symbole*'	*x*	➧	'%T(*symbole*,*x*)'
'$symbole_1$'	'$symbole_2$'	➧	'%T($symbole_1$,$symbole_2$)'

%T calcule quelle fraction (en pourcent) de l'argument à valeur réelle *x* (au niveau 2) représente l'argument *y* se trouvant au niveau 1. C'est-à-dire que %T donne $100y/x$.

SOLVE

STEQ	RCEQ	SOLVR	ISOL	QUAD	SHOW
ROOT					

Le menu SOLVE ([SOLV]) contient des commandes qui vous permettent de trouver la solution d'expressions algébriques et d'équations. Par *solution* nous entendons une *racine* mathématique d'une expression, c'est-à-dire une valeur d'une variable contenue dans l'expression et pour laquelle l'expression a la valeur zéro. Pour une équation, cela signifie que les deux termes de l'équation ont la même valeur numérique.

La commande ROOT est une fonction trés élaborée qui permet de trouver une racine numérique pour n'importe quelle expression mathématique raisonnable. Vous pouvez utiliser ROOT comme une commande ordinaire, ou accéder à la fonction de résolution d'équations par l'intermédiaire de SOLVR, qui valide une version interactive de la fonction de résolution d'équations. Cette version interactive fournit un menu pour la saisie des données et le choix d'une variable de résolution, et affiche des résultats identifiés par des libellés, avec des messages pour vous aider à interpréter les résultats.

Il est également possible de résoudre de nombres expressions symboliquement, c'es-à-dire d'obtenir des valeurs symboliques, et non pas numériques, pour les racines d'une expression. La commande ISOL (isoler) trouve une solution symbolique en isolant la première occurrence, dans une expression, d'une variable donnée. QUAD donne la solution symbolique d'une équation du second degré.

Dans de nombreux cas, un résultat symbolique est préférable à un résultat numérique. La forme fonctionnelle du résultat symbolique donne beaucoup plus de reseignements sur le comportement du système constitué par une expression mathématique que ne peut le faire un simple nombre. En outre, une solution symbolique peut contenir *toutes* les racines multiples d'une expression. Même si vous n'êtes intéressé que par des résultats numériques, résoudre une expression symboliquement avant d'utiliser SOLVR peut économiser beaucoup de temps lors du calcul des racines numériques.

...SOLVE

Résolution interactive numérique : SOLVR

Cette opération interactive automatise le processus de stockage des valeurs dans les variables de l'équation et de calcul de l'une des variables. La procédure générale d'utilisation est la suivante :

1. Utilisez STEQ (« stocker équation ») pour choisir l'*équation en cours*.
2. Appuyez sur SOLVR pour faire apparaître le *menu des variables*.
3. Utilisez les touches de ce menu pour affecter des valeurs aux variables de l'équation, avec votre « meilleure » estimation pour la variable inconnue.
4. Calculez l'inconnue en appuyant sur la touche préfixe rouge (■) puis sur la touche de menu correspondant à l'inconnue.

Chacune de ces étapes est décrite en détail dans les paragraphes suivants.

L'équation en cours

L'équation en cours est définie comme la procédure qui est stockée dans la variable utilisateur EQ. Le terme *équation en cours* (et le nom EQ) a été choisi pour refléter l'utilisation typique de la procédure ; mais cette procédure peut être une équation, une expression algébrique, ou un programme. Lorsque vous utilisez SOLVR avec un programme, ce dernier doit être équivalent à une expression algébrique ; c'est-à-dire qu'il ne doit pas prendre d'arguments dans la pile doit y renvoyer un résultat.

Vous pouvez vous représenter l'équation en cours comme un argument « implicite » de SOLVR (c'est aussi l'argument de DRAW). Cet argument implicite vous évite d'avoir à placer une procédure dans la pile chaque fois que vous utilisez SOLVR ou DRAW.

En matière de résolution des équations et expressions, vous pouvez considérer une expression comme le terme gauche d'une équation dont le terme droit est 0. De même, vous pouvez interpréter une équation comme une expression en traitant le signe = comme équivalant à − (soustraction).

Nous décrivons ci-après STEQ et RCEQ, qui sont des commandes pour le stockage et le rappel du contenu de EQ.

STEQ — *Stocker équation* — Commande

Niveau 1	
objet ➧	

STEQ prend un objet dans la pile et le stocke dans la variable EQ (EQuation). EQ sert à stocker l'*équation en cours*, utilisée par SOLVR et les applications de traçage ; l'argument de STEQ doit donc normalement être une procédure.

RCEQ — *Rappeler équation* — Commande

	Niveau 1
	➧ *objet*

RCEQ (RECALL EQUATION) donne le contenu de la variable EQ. C'est l'équivalent de `'EQ' RCL`.

...SOLVE

Activer le menu des variables

Une pression sur SOLVR active le *menu des variables* de SOLVR, qui est dérivé de l'équation en cours. Ce menu contient :

- Un libellé pour chaque variable indépendante de l'équation en cours. S'il y a plus de six variables indépendantes, vous pouvez utiliser les touches NEXT et ■ PREV pour faire apparaître les six libellés suivants.
- Un ou deux libellés pour l'évaluation de l'équation en cours. Si EQ contient une expression algébrique ou un programme, le libellé EXPR= apparaît, pour en permettre l'évaluation. Si EQ contient une équation algébrique, les libellés LEFT= (terme gauche =) et RT= (terme droit =) vous permettent d'évaluer séparément les termes droit et gauche de l'équation.

Comment le menu des variables est configuré. **Une variable** *indépendante* nommée dans l'équation en cours est soit une variable formelle, soit une variable contenant des données, généralement un nombre réel. Une variable contenant une procédure n'apparaît pas dans le menu des variables. Au contraire, les noms apparaissant dans cette procédure sont considérés comme pouvant être des variables indépendantes ; celles qui contiennent des données sont ajoutées qux variables du menu. Le processus se poursuit jusqu'à ce que toutes les variables indépendantes soit identifiées dans le menu. Le menu est continuellement mis à jour, de sorte que si vous stockez une procédure dans l'une des variables du menu, cette variable sera remplacée, dans le menu, par les nouvelles variables indépendantes contenues dans la procédure.

Par exemple, si l'équation en cours est `'A+B=C'`, le menu des variables :

est affiché si A, B, et C ne contiennent pas de procédures. Mais si nous stockons `'D+E'` dans C, le menu deviendra

(Si une variable de l'équation en cours contient elle-même une équation, cette dernière est traitée comme une expression dans laquelle le = est remplacé par un −, pour que la variable puisse être définie.)

Stocker des valeurs dans les variables indépendantes

Appuyer sur une touche nom du menu des variables équivaut à exécuter la séquence `'nom' STO`. C'est-à-dire que nom prend un objet dans la pile et le stocke en tant que valeur de la variable *nom*.

Pour confirmer cette opération, le calculateur affiche aussi le message *nom*: *objet* sur la ligne 1 de l'affichage, *objet* étant ici l'objet pris dans la pile. Ce message disparaît dès qu'une autre touche est actionnée.

Vous pouvez vérifier le contenu d'une variable à tout moment en appuyant sur ' nom puis sur ■RCL, ■VISIT, ou EVAL.

Choix d'une estimation initiale

En général, les procédures et expressions algébriques admettent plus d'une racine. L'expression $(x - 3)(x - 2)$, par exemple, admet des racines pour $x = 3$ et pour $x = 2$. La racine fournie par SOLVR dépend du point de départ de la recherche, appelé *estimation initiale*.

Vous devriez toujours fournir une estimation initiale à SOLVR. L'estimation est l'un des arguments requis par la commande ROOT. Pour SOLVR, c'est la variable en cours de la variable inconnue qui est prise comme estimation initiale. Si la variable inconnue n'a pas de valeur, SOLVR lui affectera 0 comme estimation initiale lorsque vous essaierez de la calculer, mais rien ne garantit que cette estimation initiale donnera la racine que vous recherchez.

...SOLVE

Vous pouvez accélérer le calcul de la racine, ou le guider vers une racine particulière, en donnant une estimation initiale appropriée. Cette estimation peut être l'un des objets suivants :

- Un nombre, ou une liste contenant un nombre. Ce nombre est converti en deux estimations initiales, comme décrit plus loin, par duplication et légère altération de l'une des copies.
- Une liste contenant deux nombres. Les deux nombres déterminent un intervalle dans lequel la recherche commence. Si ces deux nombres définissent un nombre impair de racine (la procédure a alors des valeurs de signes opposés pour ces deux nombres), SOLVR peut généralement trouver une racine dans l'intervalle assez rapidement. Si les valeurs de la procédure est de même signe pour les deux nombres, SOLVR doit rechercher une zone comprenant une racine. Le choix de nombres aussi proches que possible de la racine accélère cette recherche.
- Une liste contenant trois nombres. Dans ce cas, le premier nombre doit représenter votre meilleure estimation de la racine recherchée. Les deux autres doivent « encadrer » votre meilleure estimation et définir un intervalle dans lequel la recherche doit commencer. La liste de trois nombres que vous renvoie le calculateur lorsque vous interrompez le calcul de la racine par pression sur la touche [ON] correspond à l'estimation en cours, dans ce même format.

N'importe lequel des nombres décrit ci-dessus peut être un complexe ; seules les parties réelles sont alors utilisées.

La meilleure façon de choisir une estimation initiale consiste à tracer l'équation en cours. Le tracé vous donne une idée du comportement global de l'équation et vous fait voir où se situent les racines. Avec une équation, les racines sont les valeurs de la variable indépendante (horizontale) pour laquelle les deux courbes représentant l'équation se coupent ; avec une expression (ou un programme), les racines sont les points auxquels la courbe coupe l'axe horizontal (coordonnée verticale = 0). Si vous utilisez le traçage interactif (DRAW), vous pouvez déplacer le curseur sur la racine recherchée et numériser un ou plusieurs points. Vous pouvez ensuite utiliser les coordonnées du (des) point(s) comme estimation(s) initiale(s) pour SOLVR.

Calcul d'une variable inconnue

Pour *résoudre* l'équation en cours, c'est-à-dire calculer une variable « inconnue » *nom*, appuyez sur la touche préfixe ■ puis sur la touche de menu nom. Ceci active la fonction de résolution numérique, en vue de déterminer la valeur de la variable inconnue qui est une racine de l'équation en cours (c.-à-d. qui donne la valeur 0 à l'équation en cours). Pendant l'exécution du calcul, le message

`Solving for` *nom* (calcul de *nom* en cours)

apparaît sur la première ligne de l'affichage. Lorsque l'exécution est terminée, le résultat est envoyé dans la pile et la première ligne affiche

nom: `result`

(jusqu'à ce que vous appuyez sur une touche). La deuxième ligne affiche un message qui qualifie le résultat.

Pendant le calcul de la racine, vous pouvez :

- Appuyer sur ON pour arrêter l'itération du calcul de la racine et revenir à l'affichage normal de la pile. Lorsque vous interrompez le calcul de cette façon, la meilleure estimation en cours de la racine est affichée, ainsi qu'une liste contenant la meilleure estimation en cours plus deux nombres réels définissant l'intervalle de recherche de la racine. Si vous voulez relancer le calcul de la racine, il vous suffit d'appuyer sur la touche de menu correspondant à la variable recherchée (pour stocker la liste dans la variable), puis sur la touche préfixe rouge et sur la touche de la variable à nouveau. En utilisant la liste comme une estimation, vous reprenez le calcul au point où vous l'aviez interrompu.
- Appuyer sur n'importe quelle autre touche pour afficher les résultats intermédiaires du calcul de la racine. Les lignes 2 et 3 de l'affichage montreront les deux estimations en cours utilisées par la fonction, plus le signe des valeurs de l'équation en cours pour chacune des estimations. Si l'équation en cours n'est pas définie pour une estimation, un point d'interrogation (?) apparaît à la place du signe.

...SOLVE

Les résultats intermédiaires qui sont affichés sont deux valeurs de la variable inconnue qui caractérisent l'intervalle la recherche de la racine s'effectue. Bien que leur signification précise varie en fonction de la nature de l'équation en cours, vous pouvez les considérer comme une indication du progrès de la recherche de la racine. Généralement, l'algorithme recherche une racine dans l'intervalle jusqu'à ce qu'il rencontre un changement de signe (lorsque deux estimations ont des signes opposés). Vous voyez alors les deux estimations venir « encadrer » la racine et converger vers cette valeur de deux côtés opposés. Si vous observez au contraire les deux estimations diverger, cela signifie que l'algorithme n'a pas encore déterminé un intervalle contenant une racine possible. Si la divergence se poursuit, cela indique très probablement qu'il n'existe pas de racine finie.

Interpréter les résultats

L'algorithme du HP-28C/S recherche une racine réelle pour une procédure, en commençant par la première estimation que vous lui avez fournie. Dans la plupart des cas, l'algorithme fournit un résultat. La commande ROOT envoie simplement le résultat dans la pile. SOLVR envoie le résultat dans la pile, identifie ce résultat par un libellé en ligne 1, et donne en ligne 2 un message *qualifiant* le résultat. Ce message est une indication sur la nature de la racine obtennue :

Message	Signification
Zero	Une racine « exacte » de la procédure a été trouvée : évaluer la procédure avec cette racine donne la valeur 0.
Sign Reversal (changement de signe)	Une approximation d'une racine a été trouvée, avec une précision de 12 chiffres. L'algorithme a trouvé des points voisins pour lesquels la valeur de la procédure change de signe, mais il n'a pas trouvé de point pour lequel la procédure est égale à 0. Cette racine peut être ou ne pas être une racine normale, ou elle peut être une discontinuité dans la valeur de la procédure, à l'occasion de laquelle la valeur change de signe.
Extremum	L'algorithme a trouvé une approximation d'un minimum ou maximum local de la valeur numérique absolue de la procédure. Si la « racine » est ±9,99999999999E499, elle correspond à un extrême asymptotique.

Une fois que vous avez obtenu un résultat avec SOLVR ou ROOT, vous devez évaluer la procédure pour laquelle le résultat a été obtenu, afin d'interpréter les résultats (si vous avez utilisé le menu des variables, vous pouvez avoir recours à EXPR= dans le cas d'une expression ou d'un programme, ou à LEFT= et RT= dans le cas d'une équation). Il y a deux possibilités : soit la valeur de la procédure est proche de zéro pour la valeur de la variable inconnue donnée par l'algorithme de SOLVR et de ROOT, soit elle n'est pas proche de zéro. C'est à vous de décider à partir de quand la valeur de la procédure est assez proche de zéro pour que la valeur fournie par l'algorithme soit considérée comme une racine.

La meilleure façon de comprendre la nature de la racine consiste à tracer la procédure à proximité de la racine. Le tracé vous montrera si la racine est convenable ou s'il y a une discontinuité, beaucoup plus clairement que n'importe quel message affiché par le calculateur.

Pendant sa recherche d'une racine, il se peut que l'algorithme évalue la procédure pour des valeurs de l'inconnue qui génèrent des exceptions mathématiques. Il ne s'agit pas d'erreurs, mais les indicateurs binaires d'utilisateur correspondant à ces exceptions mathématiques seront alors armés.

Erreurs

Dans deux cas, l'algorithme échouera, identifiant le problème à l'aide d'un message d'erreur :

Mesage d'erreur	Signification
Bad Guess(es)	Mauvaise(s) estimation(s). L'une des estimations initiales, ou les deux, se trouve en dehors du domaine de résolution de la procédure. C'est-à-dire que la procédure génère une erreur lorsqu'elle est évaluée pour la (les) valeur(s) estimée(s).
Constant?	Constante ? La procédure donne la même valeur pour tous les points échantillonnés par l'algorithme.

...SOLVE

ROOT — *Trouver les racines* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
«programme»	' nom '	estimation	➧	racine
«programme»	' nom '	{ estimations }	➧	racine
' symbole '	' nom '	estimation	➧	racine
' symbole '	' nom '	{ estimations }	➧	racine

ROOT prend une procédure, un nom, et soit une simple estimation (un nombre réel ou complexe) soit une liste de une, deux ou trois estimations, et donne un nombre réel *racine*. *Racine* est une valeur de la variable *nom* qui est fournie par l'algorithme de résolution numérique du HP-28C/S. Lorsque le comportement mathématique de la procédure est approprié, *racine* est une racine mathématique — une valeur de la variable pour laquelle la procédure a une valeur numérique nulle. Consultez « Interpréter les résultats », en page 282, pour plus de renseignements sur la façon d'interpréter les résultats fournis par l'algorithme.

L'estimation, ou la liste des estimations, sont des estimations de la valeur de la racine, que vous devez fournir pour indiquer à l'algorithme dans quel intervalle il doit commencer sa recherche d'une racine. « Choisir des estimations initiales », en page 279, explique comment procéder.

Si vous interrompez ROOT en appuyant sur [ON], la procédure est affichée au niveau 3, le nom au niveau 2 et une liste contenant trois valeurs intermédiaires de la variable inconnue au niveau 1. La meilleure estimation en cours est stockée dans la variable inconnue. Cette liste peut être utilisée comme estimation initiale si vous voulez relancer le calcul de la racine.

Solutions symboliques

ISOL *Isoler* Commande

Niveau 2	Niveau 1		Niveau 1
' *symbole*$_1$ '	' *nom* '	➧	' *symbole*$_2$ '

ISOL donne une expression *symbole*$_2$ qui représente une réorganisation de son argument algébrique *symbole*$_1$ dans le but « d'isoler » la première occurrence de la variable *nom*. Si la variable n'apparaît qu'une fois, dans la définition de *symbole*$_1$, *symbole*$_2$ est alors une racine symbolique de *symbole*$_1$. Si *nom* apparaît plus d'une fois, *symbole*$_2$ est alors le membre droit d'une équation obtenue par réorganisation et résolution de *symbole*$_1$ en vue d'isoler la première occurence de *nom* dans le membre gauche de l'équation (si *symbole*$_1$ est une expression, considérez-la comme le membre gauche d'une équation *symbole*$_1$ = 0).

Si *nom* apparaît dans l'argument d'une fonction à l'intérieur de *symbole*$_1$, cette fonction doit être une *fonction analytique* — le HP-28C/S doit pouvoir calculer son inverse. Par exemple, ISOL ne peut pas résoudre IP(X) = 0 en X, car IP n'a pas d'inverse. Les commandes pour lesquelles le HP-28C/S peut calculer un inverse algébrique sont identifiées dans ce manuel comme des *fonctions analytiques*.

...SOLVE

QUAD — *Forme quadratique* — Commande

Niveau 2	Niveau 1		Niveau 1
'*symbole*$_1$'	'*nom*'	➡	'*symbole*$_2$'

QUAD résoud une expression algébrique *symbole*$_1$ pour une variable nom, et donne une expression *symbole*$_2$ représentant la solution. QUAD calcule l'approximation de *symbole*$_1$ par une série de Taylor du second degré, afin de convertir l'expression en une forme quadratique (ce sera exact si *symbole*$_1$ est déjà un polynôme du second degré dans *nom*).

QUAD évalue *symbole*$_2$ avant de le renvoyer dans la pile. Si vous voulez une solution symbolique, vous devez supprimer les variables que vous voulez conserver dans la solution sous la forme de variables formelles.

SHOW — *Montrer variable* — Commande

Niveau 2	Niveau 1		Niveau 1
'*symbole*$_1$'	'*nom*'	➡	'*symbole*$_2$'

SHOW donne *symbole*$_2$, qui est équivalent à *symbole*$_1$, à cette différence près que toutes les références implicites à une variable *nom* sont rendues explicites. Par exemple, si nous définissons

```
'X+1' 'A' STO 'Y+2' 'B' STO,
```

alors

```
'A*B' 'Y' SHOW donne 'A*(Y+2)'
```

et

```
'A*B' 'X' SHOW donne '(X+1)*B'.
```

Solutions générales

Les fonctions du HP-28C/S sont des fonctions au sens mathématique strict, c'est-à-dire qu'elles donnent toujours un, et un seul, résultat, lorsqu'elles sont évaluées. Cela signifie par exemple que $\sqrt{4}$ donne toujours $+2$, et non pas -2 ou ± 2. Pour d'autres fonctions, comme ASIN, c'est une valeur principale qui est fournie, conformément aux conventions mathématiques courantes.

Cela implique toutefois que les couples de fonctions comme $\sqrt{}$ et SQ, ou SIN et ASIN, ne représentent pas nécessairement la *relation* inverse générale sous-entendue par leur nom. Considérons l'équation $x^2 = 2$. Si l'on extrait la racine carrée des deux membres, on obtient les « solutions »

$$x = +\sqrt{2} \text{ et } x = -\sqrt{2}.$$

Sur le HP-28C/S, l'équation '𝕏=√2' ne peut pas représenter correctement les deux solutions — la fonction $\sqrt{}$ donne toujours la racine carrée positive. De même, si l'on résout sin $x = 0{,}5$ en x, il y a un nombre infini de solutions $x = 30° + 360n°$, où n est un entier quelconque. Mais si vous appliquez la fonction ASIN à 0,5 , vous n'obtiendrez que le résultat 30°.

L'indicateur binaire de valeur principale (indic. 34) détermine la nature des solutions obtenues à l'aide de ISOL et QUAD. Si cet indicateur binaire est armé, tous les signes et entiers arbitraires sont choisis automatiquement comme étant ceux des valeurs principales. Si l'indicateur est désarmé, les solutions obtenues sont les solutions générales.

...SOLVE

Le mode « Solutions générales »

Lorsque le HP-28C/S est en mode « Solutions générales », c'est-à-dire lorsque l'indicateur binaire 34 est désarmé, les commandes QUAD et ISOL résolvent les expressions dans toute leur généralité en introduisant, là où cela est nécessaire, des variables utilisateur spéciales représentant les signes et entiers arbitraires. Vous pouvez affecter des valeurs à ces variables de la manière habituelle (en stockant ces valeurs dans les variables) puis évaluer l'expression. QUAD et ISOL introduisent des variables de la manière suivante :

- Lorsqu'une commande donne un résultat contenant un ou plusieurs *signes arbitraires*, le premier de ces signes est représenté par une variable ≤1, le second par ≤2, et ainsi de suite. Par exemple :

  `'X^2+5*X+4' 'X' QUAD` donne `'(-5+≤1*3)/2'`.

  Le ≤1 représente le symbole conventionnel ±. Vous pouvez choisir l'une ou l'autre racine en stockant +1 ou −1 dans ≤1, puis en exécutant EVAL.

- Si ISOL donne un résultat contenant un ou plusieurs *entiers arbitraires*, le premier est représenté par une variable n1, le second par n2, et ainsi de suite. Par exemple :

  `'X^4=Y' 'X' ISOL` donne `'EXP(2*π*i*n1/4)*Y^.25'`.

  L'exponentielle représente le signe arbitraire complexe du résultat ; il y a trois valeurs uniques, correspondant à n1 = 0, 1, et 2. Vous pouvez choisir l'une de ces valeurs en stockant le nombre approprié dans n1, puis en évaluant l'expression.

