CALCULATRICES

Matthieu Cornillault
M. de Courville E. Lesueur

HP 48 G • GX • S • SX en prépa

Tous les programmes-clés pour les concours, prêts à l'emploi ... et prêts à charger sur la disquette PC/Mac incluse !

avec disk !

2e édition 500 pages

DUNOD TECH

Collection

CALCULATRICES EFFICACES

Les grands atouts de la "petite informatique"

- **Votre CASIO fx 6800 G**
- **CASIO fx : programmez votre succès !**
- **CASIO fx : 300 programmes**
- **CASIO fx : faites vos jeux !**
- **Jeux et graphisme sur CASIO fx**
- **Maths au lycée avec votre CASIO fx**
- **Programmation efficace sur CASIO fx**
- **Trucs et astuces pour CASIO fx**
- **TI-81 : programmez votre succès !**
- **300 programmes TI-81**
- **Jeux et graphisme sur TI-81**
- **TI-81 : le "top" des jeux !**
- **TI-82 : mathématiques au lycée**
- **TI-82 : programmes pour le lycée**
- **Ti-82 : le "top" des jeux !**
- **TI-85 par l'exemple**
- **TI-85 du lycée à la prépa**
- **TI-85 : le "top" des jeux !**
- **SHARP EL 9200/9300 : faites vos jeux !**
- **HP 48 en classes préparatoires**
- **Finance et gestion sur calculatrices**

DUNOD TECH

CALCULATRICES

HP 48 en prépa

M. Cornillault, M.de Courville, E. Lesueur

DUNOD TECH

HP 48, Hewlett-Packard, ainsi que tous les noms de produits cités dans cet ouvrage sont des marques déposées de leurs propriétaires respectifs.

Ce livre n'est pas le manuel des calculateurs Hewlett-Packard 48 G, GX, S et SX.
Son contenu n'engage pas la société Hewlett-Packard, ni ses distributeurs.

Illustrations de couverture : Rachid Maraï

ISBN 2 10 002012 9

Sommaire

Les auteurs tiennent à remercier Monsieur Alain Mézard, professeur de Mathématiques au Lycée Saint-Louis à Paris.

Les auteurs et l'éditeur remercient également Messieurs Jean-Paul Barnier et Michel Serge, de Hewlett-Packard France, pour l'intérêt qu'ils ont porté à ce projet.

Avant-propos

Les machines Hewlett-Packard sont depuis longtemps des références dans le domaine des calculateurs scientifiques programmables. Avec ses nouvelles fonctions, sa puissance et ses possibilités d'extension, la HP 48 est la digne héritière de la dynastie HP.

Très appréciées par les étudiants en classes préparatoires scientifiques, les HP 48 demeurent d'un emploi peu aisé pour le débutant. Ce livre a pour but d'apporter à l'étudiant une foule d'outils indispensables afin de lui faire gagner un temps précieux.

Les élèves de classes préparatoires trouveront des programmes adaptés à leurs besoins alors que l'utilisateur maîtrisant bien sa machine pourra s'aider des différentes astuces présentées dans cet ouvrage. Le spécialiste disposera d'une liste des "internals" de la HP 48 et pourra ainsi optimiser ses programmes.

Ce livre comprend une initiation aux fonctions de base de la HP 48. Les pages 19 à 474 proposent de très nombreux programmes qui assisteront efficacement l'étudiant dans son travail quotidien et à l'occasion des concours. De nombreux programmes sont capables de présenter les résultats sous forme rationnelle, et ce, de façon exacte : les calculs ne sont donc plus effectués avec de quelconques nombres décimaux comme c'est le cas dans l'immense majorité des programmes proposés aux étudiants. Cette possibilité différencie notre livre de ceux proposés jusqu'à aujourd'hui aux utilisateurs de HP 48.

Ainsi, votre HP 48 ne vous donnera désormais plus une vague valeur décimale approximative mais un résultat rationnel exploitable que vous n'aurez plus qu'à reporter sur votre copie...