Une autre méthode pour remplacer les variables arbitraires dans une expression résultant de l'exécution de ISOL ou QUAD consiste à utiliser le clavier pour modifier (EDIT) l'expression et transformer les variables arbitraires en variables temporaires dont vous fournissez la valeur. Par exemple, pour choisir la racine négative dans l'exemple ci-dessus utilisant QUAD, appuyez sur ■ EDIT pour copier l'expression résultat dans la ligne de commande, puis appuyez sur

INS −1 → ≤1 ENTER.

Ceci fait de ≤1 une variable locale, lui affecte la valeur −1, puis évalue l'expression. Cette méthode a l'avantage d'éviter la création, en mémoire utilisateur, de variables « permanentes » qui correspondent aux variables arbitraires.

Le mode « Valeur principale »

Si vous armez l'indicateur binaire 34, QUAD et ISOL donnent les valeurs « principales » des solutions. C'est-à-dire que :

- Les signes arbitraires sont choisis comme positifs. Ceci s'applique à la fois au $\pm$ ordinaire et au « signe » complexe plus général $\exp(2n\pi i/x)$ qui résulte de l'inversion d'expressions de la forme y^x. Dans ce dernier cas, l'entier arbitraire n est choisi comme étant 0.
- Les entiers arbitraires sont choisis comme étant égaux à 0. Ansi

  `'SIN(X)=Y' 'X' ISOL` donne `'ASIN(Y)'`,

  qui est toujours compris entre 0 et 180 degrés.

Vous devez comprendre que ces choix de valeurs « principales » ont essentiellement pour but de simplifier les expressions résultats. Mathématiquement, elles ne sont ni meilleures ni pires que n'importe quelles autres racines d'une expression. Si vous voulez des résultats symboliques qui peuvent par la suite être évalués à d'autres fins que la simple inspection visuelle, vous devez travailler avec l'indicateur binaire 34 désarmé, de manière à avoir des résultats absolument généraux.

STACK

DUP	OVER	DUP2	DROP2	ROT	LIST→
ROLLD	PICK	DUPN	DROPN	DEPTH	→LIST

Ce menu présente les commandes qui manipulent le contenu de la pile opérationnelle. Les commandes les plus utilisées sont disponibles sur le clavier ; les autres sont disponibles en tant que touches de menu dans le menu STACK.

Les commandes disponibles sur le clavier sont [DROP], ■[SWAP], ■[ROLL] et ■[CLEAR].

Commandes du clavier

DROP *Supprimer* Commande

Niveau 1	
objet	➧

DROP retire le premier objet de la pile. Les autres éléments de la pile descendent d'un niveau.

L'objet qui a été retiré de la pile peut être récupéré en exécutant LAST, si cette commande est active.

SWAP *Permuter* Commande

Niveau 2	Niveau 1		Niveau 2	Niveau 1
$objet_1$	$objet_2$	➧	$objet_2$	$objet_1$

SWAP permute les deux premiers objets de la pile.

ROLL — *Déplacer* — Commande

Niveau n+1 ... Niveau 2	Niveau 1		Niveau n ... Niveau 2	Niveau 1
$objet_1$ ... $objet_n$	n	➡	$objet_2$ ... $objet_n$	$objet_1$

ROLL prend un nombre entier n dans la pile opérationnelle, puis « déplace » les n premiers objets restant sur la pile de sorte que l'obj_2 (qui se trouvait à l'origine au niveau n, avant que l'argument n ne soit placé sur la pile) est descendu au niveau 1, et les objets de obj_2 à obj_n sont déplacés d'un niveau vers le haut.

Vous pouvez utiliser la séquence de commandes LAST ROLLD pour inverser l'effet de la commande ROLL.

CLEAR — *Effacer* — Commande

Niveau n ... Niveau 1		
$objet_1$ ... $objet_n$	➡	

CLEAR retire tous les objets de la pile.

Si UNDO est active, il est possible de retrouver le contenu de la pile en appuyant sur ■ UNDO tout de suite après avoir appuyé sur CLEAR.

DUP OVER DUP2 DROP2 ROT LIST→

DUP — *Dupliquer* — Commande

	Niveau 1		Niveau 2	Niveau 1
	objet	➡	*objet*	*objet*

DUP renvoie une copie de l'objet placé au niveau 1. Le fait d'appuyer sur ENTER lorsqu'aucune ligne de commande n'est affichée exécute la commande DUP.

...STACK

OVER — *Copier niveau 2* — Commande

	Niveau 2	Niveau 1		Niveau 3	Niveau 2	Niveau 1
	$objet_1$	$objet_2$	➡	$objet_1$	$objet_2$	$objet_1$

OVER renvoie une copie de l'objet placé au niveau 2 ;

DUP2 — *Dupliquer deux objets* — Commande

	Niveau 2	Niveau 1		Niveau 4	Niveau 3	Niveau 2	Niveau 1
	$objet_1$	$objet_2$	➡	$objet_1$	$objet_2$	$objet_1$	$objet_2$

DUP2 renvoie des copies des deux premiers objets de la pile.

DROP2 — *Supprimer deux objets* — Commande

	Niveau 2	Niveau 1	
	$objet_1$	$objet_2$	➡

DROP2 retire les deux premiers objets de la pile. Ils sont sauvegardés dans les arguments de LAST. Ils peuvent être récupérés par LAST si cette commande est activée.

ROT — *Permuter circulairement* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 3	Niveau 2	Niveau 1
$objet_1$	$objet_2$	$objet_3$	➡	$objet_2$	$objet_3$	$objet_1$

ROT effectue une permutation circulaire des trois premiers objets de la pile, amenant le troisième objet en première position. ROT est équivalent à `3 ROLL`.

LIST→ *De la liste vers la pile* Commande

Niveau 1		Niveau n+1 ... Niveau 2	Niveau 1
{ $objet_1$ ... $objet_n$ }	➡	$objet_1$ ... $objet_n$	n

LIST→ prend une liste de n objets dans la pile et renvoie les objets de la liste séparément dans les niveaux 2 à n+1. Le nombre n est renvoyé au niveau 1.

ROLLD PICK DUPN DROPN DEPTH →LIST

ROLLD *Défiler vers le bas* Commande

Niveau n+1 ... Niveau 2	Niveau 1		Niveau n	Niveau n−1 ... Niveau 1
$objet_1$ ... $objet_n$	n	➡	$objet_n$	$objet_1$ ... $objet_{n-1}$

ROLLD prend un nombre entier n dans la pile et fait défiler les n premiers objets restants de la pile ; $objet_1$, qui se trouvait à l'origine au niveau n avant que l'argument n soit placé dans la pile, est mis au niveau 1 et les objets $objet_2$ à $objet_n$, qui se trouvaient à l'origine aux niveaux n - 1 à 2, remontent d'un niveau vers le haut.

Vous pouvez utiliser la séquence de commande `LAST ROLL` pour revenir en arrière et inverser l'effet de ROLLD.

PICK *Choisir* Commande

Niveau n+1 ... Niveau 2	Niveau 1		Niveau n+1 ... Niveau 2	Niveau 1
$objet_1$ ... $objet_n$	n	➡	$objet_1$ ... $objet_n$	$objet_1$

PICK prend un nombre entier n dans la pile et copie $objet_1$ (qui se trouvait au niveau 1, avant que l'argument n soit placé dans la pile) au niveau 1. Les objets de $objet_1$ à $objet_n$ (qui se trouvaient aux niveaux n à 1) remontent d'un niveau chacun.

...STACK

DUPN — *Dupliquer n objets* — Commande

Niveau $n+1$ . . . Niveau 2	Niveau 1	Niveau $2n$ . . . Niveau $n+1$	Niveau n . . . Niveau 1
$objet_n$ . . . $objet_1$	n ➡	$objet_n$ . . . $objet_1$	$objet_n$ . . . $objet_1$

DUPN prend un nombre entier n dans la pile et renvoie des copies des n premiers objets de la pile (ceux qui se trouvent aux niveaux compris entre 2 et $n + 1$ lorsque n est dans la pile).

DROPN — *Supprimer n objets* — Commande

Niveau $n+1$ . . . Niveau 2	Niveau 1	
$objet_1$ . . . $objet_n$	n ➡	

DROPN retire les $n + 1$ premiers objets de la pile (les n premiers objets, à l'exception du nombre n lui-même). Le nombre n est sauvegardé dans les arguments de LAST et peut être récupéré par cette commande. La commande ■ UNDO peut être utilisée pour récupérer les objets supprimés.

DEPTH — *Nombre d'objets dans la pile* — Commande

	Niveau 1
➡	n

DEPTH renvoie un nombre réel n représentant le nombre d'objets présents dans la pile (avant l'exécution de DEPTH).

→LIST *De la pile vers la liste* Commande

Niveau $n+1$ ... Niveau 2	Niveau 1		Niveau 1
$objet_1$ ... $objet_n$	n	➧	{ $objet_1$ ... $objet_n$ }

→LIST prend un nombre entier n au niveau 1 de la pile, plus n objets supplémentaires des niveaux 2 à $n + 1$ et renvoie une liste contenant les n objets.

Si vous exécutez ensemble `DEPTH →LIST` vous combinerez le contenu entier de la pile en une liste que vous pouvez par exemple stocker dans une variable pour utilisation ultérieure.

STAT

Σ+	Σ−	NΣ	CLΣ	STOΣ	RCLΣ
TOT	MEAN	SDEV	VAR	MAXΣ	MINΣ
COLΣ	CORR	COV	LR	PRDEV	
UTPC	UTPF	UTPN	UTPT		

Les commandes de statistiques du HP-28C/S fonctionnent sur base de données statistiques rassemblées en une matrice $n \times m$ nommée la *matrice de statistiques en cours.* Elle est définie comme étant une matrice stockée dans la variable ΣDAT.

Cette matrice statistique en cours ΣDAT est créée automatiquement, si elle n'existe pas encore, dès que des *observations* sont saisies par la commande Σ+. Une observation est un vecteur ayant *m valeurs coordonnées* (nombres réels) et constitue une ligne de la matrice de statistiques. La dimension *n* (le nombre de lignes) est le nombre d'observations qui ont été saisies, comme illustré ci-dessous :

Observation	Nombre coordonné			
	1	**2**	...	***m***
1	X_{11}	X_{12}	...	X_{1m}
2	X_{21}	X_{22}	...	X_{2m}
⋮	⋮	⋮		⋮
n	X_{n1}	X_{n2}	...	X_{nm}

Certaines commandes de statistiques combinent des données provenant de deux colonnes spécifiques de la matrice de statistiques. La variable utilisateur ΣPAR contient une liste de quatre nombres réels, dont les deux premiers identifient ces deux colonnes. Elles sont sélectionnées par la commande COLΣ. Les deux derniers nombres de la liste sont la pente et l'ordonnée à l'origine calculées pour l'exécution la plus récente de la commande de régression linéaire LR (LINEAR REGRESSION).

Parce que ΣDAT et ΣPAR sont des variables ordinaires, vous pouvez utiliser des commandes ordinaires pour rappeler, visualiser ou modifier leur contenu, en plus des commandes spécifiques de statistiques qui agissent sur les variables.

Les commandes SDEV (STANDARD DEVIATION, écart-type), VAR (VARIANCE) et COV (COVARIANCE) calculent des statistiques effectuées sur des observations représentant un *échantillon* de la population. Ces commandes sont décrites en détail ci-dessous. Si ces données représentent la population *entière*, vous pouvez calculer les *statistiques de la population* de la manière suivante :

1. Exécutez MEAN pour obtenir une observation représentant la moyenne des observations.
2. Exécutez Σ+ pour ajouter cette moyenne à l'ensemble des autres données.
3. Exécutez SDEV, VAR ou COV. Le résultat constitue les statistiques de la population.
4. Exécutez Σ− DROP pour extraire la moyenne du reste des données.

Σ+ Σ− NΣ CLΣ STOΣ RCLΣ

Ces commandes vous permettent de sélectionner une matrice de statistiques et d'ajouter ou de retrancher des données de la matrice.

Σ+ — *Sigma plus* — Commande

Niveau 1	
x	➡
$[x_1\ x_2\ \ldots\ x_m]$	➡
$[[x_{11}\ x_{12}\ \ldots\ x_{1m}]$ ⋮ $[x_{n1}\ x_{n2}\ \ldots\ x_{nm}]]$	➡

Σ+ ajoute une ou plusieurs observations à la matrice de statistiques en cours ΣDAT.

Dans le cas d'une matrice de statistiques de m colonnes, il vous est possible d'introduire l'argument de $\Sigma+$ de plusieurs façons :

Saisie d'une seule observation avec une seule valeur coordonnée. L'argument de $\Sigma+$ est un nombre réel.

Saisie d'une observation avec plusieurs valeurs coordonnées. L'argument de $\Sigma+$ est un vecteur avec m valeurs coordonnées réelles.

Saisie de plusieurs observations. L'argument de $\Sigma+$ est une matrice de n lignes de m valeurs coordonnées réelles.

Dans tous ces cas, les valeurs coordonnées sont ajoutées sous forme de nouvelles lignes à la matrice de statistiques stockée dans ΣDAT. Si ΣDAT n'existe pas, $\Sigma+$ la créée sous la forme d'une matrice de $n \times m$, stockée dans la variable ΣDAT. Si ΣDAT existe, une erreur se produit si elle ne contient pas une matrice réelle ou si le nombre de valeurs coordonnées dans chaque observation saisie avec $\Sigma+$ ne correspond pas au nombre de colonnes de ΣDAT.

$\Sigma-$ *Sigma moins* **Commande**

	Niveau 1
➡	x
➡	$[x_1\ x_2\ \ldots\ x_m]$

$\Sigma-$ renvoie un vecteur de m nombres réels ou un seul nombre si $m = 1$, correspondant aux valeurs coordonnées contenues dans la dernière observation saisie par $\Sigma+$ dans la matrice de statistiques ΣDAT. La dernière ligne de la matrice statistiques est supprimée.

Le vecteur renvoyé par $\Sigma-$ peut être corrigé, modifié ou remplacé, puis ré-introduit dans la matrice de statistiques par $\Sigma+$.

NΣ ***Sigma N*** **Commande**

	Niveau 1
	➡ *n*

NΣ renvoie le nombre d'observations saisies dans la matrice de statistiques stockée dans ΣDAT. Le nombre d'observations est égal au nombre de lignes de la matrice.

CLΣ ***Effacer sigma*** **Commande**

	➡

CLΣ efface la matrice de statistiques en effaçant ΣDAT.

STOΣ ***Stocker sigma*** **Commande**

Niveau 1	
[*matrice*] ➡	

STOΣ prend une matrice dans la pile et la stocke dans la variable ΣDAT.

RCLΣ ***Rappeler sigma*** **Commande**

	Niveau 1
	➡ *objet*

RCLΣ renvoie le contenu de la variable ΣDAT. RCLΣ est équivalent à `'ΣDAT' RCL`.

...STAT

TOT MEAN SDEV VAR MAXΣ MINΣ

Ces commandes permettent des calculs statistiques élémentaires pour les données de chaque colonne de la matrice de statistiques en cours.

TOT *Total* Commande

	Niveau 1
➡	x
➡	$\lceil x_1\ x_2\ \ldots\ x_m \rceil$

TOT calcule la somme de chacune des m colonnes de valeurs coordonnées présentes dans la matrice de statistiques ΣDAT. Les sommes sont renvoyées sous la forme d'un vecteur de m nombres réels ou, si $m = 1$, sous celle d'un seul nombre réel.

MEAN *Moyenne* Commande

	Niveau 1
➡	x
➡	$\lceil x_1\ x_2\ \ldots\ x_m \rceil$

MEAN calcule la moyenne de chacune des m colonnes de valeurs coordonnées qui se trouvent dans la matrice ΣDAT et renvoie la moyenne sous forme d'un vecteur de m nombres réels ou sous la forme d'un seul nombre réel si $m = 1$. La moyenne est calculée par la formule

$$\text{Moyenne} = \sum_{i=1}^{n} x_i/n$$

dans laquelle x_i est la $i$ème valeur coordonnée dans une colonne et n le nombre d'observations.

SDEV — *Ecart-type* — Commande

	Niveau 1
➡	x
➡	$[x_1\ x_2\ \ldots\ x_m]$

SDEV calcule l'écart-type de chacune des valeurs coordonnées des m colonnes de la matrice de statistiques en cours. L'écart-type est alors renvoyé sous forme d'un vecteur de nombres reéls ou sous la forme d'un seul nombre réel si $m = 1$. Les écarts-types sont calculés par la formule :

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

dans laquelle x_i est la $i$ème valeur coordonnée d'une colonne, $\bar{x}$ est la moyenne des données présentes dans cette colonne et n le nombre d'observations.

VAR — *Variance* — Commande

	Niveau 1
➡	x
➡	$[x_1\ x_2\ \ldots\ x_m]$

VAR calcule la variance des valeurs coordonnées de chacune des m colonnes de la matrice de statistiques en cours. La variance est donnée sous forme d'un vecteur de m nombres réels ou sous la forme d'un seul nombre réel si $m = 1$. Elle est calculée par la formule

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

dans laquelle x_i est la $i$ème valeur coordonnée d'une colonne, $\bar{x}$ est la moyenne des données de cette colonne et n est le nombre d'observations.

...STAT

MAXΣ *Sigma maximum* Commande

	Niveau 1
➡	x
➡	$[x_1\ x_2\ \ldots\ x_m]$

MAXΣ trouve la valeur coordonnée maximale de chacune des m colonnes de la matrice de statistiques en cours. Les maximums sont renvoyés sous la forme d'un vecteur comportant m nombres réels ou un seul nombre réel, si $m = 1$.

MINΣ *Sigma minimum* Commande

	Niveau 1
➡	x
➡	$[x_1\ x_2\ \ldots\ x_m]$

MINΣ trouve la valeur coordonnée minimum de chacune des m colonnes de la matrice de statistiques en cours. Les minimums sont renvoyés sous la forme d'un vecteur de m nombres réels ou comme un seul nombre réel si $m = 1$.

COLΣ CORR COV LR PREDV

COLΣ *Colonnes sigma* Commande

Niveau 2	Niveau 1	
n_1	n_2	➡

COLΣ prend deux numéros de colonnes dans la pile, n_1 et n_2, et les stocke sous la forme des deux premiers objets de la liste contenue dans la variable ΣPAR. Les nombres identifient des colonnes dans la matrice de statistiques en cours ΣDAT et sont utilisés par les commandes de statistiques qui opère sur les couples de colonnes. n_1 désigne la colonne correspondant à la variable indépendante de LR ou à la coordonnée horizontale de DRWΣ ou de SCLΣ. n_2 désigne la variable dépendante ou coordonnée verticale. Pour CORR et COV, l'ordre des deux numéros de colonnes n'est pas important.

Si l'une des commandes opérant sur deux colonnes est exécutée lorsque ΣPAR n'existe pas encore, ΣPAR est automatiquement créée avec les valeurs par défaut $n_1 = 1$ et $n_2 = 2$.

CORR ***Corrélation*** **Commande**

	Niveau 1
	➡ *corrélation*

CORR renvoie dans la matrice statistique en cours la corrélation de deux colonnes de valeurs coordonnées. Les colonnes sont spécifiées par les deux premiers éléments de ΣPAR (par défaut 1 et 2). La corrélation est calculée par la formule

$$\frac{\sum_{i=1}^{n} (x_{in_1} - \bar{x}_{n_1})(x_{in_2} - \bar{x}_{n_2})}{\sqrt{\sum_{i=1}^{n} (x_{in_1} - \bar{x}_{n_1})^2 \sum_{i=1}^{n} (x_{in_2} - \bar{x}_{n_2})^2}}$$

dans laquelle x_{in_1} est la $i$ème valeur coordonnée de la colonne n_1, x_{in_2} est la $i$ème valeur coordonnée de la colonne n_2, $\bar{x}_{n_1}$ est la moyenne des observations de la colonne n_1, $\bar{x}_{n_2}$ est la moyenne des observations de la colonne n_2 et n est le nombre d'observations.

...STAT

COV — Covariance — Commande

	Niveau 1
	➡ *covariance*

COV renvoie dans deux colonnes de la matrice de statistiques en cours la covariance d'échantillonnage des valeurs coordonnées. Les colonnes sont désignées par les deux premiers éléments de ΣPAR (1 et 2 par défaut). La covariance est calculée par la formule

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_{in_1}-\bar{x}_{n_1})(x_{in_2}-\bar{x}_{n_2})$$

dans laquelle x_{in_1} est la $i$ème valeur coordonnée de la colonne n_1, x_{in_2} est la $i$ème valeur coordonnée de la colonne n_2, $\bar{x}_{n_1}$ est la moyenne des données de la colonne n_1, $\bar{x}_{n_2}$ est la moyenne des données de la colonne n_2 et n est le nombre d'observations.

LR — *Régression linéaire* — Commande

	Niveau 2	Niveau 1
	➡ *ord. à l'orig.*	*pente*

LR calcule la régression linéaire d'une colonne de données dépendantes sur une colonne de données indépendantes, lorsque les colonnes de données existent dans la matrice de statistiques en cours. Les colonnes de données dépendantes et indépendantes sont spécifiées par les deux premiers éléments de ΣPAR (1 et 2 par défaut).

L'*ordonnée à l'origine* et la *pente* de la droite de régression sont renvoyées respectivement vers les niveaux 2 et 1 de la pile. LR stocke également ces coefficients de régression en tant que troisième (ordonnée à l'origine) et quatrième (pente) éléments de la liste contenue dans la variable ΣPAR.

PREDV ***Valeur estimée*** **Commande**

Niveau 1	Niveau 1
x ➡	*valeur estimée*

PREDV calcule une valeur estimée à partir d'un nombre réel, l'argument *x*, utilisant les coefficients de régression linéaire les plus récemment calculés par LR et stockés dans la variable ΣPAR :

$$\textit{valeur estimée} = (x \times \textit{pente}) + \textit{ordonnée à l'origine.}$$

Les coefficients de régression *ordonnée à l'origine* et *pente* sont stockés par LR dans la variable ΣPAR, respectivement en tant que troisième et quatrième éléments. Si vous exécutez PREDV sans avoir auparavant exécuté LR, une valeur par défaut de zéro est utilisée pour les deux coefficients, de sorte que PREDV renverra zéro dans ce cas.

UTPC UTPF UTPN UTPT

Le HP-28C/S offre quatre commandes de probabilités (*Upper-Tail Probability*, abrégé en UTP), que vous pouvez utiliser pour déterminer le niveau de signification de tests statistiques. La fonction de probabilité d'une variable aléatoire X est définie comme la probabilité que X soit *plus grand que* le nombre *x* et soit égal à 1 − F(*x*), où F(*x*) est la fonction de répartition de X.

Les inverses des fonctions de répartition sont utiles pour construire des intervalles de confiance. L'argument de l'inverse d'une de ces fonctions de probabilité est une valeur comprise entre 0 et 1 ; lorsque l'argument est exprimé en pourcentage, les valeurs de la fonction inverse sont appelés des percentiles. Par exemple, le 90ème percentile d'une distribution est le nombre *x* pour lequel la probabilité que la variable aléatoire X soit plus grande que *x* est 100 % − 90 % = 10 %.

...STAT

Vous pouvez utiliser l'algorithme de résolution des équations pour obtenir les inverses des fonctions de probabilité. Supposons que vous désiriez déterminer un percentile de la distribution normale. Si

$$P = \text{percentile}/100$$

$$M = \text{moyenne de la distribution}$$

$$V = \text{variance}$$

$$X = \text{variable aléatoire}$$

UTPN (UPPER-TAIL PROBABILITY NORMAL DISTRIBUTION — décrite ci-dessous) donne la probabilité pour la loi normale (loi de Laplace-Gauss). Pour résoudre en X l'équation

$$1 - P = \text{utpn}(M, V, X),$$

créez le programme

« 1 P - M V X UTPN - »,

et stockez-le en tant qu'équation en cours en appuyant sur [SOLV] **STEQ**. Appuyez ensuite sur **SOLVR** pour afficher le menu de résolution :

P **M** **V** **X** **EXPR=** **'**.

Essayez $M = 0$, $V = 1$:

0 **M** **1** **V**.

Calculez maintenant le 95ème percentile :

0,95 **P** **X**

Le résultat obtenu est $X = 1.6449$.

UTPC — *Loi de χ^2 (de Khi carré)* — Commande

Niveau 2	Niveau 1		Niveau 1
n	*x*	➡	*utpc(n, x)*

UTPC (UPPER-TAIL PROBABILITY CHI-SQUARE DISTRIBUTION) donne la probabilité utpc(*n, x*) qu'une variable aléatoire χ^2 (khi carré) soit plus grande que *x*, où *n* est le nombre de degrés de liberté de la distribution. *n* doit être un nombre entier positif.

UTPF — *Loi de F* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
n_1	n_2	*x*	➡	*utpf(n_1, n_2, x)*

UTPF (UPPER-TAIL PROBABILITY F DISTRIBUTION) donne la probabilité utpf(n_1, n_2, *x*) qu'une variable aléatoire F soit plus grande que *x*, n_1 et n_2 étant les nombres de degrés de liberté du numérateur et du dénominateur de la loi de F. n_1 et n_2 doivent être des nombres entiers positifs.

UTPN — *Loi normale (Loi de Laplace-Gauss)* — Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 1
m	*v*	*x*	➡	*utpn(m, v, x)*

UTPN (UPER-TAIL PROBABILITY NORMAL DISTRIBUTION) donne la probabilité utpn(*m, v, x*) qu'une variable aléatoire normale soit plus grande que *x*, avec *m* et *v* représentant respectivement la moyenne et la variance de la loi normale. *v* doit être un nombre non négatif.

...STAT

UTPT — *Loi de t (loi de Student)* — Commande

Niveau 2	Niveau 1		Niveau 1
n	*x*	➧	*utpt*(*n*, *x*)

UTPT (UPPER-TAIL PROBABILITY STUDENT'S t DISTRIBUTION) donne la probabilité utpt(*n*, *x*) qu'une variable aléatoire *t* (ou variable de Student) soit plus grande que *x*, *n* étant le nombre de degrés de liberté de la loi. *n* doit être un nombre entier positif.

STORE

STO+	STO−	STO*	STO/	SNEG	SINV
SCONJ					

Le menu STORE contient des commandes d'arithmétique directe qui vous permettent d'effectuer additions, soustractions multiplications, divisions, inversions, changements de signes, et calculs du conjugué sur les tableaux et nombres réels et complexes stockés dans les variables, sans avoir à rappeler le contenu de la variable dans la pile. Outre le fait qu'elle minimise le nombre de touches à actionner, la commande STORE offre une méthode directe de modification du contenu d'un tableau, qui utilise moins de mémoire qu'une manipulation des tableaux dans la pile opérationnelle.

L'arithmétique directe est limitée aux variables ordinaires — vous ne pouvez pas utiliser un nom local comme argument de l'une des commandes du menu STORE.

STO+ STO− STO* STO/ SNEG SINV

STO+ *Stocker somme* Commande

Niveau 2	Niveau 1	
z	' *nom* '	➧
' *nom* '	*z*	➧
⟦ *tableau* ⟧	' *nom* '	➧
' *nom* '	⟦ *tableau* ⟧	➧

...STORE

STO+ ajoute un nombre ou un tableau au contenu de la variable. Le nom de la variable et le nombre ou le tableau peuvent être placés dans n'importe quel ordre dans la pile.

L'objet dans la pile et l'objet dans la variable doivent être de nature à pouvoir être additionnés — vous pouvez additionner n'importe quelle combinaison de nombres réels et complexes, ou n'importe quelle combinaison de tableaux réels et complexes compatibles.

STO− **Stocker différence** **Commande**

Niveau 2	Niveau 1	
z	*'nom'*	➡
'nom'	*z*	➡
[tableau]	*'nom'*	➡
'nom'	*[tableau]*	➡

STO− calcule la différence entre deux nombres ou deux tableaux. Un objet est pris dans la pile et l'autre est le contenu de la variable spécifiée par son nom dans la pile. La différence est stockée dans la variable et en constitue la nouvelle valeur.

Le résultat dépend de l'ordre des arguments :

- Si *nom* est au niveau 1, la différence

  (valeur au niveau 2) − (valeur dans *nom*)

  devient la nouvelle valeur de *nom*.
- Si *nom* est au niveau 2, la différence

  (valeur dans *nom*) − (valeur au niveau 1)

  devient la nouvelle valeur de *nom*.

L'objet se trouvant dans la pile et celui se trouvant dans la variable doivent être de nature à pouvoir être soustraits l'un de l'autre — vous pouvez effectuer la soustraction sur n'importe quelle combinaison de nombres réels et complexes, ou de tableaux réels et complexes compatibles.

STO*

Stocker produit — **Commande**

Niveau 2	Niveau 1	
z	*'nom'*	➧
'nom'	*z*	➧
[tableau]	*'nom'*	➧
'nom'	*[tableau]*	➧

STO* multiplie le contenu de la variable par un nombre ou un tableau se trouvant dans la pile. Lors de la multiplication de deux nombres ou d'un nombre et d'un tableau, le nom de la variable et l'autre objet peuvent être dans n'importe quel ordre dans la pile. Lors de la multiplication de deux tableaux, le résultat dépend de l'ordre des arguments :

- Si *nom* est au niveau 1, le produit

  (tableau au niveau 2) × (tableau dans *nom*)

  devient la nouvelle valeur de *nom*.
- Si *nom* est au niveau 2, le produit

  (tableau dans *nom*) × (tableau au niveau 1)

  devient la nouvelle valeur de *nom*.

Les tableaux doivent être de nature à pouvoir être multipliés.

STO/

Stocker quotient — **Commande**

Niveau 2	Niveau 1	
z	*'nom'*	➧
'nom'	*z*	➧
[tableau]	*'nom'*	➧
'nom'	*[tableau]*	➧

STO/ calcule le quotient de deux nombres ou tableaux. Un objet est pris dans la pile, et l'autre est le contenu d'une variable spécifiée par son nom dans la pile. Le quotient est stocké en tant que nouvelle valeur de la variable.

...STORE

Le résultat dépend de l'ordre des arguments :

- Si *nom* est au niveau 1, le quotient

  (valeur au niveau 2)/(valeur dans *nom*)

  devient la nouvelle valeur de *nom*.
- Si *nom* est au niveau 2, le quotient

  (valeur dans *nom*)/(valeur dans niveau 1)

  devient la nouvelle valeur dans *nom*.

L'objet dans la pile et l'objet dans la variable doivent être de nature à pouvoir être divisés l'un par l'autre. En particulier, si les deux objets sont des tableaux, le diviseur (niveau 1) doit être une matrice carrée, et le dividende (niveau 2) doit avoir le même nombre de colonnes que le diviseur.

SNEG **Stocker opposé** **Commande**

Niveau 1	
' *nom* ' ➧	

SNEG (STORE NEGATE) calcule l'opposé du contenu de la variable nommée dans la pile ; le résultat remplace le contenu initial de la variable. Cette dernière peut contenir un nombre réel, complexe, ou un tableau.

SINV **Stocker inverse** **Commande**

Niveau 1	
' *nom* ' ➧	

SINV (STORE INVERSE) calcule l'inverse du contenu de la variable nommée dans la pile ; le résultat remplace le contenu initial de la variable. Cette dernier peut contenir un nombre réel, complexe, ou une matrice carrée.

SCONJ

SCONJ *Stocker conjugué* Commande

Niveau 1	
' nom ' ➧	

SCONJ (STORE CONJUGATE) calcule le conjugué du contenu de la variable nommée dans la pile ; le résultat remplace le contenu initial de la variable. Cette dernière peut contenir un nombre réel, complexe, ou un tableau.

STRING

→STR	STR→	CHR	NUM	POS	DISP
SUB	SIZE				

Une chaîne (*string*) est un des « objets » du HP-28C/S ; c'est une série de caractères délimitée par des *guillemets anglais*, ", à chaque extrémité. N'importe quel caractère généré par le HP-28C/S peut être inclus dans une chaîne, y compris les délimiteurs d'objets ‹, ›, ⟦, ⟧, ‹, ›, #, ", ', » et «. Les caractères qui ne peuvent être trouvés sur le clavier peuvent être saisis par l'intermédiaire de la commande CHR.

Bien que vous puissiez inclure les caractères " dans une chaîne, par la commande CHR et +, il est impossible de corriger ou de modifier une chaîne contenant le signe " de la manière habituelle, et ceci parce que ENTER cherche les paires de " sur la ligne de commande - les signes " excédentaires présents à l'intérieur d'une chaîne provoqueront sa cassure en plusieurs tronçons ne contenant pas le signe ".

Les chaînes sont le plus souvent utilisées pour des raisons d'affichage : messages de sollicitation, libellés des résultats obtenus etc. Les commandes incluses dans le menu STRING permettent des opérations simples sur les chaînes et les caractères. Les commandes →STR et STR→, toutefois, sont d'une grande utilité dans le cas des chaînes - elles peuvent convertir tout objet ou suite d'objets en chaîne de caractères et vice-versa. Dans bien des cas, un objet sous forme de chaîne utilisera moins de mémoire que s'il est sous sa forme normale. Les objets peuvent être stockés dans des variables lorsqu'ils sont sous forme de chaînes et peuvent n'être remis sous leur forme normale que lorsque c'est nécessaire. Voyez à ce sujet les descriptions de →STR et de STR→, ci-après.

Fonction disponible au clavier

+ ***Ajouter*** **Fonct. analyt.**

Niveau 2	Niveau 1		Niveau 1
"*chaîne*$_1$"	"*chaîne*$_2$"	➡	"*chaîne*$_1$ *chaîne*$_2$"

+ effectue une concaténation des caractères de la chaîne au niveau 1 et de ceux appartenant à la chaîne se trouvant au niveau 2, avec pour résultat une chaîne.

→STR **STR→** **CHR** **NUM** **POS** **DISP**

→STR ***Convertir objet en chaîne*** **Commande**

Niveau 1		Niveau 1
objet	➡	"*chaîne*"

→STR convertit un objet arbitraire en chaîne de caractères. La chaîne produite est essentiellement la même que celle obtenue si l'objet était au niveau 1 et si le mode multi-lignes était actif :

- La chaîne résultante comprend l'objet entier, même si la forme affichée de cet objet est trop importante pour se loger sur l'affichage.
- Si l'objet est affiché sur deux ou plusieurs lignes, la chaîne résultat contiendra des caractères « nouvelle ligne » (caractère 10) à la fin de chaque ligne. Les nouvelles lignes sont affichées sous forme du caractère par défaut ▪.

- Les nombres sont convertis en chaînes selon le mode d'affichage numérique en cours (STD, FIX, SCI ou ENG) ou la base d'entiers binaires (DEC, BIN, OCT ou HEX) et la taille de mot. La forme interne de pleine précision n'est pas nécessairement représentée dans la chaîne-résultat. Vous pouvez être assuré que →STR préserve la pleine précision d'un nombre en sélectionnant le mode STD ou une taille de mot de 64 bits, ou les deux, avant d'exécuter →STR.
- Si l'objet est déjà sous la forme d'une chaîne, →STR renvoie la chaîne.

Vous pouvez utiliser →STR pour créer des affichages spéciaux pour étiqueter les résultats produits par un programme ou pour créer des messages de sollicitation de saisies. Par exemple, la séquence

```
"Resultat = " SWAP →STR + 1 DISP
```

affiche "Resultat = *objet*" en ligne 1 de l'affichage, *objet* étant un objet sous forme de chaîne, pris au niveau 1.

STR→ ***Convertir chaîne en objets*** **Commande**

Niveau 1	
"*chaîne*" ➡	

STR→ est la forme « commande » de ENTER. Les caractères de la chaîne (l'argument) sont analysés et évalués comme contenu de la ligne de commande. La chaîne peut définir un seul objet ou bien il peut s'agir d'une série d'objets qui seront évalués comme le seraient des programmes.

STR→ peut aussi être utilisée pour ramener à leur forme d'origine des objets qui ont été convertis en chaînes par →STR. La combinaison →STR STR→ laisse les objets inchangés si ce n'est que →STR convertit les nombres en chaînes selon le format d'affichage numérique en cours, la base des entiers binaires et la taille de mot. STR→ ne reproduira un nombre qu'avec la précision représentée dans la chaîne.

CHR — *Caractère* — Commande

Niveau 1	Niveau 1
n ➡	"*chaîne*"

CHR renvoie une chaîne d'un seul caractère contenant le caractère du HP-28C/S correspondant au code de caractère *n* pris au niveau 1. Le caractère par défaut ▫ est utilisé pour tous les codes de caractères qui ne font pas partie du jeu de caractères normal du HP-28C/S.

Le code de caractère 0 est réservé pour utilisation spéciale en ligne de commande. Ce caractère peut être introduit dans les chaînes de caractères en utilisant CHR, mais un essai de correction ou de modification d'une chaîne contenant ce caractère provoquera l'erreur `Can't Edit CHR(0)`.

NUM — *Numéro du caractère* — Commande

Niveau 1	Niveau 1
"*chaîne*" ➡	*n*

NUM renvoie le code du premier caractère d'une chaîne.

Le tableau suivant montre la relation existant entre les codes de caractères (résultats de NUM, arguments de CHR) et les caractères eux-mêmes (résultats de CHR et arguments de NUM). Pour les codes de caractères de 0 à 147, le tableau montre les caractères comme ils sont affichés dans une chaîne de caractères. Les caractères correspondant aux codes 148 à 255 figurent dans le tableau sous la forme imprimée par l'imprimante HP 82240A ; ces caractères sont affichés par le HP-28C/S comme le caractère par défaut ▪.

...STRING

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		224	
129		161		193		225	
130		162		194		226	
131		163		195		227	
132		164		196		228	
133		165		197		229	
134		166		198		230	
135		167		199		231	
136		168		200		232	
137		169		201		233	
138		170		202		234	
139		171		203		235	
140		172		204		236	
141		173		205		237	
142		174		206		238	
143		175		207		239	
144		176		208		240	
145		177		209		241	
146		178		210		242	
147		179		211		243	
148		180		212		244	
149		181		213		245	
150		182		214		246	
151		183		215		247	
152		184		216		248	
153		185		217		249	
154		186		218		250	
155		187		219		251	
156		188		220		252	
157		189		221		253	
158		190		222		254	
159		191		223		255	

...STRING

POS — *Position* — Commande

Niveau 2	Niveau 1		Niveau 1
"*chaîne*$_1$"	"*chaîne*$_2$"	➡	*n*

POS renvoie un nombre réel *n* qui est la position de *chaîne*$_2$ au sein de *chaîne*$_1$. La position est, en partant de la gauche, celle du premier caractère de la sous-chaîne à l'intérieur de *chaîne*$_1$ (une partie de *chaîne*$_1$) qui correspond à *chaîne*$_2$. Un résultat zéro indique que *chaîne*$_2$ n'est pas reproduite dans *chaîne*$_1$.

DISP — *Afficher* — Commande

Niveau 2	Niveau 1		
objet	*n*	➡	

DISP affiche *objet* sur la *n*ième ligne de l'affichage, *n* étant un nombre réel entier. $n = 1$ indique la ligne supérieure de l'affichage ; $n = 4$ indique la ligne inférieure. DISP arme l'indicateur binaire du système de façon que l'affichage normal de la pile soit supprimé.

Les chaînes sont affichées sans les délimiteurs habituels ". Les autres objets sont affichés sous la même forme qu'ils le sont au niveau 1 en mode d'affichage à plusieurs lignes. Si l'affichage de l'objet nécessite plus d'une ligne d'affichage, celui-ci commence à la ligne *n* et continue vers le bas de l'affichage, jusqu'à la fin de l'objet ou la fin de l'affichage.

SUB SIZE

SUB

Sous-ensemble **Commande**

Niveau 3	Niveau 2	Niveau 1		Niveau 1
"*chaîne*$_1$"	n_1	n_2	➡	"*chaîne*$_2$"
⟨ *liste*$_1$ ⟩	n_1	n_2	➡	⟨ *liste*$_2$ ⟩

SUB prend dans la pile une chaîne et deux nombres entiers n_1 et n_2 et renvoie une nouvelle chaîne de caractères contenant les caractères placés aux positions n_1 à n_2 dans la chaîne d'origine. Si $n_2 < n_1$, SUB renvoie une chaîne vide.

Les arguments inférieurs à 1 sont changés en 1 ; les arguments plus grands que la taille de la chaîne sont mis à la taille de la chaîne.

Référez-vous à « LIST » pour l'utilistation de SUB avec des listes.

SIZE

Taille d'une chaîne **Commande**

Niveau 1		Niveau 1
"*chaîne*"	➡	*n*
⟦ *tableau* ⟧	➡	⟨ *liste* ⟩
⟨ *liste* ⟩	➡	*n*
' *symbole* '	➡	*n*

SIZE renvoie un nombre *n* qui est le nombre de caractères que contient une chaîne.

Référez-vous à « ALGEBRA », « ARRAY » et « LIST » pour l'utilisation de SIZE avec d'autres types d'objet.

TRIG

SIN	**ASIN**	**COS**	**ACOS**	**TAN**	**ATAN**
P→R	**R→P**	**R→C**	**C→R**	**ARG**	
→HMS	**HMS→**	**HMS+**	**HMS−**	**D→R**	**R→D**

Le menu TRIG (trigonométrie) contient les commandes ayant trait à la mesure des angles et à la trigonométrie : fonctions circulaires, conversions de coordonnées polaires en coordonnées rectangulaires et inversement, conversions degrés/radians, et calculs de valeurs exprimées en degrés-minutes-secondes ou en heures-minutes-secondes.

SIN ASIN COS ACOS TAN ATAN

Ce sont les fonctions circulaires et leurs inverses. SIN, COS et TAN interprètent les arguments réels en fonction du mode angulaire choisi (DEG ou RAD), et leurs résultats sont des nombres réels. ACOS, ASIN et ATAN expriment leurs résultats à valeur réelle en fonction du mode angulaire choisi.

Les six fonctions admettent des arguments complexes, et leurs résultats sont alors des nombres complexes. Pour ACOS et ASIN, les arguments réels ayant une valeur absolue supérieure à 1 donnent également des résultats qui sont des nombres complexes. Les nombres complexes sont interprétés et exprimés en radians.

ASIN, ACOS et ATAN donnent les valeurs principales des relations inverses, comme expliqué dans « COMPLEX ».

SIN — *Sinus* — Fonct. analyt.

Niveau 1		Niveau 1
z	➡	sin *z*
'*symbole*'	➡	'SIN(*symbole*)'

SIN donne le sinus de son argument. Pour les arguments complexes,

$$\sin (x + iy) = \sin x \cosh y + i \cos x \sinh y.$$

ASIN — *Arc sinus* — Fonct. analyt.

Niveau 1		Niveau 1
z	➡	arc sin *z*
'*symbole*'	➡	'ASIN(*symbole*)'

ASIN donne la valeur principale de l'angle dont un sinus est égal à l'argument. Pour les arguments réels, le résultat est compris entre -90 et $+90$ degrés ($-\pi/2$ et $+\pi/2$ radians). Pour les arguments complexes, c'est la valeur principale complexe de l'arc sinus qui est obtenue :

$$\text{arc sin } z = -i \ln (iz + \sqrt{1 - z^2})$$

Un argument réel x extérieur au domaine $-1 \leqslant x \leqslant 1$ est converti en un argument complexe $z = x + 0i$, et l'on obtient la valeur principale complexe.

...TRIG

COS *Cosinus* Fonct. analyt.

Niveau 1		Niveau 1
z	➡	cos z
'symbole'	➡	'COS(symbole)'

COS donne le cosinus de son argument. Pour les arguments complexes,

$$\cos (x + iy) = \cos x \cosh y - i \sin x \sinh y$$

ACOS *Arc cosinus* Fonct. analyt.

Niveau 1		Niveau 1
z	➡	arc cos z
'symbole'	➡	'ASIN(symbole)'

ACOS donne la valeur principale de l'angle dont un cosinus est égal à l'argument. Pour les arguments réels, le résultat est compris entre 0 et 180 degrés (0 et π radians). Pour les arguments complexes, ACOS donne la valeur principale complexe de l'arc cosinus :

$$\text{arc cos } z = -i \ln (z + \sqrt{z^2 - 1})$$

Un argument réel x à l'extérieur du domaine $-1 \leqslant x \leqslant 1$ est converti en un argument complexe $z = x + 0i$, et l'on obtient la valeur principale complexe.

TAN — *Tangente* — Fonct. analyt.

Niveau 1		Niveau 1
z	➡	tan *z*
'*symbole*'	➡	'TAN(*symbole*)'

TAN donne la tangente de son argument. Pour les arguments complexes,

$$\tan(x + iy) = \frac{\sin x \cos x + i \sinh y \cosh y}{(\sinh y)^2 + (\cos x)^2}$$

Si un argument réel est un entier impair multiple de 90, et si le mode angulaire choisi est DEG, il se produit une exception mathématique identifiée par le message `Infinite Result` (résultat infini). Si l'indicateur binaire 59 est désarmé, le signe du résultat (MAXR) est celui de l'argument.

ATAN — *Arc tangente* — Fonct. analyt.

Niveau 1		Niveau 1
z	➡	arc tan *z*
'*symbole*'	➡	'ATAN(*symbole*)'

ATAN donne la valeur principale de l'angle dont une tangente est égale à l'argument. Pour des arguments réels, le résultat est compris entre -90 et $+90$ degrés ($-\pi/2$ et $+\pi/2$ radians). Pour les arguments complexes, ATAN donne la valeur principale complexe de l'arc tangente :

$$\text{arc tan } z = \frac{i}{z} \ln\left(\frac{i + z}{i - z}\right)$$

...TRIG

P→R R→P R→C C→R ARG

Les fonctions de conversion P→R (*polaires en rectangulaires*) et R→P (*rectangulaires en polaires*), et la fonction ARG (*argument*) opèrent sur les nombres complexes qui représentent les coordonnées de points dans des systèmes à deux dimensions. Les fonctions R→C (*réels en complexes*) et C→R (*complexes en réels*) convertissent des couples de nombres réels en leur équivalent en notation complexe, et vice versa.