Les polynômes, les matrices, les développements limités et autres gâteries en deviennent presque trop faciles à traiter...

Avertissement

• Ce livre est destiné aux utilisateurs des HP 48 s, sx, g et gx. Tous les programmes présentés fonctionnent sur ces modèles à l'exception de certaines astuces exploitant les spécificités des versions sx et gx.

• L'adaptation à la HP 28 de la plupart des programmes présentés dans ce livre est possible et aisée.

• Certains des programmes présentés dans cet ouvrage utilisent des sous-programmes. Il vous faut respecter la structure des répertoires présentée page 15 et vous reporter à l'annexe page 475 afin de savoir quels sont les sous-programmes indispensables au fonctionnement d'un programme principal donné. Certains petits sous-programmes ne sont pas signalés dans la table des matière. Cela signifie qu'ils sont placés dans ce livre immédiatement après le programme qu'ils complètent.

• Nous vous proposons une disquette comportant tous les programmes de ce livre. Son utilisation recquiert le kit de connexion HP/PC (câble série HP 48/PC) ou HP/Mac, le logiciel de transfert série, un ordinateur compatible IBM PC™ ou un ordinateur Macintosh™ , un lecteur de disquette au format $3^{1/2}$ pouces et, selon le type de votre connecteur série, un adaptateur 9/25 broches.

• Le nom des touches sur lesquelles il vous faut appuyer est placé entre crochets, par exemple, [ENTER] signifie "*appuyez sur la touche nommée ENTER*".

Introduction

Prêt pour le prêt-à-programmer ?

L'utilisation des programmes présentés dans ce livre vous oblige à consulter l'annexe placée en fin d'ouvrage afin de connaitre les sous-programmes d'un programme donné.

Signification des symboles utilisés dans cet ouvrage

Dans nos exemples, les touches à presser sont notées entre deux crochets.

• Les six touches grises de sélection placées sous l'écran (notées "**A**", "**B**"... "**F**") seront représentées par [**A**], [**B**], [**C**], [**D**], [**E**] et [**F**].

• Les quatre flèches de déplacement seront notées :

flèche vers le haut...	[**FH**],
flèche vers le bas...	[**FB**],
flèche vers la gauche...	[**FG**],
flèche vers la droite...	[**FD**].

• Les touches de multiplication (×) et de division (÷) seront respresentées respectivement par [*] et par [/].

• Les touches colorées en orange et en bleu sur les HP48 s et sx permettant d'accéder aux fonctions secondaires de chaque touche seront représentées dans notre propos par [ORANGE] et [BLEU]. De la

même manière, nous les représenteront par [VIOLET] et [VERT] pour les HP48 g et gx.

1
UTILISATION DE LA HP 48

Les paragraphes suivants sont destinés à vous rappeler les bases du fonctionnement de votre calculateur.

1
CREER UN OBJET

Un objet consiste en un élément quelconque manipulé par la machine qu'il s'agisse d'un programme, d'une matrice ou encore d'une chaîne de caractères.

1
CREER UNE CHAINE DE CARACTERES

Appuyer sur [BLEU] puis [-] sur les HP48 s et sx. Réciproquement appuyez sur [VERT] puis [-] sur les HP48 g et gx.

Sur la ligne de commande apparaît :

" "

Appuyer ensuite sur [α] une fois pour taper un seul caractère ou deux fois si vous désirez en taper plusieurs :

Par exemple : [α] [α] [A] [B] [C] [ENTER]

Les touche A, B, C sont sur la première ligne. Les lettres de l'alphabet sont écrites au coin inférieur droit de chaque touche.
On obtient le résultat suivant:

```
1:  "ABC"
```

2
CREER UN PROGRAMME

Appuyer sur [ORANGE] puis [-] sur les HP48 s et sx. Réciproquement appuyez sur [VIOLET] puis [-] sur les HP48 g et gx.