Les fonctions P→R et R→P peuvent également opérer sur les deux premières composantes d'un vecteur réel.

P→R ***Polaires en rectangulaires*** **Fonction**

Niveau 1		Niveau 1
x	➡	(*x*,0)
(*r*,*θ*)	➡	(*x*,*y*)
[*r θ*...]	➡	[*x y*...]
'*symbole*'	➡	'P→R(*symbole*)'

P→R convertit un nombre complexe (r, θ) ou un vecteur à deux composantes $[\, r\ \theta\,]$, représentant des coordonnées polaires, en un nombre complexe (x, y) ou en un vecteur à deux composantes $[\, x\ y\,]$, représentant des coordonnées rectangulaires, avec :

$$x = r \cos \theta, \quad y = r \sin \theta.$$

Le mode angulaire choisi détermine si θ est interprété en degrés ou en radians.

Si un vecteur a plus de deux composantes, P→R convertit les deux premières composantes et laisse les autres inchangées. Pour les vecteurs à trois composantes, P→R convertit un vecteur $[\, \rho\ \theta\ z\,]$ à coordonnées cylindriques (où ρ est la distance par rapport à l'axe z, et θ l'angle, dans le plan xy, entre l'axe des x et le vecteur projeté) en un vecteur $[\, x\ y\ z\,]$ à coordonnées rectangulaires.

Vous pouvez représenter un vecteur en coordonnées sphériques sous la forme [r ϕ θ], où r est la longueur du vecteur, ϕ l'angle entre l'axe des z et le vecteur, et θ l'angle dans le plan xy entre l'axe des x et le vecteur projeté. Pour convertir les coordonnées sphériques d'un vecteur en coordonnées rectangulaires, exécutez :

```
P→R ARRY→ DROP ROT ⟨3⟩ →ARRY P→R
```

R→P ***Rectangulaires en polaires*** **Fonction**

Niveau 1		Niveau 1
z	➡	⟨*r* , *θ*⟩
[*x y*...]	➡	[*r θ*...]
'*symbole*'	➡	'R→P⟨*symbole*⟩'

R→P convertit un nombre complexe (x, y) ou un vecteur à deux composantes [x y], représentant des coordonnées rectangulaires, en un nombre complexe (r, θ) ou en un vecteur à deux composantes [r θ], représentant des coordonnées polaires, où :

$$r = \text{abs}\ (x, y), \quad \theta = \arg\ (x, y)$$

Le mode angulaire choisi détermine si θ est exprimé en degrés ou en radians. Un argument réel x est traité comme l'argument complexe (x, 0).

...TRIG

Si un vecteur a plus de deux composantes, R→P convertit les deux premières composantes et laisse les autres inchangées. Pour les vecteur à trois composantes, R→P convertit un vecteur [x y z] à coordonnées rectangulaires en un vecteur [ρ θ z] à coordonnées cylindriques, où ρ est la distance par rapport à l'axe des z et θ l'angle dans le plan xy entre l'axe des x et le vecteur projeté.

Vous pouvez représenter un vecteur à coordonnées sphériques sous la forme [r ϕ θ], où r est la longueur du vecteur, ϕ l'angle entre l'axe des z et le vecteur, et θ l'angle dans le plan xy entre l'axe des x et le vecteur projeté. Pour convertir les coordonnées rectangulaires d'un vecteur en coordonnées sphériques, exécutez :

```
R→P ARRY→ DROP ROT ROT ⟨3⟩ →ARRY R→P
```

R→C

Réels en complexe **Commande**

Niveau 2	Niveau 1		Niveau 1
x	y	➡	(x,y)

R→C combine les nombres réels x et y en un couple de coordonnées (x, y).

Consultez « ARRAY » pour utiliser R→C avec des tableaux.

C→R

Complexe en réels **Commande**

Niveau 1		Niveau 2	Niveau 1
(x,y)	➡	x	y

C→R convertit un couple de coordonnées (x, y) en deux nombres réels x et y.

Consultez « ARRAY » pour utiliser C→R avec des tableaux.

ARG

Argument **Fonction**

Niveau 1		Niveau 1
z	➡	θ
'*symbole*'	➡	'ARG(*symbole*)'

ARG donne l'angle polaire θ d'un couple de coordonnées (x, y) où

$$\theta = \begin{cases} \text{arc tan } y/x & \text{pour } x \geqslant 0, \\ \text{arc tan } y/x + \pi \text{ signe } y & \text{pour } x < 0, \text{ mode RADIANS,} \\ \text{arc tan } y/x + 180 \text{ signe } y & \text{pour } x < 0, \text{ mode DEGRES.} \end{cases}$$

Le mode angulaire en cours détermine si θ est exprimé en degrés ou en radians. Un argument réel x est traité comme l'argument complexe $(x, 0)$.

→HMS HMS→ HMS+ HMS− D→R R→D

Les commandes →HMS, HMS→, HMS+, et HMS− opèrent sur des quantités de temps (ou des quantités angulaires) exprimées à l'aide de nombres réels sous la forme HMS (*heures-minutes-secondes*).

...TRIG

Le format HMS est *h,MMSSs* (ou *h.MMSSs* si vous avez choisi le point comme séparateur décimal), où :

- *h* = zéro ou plusieurs chiffres représentant la partie entière du nombre.
- *MM* = deux chiffres représentant le nombre de minutes.
- *SS* = deux chiffres représentant le nombre de secondes.
- *s* = zéro ou plusieurs chiffres représentant la partie fractionnaire décimale de secondes.

Voici quelques exemples de quantités de temps (ou de quantité angulaires) exprimées en format HMS.

Quantité	Format HMS
12 h 32 mn 46 s (12° 32′ 46″)	12,3246
− 6 h 00 mn 13,2 s (− 6° 00′ 13,2″)	−6,00132
36mn (36′)	0,36

→HMS

Décimal en H-M-S

Commande

Niveau 1		Niveau 1
x	➡	*hms*

→HMS convertit un nombre réel représentant des heures décimales (ou des degrés) en format HMS.

HMS→

H-M-S en décimal

Commande

Niveau 1		Niveau 1
hms	➡	*x*

HMS→ convertit un nombre réel exprimé en format HMS en son équivalent décimal.

HMS+ *Heures-Minutes-Secondes : Plus* Commande

Niveau 2	Niveau 1		Niveau 1
hms_1	hms_2	➡	$hms_1 + hms_2$

HMS+ additionne deux nombres en format HMS et donne leur somme en format HMS.

HMS− *Heures-Minutes-Secondes : Moins* Commande

Niveau 2	Niveau 1		Niveau 1
hms_1	hms_2	➡	$hms_1 - hms_2$

HMS− soustrait deux nombres réels en format HMS et donne leur différence en format HMS.

D→R *Degrés en radians* Fonction

Niveau 1		Niveau 1
x	➡	$(\pi/180)\ x$
'*symbole*'	➡	'D→R(*symbole*)'

D→R convertit en radians un nombre réel qui était exprimé en degrés.

R→D *Radians en degrés* Fonction

Niveau 1		Niveau 1
x	➡	$(180/\pi)\ x$
'*symbole*'	➡	'R→D(*symbole*)'

R→D convertit en degrés un nombre réel qui était exprimé en radians.

UNITS

La valeur d'une mesure physique implique l'utilisation d'une unité, en plus de la valeur numérique. Pour faire passer une mesure physique d'un système d'unités à un autre, vous multipliez la valeur numérique par un facteur de conversion qui est le rapport entre la nouvelle unité et l'ancienne. Le HP-28C/S a automatisé cette opération grâce à la commande CONVERT. Vous spécifiez une valeur numérique, l'ancienne unité, et la nouvelle unité, et CONVERT calcule le facteur de conversion approprié et multiplie la valeur numérique par ce facteur de conversion.

Le système de conversion d'unités du HP-28C/S est basé sur le système internationnal d'unités (SI). 120 unités ont été incluses en mémoire permanente du HP-28C/S. CONVERT identifie n'importe quel mutiple de ces unités, ainsi que les unités supplémentaires que vous avez la possibilité de définir. Le catalogue UNITS est la liste des unités intégrées au calculateur et donne leurs valeurs, exprimées en quantités de base standard.

Le système internationnal définit sept quantités de base : longueur (mètre), masse (kilogramme), temps (seconde), courant électrique (ampère), température thermodynamique (degré kelvin), intensité lumineuse (candela), et quantité de substance (mole). En outre, le HP-28C/S reconnaît une quantité de base indéfinie que vous pouvez spécifier en tant qu'une des unités définies par l'utilisateur.

CONVERT *Convertir* Commande

Niveau 3	Niveau 2	Niveau 1		Niveau 2	Niveau 1
x	"*ancienne*"	"*nouvelle*"	➡	*y*	"*nouvelle*"
x	"*ancienne*"	'*nouvelle*'	➡	*y*	'*nouvelle*'
x	'*ancienne*'	"*nouvelle*"	➡	*y*	"*nouvelle*"
x	'*ancienne*'	'*nouvelle*'	➡	*y*	'*nouvelle*'

CONVERT multiplie un nombre réel x par un facteur de conversion qui est calculé à partir des deux arguments représentant l'ancienne et la nouvelle unité. Le nombre réel résultat, y, est envoyé au niveau 2 et le nom de la nouvelle unité apparaît au niveau 1.

Généralement, la nouvelle unité et l'ancienne sont représentées par des chaînes de caractères. Mais par commodité dans les conversions simples, vous pouvez utiliser un nom pour représenter une unité. Par exemple, à condition de n'avoir pas créé de variable nommée 'ft' ou 'm', vous pouvez convertir 320 pieds en mètres en exécutant :

```
320 ft m CONVERT.
```

Les chaînes d'unités sont des objets qui représentent des expressions algébriques contenant les abréviations des unités. Elles peuvent contenir :

- N'importe quelle unité intégrée au calculateur ou définie par l'utilisateur. Les unités intégrées sont représentées par leurs abréviations (voir « Le catalogue des unités », p. 335). Les unités définies par l'utilisateur sont représentées par leurs noms de variable (voir « Unités définies par l'utilisateur », p. 343).
- Une unité, suivie du symbole ^, plus un chiffre unique compris entre 1 et 9. Par exemple : `"m^2"` (mètres carrés), `"g*s^3"` (grammes-secondes au cube).
- Une unité précédée d'un préfixe représentant une puissance multiple de 10. Par exemple `"Mpc"` (Megaparsec), `"nm"` (nanomètre). Voir « Préfixes d'unités ».
- Deux ou plusieurs unités multipliées l'une par l'autre (ou les unes par les autres), à l'aide du symbole *. Par exemple : `"g*cm"` (grammes-centimètres), `"ft*lb"` (pieds-livres), `"m*kg*s"` (mètres-kilogrammes-secondes).
- Un symbole / pour indiquer les puissances inverses des unités. Si, dans une chaîne, toutes les unités ont des puissances inverses, la chaîne peut commencer par `"1/"`. Par exemple : `"m/sec"` (mètres par secondes), `"1/m"` (mètres inverses), `"g*cm/s^2*°K"` (grammes-centimètres par secondes au carré par degrés Kelvin).
- Le symbole ', qui est ignoré. Ceci vous permet de créer une expression algébrique dans la pile puis d'utiliser →STR pour changer l'expression en une chaîne d'unités. Toutefois, les parenthèses sont interdites dans les chaînes d'unités.

...UNITS

Les deux chaînes d'unités doivent représenter une conversion d'unités dimensionnellement compatibles. Par exemple, vous pouvez convertir "l" (litres) en "cm^3" (centimètres cubes), mais pas en "acre". CONVERT vérifie que les puissances de chacune des huit quantités de base (sept quantités de base du système international plus une définie par l'utilisateur) sont les mêmes dans les deux chaînes d'unités (compatibilité dimensionnelle vérifiée par modulo 256).

Voici quelques exemples d'utilisation de CONVERT (nombres représentés en format STD) :

Ancienne valeur	Ancienne unité	Nouvelle unité		Nouvelle valeur	Nouvelle unité
1	"m"	"ft"	➧	3,28083989501	"ft"
1	"b*Mpc"	"cm^3"	➧	3,085678	"cm^3"
12,345	"kg*m/s^2"	"dyn"	➧	18585	"dyn"

Conversion de températures

Les conversions entre les quatre échelles de températures (°K, °C, °F, et °R) font intervenir l'addition de constantes et la multiplication par certains facteurs. Si les deux arguments (chaînes d'unités) contiennent une seule unité de température, sans préfixe ni exposant, CONVERT effectue une conversion absolue, en incluant les constantes à ajouter. Par exemple, pour convertir 50 degrés Fahrenheit en degrés Celsius, exécutez :

```
50 °F °C CONVERT
```

Si l'une des chaînes d'unités comprend un préfixe, un exposant ou une unité qui ne soit pas une unité de mesure de la température, CONVERT effectue une conversion relative, qui ignore les constantes à ajouter.

Le catalogue des unités

Une pression sur ■ [UNITS] active le catalogue des unités, qui est analogue au catalogue des commandes obtenu par pression sur ■ [CATALOG]. Ce catalogue énumère toutes les unités contenues dans le HP-28C/S, avec l'abréviation reconnue par CONVERT et la valeur de l'unité exprimée dans l'une des quantités de base du système international.

Lorsque vous appuyez sur ■ [UNITS], l'affichage normal du HP-28C/S est remplacé par l'affichage du catalogue :

La ligne supérieure affiche l'abréviation de l'unité choisie, a dans l'exemple ci-dessus, suivie du nom complet, are. Are est la première unité du catalogue alphabétique d'unités du HP-28C/S. La seconde ligne affiche la valeur de l'unité exprimée dans l'une des quantités de base du système international, laquelle apparaît sur la troisième ligne. Cet affichage montre donc que are est abrégée a et vaut 100 mètres carrés.

...UNITS

Le menu UNITS est affiché sur la dernière ligne. Les cinq touches de menu sont les suivantes :

Touche de menu	Description
NEXT	Fait avancer le catalogue à l'unité suivante.
PREV	Fait passer le catalogue à l'unité précédente.
SCAN	Fait défiler les unités du catalogue par ordre alphabétique, en montrant brièvement le nom de chacune. Une pression sur SCAN transforme ce libellé en STOP ; une pression sur STOP arrête le défilement sur l'unité en cours d'affichage. Le défilement s'arrête automatiquement à la dernière unité du catalogue (l'unité « 1 »).
FETCH	Quitte le catalogue et ajoute l'abréviation de l'unité en cours dans la ligne de commande, à l'emplacement du curseur (commence une nouvelle ligne de commande si aucune n'était affichée).
QUIT	Quitte le catalogue, en laissant inchangée la ligne de commande éventuellement en cours.

En plus des opérations disponibles dans le menu UNITS, vous pouvez :

- Appuyer sur n'importe quelle touche alphabétique pour placer l'affichage du catalogue sur la première unité commençant par cette lettre.
- Appuyer sur n'importe quelle touche non-alphabétique du clavier gauche pour faire passer le catalogue directement à « ° », la première unité non-alphabétique.
- Appuyer sur [1] pour amener le catalogue sur « 1 », la dernière unité non-alphabétique.
- Appuyer sur [ON] pour quitter le catalogue et effacer la ligne de commande.

Le tableau suivant donne toutes les unités du catalogue UNITS, avec une description de chacune d'entre elles.

Unités du HP-28C/S

Unité	Nom complet	Description	Valeur
a	Are	Superficie	100 m^2
A	Ampère	Courant électrique	1 A
acre	Acre	Superficie	4 046,87260987 m^2
arcmin	Minute d'arc	Angle plan	2,90888208666E−4
arcs	Seconde d'arc	Angle plan	4,8481368111E−6
atm	Atmosphère	Pression	101 325 Kg/m*s^2
au	Unité astronomique	Longueur	149 597 900 000 m
A°	Angström	Longueur	0,0000000001 m
b	Barn	Section efficace	1,E−28 m^2
bar	Bar	Pression	100 000 Kg/m*s^2
bbl	Baril, pétrole	Volume	0,158987294928 m^3
Bq	Becquerel	Activité	1 1/s
Btu	Btu du tableau international	Energie	1 055,05585262 Kg*m^2/s^2
bu	Boisseau US (Bushel)	Volume	0,03523907 m^3
C	Coulomb	Charge électrique	1 A*s
cal	Calorie du tableau international	Energie	4,1868 Kg*m^2/s^2
cd	Candela	Intensité lumineuse	1 cd
chain	Chain	Longueur	20,1168402337 m
Ci	Curie	Activité	3,7E10 1/s
ct	Carat	Masse	0,0002 Kg
cu	Tasse US (US cup)	Volume	2,365882365E−4 m^3
d	Jour (day)	Temps	86 400 s
dyn	Dyne	Force	0,00001 Kg*m/s^2
erg	Erg	Energie	0,0000001 Kg*m^2/s^2

...UNITS

Unités du HP-28C/S (suite)

Unité	Nom complet	Description	Valeur
eV	Electron volt	Energie	1,60219E−19 Kg*m^2/s^2
F	Farad	Capacitance	1 A^2*s^4/Kg*m^2
fath	Brasse (Fathom)	Longueur	1,82880365761 m
fbm	Board foot	Volume	0,002359737216 m^3
fc	Footcandle	Luminance	10,7639104167 cd/m^2
Fdy	Faraday	Charge électrique	96 487 A*s
fermi	Fermi	Longueur	1,E−15 m
flam	Footlambert	Luminance	3,42625909964 cd/m^2
ft	Pied international (foot)	Longueur	0,3048 m
ftUS	Pied topographique (foot)	Longueur	0,304800609601 m
g	Gramme	Masse	0,001 Kg
ga	Accélération de chute libre standard	Accélération	9,80665 m/s^2
gal	Gallon US	Volume	0,003785411784 m^3
galC	Gallon canadien	Volume	0,00454609 m^3
galUK	Gallon du Royaume Uni	Volume	0,004546092 m^3
gf	Gramme-force	Force	0,00980665 Kg*m/s^2
grad	Grade	Angle plan	1,57079632679E−2
grain	Grain	Masse	0,00006479891 Kg
Gy	Gray	Dose absorbée	1 m^2/s^2
h	Heure	Temps	3 600 s
H	Henry	Inductance	1 Kg*m^2/A^2*s^2
hp	Cheval-vapeur (Horsepower)	Puissance	745,699871582 Kg*m^2/s^3
Hz	Hertz	Fréquence	1 1/s

Unités du HP-28C/S (suite)

Unité	Nom complet	Description	Valeur
in	Pouce (inch)	Longueur	0,0254 m
inHg	Pouce de mercure	Pression	3 386,38815789 Kg/m*s^2
inH2O	Pouce d'eau	Pression	248,84 Kg/m*s^2
J	Joule	Energie	1 Kg*m^2/s^2
kip	Kilolivre-force	Force	4 448,22161526 Kg*m/s^2
knot	Nœeud	Vitesse	0,514444444444 m/s
kph	Kilomètre par heure	Vitesse	0,277777777778 m/s
l	Litre	Volume	0,001 m^3
lam	Lambert	Luminance	3 183,09886184 cd/m^2
lb	Livre-avoirdupois	Masse	0,45359237 Kg
lbf	Livre-force	Force	4,44822161526 Kg*m/s^2
lbt	Troy lb	Masse	0,3732417 Kg
lm	Lumen	Flux lumineux	1 cd
lx	Lux	Eclairement lumineux	1 cd/m^2
lyr	Année lumière (light year)	Longueur	9,46055E15 m
m	Mètre	Longueur	1 m
mho	Mho	Conductance électrique	1 A^2*5s^3/Kg*5m^2
mi	Mille terrestre internat.	Longueur	1 609,344 m
mil	Mil	Longueur	0,0000254 m
min	Minute	Temps	60 s
miUS	Mille terrestre US	Longueur	1 609,34721869 m
mmHg	Millimètre de mercure	Pression	133,322368421 Kg/m*s^2
mol	Mole	Quantité de matière	1 mol

...UNITS

Unités du HP-28C/S (suite)

Unité	Nom complet	Description	Valeur
mph	Mille terrestre par heure	Vitesse	0,44704 m/s
N	Newton	Force	1 Kg*m/s^2
nmi	Mille marin (nautical mile)	Longueur	1852 m
ohm	Ohm	Résistance électrique	1 Kg*m^2/A^2*s^3
oz	Once (ounce)	Masse	0,028349523125 Kg
ozfl	US fluid oz	Volume	2,95735295625E−5 m^3
ozt	Troy oz	Masse	0,031103475 Kg
ozUK	UK fluid oz	Volume	0,000028413075 m^3
P	Poise	Viscosité dynamique	0,1 Kg/m*s
Pa	Pascal	Pression	1 Kg/m*s^2
pc	Parsec	Longueur	3,08567818585E16 m
pdl	Poundal	Force	0,138254954376 Kg*m/s^2
ph	Phot	Luminance	10 000 cd/m^2
pk	Peck	Volume	0,0088097675 m^3
psi	Livres par pouces carrés (Pounds per square inch)	Pression	6 894,75729317 Kg/m*s^2
pt	Pinte	Volume	0,000473176473 m^3
qt	Quart	Volume	0,000946352946 m^3
r	Radian	Angle plan	1
R	Roentgen	Exposition aux rayonnements	0,000258 A*s/Kg
rad	Rad	Dose absorbée	0,01 m^2/s^2
rd	Rod	Longueur	5,02921005842 m
rem	Rem	Equivalent de dose	0,01 m^2/s^2
s	Seconde	Temps	1 s
S	Siemens	Conductance électrique	1 A^2*s^3/Kg*m^2
sb	Stilb	Luminance	10 000 cd/m^2

Unités du HP-28C/S (suite)

Unité	Nom complet	Description	Valeur
slug	Slug	Masse	14,5939029372 Kg
sr	Stéradian	Angle solide	1
st	Stère	Volume	1 m^3
St	Stokes	Viscosité cinématique	0,0001 m^2/s
Sv	Sievert	Equivalent de dose	1 m^2/s^2
t	Tonne métrique	Masse	1 000 Kg
T	Tesla	Induction magnétique	1 Kg/A*s^2
tbsp	Cuiller à soupe (tablespoon)	Volume	1,47867647813E−5 m^3
therm	Thermie européenne	Energie	105 506 000 Kg*m^2/s^2
ton	Short ton	Masse	907,18474 Kg
tonUK	Long ton	Masse	1 016,0469088 Kg
torr	Torr	Pression	133,322368421 Kg/m*s^2
tsp	Cuiller à café (teaspoon)	Volume	4,92892159375E−6 m^3
u	Masse atomique unifiée	Masse	1,66057E−27 Kg
V	Volt	Différence de potentiel électrique	1 Kg*m^2/A*s^3
W	Watt	Puissance	1 Kg*m^2/s^3
Wb	Weber	Flux d'induction magnétique	1 Kg*m^2/A*s^2
yd	Yard international	Longueur	0,9144 m
yr	Année (year)	Temps	31 536 000 s
°	Degré	Angle	1,74532925199E−2
°C	Degré Celsius	Température	1 °K
°F	Degré Fahrenheit	Température	0,555555555556 °K
°K	Degré Kelvin	Température	1 °K

...UNITS

Unités du HP-28C/S (suite)

Unité	Nom complet	Description	Valeur
°R	Degré Rankine	Température	0,555555555556 °K
μ	Micron	Longueur **ufcol4**0,000001 m	
?	Quantité définie par l'utilisateur		1 ?
1	Unité sans dimension		1

Sources : The National Bureau of Standards Special Publication 330, *The International System of Units (SI), Fourth Edition,* Washington D.C., 1981.

The Institute of Electrical and Electronics Engineers, Inc., *American National Standard Metric Practice ANSI/IEEE Std. 268-1982,* New York, 1982.

American Society for Testing and Materials, *ASTM Standard for Metric Practice E380-84,* Philadelphia, 1984.

Aerospace Industries Association of America, Inc., *National Aerospace Standard,* Washington D.C., 1977.

Handbook of Chemistry and Physics, 64th Edition, 1983-1984, CRC Press, Inc., Boca Raton, FL, 1983.

Unités définies par l'utilisateur

Vous pouvez créer une variable contenant une liste que CONVERT acceptera comme une *unité définie par l'utilisateur* dans une chaîne d'unités. La liste doit contenir un nombre réel et une chaîne d'unités (similaire aux seconde et troisième lignes de l'affichage de UNITS). Supposons par exemple que vous utilisez souvent la semaine comme unité de temps. Exécuter

```
{7 "d"} 'SEM' STO
```

vous permet d'utiliser "SEM" dans les conversions ou lorsque vous créez des unités plus complexes.

Les chaînes d'unités définies par l'utilisateur peuvent contenir n'importe quel élément d'une chaîne d'unités, plus deux autres unités spéciales :

- Pour définir une unité sans dimension, spécifier une chaîne "1".
- Pour définir une nouvelle unité qui ne peut pas être exprimée en unités du SI (système international), spécifiez une chaîne "?". CONVERT vérifie la compatibilité des dimensions de cette unité avec celles des unités du SI. Par exemple, pour convertir de l'argent en trois monnaies différentes (dollars, livres et francs), définissez :

```
{1 "?"} 'DOLLAR' STO
{2,25 "?"} 'LIVRE' STO
{0,4 "?"} 'FRANC' STO
```

  puis, pour une somme dans l'une de ces monnaies, calculez l'équivalent dans les deux autres monnaies (les valeurs choisies n'ont été prises qu'à titre d'exemple).

Préfixes d'unités

Dans une chaîne d'unités, vous pouvez faire précéder une unité intégrée d'un préfixe indiquant une puissance de dix. Par exemple, "mm" indique des "millimètres", c'est-à-dire des mètres $\times 10^{-3}$. Le tableau ci-après dresse la liste des préfixes reconnus par CONVERT.

...UNITS

Préfixes d'unités

Préfixe	Nom	Exposant
E	exa	+18
P	peta	+15
T	téra	+12
G	giga	+9
M	méga	+6
k ou K	kilo	+3
h ou H	hecto	+2
D	déca	+1
d	déci	−1
c	centi	−2
m	milli	−3
μ	micro	−6
n	nano	−9
p	pico	−12
f	femto	−15
a	atto	−18

La plupart des préfixes utilisés par le HP-28C/S correspondent aux notations standard du SI (système international). La seule exception est « déca », qui indique un exposant +1, et qui est représenté par « D » sur le HP-28C/S et par « da » dans le SI.

Remarque Vous ne pouvez pas utiliser un préfixe avec une unité si la combinaison qui en résulte coïncide avec une unité intégrée. Par exemple, vous ne pouvez pas utiliser `"min"` pour indiquer des milli-pouces car `"min"` est une unité intégrée représentant les minutes. Les autres combinaisons possibles qui coïncideraient avec des unités intégrées sont les suivantes : `"Pa"`, `"cd"`, `"ph"`, `"flam"`, `"nmi"`, `"mph"`, `"kph"`, `"ct"`, `"pt"`, `"ft"`, `"au"` et `"cu"`.

Bien que vous ne puissiez pas utiliser un préfixe avec une unité que vous définissez vous-même, vous pouvez créer une nouvelle unité dont le nom inclut le caractère préfixe.

USER

ORDER	CLUSR	MEM

Le menu USER (utilisateur) contient, en plus des trois commandes « permanentes » ORDER, CLUSR et MEM, un libellé pour chacune des variables d'utilisateur. Celles-ci sont les premières à apparaître dans le menu, avec à gauche la variable la plus récemment créée. S'il existe plus de six variables d'utilisateur, [USER] active les six premières ; chaque pression sur la touche [NEXT] active un autre groupe de six variables. ORDER, CLUSR et MEM apparaissent lorsque vous appuyez sur [NEXT] pendant que le dernier groupe de variables d'utilisateur est affiché, ou lorsque vous appuyez sur ■[PREV] pendant que le premier groupe de variables est actif.

Une pression sur une touche du menu USER correspondant à une variable provoque l'évaluation du nom de la variable, en mode exécution, ou l'ajout de cette variable en ligne de commande, en mode de saisie alphabétique ou algébrique.

Les libellés apparaissant dans le menu sont dérivés des noms des variables d'utilisateur. Un libellé reprendra les caractères principaux composant le nom de la variable en fonction de ce qui pourra être affiché dans le libellé, avec un maximum de cinq caractères. Un nom de variable peut contenir des minuscules, mais celles-ci apparaîtront sous forme de majuscules dans le libellé de la touche de menu.

ORDER CLUSR MEM

ORDER *Réorganiser le menu* Commande

Niveau 1	
{ nom_1 nom_2 ... } ➡	

ORDER prend une liste de noms de variables et réarrange la mémoire allouée aux variables de façon qu'elles apparaissent dans le menu USER dans le même ordre que dans la liste. Les variables contenues dans la liste sont placées au début du menu utilisateur (USER). Les autres restent dans l'ordre où elles se trouvent et suivent les variables contenues dans la liste.

CLUSR *Effacer la mémoire utilisateur* Commande

➡

CLUSR (CLEAR USER) supprime toutes les variables en cours. Pour éviter une suppression accidentelle, une pression sur CLUSR exécute toujours ENTER et place `CLUSR` en ligne de commande. Vous pouvez alors appuyer sur [ENTER] pour exécuter CLUSR.

MEM *Mémoire* Commande

	Niveau 1
➡	*x*

MEM renvoie le nombre d'octets de mémoire inutilisés. Le nombre renvoyé par MEM ne devrait cependant être utilisé que comme un indication approximative de la mémoire disponible puisque la quantité de mémoire utilisée par le calculateur peut varier en fonction des opérations en cours d'exécution ou prêtes à être exécutées, qui ne sont pas toujours directement visibles. Les mécanismes de récupération, associés à ■[COMMAND], ■[UNDO] et ■[LAST], par exemple, peuvent consommer ou libérer de grandes quantités de mémoire lorsque des opérations sont exécutées.

A

Messages

Cette annexe fait la liste de tous les messages d'erreur et d'état qu'affiche le HP-28C/S. Les messages sont normalement affichés en ligne 1 de l'affichage et disparaissent dès la première frappe suivant leur apparition (les messages concernant l'algorithme de résolution Solver apparaissent en ligne 2).

Les messages désignés *messages d'état* sont pour votre information et n'indiquent aucunement une erreur. Les messages désignés *exceptions mathématiques* n'apparaîtront pas si l'indicateur binaire d'exception est désarmé (les exceptions mathématiques sont décrites sous « Principes généraux »).

Liste alphabétique des messages

Message	N° d'erreur		Signification
	Hex	Décimal	
Bad Argument Type	202	514	Une commande nécessitait un autre type d'argument.
Bad Argument Value	203	515	Une valeur de l'argument était en dehors des limites de fonctionnement d'une commande.
Bad Guess(es)	A01	2561	La (ou les) estimation(s) fournie(s) à l'algorithme de résolution Solver ou à ROOT a (ou ont) provoqué un résultat incorrect lors de l'évaluation de l'équation en cours.
Can't Edit CHR(0)	102	258	Une tentative de corriger une chaîne contenant le caractère 0 a été faite.

Liste alphabétique des messages (suite)

Message	N° d'erreur		Signification
	Hex	Décimal	
Circular Reference	129	297	Une tentative de stocker un objet dans une variable a été faite dans le menu Solver, alors que l'objet fait, directement ou indirectement, référence à la variable.
Command Stack Disabled	125	293	Vous avez appuyé sur les touches ■ COMMAND alors que COMMAND n'était pas active.
Constant?	A02	2562	L'équation en cours a renvoyé la même valeur à chaque point échantillonné par l'algorithme de résolution.
Constant Equation	Etat		L'équation en cours a renvoyé la même valeur pour chacun des points se trouvant dans la plage définie pour DRAW.
Extremum	Etat		Le résultat renvoyé par l'algorithme de résolution Solver est un extrême plutôt qu'une racine.
HALT not Allowed	121	289	DRAW ou Solver ont rencontré une commande HALT dans le programme EQ.
Improper User Function	103	259	Une tentative a été faite d'évaluer une fonction incorrecte définie par l'utilisateur. Aidez-vous de « Programmes » pour corriger la syntaxe.
Inconsistent Units	B02	2818	CONVERT a été exécutée avec des chaînes d'unités de dimensions différentes.

Liste alphabétique des messages (suite)

Message	N° d'erreur		Signification
	Hex	Décimal	
Infinite Result	305	773	Exception mathématique. Un calcul, tel que 1/0 ou LN(0), a renvoyé un résultat infini.
Insufficient Memory	001	001	Il n'y avait pas assez de mémoire disponible pour effectuer cette opération.
Insufficient Σ Data	603	1539	Une commande de statistiques a été exécutée alors que ΣDAT ne contenait pas assez d'observations pour permettre le calcul.
Interrupted	Etat		L'algorithme de résolution a été interrompu par la touche [ON].
Invalid Dimension	501	1281	Un tableau argument n'avait pas la dimension adéquate.
Invalid PPAR	11E	286	DRAW ou DRWΣ on rencontré une saisie incorrecte dans PPAR.
Invalid Unit String	B01	2817	CONVERT a été exécutée avec une unité de chaîne incorrecte.
Invalid ΣDAT	601	1537	Une commande de statistiques a été exécutée avec un objet incorrect stocké dans ΣDAT.
Invalid ΣPAR	604	1540	ΣPAR est un type d'objet incorrect ou contient une saisie incorrecte dans sa liste - ou une saisie manque dans cette liste.
LAST Disabled	205	517	LAST a été exécutée alors que l'indicateur 31 était désarmé.
Low Memory!	Etat		Indique qu'il reste moins de 128 octets de mémoire disponible.

Liste alphabétique des messages (suite)

Message	N° d'erreur		Signification
	Hex	Décimal	
Memory Lost	005	005	La mémoire du HP-28C/S vient d'être ré-initialisée.
Negative Underflow	302	770	Exception mathématique. Un calcul a renvoyé un résultat négatif, non nul, plus grand que −MINR.
No Current Equation	104	260	SOLVR ou DRAW ont été exécutées sans variable EQ.
Nonexistent ΣDAT	602	1538	Une commande de statistiques a été exécutée sans variable ΣDAT.
Non-real Result	11F	287	Une procédure a renvoyé un résultat différent d'un nombre réel qui était nécessaire pour l'algorithme de résolution, ROOT, DRAW ou ∫.
No Room for UNDO	101	257	Pas assez de mémoire disponible pour conserver une copie de la pile. UNDO est automatiquement inactivée.
No Room to ENTER	105	261	Pas assez de mémoire disponible pour traiter la ligne de commande.
No Room to Show Stack	Etat		Pas assez de mémoire disponible pour l'affichage normal de la pile.
nom$_1$ Not in Equation	Etat		DRAW a été exécutée alors que la variable indépendante *nom*$_1$ de PPAR n'existait pas dans l'équation en cours. Ce message est suivi par **Constant Equation** ou **Using** *nom*$_2$.

Liste alphabétique des messages (suite)

Message	N° d'erreur		Signification
	Hex	Décimal	
Out of Memory			Plus de mémoire disponible pour permettre le fonctionnement du calculateur. Il vous faut supprimer un ou plusieurs objets pour continuer.
Overflow	303	771	Exception mathématique. Un calcul a donné un résultat supérieur (en valeur absolue) à MAXR.
Positive Underflow	301	769	Exception mathématique. Un calcul a donné un résultat positif, différent de zéro, inférieur à MINR.
Sign Reversal	Etat		L'algorithme de résolution n'a pu trouver un point pour lequel l'équation en cours serait égale à zéro, mais il a trouvé deux points voisins provoquant le changement de signe de l'équation.
Syntax Error	106	262	Un objet en ligne de commande a été saisi de manière non valable.
Too Few Arguments	201	513	Une commande nécessite plus d'argumentrs qu'il n'y en a sur la pile.
Unable to Isolate	120	288	Le nom spécifié ne peut pas être isolé dans l'expression. Le nom était soit absent, soit contenu dans l'argument d'une fonction sans inverse.
Undefined Local Name	003	003	Tentative d'évaluation d'un nom local auquel ne correspond pas de variable locale.

Liste alphabétique des messages (suite)

Message	N° d'erreur		Signification
	Hex	Décimal	
Undefined Name	204	516	Tentative de rappeler la valeur d'une variable indéfinie (formelle).
Undefined Result	304	772	Une fonction a été exécutée avec des arguments qui conduisent à un résultat mathématiquement indéfini, tel que 0/0 ou LNP1(*x*) pour $x<-1$.
UNDO Disabled	124	292	Vous avez appuyé sur ■ UNDO alors que UNDO était inactif.
Using *nom*	Etat		DRAW a sélectionné la variable indépendante *nom*.
Wrong Argument Count	128	296	Une fonction définie par l'utilisateur a été évaluée dans une expression, avec un nombre d'arguments entre parenthèses incorrect.
Zero	Etat		L'algorithme de résolution Solver a trouvé une valeur de la variable inconnue pour laquelle l'équation en cours vaut 0.

Liste des messages d'erreur par numéro

Hex	Décimal	Message
Erreurs causées par des opérations générales		
001	001	Insufficient Memory
003	003	Undefined Local Name
005	005	Memory Lost
101	257	No Room for UNDO
102	258	Can't Edit CHR(0)
103	259	Improper User Function
104	260	No Current Equation
105	261	No Room to ENTER
106	262	Syntax Error
11E	286	Invalid PPAR
11F	287	Non-real Result
120	288	Unable to Isolate
121	289	HALT not Allowed
124	292	UNDO Disabled
125	293	Command Stack Disabled
128	296	Wrong Argument Count
129	297	Circular Reference
Erreurs causées par des opérations sur la pile opérationnelle		
201	513	Too Few Arguments
202	514	Bad Argument Type
203	515	Bad Argument Value
204	516	Undefined Name
205	517	LAST Disabled
Erreurs causées par des opérations sur les nombres réels		
301	769	Positive Underflow
302	770	Negative Underflow
303	771	Overflow
304	772	Undefined Result
305	773	Infinite Result

Liste des messages d'erreur par numéro (suite)

Hex	Décimal	Message
Erreurs causées par des opérations sur les tableaux		
501	1281	Invalid Dimension
Erreurs causées par des opérations de statistiques		
601	1537	Invalid ΣDAT
602	1538	Nonexistent ΣDAT
603	1539	Insufficient Σ Data
604	1540	Invalid ΣPAR
Erreurs causées par l'algorithme de résolution des équations		
A01	2561	Bad Guess(es)
A02	2562	Constant?
Erreurs causées par des conversions d'unités		
B01	2817	Invalid Unit String
B02	2818	Inconsistent Units

B

Si vous connaissez déjà la notation polonaise inverse

En commençant avec le HP-35, lancé en 1972, Hewlett-Packard a mis au point une série de calculateurs scientifiques et d'affaires utilisant la notation polonaise inverse comme interface avec la pile opérationnelle. Bien que les divers calculateurs présentent de grandes différences au niveau de leurs possibilités et de leurs applications, ils ont tous en commun l'interface de base avec la pile opérationnelle, ce qui facilite aux utilisateurs de calculateurs Hewlett-Packard le passage d'une machine à l'autre.

Le HP-28C/S utilise également, pour son interface utilisateur, une pile opérationnelle et la notation polonaise inverse. Toutefois, la structure des calculateurs précédents, qui offrait une pile à quatre niveaux et des registres fixes, n'était pas appropriée aux possibilités de résolution symbolique et de traitement d'objets multiples du HP-28C/S. En conséquence, bien que le HP-28C/S soit une évolution naturelle de l'interface « initiale » à notation polonaise inverse, il présente suffisamment de différences avec ses prédécesseurs pour que cela mérite une brève présentation, à l'usage de ceux qui connaissent déjà la notation polonaise inverse telle qu'elle existe sur les calculateurs Hewlett-Packard précédents.

La pile opérationnelle dynamique

La différence la plus spectaculaire entre l'interface de base du HP-28C/S et celle des calculateurs HP précédents est la taille de la pile opérationnelle. Les autres calculateurs offrent une pile fixe, à quatre niveaux, composée des registres X, Y, Z et T, plus un seul registre LASTX ou L. Cette pile est toujours « pleine », même lorsque vous
« l'effacez » (ce qui consiste en fait à la remplir de zéros).

Le HP-28C/S, lui, n'a pas une pile de taille fixe. A mesure que vous saisissez de nouveaux objets dans la pile, de nouveaux niveaux sont dynamiquement créés. Lorsque vous supprimez des objets de la pile, la taille de cette dernière diminue, et elle peut même être vide si vous supprimez tous les objets. C'est ainsi que le HP-28C/S peut générer le message d'erreur `Too Few Arguments` (pas assez d'arguments), qui ne pouvait pas apparaître sur les autres calculateurs HP.

Cette version dynamique de la pile est la cause des différences suivantes par rapport aux calculateurs HP à pile fixe :

Niveaux numérotés. La taille infinie de la pile du HP-28C/S rend inadéquats les noms X Y Z T désignant précédemment les différents niveaux de la pile ; les niveaux de la pile du HP-28C/S sont donc numérotés. Le niveau 1 est analogue à l'ancien registre X, le niveau 2 au registre Y, le niveau 3 au registre Z et le niveau 4 au registre T. Les libellés des touches $1/x$ et x^2 ont été conservés sur le HP-28C/S parce qu'ils sont plus aisément visibles que les noms (respectivement INV et SQ) des commandes qu'ils représentent. Mais l'ancienne fonction X<>Y a été rebaptisée SWAP (permuter) sur le HP-28C/S.

Manipulation de la pile. Le HP-28C/S nécessite un jeu de commanndes de manipulation de la pile plus général que les calculateurs à pile fixe. R↑ et R↓, par exemple, sont remplacés par ROLL et ROLLD, respectivement, qui requièrent chacun un argument supplémentaire pour spécifier le nombre de niveaux à déplacer. Le menu STACK (pile opérationnelle) contient plusieurs commandes de manipulation qui n'existent pas sur les calculateurs à pile fixe.

Pas de duplication automatique du registre T. Sur les calculateurs à pile fixe, le contenu du registre T est dupliqué dans le registre Z chaque fois que la pile « descend » (c'est-à-dire lorsqu'un nombre est supprimé de la pile). Ceci fournit un moyen pratique de multiplication par une constante — vous pouvez remplir la pile de copies d'une constante, puis les multiplier par une série de nombres en saisissant chaque nombre, en appuyant sur [×], puis en appuyant sur [CLx] après avoir consigné chaque résultat par écrit. Vous ne pouvez pas faire cela avec le HP-28C/S — mais il est facile de créer un programme de la forme

```
« 12345 * » 'MULT' STO
```

où 12345 représente une constante typique. Il vous suffit ensuite d'appuyer sur [USER], de saisir un nombre et d'appuyer sur [MULT], puis de saisir un nouveau nombre et d'appuyer à nouveau sur [MULT], et ainsi de suite, pour efffectuer une multiplication par une constante. Vous pouvez laisser les résultats successifs dans la pile.

Mémoire affectée à la pile opérationnelle. Une pile opérationnelle dynamique a l'avantage de vous permettre d'utiliser autant de niveaux que vous en avez besoin, sans craindre de perdre des objets « par le haut de la pile » lorsque vous en saisissez de nouveaux « dans le bas ». Mais cela a aussi l'inconvénient « d'immobiliser » une quantité importante de mémoire avec de « vieux » objets si vous les laissez dans la pile une fois vos calculs terminés. Avec le HP-28C/S, vous devez prendre l'habitude de supprimer les objets devenus inutiles.

DROP par rapport à CLX. Dans les calculateurs à pile fixe, CLX signifie « remplacer le contenu du registre X par 0, et invalider le décalage des nombres vers le haut de la pile » (voir ci-dessous). Le but de cette opération est essentiellement de supprimer un nombre devenu inutile, avant de le remplacer par un nouveau — mais vous pouvez aussi l'utiliser pour saisir un 0. Sur le HP-28C/S, CLX est remplacée par DROP, qui supprime l'objet se trouvant au niveau 1 de la pile, et entraîne un décalage d'un niveau vers le bas de tous les autres objets. Aucun 0 n'est introduit dans la pile. De même, CLEAR supprime tous les objets de la pile au lieu de les remplacer par des zéros comme le fait CLST (CLEAR STACK) sur les calculateurs à pile fixe.

Invalidation du décalage vers le haut de pile, et ENTER

Sur les calculateurs à pile fixe, certaines commandes (ENTER↑, CLX, $\Sigma+$, $\Sigma-$) présentent la particularité d'invalider les mouvements de *décalage vers le haut de la pile*. C'est-à-dire que, après l'exécution d'une de ces commandes, le nombre suivant saisi dans la pile remplace le contenu du registre X au lieu de repousser ce dernier dans le registre Y. Cette particularité est absente du HP-28C/S. Les nouveaux objets saisis repoussent *toujours* les objets précédents d'un niveau vers le haut.

Le registre X et l'opération ENTER des calculateurs à pile fixe jouent un double rôle, qui est rendu nécessaire plus par l'affichage à une seule ligne de ces calculateurs que par la structure de la pile elle-même. Le registre X joue le rôle de registre de saisie *et* de registre ordinaire de la pile — lorsque vous saisissez un nombre, les chiffres sont créés dans le registre X, jusqu'à ce qu'une pression sur une touche non numérique mette fin à la saisie. La touche [ENTER↑] sert à séparer deux saisies numériques consécutives. Mais elle sert *également* à copier le contenu du registre X dans le registre Y et à invalider les mouvements de décalage vers le haut de la pile.

Sur le HP-28C/S, chacun de ces doubles rôles est distinct — il n'existe pas d'invalidation des mouvements de décalage vers le haut. Une ligne de commande totalement distincte du niveau 1 (l'équivalent du registre X) est utilisée pour la saisie des commandes. ENTER sert *uniquement* à traiter le contenu de la ligne de commande — elle ne duplique pas le contenu du niveau 1. Notez cependant que la touche [ENTER] exécute DUP (qui copie le niveau 1 dans le niveau 2) s'il n'y a aucune ligne de commande. Cette caractéristique de [ENTER] est destinée en partie à donner au HP-28C/S une similarité de fonctionnement avec les calculateurs précédents.