Sur la ligne de commande apparaît :

```
«
»
```

Vous pouvez soit utiliser l'alphabet par le même principe que pour les chaînes de caractères, ou bien utiliser les autres touches du calculateur et les menus.

Par exemple:
Créons un programme qui calcule 'ASINH(x)'

sur les HP48 s et sx :

```
[ORANGE]  [-]  [MTH]  [C]  [B]  [ENTER]
```

sur les HP48 g et gx :

```
[VIOLET]  [-]  [MTH]  [D]  [B]  [ENTER]
```

On obtient alors le programme:

```
« ASINH »
```

3
CREER UNE MATRICE OU UN VECTEUR

a) En ligne de commande:

Tapez [ORANGE] [*] sur les HP48 s et sx, ou [VIOLET] [*] sur les HP48 g et gx. Puis entrez les différentes composantes.
Par exemple:

[ORANGE] [*] [1] [SPC] [2] [ENTER]
sur les HP48 s et sx.

[VIOLET] [*] [1] [SPC] [2] [ENTER]
sur les HP48 g et gx.

donne :

```
[ 1 2 ]
```

[ORANGE] [*] [ORANGE] [*] [1] [SPC] [2]
[FD] [3] [SPC] [4] [ENTER]
sur les HP48 s et sx.

[VIOLET] [*] [VIOLET] [*] [1] [SPC] [2]
[FD] [3] [SPC] [4] [ENTER]
sur les HP48 g et gx.

donne :

```
[[1 2]
[3 4]]
```

b) Avec l'éditeur de matrices:

Entrez dans l'éditeur avec...

[BLEU] [ENTER] sur les HP48 s et sx
[VERT] [ENTER] sur les HP48 g et gx

On se déplace d'une case à l'autre grâce aux flèches...

[FH] [FB] [FG] [FD]

On tape ensuite la valeur désirée puis [ENTER].
On sort de l'éditeur de matrices en tapant une seconde fois [ENTER].

2
STOCKER UN OBJET

Une fois l'objet créé, on peut l'enregistrer. Pour cela, il faut d'abord lui choisir un nom:
Considérons la chaîne "ABC"

[α] [α] [A] **[B] [C] [ENTER]**

Supposons que le cet objet ait pour nom ESSAI

Appuyez sur :

['] [α] [α] **[E] [SIN] [SIN] [A] [CST] [ENTER]**

Puis tapez:

[STO]

L'objet est maintenant enregistré.
Il doit apparaître en bas de l'écran lorsque l'on appuie sur

[VAR]

On peut aussi rappeler la chaîne en tapant...

['] [α] [α] **[E] [SIN] [SIN] [A] [CST] [ENTER] [EVAL]**

ou en appuyant sur la touche blanche située en dessous de son nom (affiché en appuyant sur **[VAR]**).

3
VERIFIER UN OBJET

Dans le but de vérifier que vous n'avez pas commis d'erreur en rentrant les programmes, un CHECKSUM a été effectué :

Considérons que le programme « **1000 1 BEEP** » soit créé et stocké sous le nom de 'BIP

sur les HP48 s et sx :

```
[ORANGE] [-] [1] [0] [0] [0] [SPC] [1]
[PRG] [D] [NXT] [NXT] [E] [ENTER]
```

sur les HP48 g et gx :

```
[VIOLET] [-] [1] [0] [0] [0] [SPC] [1]
[PRG] [NXT] [B] [NXT] [A] [ENTER]

['] [α] [α] [B] [CST] [FG] [ENTER] [STO]
```

Vérifions qu'il n'y a pas d'erreur :
Tapez...

```
['] [α] [α] [B] [CST] [FG] [ENTER]
[ORANGE] [VAR] [B]  sur les HP48 s et sx
[VIOLET] [VAR] [B]  sur les HP48 g et gx
```

Le résultat est :

```
# 4092 h    (ou # 16530 d)
33
```

Ces deux nombres sont indiqués à côté des programmes et vous permettent de contrôler que vous n'avez pas commis d'erreur.