Préfixe par rapport à postfixe

Les commandes du HP-28C/S obéissent strictement à une syntaxe postfixe. C'est-à-dire que toutes les commandes utilisant des arguments nécessitent que ces arguments soient présents dans la pile avant que la commande ne soit exécutée. Ceci diffère de la convention utilisée sur les calculateurs HP précédents, sur lesquels les arguments définissant un numéro de registre, d'indicateur binaire, etc. ne sont pas saisis dans la pile mais *après* la commande elle-même ; par exemple STO 25, TONE 1, CF 03, et ainsi de suite. Ceci a l'avantage d'économiser un niveau de la pile, mais l'inconvénient de nécessiter un format immuable — STO sur le HP-41, par exemple, doit toujours être suivie d'un numéro de registre à deux chiffres.

Sur le HP-28C/S, les opérations similaires sont d'un style plus proche des opérations *indirectes* des calculateurs à pile fixe, où vous pouvez utiliser un *registre i* (ou n'importe quel registre dans le cas du HP-41) pour spécifier le numéro du registre, de l'indicateur binaire, etc., dans une commande. Vous pouvez considérer les opérations STO, RCL, etc. du HP-28C/S comme utilisant le niveau 1 en tant que *registre i*. RCL, par exemple, signifie « rappeler le contenu de la variable (registre) nommée au niveau 1 », et équivaut à RCL IND X sur le HP-41.

N'oubliez pas enfin que la plupart des commandes du HP-28C/S enlèvent leurs arguments de la pile. Si vous exécutez 123 « X » STO, par exemple, le 123 et le « X » disparaissent de la pile. Sans cette caractéristique, la pile serait surchargée de « vieux » arguments. Si vous voulez garder le 123 dans la pile, vous devez exécuter 123 DUP « X » STO.

Registres par rapport à variables

Les calculateurs à pile fixe ne peuvent traiter efficacement que des nombres réels en virgule flottante, pour lesquels la structure fixe à sept octets des registres de la pile et la mémoire à registres numérotés est appropriée (le HP-41 a introduit un objet alphabétique primitif qui respecte le format sur sept octets). Le HP-28C/S remplace les registres numérotés par des variables portant un nom. Outre le fait qu'elles ont une structure plus souple et peuvent recevoir des types d'objets plus divers, les variables ont des noms qui peuvent vous aider à mémoriser leur contenu plus facilement.

Si vous voulez reproduire la structure à registres numérotés sur le HP-28C/S, vous pouvez utiliser un vecteur :

```
« 50 » 0 CON 'REG' STO
```

crée un vecteur à 50 composantes initialisé à 0 ;

```
« 1 →LIST 'REG' SWAP GET » 'NRCL' STO
```

crée un programme NRCL qui rappelle la *n*ième composante du vecteur, *n* étant un nombre se trouvant au niveau 1 ;

```
« 1 →LIST 'REG' SWAP ROT PUT » 'NSTO' STO
```

crée un programme analogue NSTO pour le stockage de la même composante.

LASTX par rapport à LAST

Sur les calculateurs à pile fixe, la commande LASTX donne le contenu du registre LASTX (ou L), qui contient la dernière valeur utilisée dans le registre X. Ce concept est généralisé sur le HP-28C/S par la commande LAST, qui donne le dernier (ou les deux ou trois derniers) argument(s) pris dans la pile par une commande (aucune commande n'utilise plus de trois arguments). Ainsi 1 2 + LASTX renvoie 3 et 2 dans la pile d'un calculateur à pile fixe, mais 1 2 + LAST renvoie 3, 1 et 2 dans la pile du HP-28C/S.

Bien que la commande LAST du HP-28C/S soit plus souple que la commande LASTX des calculateurs précédents, vous ne devez pas oublier qu'un plus grand nombre de commandes prennent leurs arguments dans la pile sur le HP-28C/S que sur les calculateurs à pile fixe. Cela signifie que les arguments de LAST sont mis à jour plus fréquemment, et que même des commandes comme DROP et ROLL remplacent les arguments stockés dans LAST.

N'oubliez pas non plus que UNDO peut remplacer la totalité de la pile, ce qui peut être préférable à LAST pour la simple récupération de données après erreur.

C

Si vous ne connaissez pas la notation polonaise inverse

Bien des calculateurs, y compris la vaste majorité des calculateurs simples, dits « à quatre fonctions », utilisent un interface *algébrique*. C'est-à-dire que la séquence des opérations effectuées est identique à celle des calculs effectués « sur papier ». Par exemple, pour effectuer l'addition 1 + 2 + 3, on utilise la séquence de frappe [1] [+] [2] [−] [3] [=].

Cette interface fonctionne de façon satisfaisante lorsqu'il s'agit d'expressions contenant des nombres et des *opérateurs* - des fonctions comme +, −, × et /, écrites en notation infixe entre leurs arguments. Certains calculateurs plus sophistiqués permettent d'introduire, lors de la saisie, des parenthèses indiquant l'ordre dans lequel les opérations doivent s'effectuer. Toutefois, l'introduction de fonctions préfixes, telles SIN, LOG etc., mènent à deux variations distinctes :

- Les calculateurs ordinaires algébriques utilisent une combinaison de styles - les opérations infixes demeurent infixes, mais les fonctions préfixes sont introduites en mode postfixe (comme dans les calculateurs à *notation polonaise inverse*). Par exemple 1 + SIN(23) est saisi de cette manière [1] [+] [2] [3] [SIN] [=]. Cette façon de procéder a l'avantage de montrer des résultats intermédiaires et de préserver l'évaluation en une seule touche de fonctions préfixes (sans parenthèses), mais le désavantage de perdre la correspondance avec la notation mathématique classique, qui est le principal avantage de l'interface algébrique.

- Les calculateurs à « saisie directe d'équations » et les ordinateurs BASIC utilisant un mode d'exécution immédiate permettent de saisir une expression sous sa forme algébrique ordinaire, puis de calculer le résultat dès l'appui sur une touche de terminaison (qu'elle s'appelle [ENTER], [ENVOI] ou [RETOUR] etc). Cette approche a le mérite de conserver la correspondance entre les expressions telles qu'elles sont écrites sur papier et leur saisie sur le clavier, mais a en général le désavantage de ne pas fournir de résultats intermédiaires (le mode CALC du HP-71B constitue une exception). Il est nécessaire de connaître la forme complète d'une expression avant de commencer à la saisir - il est difficile dans ces conditions de s'acheminer vers une solution en modifiant les calculs en fonction des résultats intermédiaires.

S'habituer au HP-28C/S

La logique de fonctionnement du HP-28C/S est fondée sur une logique mathématique connue sous le nom de *« notation polonaise inverse »* développée par le logicien polonais Jean Łukasiweicz (1878 - 1956). La notation algébrique conventionnelle place les opérateurs mathématiques *entre* les nombres ou variables lors de l'évaluation d'expressions algébriques. La notation de Łukasiweicz spécifie les opérateurs *avant* les variables. Une variante de cette logique spécifie les opérateurs *après* les variables - ce qui s'appelle alors *« notation polonaise inverse »*, en anglais *Reverse Polish Notation*, ou RPN.

L'idée fondamentale de la notation polonaise inverse est que les nombres et les autres objets sont saisis d'abord, et après eux une commande qui agit sur ces nombres ou objets (les « arguments »). La « pile opérationnelle » est une suite de ces objets qui attendent d'être utilisés. La plupart des commandes renvoient leur résultat dans la pile opérationnelle, où ils peuvent à leur tour être utilisés comme arguments pour d'autres opérations.

Le HP-28C/S utilise cette logique polonaise inverse parce qu'elle présente la souplesse nécessaire pour permettre l'exécution uniforme des calculs permis par le HP-28C/S. Toutes les opérations, même celles qui ne peuvent pas être exprimées sous forme algébrique, sont exécutées de la même manière - les arguments sont pris dans la pile opérationnelle, les résultats y sont renvoyés.

Ceci dit, l'utilisation de la notation polonaise inverse est sans doute l'obstacle le plus grand que rencontrent les utilisateurs de calculateurs algébriques classiques lorsqu'ils abordent des calculateurs plus sophistiqués. La notation polonaise inverse est très efficace, mais demande de ré-arranger mentalement une expression avant de pouvoir calculer ses résultats. Mais la capacité que possède le HP-28C/S d'interpréter des expressions mathématiques sans traduction devrait rendre la transition plus simple que dans le cas des calculateurs à notation polonaise inverse qui existaient jusqu'à présent. Les quatre lignes de l'affichage constituent une aide visuelle qui ôte beaucoup de son mystère à la « pile opérationnelle » en en montrant en permanence le contenu, quatre niveaux à la fois.

Pour ce qui concerne l'évaluation d'expressions algébriques, le HP-28C/S est essentiellement un calculateur à « saisie directe des équations ». Pour évaluer une expression algébrique, il suffit de la faire précéder d'un signe ['] et de la saisir sous sa forme algébrique, y compris opérateurs infixes, fonctions préfixes et postfixes et parenthèses, puis d'appuyer sur la touche [EVAL] pour lire le résultat. Cette méthode peut même être utilisée pour l'arithmétique simple :

['] [1] [+] [2] [−] [3] [EVAL] donne 0.

Sauf pour les signes ['], ce sont les mêmes séquences de frappe que vous utilisieriez sur un calculateur algébrique simple où [EVAL] serait remplacé par la touche [=].

Remarque Ne faites pas l'erreur de confondre la touche [=] du HP-28C/S avec celle des calculateurs algébriques ; sur le HP-28C/S, [=] est utilisée dans le seul but de créer des équations algébriques (lisez à ce sujet « ALGEBRA »).

Lorsque vous utilisez le HP-28C/S comme « calculateur à saisie directe d'équations », chaque résultat calculé est retenu sur la pile opérationnelle, qui conserve « l'historique » de vos calculs. Ceci permet de stocker d'anciens résultats indéfiniment pour les ré-utiliser plus tard. Cela permet aussi de fractionner de longs calculs en une série de calculs plus petits, conservant chaque résultat partiel dans la pile et combinant les résultats au fur et à mesure de leur apparition (portée à l'extrême, cette méthode est l'essence même de l'arithmétique en notation polonaise inverse). La pile constitue un historique bien plus complet que la simple fonction « résultat » des calculateurs algébriques ou BASIC.

Un des avantages essentiels du HP-28C/S est que vous n'avez pas à vous préoccuper de savoir si la notation polonaise inverse est meilleure ou pire que la logique algébrique. Vous pouvez choisir le système qui est le plus adapté à vos besoins et mélanger expressions algébriques classiques et manipulations en notation polonaise inverse.

Glossaire

affinement itératif : un processus d'approximations successives dans la recherche de la solution de systèmes d'équations.

algébrique : voir « Objet », « Syntaxe », « Mode ».

algorithme de résolution : le système du HP-28C/S qui construit un menu des variables à partir de la définition de l'équation en cours, vous permettant ainsi de stocker des valeurs dans ces variables et de résoudre l'équation en calculant l'une des variables.

analyser : convertir une chaîne de caractères en un programme constitué par la série d'objets définis par la chaîne. S'applique généralement à l'action de ENTER sur la ligne de commande.

argument : un objet pris dans la pile comme donnée pour une opération.

arithmétique directe : effectuer des opérations arithmétiques sur le contenu de variables, sans rappeler ce contenu dans la pile.

armer : affecter la valeur *vrai*, c'est-à-dire une valeur non nulle, à un indicateur binaire.

arrêt système : une initialisation qui cause l'arrêt de toutes les opérations en cours et l'effacement de la pile opérationnelle.

associativité, appliquer l'— : modifier l'ordre dans lequel deux fonctions sont appliquées à trois arguments, sans changer la valeur d'une expression — par exemple, (a + b) + c est réorganisé en a + (b + c) (en notation polonaise inverse, a b + c + est réorganisé en a b c + +).

base : la base (ou système) numérique dans laquelle les entiers binaires sont affichés. Les choix sont : binaire (base 2), octal (base 8), décimal (base 10) et hexadécimal (base 16).

chaîne : un objet contenant une séquence de caractères (lettres, nombres ou autres symboles) et délimité par des guillemets ".

chaîne d'unités : une chaîne qui représente les unités physiques associées à une valeur exprimée par un nombre réel. Une chaîne d'unités peut contenir des noms d'unités, des puissances, des produits, et un rapport.

clause: une séquence de programme entre deux commandes de programme, comme IF *clause-test* THEN *clause-alors* END, par exemple.

commande : toute opération du HP-28C/S qui peut être incluse dans la définition d'une procédure ou dont le nom peut être inclus dans la ligne de commande.

constante symbolique : n'importe lequel des cinq objets e, i, π, MAXR, et MINR, dont l'évaluation donne la valeur numérique ou qui garde sa forme symbolique, selon l'état des indicateurs binaires 35 et 36.

conversion d'unités : une multiplication d'un nombre réel par un facteur de conversion déterminé par les valeurs des deux chaînes d'unités représentant « l'ancienne » unité et la « nouvelle ».

couple de coordonnées : un nombre complexe servant à représenter les coordonnées d'un point dans un espace à deux dimensions. La partie réelle est la coordonnée « horizontale » et la partie imaginaire la coordonnée « verticale ».

curseur : un caractère d'affichage qui met en valeur une position sur l'affichage. (1) Le curseur de la ligne de commande indique là où le prochain caractère sera introduit dans la ligne de commande. Son aspect varie pour refléter les différents modes de saisie. (2) Le curseur de FORM met en vidéo inverse la sous-expression choisie. (3) Le curseur de DRAW/DRWΣ est un petit réticule indiquant la position d'un point à numériser.

délimiteur : un caractère qui définit le début ou la fin d'un objet sur l'affichage ou dans la ligne de commande : ', ", #, [,], { , }, (,), «, ou ».

dépassement de la limite inférieure de capacité : une exception mathématique générée par un calcul donnant un résultat non nul trop petit pour être représenté par un nombre en virgule flottante.

dépassement de la limite supérieure de capacité : une exception mathématique générée par un calcul qui donne un résultat trop grand pour être représenté par un nombre en virgule flottante.

désarmer : affecter la valeur 0 à un indicateur binaire d'utilisateur (CF).

diagramme de la pile : un tableau résumant les arguments et résultats d'une commande, et montrant leur nature et position dans la pile opérationnelle.

distributivité, appliquer la — : appliquer une fonction aux arguments de l'opérateur + avant d'effectuer l'addition : a × (b + c) donne (a × b) + (a × c) après application de la distributivité.

domaine: la plage des valeurs d'un argument sur laquelle une fonction est définie.

donnée : un type d'objet qui, lorsqu'il est évalué, se renvoie lui-même dans la pile. Les nombres réels et complexes, les tableaux, les chaînes de caractères, les entiers binaires et les listes sont des données.

effacer : (1) vider la pile (CLEAR). (2) Effacer l'affichage (CLLCD).

entier arbitraire : une variable *n1*, *n2* etc. qui apparaît dans la solution d'une expression à racines multiples. On obtient des racines différentes en stockant des entiers réels dans les variables.

entier binaire : un objet identifié par le délimiteur # et qui représente un nombre entier ayant de 1 à 64 chiffres binaires ; il est affiché sous une forme qui dépend de la *base* en cours.

entier réel : un nombre réel servant d'argument à une commande opérant sur des valeurs entières.

équation : un objet algébrique consistant en deux expressions reliées par un signe égale (=).

équation en cours : la procédure stockée dans la variable EQ et servant d'argument implicite à DRAW et à l'algorithme de résolution d'équations.

erreur : n'importe quelle erreur d'exécution, causée par une erreur mathématique, des arguments inappropriés, une quantité insuffisante de mémoire, et ainsi de suite, et qui interrompt une exécution normale tout en affichant un message d'erreur.

estimation initiale : un ou plusieurs nombres fournis à l'algorithme de résolution des équations pour définir la région dans laquelle il doit rechercher une racine.

évaluation : c'est l'opération fondamentale du calculateur. (1) L'évaluation d'une donnée renvoie dans la pile cet objet lui-même. (2) L'évaluation d'un nom renvoie dans la pile l'objet stocké dans la variable correspondant à ce nom et l'évalue, si cet objet est un nom ou un programme. (3) L'évaluation d'une procédure renvoie dans la pile chaque objet contenu dans la procédure et l'évalue s'il s'agit d'une commande ou d'un nom sans guillemets.

exception : un type spécial d'erreur mathématique pour laquelle vous pouvez choisir, à l'aide d'un indicateur binaire d'utilisateur, si le calculateur renvoie un résultat par défaut ou s'il interrompt le calcul en cours et affiche un message d'erreur.

exécuter : évaluer une procédure ou une partie d'une procédure, y compris les opérations du HP-28C/S, qui sont des procédures stockées en mémoire morte (ROM).

exposant : la puissance de 10 comprise dans la représentation en notation exponentielle d'un nombre à virgule flottante ; spécifiquement, le nombre à un, deux ou trois chiffres, et avec signe, suivant le « E » dans l'affichage d'un nombre. L'exposant de x est IP (LOG (x)).

expression : un objet algébrique qui ne contient pas le signe égale (=).

faux : une des deux valeurs (ou « états ») d'un indicateur binaire, représentée par le nombre réel 0.

fonction : une fonction du HP-28C/S qui peut être incluse dans la définition d'un objet algébrique. Certaines fonctions peuvent nécessiter jusqu'à trois arguments, mais elles donnent toutes un seul résultat.

fonction analytique : une fonction qui peut être dérivée ou dont on peut calculer l'argument.

format HMS : un format de représentation d'un nombre réel dans lequel les chiffres à gauche du séparateur décimal représentent un nombre entier d'heures (ou de degrés), les deux premiers chiffres à droite du séparateur décimal représentent des minutes (d'arc ou de temps), les deux chiffres suivants un nombre entier de secondes, et les chiffres restant des fractions de secondes.

forme quadratique : un polynôme du second degré dans une variable donnée.

hiérarchie : la structure d'une expression mathématique, qui peut être organisée en une série de sous-expressions de différents niveaux dont chacune peut être l'argument d'une fonction.

inconnue : la variable pour laquelle l'algorithme de résolution des équations, ROOT, QUAD ou ISOL, essaie de trouver un racine symbolique ou numérique.

indicateur binaire : un nombre réel utilisé comme indicateur pour déterminer une décision du type vrai/faux. Le nombre 0 représente l'état *faux* et tout autre nombre, généralement +1, l'état *vrai*.

indicateur binaire d'utilisateur : un emplacement mémoire d'un bit, dont la valeur peut être fixée à 0 ou à 1 et contrôlée. Le HP-28C/S contient 64 indicateurs binaires d'utilisateur, numérotés de 1 à 64.

indicateur binaire système : un indicateur binaire interne qui détermine si l'affichage normal de la pile opérationnelle apparaît lorsque l'exécution de toutes les opérations en cours est terminée. L'indicateur binaire système est armé par des erreurs et des commandes qui génèrent des affichages spéciaux.

interface avec l'utilisateur : les procédures, séquences de touches, affichages, etc., par lesquels un utilisateur a une interaction avec un calculateur.

inverse : (1) l'inverse d'un nombre ou d'un tableau. (2) Une fonction qui, lorsqu'elle est appliquée à une seconde fonction, donne l'argument de la seconde fonction. Ainsi, SIN est l'inverse de ASIN.

ligne de commande : la chaîne de saisie contenant des caractères, nombres, objets, commandes etc. qui ne sont pas destinés à une exécution immédiate et qui sont saisis à partir du clavier. ENTER convertit la chaîne de la ligne de commande en un programme et en entraîne l'évaluation.

liste : un objet consistant en une collection d'autres objets.

mantisse : la portion d'un nombre représentée par la partie décimale de son logarithme. Spécifiquement, la partie du nombre à gauche du « E » lorsque ce nombre est affiché en notation exponentielle.

matrice : un tableau à deux dimensions.

matrice statistique en cours : la matrice stockée dans la variable ΣDAT et contenant les données statistiques accumulées à l'aide de Σ+.

mémoire tampon pour touches : un emplacement de la mémoire qui peut contenir jusqu'à 15 codes de touches, qui représentent les touches qui ont été actionnées mais pas encore traitées.

mémoire utilisateur : la zone de la mémoire dans laquelle sont stockées les variables utilisateur.

menu : un ensemble d'opérations ayant des caractéristiques communes et qui sont affectées, six par six, aux touches de menu.

menu des variables : le menu créé par l'algorithme de résolution des équations, dans lequel chaque variable contenue dans l'équation en cours est représentée par une touche de menu.

message d'état : un message affiché par le calculateur pour vous informer d'un état qui n'est pas une erreur.

mode de saisie : le mode de fonctionnement du calculateur qui détermine si une pression sur les touches entraîne l'exécution immédiate des commandes ou introduit simplement leur nom dans la ligne de commande. Le mode de saisie peut être le mode exécution, le mode algébrique, ou le mode alphabétique.

mode de saisie algébrique : le mode de saisie dans lequel une touche correspondant à une *fonction* ajoute, lorsqu'elle est actionnée, le nom de la fonction et une ouverture de parenthèses (le cas échéant) dans la ligne de commande. Les touches correspondant à d'autres commandes exécutent leur commande immédiatement.

mode de saisie alphabétique : le mode de saisie dans lequel toutes les touches correspondant à des commandes ajoutent, lorsqu'elles sont actionnées, le nom de leur commande dans la ligne de commande.

mode numérique : un mode dans lequel l'évaluation des fonctions entraîne l'évaluation répétée de leurs arguments jusqu'à ce que ces arguments renvoient des nombres dans la pile.

mode symbolique : le mode du calculateur dans lequel les fonctions d'arguments symboliques donnent des résultats symboliques.

niveau : une position dans la pile opérationnelle, pouvant contenir un objet.

nom : un objet consistant en une séquence de caractères représentant le nom d'une variable. (1) L'évaluation d'un nom renvoie dans la pile l'objet stocké dans la variable correspondante et l'évalue si cet objet est un nom ou un programme. (2) L'évaluation d'un nom local renvoie dans la pile l'objet stocké dans la variable locale correspondante.

nom local : un nom désignant une variable locale. Les noms locaux constituent un type d'objet différent (type 7) des noms ordinaires (type 6). L'évaluation d'un nom local renvoie dans la pile le contenu *non-évalué* de la variable locale correspondante.

nombre complexe : un objet délimité par les symboles ( ) et consistant en deux nombres réels représentant les parties réelle et imaginaire d'un nombre complexe.

nombre réel : un objet consistant en seul nombre réel en virgule flottante, affiché en base 10.

notation exponentielle : une représentation d'un nombre sous la forme d'un signe, d'une mantisse comprise entre 1 et 9,99999999999, et d'un exposant « E » suivi d'un entier à trois chiffres avec un signe.

notation polonaise : une notation mathématique dans laquelle toutes les fonctions et opérateurs sont écrits sous la forme préfixe. Par exemple, « 1 plus 2 » s'écrit sous la forme « +(1, 2) ».

notation polonaise inverse : une modification de la notation polonaise dans laquelle les fonctions suivent leurs arguments : 1 2 + signifie 1 plus 2. Cette notation mathématique est celle de l'interface du calculateur, dans laquelle les fonctions prennent leurs arguments dans une pile et renvoient leurs résultats dans une pile.

nuage de points : un tracé d'observations provenant de la matrice statistique, généré par DRWΣ.

numéro d'ordre : un classement des éléments d'un tableau, de gauche à droite dans chaque ligne, et de la ligne supérieure vers la ligne inférieure.

objet : l'élément de base dans le fonctionnement du calculateur. Les différents types d'objets sont : les données, qui sont des quantités ayant une « valeur » simple ; les noms, qui servent à nommer les variables, lesquelles contiennent d'autres objets ; les procédures, qui sont des séquences d'objets et de commandes.

objet algébrique : une procédure, saisie et affichée entre des délimiteurs ' ', contenant nombres, variables, opérateurs et fonctions combinés selon la syntaxe algébrique pour représenter une équation ou une expression mathématique.

objet numérique : un nombre ou tableau réel ou complexe.

opérateur : une fonction qui est soumise à des règles spéciales de priorité lorsqu'elle est incluse dans une expression algébrique.

opération : toute caractéristique intégrée dans le HP-28C/S et disponible à l'utilisateur ; ce qui comprend les séquences de touches non programmables et les commandes programmables.

ordonnée à l'origine : la coordonnée verticale pour laquelle la droite déterminée par une régression linéaire coupe l'axe vertical (variable dépendante).

paramètres de traçage : le contenu de la variable PPAR, qui détermine la position et l'échelle d'un tracé, et le nom de la variable indépendante.

pas-à-pas, exécution — : exécuter une structure ou un objet compris dans la définition d'un programme.

pile : la série d'objets présentée sous la forme d'une liste « verticale » du type « dernier entré, premier sorti », et qui fournit une interface uniforme pour traiter les arguments et résultats des commandes.

pile des commandes : jusqu'à quatre lignes de commandes précédemment saisies par vous, et qui sont stockées pour réutilisation ultérieure par pression sur ■ COMMAND.

pile fixe, calculateur à — : un calculateur à notation polonaise inverse avec pile opérationnelle fixe, généralement à quatre niveaux.

pixel : le plus petit élément individuel de l'affichage à cristaux liquides.

priorité d'exécution : règles qui déterminent l'ordre d'exécution des opérateurs dans les expressions où l'omission de parenthèses entraînerait sinon des ambiguïtés.

procédure : une catégorie d'objet qui comprend des programmes et des équations et expressions algébriques, et dont l'évaluation signifie que chaque objet dont il est constitué est placé dans la pile et évalué s'il est une commande ou un nom sans guillemets.

programme : une procédure définie selon la notation polonaise inverse et identifiée par les délimiteurs « ».

programme suspendu : un programme dont l'exécution a été arrêtée par HALT et peut être relancée par pression sur SST ou sur ■ CONT.

précision : pour l'intégration numérique, la précision numérique de l'expression à intégrer, qui détermine les intervalles sur lesquels s'effectue le calcul de l'approximation de l'intégrale.

racine : une valeur d'une variable pour laquelle une expression a la valeur 0 ou pour laquelle une équation est satisfaite (les deux membres de l'équation ont la même valeur).

rang : la position d'une sous-expression dans la hiérarchie d'une expression algébrique.

réinitialisation mémoire : un effacement du système dans lequel tous les modes du calculateur et emplacements mémoire sont réinitialisés (remis à leurs valeurs ou contenus par défaut), ce qui implique l'effacement de la pile opérationnelle, de la pile des commandes, de la pile tampon de UNDO, des arguments de LAST, et de la mémoire affectée aux variables utilisateur.

résolution : dans un tracé, l'espacement des points de l'abscisse pour lesquels des valeurs sont calculées sur l'ordonnée. La résolution 1 signifie que ce calcul s'effectue pour chaque point, la résolution 2 qu'il s'effectue pour un point sur deux, et ainsi de suite.

résultat infini : une exception mathématique résultant d'une opération qui donnerait un résultat infini, comme une division par zéro par exemple.

saisie directe des formules, calculateur avec — : un calculateur sur lequel vous effectuez les calculs numériques en saisissant une formule complète sous forme mathématique ordinaire, sans obtenir de résultats intermédiaires.

séparateur décimal : le signe de ponctuation séparant les parties entière et fractionnaire d'un nombre décimal.

signe arbitraire : une variable *s1, s2* etc. qui apparaît dans la solution d'une expression à racines multiples. On obtient des racines différentes en stockant $+1$ ou -1 dans les variables.

simplification : ré-écrire une expression algébrique sous une forme qui préserve la valeur initiale de l'expression mais se présente d'une manière plus simple. La simplification peut impliquer la combinaison de termes ou l'évaluation partielle d'une expression.

singulière : Fait référence à une quantité mathématique dont l'évaluation donne 0 en un point donné ou qui a des dérivées égales à 0, et telle qu'elle ne peut pas être évaluée ou inversée sans donner un résultat infini. Une matrice singulière a un déterminant nul et ne peut donc pas être inversée.

singulière pour le calculateur : décrit une valeur numérique qui est trop grande pour être représentée par un nombre en virgule flottante sur le HP-28C/S.

sous-expression : une partie d'une expression algébrique consistant en un nombre, un nom, ou une fonction et ses arguments. Toute sous-expression peut avoir d'autres sous-expressions comme arguments, et peut elle-même être l'argument d'une autre sous-expression.

sous-expression choisie : la sous-expression qui est le sujet du menu actif des opérations de FORM ; elle est identifiée par le curseur en vidéo inverse qui met en valeur l'objet définissant la sous-expression.

structure de programme : un jeu de commandes qui doivent se succéder dans un ordre particulier à l'intérieur d'un programme. Plus spécifiquement, des clauses de programme, délimitées par les commandes, qui comprennent des unités logiques de décision et de branchement.

symbolique : représenter une valeur par un nom ou un symbole plutôt que par une valeur numérique explicite.

syntaxe algébrique : les restrictions s'appliquant à une procédure et selon lesquelles : (1) lorsqu'elle est évaluée, la procédure ne prend pas d'arguments dans la pile et y envoie un résultat ; et (2) elle peut être subdivisée en une hiérarchie de sous-expressions. Tous les objets algébriques et certains programmes satisfont à ces conditions.

tableau : un objet défini par les délimiteurs `[ ]` et qui représente une matrice ou un vecteur réel ou complexe.

tableau complexe : un tableau dont les éléments sont des nombres complexes.

tableau réel : un tableau dont les éléments sont uniquement des nombres réels.

taille de mot : le nombre de bits auquel sont tronqués les résultats des commandes opérant sur des entiers binaires.

tester : prendre une décision de branchement dans un programme en fonction de la valeur d'un indicateur binaire.

touches de menu : les six touches gris clair, sans identification, en haut du clavier de droite, dont la fonction est déterminée par le menu affiché en bas de l'écran.

touches de sélection des menus : toute touche activant un menu d'opérations qui peuvent être exécutées par pression sur les touches de menu.

tracé d'une fonction : un tracé produit par DRAW, pour lequel l'équation en cours est évaluée pour un maximum de 137 valeurs d'une variable (indépendante) donnée.

témoins : les pictogrammes en haut de l'affichage, qui indiquent l'état de certains modes du calculateur.

unité de base : l'une des sept unités utilisées comme base pour les conversions d'unités sur le HP-28C/S. Les unités de base sont le mètre (longueur), le kilogramme (masse), la seconde (temps), l'ampère (courant électrique), le degré Kelvin (température thermodynamique), la candela (intensité lumineuse) et la mole (quantité de matière).

valeur : le contenu numérique, logique ou symbolique d'un objet. Lorsqu'il se réfère aux variables, le terme *valeur* signifie l'objet stocké dans la variable.

valeur principale : un choix particulier parmi les valeurs multiples d'une relation ou solution mathématique, choisi pour son unicité ou sa simplicité. Par exemple, ASIN (0,5) donne 30°, qui est la valeur principale du résultat plus général $(-1)^n\ 30° + 180n°$, où n est un entier quelconque.

variable : une combinaison d'un nom (celui de la variable) et de tout autre objet (la valeur de la variable), qui sont stockés ensemble en mémoire.

variable dépendante : une variable dont la valeur est calculée à partir des valeurs d'autres variables (indépendantes) au lieu d'être fixée arbitrairement. Réfère également à la coordonnée verticale dans les tracés de courbes.

variable formelle : une variable qui a un nom mais n'existe pas, c'est-à-dire qui n'a pas une valeur.

variable indépendante : une variable dont la valeur peut être fixée arbitrairement au lieu d'être calculée à partir des valeurs d'autres variables. Dans un tracé, la coordonnée horizontale. Pour l'algorithme de résolution des équations, une variable qui ne contient pas une procédure comprenant des noms.

variable locale : une variable créée par les commandes → ou FOR pour utilisation temporaire dans une structure de programme. La variable est automatiquement supprimée lorsque l'exécution de la procédure est terminée.

vecteur : un tableau à une dimension.

vrai : une valeur (ou « état ») d'un indicateur binaire, représentée par un nombre réel d'une valeur quelconque différente de 0. Lorsqu'une commande renvoie un indicateur binaire vrai, ce dernier est représenté par le nombre 1.

Index des opérations

Cet index contient des informations de base et de référence pour toutes les opérations possibles sur le HP-28C/S :

Nom. Pour les opérations, la touche ou libellé de menu associé à cette opération. Pour les commandes, il montre la manière dont la commande apparaît en ligne de commande.

Description. Ce que fait l'opération.

Type. Comment l'opération peut être utilisée et comment elle correspond aux frappes au clavier. Ces informations sont données selon le code suivant :

Code	Description
A	Fonction analytique. Peut être résolue ou dérivée.
F	Fonction. Peut être incluse dans objets algébriques ou programmes.
C	Commande. Peut être incluse dans des programmes mais non dans des expressions algébriques.
O	Opération. Ne peut être incluse en ligne de commande ou dans une procédure.
*	La touche de clavier ou de menu correspondante n'effectue pas ENTER en mode exécution.
†	La touche de clavier ou de menu correspondante ajoute toujours le nom de la commande en ligne de commande.

Entrée. Nombre des objets nécessaires dans la pile opérationnelle (rubrique laissée en blanc pour les opérations n'utilisant pas la pile).

Sortie. Nombre d'objets renvoyés à la pile (en blanc pour les opérations qui n'utilisent pas la pile).

Référence. Endroit du manuel où cette commande est décrite.

HP-28C/S Index des opérations

Nom	Description	Type	Entrée	Sortie	Référence
ABORT	Met fin à l'exécution du programme.	C	0	0	PROGRAM CONTROL 246
ABS	Valeur absolue.	F	1	1	ARRAY 125 COMPLEX 165 REAL 269
ACOS	Arc cosinus.	A	1	1	TRIG 324
ACOSH	Arc cosinus hyperbolique.	A	1	1	LOGS 185
AF	Addtionne des fractions.	O*			ALGEBRA (FORM) 89
ALGEBRA	Sélectionne le menu ALGEBRA.	O*			ALGEBRA 59
ALOG	Antilogarithme (10 à une puissance).	A	1	1	LOGS 181
AND	ET logique ou binaire.	F	2	1	BINARY 138 PROGRAM TEST 258
ARG	Argument.	F	1	1	COMPLEX 165 TRIG 329
ARRAY	Selectionne le menu ARRAY.	O*			ARRAY 106
ARRY→	Remplace un tableau par ses éléments sous forme d'objets indépendants dans la pile.	C	1	$n+1$	ARRAY 114
ASIN	Arc sinus.	A	1	1	TRIG 323
ASINH	Arc sinus hyperbolique.	A	1	1	LOGS 184
ASR	Déplacement arithmétique d'un bit à droite.	C	1	1	BINARY 137

ATAN	Arc tangente.	A	1	1	TRIG	325
ATANH	Arc tangente hyperbolique.	A	1	1	LOGS	186
ATTN (ON)	Arrête l'exécution du programme ; efface la ligne de commande ; permet de quitter les affichages catalogue, FORM et de traçage de courbes.	O*			Opérations de base	53
AXES	Définit l'intersection des axes.	C	1	0	PLOT	208
A→	Associativité à droite.	O*			ALGEBRA (FORM)	82
BEEP	Fait sonner une tonalité.	C	2	0	PROGRAM CONTROL	249
BIN	Sélectionne la base binaire.	C*	0	0	BINARY	133
BINARY	Sélectionne le menu BINARY.	O*			BINARY	130
BRANCH	Sélectionne le menu PROGRAM BRANCH.	O*			PROGRAM BRANCH	232
B→R	Conversion de binaire en réel.	C	1	1	BINARY	135
CATALOG	Commence le catalogue des commandes.	O*			CATALOG	156
CEIL	Entier plus grand suivant.	F	1	1	REAL	271
CENTR	Définit le centre de l'affichage lors du traçage de courbes.	C	1	0	PLOT	209
CF	Désarme un indicateur binaire d'utilisateur.	C	1	0	PROGRAM TEST	255
CHR	Crée une chaîne à un seul caractère.	C	1	1	STRING	317
CHS	Change le signe d'un nombre en ligne de commande ou exécute NEG.	O*			Opérations de base	32
CLEAR	Efface la pile opérationnelle.	C	*n*	0	STACK	291

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence
CLLCD	Efface l'affichage.	C	0	0	PLOT 212 PROGRAM CONTROL 249
CLMF	Désarme l'indicateur binaire système.	C	0	0	PLOT 214 PROGRAM CONTROL 250
CLUSR	Efface toutes les variables d'utilisateur.	C†	0	0	USER 347
CLΣ	Supprime la matrice de statistiques.	C	0	0	STAT 299
■ CMPLX	Sélectionne le menu COMPLEX.	O*			COMPLEX 160
CNRM	Calcule une norme de colonne.	C	1	1	ARRAY 126
COLCT	Met en facteur les termes semblables.	C	1	1	ALGEBRA 71
COLCT	Met en facteur les termes semblables dans une sous-expression.	O*			ALGEBRA (FORM) 80
COLΣ	Sélectionne des colonnes de la matrice de statistiques.	C	2	0	PLOT 211 STAT 302
■ COMMAND	Déplace une commande de la pile des commandes vers la ligne de commande.	O			Opérations de base 48
CON	Crée une matrice constante.	C	2	0, 1	ARRAY 121
CONJ	Calcul du complexe conjugué.	F	1	1	ARRAY 128 COMPLEX 163
■ CONT	Relance un programme qui a été suspendu.	O			PROGRAM CONTROL 244

CONVERT	Exécute une conversion d'unités.	C	3	2	UNITS	332
CORR	Coefficient de corrélation.	C	0	1	STAT	303
COS	Cosinus.	A	1	1	TRIG	324
COSH	Cosinus hyperbolique.	A	1	1	LOGS	185
COV	Covariance.	C	0	1	STAT	304
CR	Imprime le contenu de la mémoire-tampon de l'imprimante et place la tête d'impression à droite.	C	0	0	PRINT	222
CROSS	Donne le produit vectoriel de deux vecteurs à trois composantes.	C	2	1	ARRAY	124
■ CTRL	Sélectionne le menu PROGRAM CONTROL.	O*			PROGRAM CONTROL	243
C→R	Conversion complexe-réel.	C	1	2	ARRAY COMPLEX TRIG	127 162 328
■ d/dx	Dérivée (fonction ∂).	F	2	1	Calcul différentiel et intégral	141
DEC	Sélectionne une base décimale.	C*	0	0	BINARY	132
DEG	Sélectionne le mode degrés.	C*	0	0	MODE	191
DEL	Supprime le caractère sous le curseur ; numérise le point.	O*	0	0, 2	Opérations de base PLOT	39 203
■ DEL	Détruit le caractère sous le curseur et tous les caractères se trouvant à sa droite.	O*			Opérations de base	39
DEPTH	Compte les objets de la pile.	C	0	1	STACK	294

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
DET	Déterminant de son argument (une matrice carrée).	C	1	1	ARRAY	125
DINV	Double inversion.	O*			ALGEBRA (FORM)	86
DISP	Affiche un objet.	C	2	0	PLOT PROGRAM CONTROL STRING	213 250 320
DNEG	Double changement de signe.	O*			ALGEBRA (FORM)	85
DO	Elément de DO...UNTIL...END.	C†			PROGRAM BRANCH	233
DOT	Produit scalaire de deux vecteurs.	C	2	1	ARRAY	124
DRAW	Crée un tracé de fonction mathématique.	C	0	0	PLOT	206
DRAX	Trace des axes.	C	0	0	PLOT	213
DROP	Supprime un objet dans la pile.	C	1	0	STACK	290
DROPN	Supprime $n+1$ objets de la pile.	C	$n+1$	0	STACK	294
DROP2	Supprime deux objets de la pile.	C	2	0	STACK	292
DRWΣ	Crée un nuage de points (statistiques).	C	0	0	PLOT	212
DUP	Copie un objet de la pile.	C	1	2	STACK	291
DUPN	Copie n objets de la pile.	C	$n+1$	$2n$	STACK	294
DUP2	Copie deux objets de la pile.	C	2	4	STACK	292
D→	Distributivité à droite.	O*			ALGEBRA (FORM)	83

D→R	Conversion degrés-radians.	F	1	1	TRIG	331
e	Constante symbolique *e*.	F†	0	1	ALGEBRA REAL	70 266
■ EDIT	Copie l'objet du niveau 1 en ligne de commande pour en permettre la correction ou modification.	O	1	1	Opérations de base	41
EEX	Introduit l'exposant en ligne de commande.	O*			Opérations de base	32
ELSE	Commence la clause ELSE.	C†			PROGRAM BRANCH	232
END	Termine les programmes.	C†	1	0	PROGRAM BRANCH	232 233
ENG	Définit le format d'affichage ingénieur.	C	1	0	MODE	191
ENTER	Analyse et évalue la ligne de commande ou exécute DUP.	O			Opérations de base	40
ERRM	Renvoie le dernier message d'erreur.	C	0	1	PROGRAM CONTROL	251
ERRN	Renvoie le dernier numéro d'erreur.	C	0	1	PROGRAM CONTROL	251
EVAL	Evalue un objet.	C	1	0	Opérations de base	42
EXGET	Appelle une sous-expression.	C	2	1	ALGEBRA	76
EXGET	Appelle une sous-expression.	O*		2	ALGEBRA (FORM)	80
EXP	Exponentielle.	A	1	1	LOGS	182
EXPAN	Développe une expression algébrique.	C	1	1	ALGEBRA	72
EXPAN	Développe une sous-expression.	O*			ALGEBRA (FORM)	80

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
EXPM	Exponentielle moins 1.	A	1	1	LOGS	183
EXPR=	Evalue l'équation en cours.	O	0	1	SOLVE	278
EXSUB	Remplace par une sous-expression.	C	3	1	ALGEBRA	75
E^	Remplace puissance d'un produit par puissance d'une puissance.	O*			ALGEBRA (FORM)	89
E( )	Remplace puissance d'une puissance par puissance d'un produit.				ALGEBRA (FORM)	89
FACT	Factorielle ou fonction gamma.	F	1	1	REAL	267
FC?	Teste un indicateur binaire d'utilisateur.	C	1	1	PROGRAM TEST	256
FC?C	Teste et désarme un indicateur binaire d'utilisateur.	C	1	1	PROGRAM TEST	257
FETCH	Met fin à CATALOG ou UNITS, écrit la commande ou l'unité en cours en ligne de commande.	O*			CATALOG UNITS	336 157
FIX	Sélectionne le format d'affichage FIX.	C	1	0	MODE	189
FLOOR	Entier plus petit suivant.	F	1	1	REAL	271
FOR	Commence une boucle définie.	C†	2	0	PROGRAM BRANCH	233
FORM	Change la forme d'une expression algébrique.	C	1	1, 3	ALGEBRA ALGEBRA (FORM)	74 77
FP	Partie fractionnelle.	F	1	1	REAL	271

FS?	Teste un indicateur binaire d'utilisateur.	C	1	1	PROGRAM TEST	256
FS?C	Teste et désarme un indicateur binaire d'utilisateur.	C	1	1	PROGRAM TEST	256
GET	Appelle un élément d'un objet.	C	2	1	ARRAY LIST	116 176
GETI	Appelle un élément d'un objet et augmente l'index d'une unité.	C	2	3	ARRAY LIST	118 178
HALT	Suspend l'exécution d'un programme.	C†			PROGRAM CONTROL	246
HEX	Sélectionne une base hexadécimale.	C*	0	0	BINARY	132
HMS+	Additionne en format H.M.S.	C	2	1	TRIG	331
HMS−	Soustrait en format H.M.S.	C	2	1	TRIG	331
HMS→	Convertit à partir d'un format H.M.S.	C	1	1	TRIG	330
i	Constante symbolique *i*.	F†	0	1	ALGEBRA	70
IDN	Crée une matrice-unité.	C	1	0, 1	ARRAY	122
IF	Commence la clause IF.	C†	0	0	PROGRAM BRANCH	232
IFERR	Commence la clause d'erreur IF.	C†	0	0	PROGRAM BRANCH	232
IFT	Commande If-then.	C	2	0	PROGRAM BRANCH	241
IFTE	Fonction If-then-else.	F	3	0	PROGRAM BRANCH	241
IM	Renvoie la partie imaginaire d'un nombre ou d'un tableau.	F	1	1	ARRAY COMPLEX	128 162
INDEP	Sélectionne la variable indépendante d'un tracé.	C	1	0	PLOT	206

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence
INS	Fait passer du mode d'insertion au mode de remplacement et inversement ; numérise le point.	O*	0	0, 1	Opérations de base 39 PLOT 203
■ INS	Supprime tous les caractères à la gauche du curseur.	O*			Opérations de base 39
INV	Inverse.	A	1	1	Arithmétique 103 ARRAY 113
IP	Partie entière.	F	1	1	REAL 270
ISOL	Résout une expression ou une équation.	C	2	1	ALGEBRA 76 SOLVE 285
KEY	Renvoie une chaîne représentant la plus ancienne touche stockée en mémoire-tampon.	C	0	1, 2	PROGRAM CONTROL 247
KILL	Met fin à tous les programmes suspendus.	C	0	0	PROGRAM CONTROL 247
LAST	Renvoie les derniers arguments.	C	0	1, 2, 3	Opérations de base 49
LC	Echange entre les modes « capitales » et « minuscules ».	O*			Opérations de base 33
LEFT=	Evalue la partie gauche d'une équation en cours.	O	0	1	SOLVE 278
LEVEL	Affiche le rang de la sous-expression spécifiée.	O*			ALGEBRA (FORM) 80
■ LIST	Sélectionne le menu LIST.	O*			LIST 174
LIST→	Déplace les éléments de la liste vers la pile opérationnelle.	C	1	$n+1$	LIST 175 STACK 293

LN	Logarithme népérien.	A	1	1	LOGS	182
LNP1	Logarithme népérien de (*argument*+1).	A	1	1	LOGS	183
LOG	Logarithme (base 10).	A	1	1	LOGS	181
■ LOGS	Sélectionne le menu LOGS.	O*			LOGS	181
LR	Calcule une régression linéaire.	C	0	2	STAT	304
L( )	Remplace multiplication par log. par log. d'une puissance.	O*			ALGEBRA (FORM)	88
L*	Remplace log. d'une puissance par multiplication par un log.	O*			ALGEBRA (FORM)	88
MANT	Renvoie la mantisse d'un nombre.	F	1	1	REAL	270
MAX	Renvoie le maximum de deux nombres.	F	2	1	REAL	272
MAXR	Constante symbolique réelle maximale.	F	0	1	ALGEBRA REAL	70 268
MAXΣ	Trouve les valeurs maximales de valeurs coordonnées dans la matrice de statistiques.	C	0	1	STAT	302
MEAN	Calcule des moyennes statistiques.	C	0	1	STAT	300
MEM	Renvoie la quantité de mémoire disponible.	C	0	1	USER	347
MIN	Renvoie le minimum de deux nombres.	F	2	1	REAL	273
MINR	Constante symbolique réelle minimale.	F	0	1	ALGEBRA REAL	70 268
MINΣ	Trouve les valeurs coordonnées minimales dans la matrice de statistiques.	C	0	1	STAT	302

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
MOD	Modulo.	F	2	1	REAL	273
■ MODE	Sélectionne le menu MODE.	O*			MODE	187
M→	Met en facteur l'argument droit.	O*			ALGEBRA (FORM)	85
NEG	Changement de signe d'un argument.	A	1	1	Arithmétique ARRAY COMPLEX REAL	104 129 165 266
NEXT	Met fin à une boucle sans fin.	C†	0	0	PROGRAM BRANCH	233
NEXT	Affiche la ligne suivante de libellés de menu.	O*				57
NEXT	Avance à la commande ou à l'unité suivante dans un catalogue.	O*			CATALOG UNITS	157 336
NEXT	Avance à l'option d'argument suivante dans USAGE.	O*			CATALOG	158
NO	Pas de suppression si Out of Memory.	O*			Opérations de base	52
NORM	Désactive le mode TRACE.	O*			PRINT	221
NOT	NON logique ou binaire.	F	1	1	BINARY PROGRAM TEST	140 260
NUM	Renvoie le code de caractère.	C	1	1	STRING	317
NΣ	Renvoie le nombre d'observations contenues dans la matrice de statistiques.	C	0	1	STAT	299

OBGET	Extrait un objet d'une expression algébrique.	C	2	1	ALGEBRA	76
OBSUB	Remplace un objet dans une expr. algébrique.	C	3	1	ALGEBRA	75
OCT	Sélectionne la base octale.	C*	0	0	BINARY	132
ON (ATTN)	Met le calculateur sous tension ; arrête exécution d'un programme ; efface ligne de commande ; arrête catalogues, FORM et affichages graphiques.	O*			Opérations de base	53
ON DEL	Annule un arrêt système ou une réinitialisation si vous appuyez dessus sans relâcher ON.	O*			Opérations de base	55
ON INS ▶	(Réinitialisation) Arrête l'exécution d'un programme, efface les variables locales et de l'utilisateur, efface la pile opérationnelle, réinitialise les indicateurs binaires d'utilisateur.	O*			Opérations de base	54
ON +	Augmente le contraste de l'affichage.	O*			Opérations de base	53
ON −	Diminue le contraste de l'affichage.	O*			Opérations de base	53
ON ▲	(Arrêt système) Arrête l'exécution du programme, efface variables locales et pile opérationnelle.	O*			Opérations de base	54
ON ▼	Lance le test système.	O*			Opérations de base	55
ON ◀	Lance un test système continuel.	O*			Opérations de base	55
■ OFF	Eteint le calculateur.	O*			Opérations de base	53
OR	OU logique ou binaire.	F	2	1	BINARY PROGRAM TEST	139 258
ORDER	Ré-arrange le menu USER.	C	1	0	USER	346
OVER	Duplique l'objet placé en niveau 2.	C	2	3	STACK	292

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence
PICK	Duplique le *n*ième objet.	C	*n*+1	*n*+1	STACK 293
PIXEL	Allume un pixel de l'affichage.	C	1	0	PLOT 213
■ PLOT	Sélectionne le menu PLOT.	O*			PLOT 198
PMAX	Définit les coordonnées de l'angle supérieur droit de l'affichage graphique.	C	1	0	PLOT 206
PMIN	Définit les coordonnées de l'angle inférieur gauche de l'affichage graphique.	C	1	0	PLOT 205
POS	Trouve un sous-ensemble dans une chaîne.	C	2	1	STRING 320
PPAR	Rappelle la liste des paramètres de traçage.	C	0	1	PLOT 208
PREDV	Valeur estimée.	C	1	1	STAT 305
■ PREV	Affiche la ligne précédente de libellés de menu.	O*			57
PREV↓	Affiche la commande ou l'unité précédente dans un catalogue.	O*			CATALOG 157 UNITS 336
PREV	Affiche l'option d'argument précédent dans USAGE.	O*			CATALOG 158
■ PRINT	Sélectionne le menu PRINT.	O*			PRINT 215
PRLCD	Imprime une image de l'affichage.	C	0	0	PLOT 214 PRINT 220
PRMD	Imprime et affiche les modes en cours.	C	0	0	MODE 196 PRINT 222

PROGRAM						
[BRANCH]	Sélectionne le menu PROGRAM BRANCH.	O*			PROGRAM BRANCH	252
[CTRL]	Sélectionne le menu PROGRAM CONTROL.	O*			PROGRAM CONTROL	232
[TEST]	Sélectionne le menu PROGRAM TEST.	O*			PROGRAM TEST	243
PRST	Imprime la pile opérationnelle.	C	0	0	PRINT	219
PRSTC	Imprime la pile opérationnelle en format compact.	C	0	0	PRINT	221
PRUSR	Imprime une liste de variables.	C	0	0	PRINT	222
PRVAR	Imprime le contenu d'une variable.	C	1	0	PRINT	220
PR1	Imprime l'objet en niveau 1.	C	0	0	PRINT	218
PURGE	Supprime une ou plusieurs variables.	C	1	0	Opérations de base	47
PUT	Place un élément dans un tableau ou dans une liste.	C	3	0, 1	ARRAY LIST	115 175
PUTI	Place un élément dans un tableau ou dans une liste et incrémente l'index en conséquence.	C	3	2	ARRAY LIST	117 177
P→R	Conversion polaires en rectangulaires.	F	1	1	COMPLEX TRIG	164 326
QUAD	Résout un polynôme du second degré.	C	2	1	ALGEBRA SOLVE	76 286
QUIT	Met fin à CATALOG ou à UNITS.	O*			CATALOG UNITS	336 157
QUIT	Met fin à l'affichage USAGE.	O*			CATALOG	159

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
RAD	Sélectionne le mode radians.	C*	0	0	MODE	192
RAND	Renvoie un nombre aléatoire.	C	0	1	REAL	267
RCEQ	Rappelle l'équation en cours.	C	0	1	PLOT SOLVE	205 277
RCL	Rappelle le contenu non évalué d'une variable.	C	1	1	Opérations de base	46
RCLF	Renvoie un nombre entier binaire représentant les indicateurs binaires d'utilisateur.	C	0	1	PROGRAM TEST	262
RCLΣ	Rappelle la matrice de statistiques en cours.	C	0	1	PLOT STAT	210 299
RCWS	Rappelle la taille de mot, un nombre entier binaire.	C	0	1	BINARY	133
RDM	Redimensionne un tableau.	C	2	0, 1	ARRAY	120
RDX.	Définit «.» comme séparateur décimal.	O*			MODE	196
RDX,	Définit «,» comme séparateur décimal.	O*			MODE	196
RDZ	Définit la racine des nombres aléatoires.	C	1	0	REAL	268
RE	Renvoie la partie réelle d'un nombre complexe ou d'un tableau.	F	1	1	ARRAY COMPLEX	127 162
[REAL]	Sélectionne le menu REAL.	O*			REAL	264
REPEAT	Un des éléments de WHILE...REPEAT...END.	C†	1	0	PROGRAM BRANCH	233
RES	Définit la résolution du tracé.	C	1	0	PLOT	208

RL	Effectue une permutation circulaire d'un bit vers la gauche.	C	1	1	BINARY	134
RLB	Effectue une permutation circulaire d'un octet vers la gauche.	C	1	1	BINARY	134
RND	Arrondit en fonction du mode d'affichage des nombres reéls.	F	1	1	REAL	272
RNRM	Calcule la norme de ligne de son argument.	C	1	1	ARRAY	125
ROLL	Déplace l'objet au niveau $n+1$ vers le niveau 1.	C	$n+1$	n	STACK	291
ROLLD	Déplace l'objet au niveau 2 vers le niveau n.	C	$n+1$	n	STACK	293
ROOT	Trouve une racine numérique.	C	3	1, 3	SOLVE	284
ROT	Déplace l'objet placé au niveau 3 vers le niveau 1.	C	3	3	STACK	292
RR	Effectue une permutation circulaire d'un bit vers la droite .	C	1	1	BINARY	134
RRB	Effectue une permutation d'un octet vers la droite.	C	1	1	BINARY	135
RSD	Calcule une correction à la solution d'un système d'équations.	C	3	1	ARRAY	123
RT=	Evalue la partie droite de l'équation en cours.	O	0	1	SOLVE	278
R→B	Conversion réel en binaire.	C	1	1	BINARY	135
R→C	Conversion réel en complexe.	C	2	1	ARRAY COMPLEX TRIG	126 161 328
R→D	Conversion radians en degrés.	F	1	1	TRIG	331

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence
R→P	Conversion de coordonnées rectangulaires en polaires.	F	1	1	COMPLEX 164 TRIG 327
SAME	Teste l'égalité de deux objets.	C	2	1	PROGRAM TEST 260
SCAN	Défilement automatique de CATALOG ou de UNITS.	O*			CATALOG 336 UNITS 157
SCI	Sélectionne le format d'affichage scientifique.	C	1	0	MODE 190
SCLΣ	Met automatiquement à l'échelle les paramètres de traçage en fonction des données statistiques.	C	0	0	PLOT 211
SCONJ	Calcule le conjugué du contenu d'une variable.	C	1	0	STORE 313
SDEV	Calcule l'écart-type.	C	0	1	STAT 301
SF	Définit un indicateur binaire d'utilisateur.	C	1	0	PROGRAM TEST 255
SHOW	Résout toutes références à un nom implicite dans une expression algébrique.	C	2	1	ALGEBRA 76 SOLVE 286
SIGN	Signe d'un nombre.	F	1	1	COMPLEX 163 REAL 269
SIN	Sinus.	A	1	1	TRIG 323
SINH	Sinus hyperbolique.	A	1	1	LOGS 184
SINV	Inverse le contenu de la variable nommée dans la pile.	C	1	0	STORE 312

SIZE	Donne les dimensions d'une liste, d'un tableau, d'une chaîne ou d'une expression algébrique.	C	1	1	ALGEBRA 74 ARRAY 119 LIST 180 STRING 321
SL	Décale vers la gauche d'un bit.	C	1	1	BINARY 136
SLB	Décale vers la gauche d'un octet.	C	1	1	BINARY 137
SNEG	Donne l'opposé de la variable nommée dans la pile.	C	1	0	STORE 312
[SOLV]	Sélectionne le menu SOLVE.	O*			SOLVE 275
SOLVR	Sélectionne le menu des variables dans SOLVE.	O			SOLVE 278
SQ	Met au carré un nombre ou une matrice.	A	1	1	Arithmétique 104 ARRAY 113
SR	Déplace d'un bit vers la droite	C	1	1	BINARY 136
SRB	Déplace d'un octet vers la droite.	C	1	1	BINARY 137
SST	Exécute pas-à-pas un programme suspendu.	O			PROGRAM CONTROL 245
■ [STACK]	Sélectionne le menu STACK.	O*			STACK 290
START	Commence une boucle définie.	C†	2	0	PROGRAM BRANCH 233
■ [STAT]	Sélectionne le menu STAT.	O*			STAT 296
STD	Sélectionne le format d'affichage standard.	C*			MODE 188
STEP	Met fin à une boucle définie.	C†	1	0	PROGRAM BRANCH 233
STEQ	Stocke l'équation en cours.	C	1	0	PLOT 205 SOLVE 277

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
STO	Stocke un objet dans une variable.	C	2	0	Opérations de base	46
STOF	Définit tous les indicateurs binaires d'utilisateur en fonction de la valeur d'un entier binaire.	C	1	0	PROGRAM TEST	262
STOP	Arrête le défilement de CATALOG ou de UNITS.	O*			CATALOG UNITS	336 157
■ STORE	Sélectionne le menu STORE.	O*			STORE	309
STO∗	Stocke un produit arithmétique.	C	2	0	STORE	311
STO+	Stocke une somme arithmétique.	C	2	0	STORE	309
STO−	Stocke une soustraction arithmétique.	C	2	0	STORE	310
STO/	Stocke une division arithmétique.	C	2	0	STORE	311
STOΣ	Stocke la matrice statistique en cours.	O*	1	0	PLOT STAT	210 299
■ STRING	Sélectionne le menu STRING.	O*			STRING	314
STR→	Analyse et évalue les commandes définies par une chaîne.	C	1	0	STRING	315
STWS	Définit la taille de mot des entiers binaires.	C	1	0	BINARY	133
SUB	Extrait une partie d'une liste ou d'une chaîne.	C	3	1	LIST STRING	180 321

SWAP	Permute les deux premiers objets de la pile.	C	2	2	STACK	290
SYSEVAL	Evalue un objet système.	C	1	0	Opérations de base	43
TAN	Tangente.	A	1	1	TRIG	325
TANH	Tangente hyperbolique.	A	1	1	LOGS	185
TAYLR	Calcule une approximation par une série de Taylor.	C	3	1	ALGEBRA Calcul différentiel et intégral	76 151
■ TEST	Sélectionne le menu PROGRAM TEST.	O*			PROGRAM TEST	252
THEN	Commence la clause THEN.	C†	1	0	PROGRAM BRANCH	232
TOT	Fait la somme des valeurs coordonnées dans la matrice de statistiques.	C	0	1	STAT	300
TRACE	Active le mode TRACE de l'imprimante.	O*			PRINT	221
TRIG	Sélectionne le menu TRIG.	O*			TRIG	322
TRN	Transpose une matrice.	C	1	0, 1	ARRAY	121
TYPE	Renvoie le type d'un objet.	C	1	1	PROGRAM TEST	263
■ UNDO	Remplace le contenu de la pile opérationnelle.	O*			Opérations de base	48
■ UNITS	Sélectionne le catalogue des unités.	O*			UNITS	332
UNTIL	Un élément de BEGIN...UNTIL...END.	C†			PROGRAM BRANCH	233
USE	Affiche USAGE pour la commande en cours de CATALOG.	O*			CATALOG	157
USER	Sélectionne le menu USER.	O*			USER	346

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
UTPC	Loi de χ^2 (de Khi carré).	C	2	1	STAT	307
UTPF	Loi de F.	C	3	1	STAT	307
UTPN	Loi normale (loi de Laplace-Gauss).	C	3	1	STAT	307
UTPT	Loi de t (loi de Student).	C	2	1	STAT	308
VAR	Calcule les variances statistiques.	C	0	1	STAT	301
■ VIEW↑	Déplace affichage d'une ligne vers le haut.	O*			Opérations de base	40
■ VIEW↓	Déplace l'affichage d'une ligne vers le bas.	O*			Opérations de base	40
■ VISIT	Copie un objet en ligne de commande pour modification ou correction.	O	1	0	Opérations de base	42
WAIT	Fait une pause dans l'exécution du programme.	C	1	0	PROGRAM CONTROL	247
WHILE	Commence WHILE...REPEAT...END.	C†	0	0	PROGRAM BRANCH	233
XOR	OU exclusif logique ou binaire.	F	2	1	BINARY PROGRAM TEST	139 259
XPON	Renvoie l'exposant d'un nombre.	F	1	1	REAL	270
■ x^2	Exécute la fonction SQ.	A	1	1	Arithmétique ARRAY	104 113
YES	Suppression si `Out of Memory`.	O*			Opérations de base	52

[1/x]	Exécute la fonction INV.	A	1	1	Arithmétique ARRAY	103 113
1/( )	Double inversion et distributivité.	O*			ALGEBRA (FORM)	87
+	Additionne deux objets.	A	2	1	Arithmétique ARRAY LIST STRING	96 108 174 315
+CMD	Active COMMAND.	O*			MODE	193
+LAST	Active LAST.	O*			MODE	193
+ML	Sélectionne le mode d'affichage multi-lignes.	O*			MODE	195
+UND	Active UNDO.	O*			MODE	194
+1-1	Ajoute et soustrait 1.	O*			ALGEBRA (FORM)	88
–	Soustrait deux objets.	A	2	1	Arithmétique ARRAY	98 108
-CMD	Désactive COMMAND.	O*			MODE	193
-LAST	Désactive LAST.	O*			MODE	193
-ML	Sélectionne le mode d'affichage à 1 ligne.	O*			MODE	195
-UND	Désactive UNDO.	O*			MODE	194
-( )	Double changement de signe et distributivité.	O*			ALGEBRA (FORM)	86
*	Multiplie deux objets.	A	2	1	Arithmétique ARRAY	99 109

*

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence	
*H	Ajuste la hauteur d'un tracé.	C	1	0	PLOT	210
*W	Ajuste la largeur d'un tracé.	C	1	0	PLOT	209
*1	Multiplie par 1.	O*			ALGEBRA (FORM)	87
/	Divise deux objets.	A	2	1	Arithmétique ARRAY	101 109
/1	Divise par 1.	O*			ALGEBRA (FORM)	87
%	Pourcent.	F	2	1	REAL	265
%CH	Différence en pourcent.	F	2	1	REAL	265
%T	Pourcentage du total.	F	2	1	REAL	174
^	Elève un nombre à une puissance.	A	2	1	Arithmétique	102
^1	Elève à la puissance 1.	O*			ALGEBRA (FORM)	88
√	Prend la racine carrée.	A	1	1	Arithmétique	103
∫	Intégrale définie ou indéfinie.	C	3	1, 2	Calc. différ. et intégral	145
∂	Dérivée.	F	2	1	Calc. différ. et intégral	141
<	Comparaison « inférieur à ».	F†	2	1	PROGRAM TEST	253
≤	Comparaison « plus petit ou égal ».	F†	2	1	PROGRAM TEST	254
=	Egale.	A†	2	1	ALGEBRA	64

==	Comparaison d'égalité.	F	2	1	PROGRAM TEST	261
≠	Différent de.	F†	2	1	PROGRAM TEST	252
≥	Comparaison « plus grand ou égal ».	F†	2	1	PROGRAM TEST	254
>	Comparaison « plus grand que ».	F†	2	1	PROGRAM TEST	253
■	Touche « préfixe », également nommée « Shift ».	O*				
◀✥▶	Choisit menu du curseur ou rétablit menu préc.	O*			Opérations de base	38
▲	Déplace le curseur vers le haut.	O*			Opérations de base	39
■ ▲	Déplace le curseur en première ligne de la ligne de commande.	O*			Opérations de base	39
▼	Déplace le curseur vers le bas.	O*			Opérations de base	39
■ ▼	Déplace le curseur en dernière ligne de la ligne de commande.	O*			Opérations de base	39
◀	Déplace le curseur vers la gauche.	O*			Opérations de base	39
■ ◀	Déplace le curseur à gauche de la ligne de commande.	O*			Opérations de base	39
▶	Déplace le curseur vers la droite.	O*			Opérations de base	39
■ ▶	Déplace le curseur à droite de la ligne de commande.	O*			Opérations de base	39
[←]	Déplace le curseur FORM vers la gauche.	O*			ALGEBRA (FORM)	80
[→]	Déplace le curseur FORM vers la droite.	O*			ALGEBRA (FORM)	80
◀	Espace arrière.	O*			Opérations de base	33
α	Alterne entre mode alpha actif et inactif.	O*			Opérations de base	37
■ α LOCK	Verrouille le mode alpha en position « actif ».	O*			Opérations de base	37

HP-28C/S Index des opérations (suite)

Nom	Description	Type	Entrée	Sortie	Référence
π	Constante symbolique π.	F†	0	1	ALGEBRA 70 REAL 266
Σ+	Ajoute une observation à la matrice de statistiques.	C	1	0	STAT 297
Σ−	Supprime la dernière observation dans la matrice de statistiques.	C	0	1	STAT 298
←A	Associativité à gauche.	O*			ALGEBRA (FORM) 81
←D	Distributivité à gauche.	O*			ALGEBRA (FORM) 83
←M	Mise en facteur des arguments gauches.	O*			ALGEBRA (FORM) 84
←→	Commutativité des arguments d'un opérateur.	O*			ALGEBRA (FORM) 81
→	Crée des variables locales.	C†	*n*	0	Programmes 228
→ARRY	Combine des nombres dans un tableau.	C	*n*+1	1	ARRAY 114
→HMS	Convertit un nombre en format H.M.S.	C	1	1	TRIG 330
→LIST	Combine des objets en une liste.	C	*n*+1	1	LIST 174 STACK 295
→NUM	Evalue un objet en mode numérique.	C	1	0	Opérations de base 43
→STR	Convertit un objet en chaîne.	C	1	1	STRING 315
→()	Distributivité de l'opérateur préfixe.	O*			ALGEBRA (FORM) 82

Termes utilisés dans les diagrammes de la pile

Terme	Description
objet	N'importe quel objet.
x ou *y*	Nombre réel.
hms	Nombre réel en format heures-minutes-secondes.
n	Nombre réel entier positif.
indic. binaire	Nombre réel, nul (état faux) ou non nul (état vrai).
z	Nombre réel ou complexe.
(*x* , *y*)	Nombre complexe sous forme rectangulaire.
(*r* , *θ*)	Nombre complexe sous forme polaire.
# *n*	Entier binaire.
"*chaîne*"	Chaîne de caractères.
[*tableau*]	Matrice ou vecteur réel ou complexe.
[*vecteur*]	Vecteur réel ou complexe.
[*matrice*]	Matrice réelle ou complexe.
[*tableau R*]	Matrice ou vecteur réel.
[*tableau C*]	Matrice ou vecteur complexe.
{ *liste* }	Liste d'objets.
{ *index* }	Liste d'un ou deux nombre(s) réel(s) définissant l'élément d'un tableau.
{ *dim* }	Liste d'un ou deux nombre(s) réel(s) définissant la (les) dimension(s) d'un tableau.
' *nom* '	Nom ou nom local.
«*programme*»	Programme.
' *symbole* '	Equation, expression, ou nom traité comme un objet algébrique.

Table des matières

HEWLETT PACKARD

N° de référence
00028-90023

00028-90114
Printed in W. Germany 2/88
Imprimé en R.F.A. 2/88.
French – Français