4
MODIFIER UN OBJET

La modification s'obtient en appuyant sur **[FB]**.

Le fonctionnement est similaire à celui de la création des objets. On se déplace avec les 4 flèches

[FH] [FB] [FG] [FD]

5
LES TOUCHES

Sur la HP 48, les touches ont de nombreux effets et tous ne sont pas signalés sur le clavier.

Tout d'abord, les caractères écrits en blanc dans l'angle inférieur droit des touches s'obtient en actionnant [α] puis la touche correspondante

Les caractères alphabétiques minuscules s'obtiennent en tapant sur [α] puis [ORANGE] puis la touche correspondante sur les HP48 s et sx. Sur les modèles g et gx, il faut taper [α] [VIOLET] puis la touche correspondante.

Pour les HP48 s et sx :
Les caractères écrits en orange au dessus des touches s'obtiennent en tapant sur [ORANGE] puis la touche correspondante.
Les caractères écrits en bleu au dessus des touches s'obtiennent en tapant sur [BLEU] puis la touche correspondante.

Pour les HP48 g et gx:
Les caractères écrits en violet au dessus des touches s'obtiennent en tapant sur [VIOLET] puis la touche correspondante.
Les caractères écrits en vert au dessus des touches s'obtiennent en tapant sur [VERT] puis la touche correspondante.

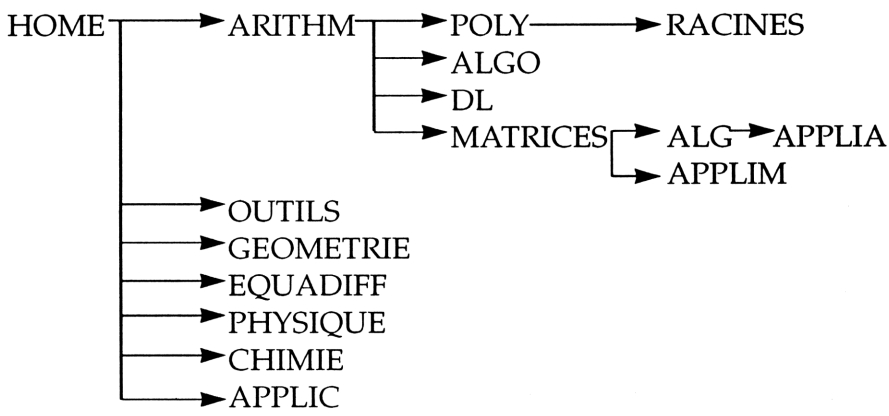
La liste qui suit concerne spécialement les HP48 s et sx. La liste des caractères des HP48 g et gx, s'affiche par [VERT] [PRG].

CARACTERES NON SIGNALES SUR LE CLAVIER DE LA 48

CARACTERE	*CE QU'IL FAUT SAISIR*
Caractère grec alpha minuscule (α)	[α] [BLEU] [A]
Caractère grec bêta minuscule (β)	[α] [BLEU] [B]
Caractère grec delta majuscule (Δ)	[α] [BLEU] [C]
Caractère grec delta minuscule (δ)	[α] [BLEU] [D]
Caractère grec epsilon (ε)	[α] [BLEU] [E]
Caractère grec thêta (ϑ)	[α] [BLEU] [F]
Caractère grec gamma minuscule (γ)	[α] [BLEU] [MTH]
Caractère grec êta minuscule (η)	[α] [BLEU] [PRG]
Caractère mathématique infini (∞)	[α] [BLEU] [CST]
Barre verticale (|)	[α] [BLEU] [VAR]
Flèche vers le haut (↑)	[α] [BLEU] [FH]
Caractère grec lambda minuscule (λ)	[α] [BLEU] [NXT]
Caractère grec mu minuscule (μ)	[α] [BLEU] [STO]
Caractère grec oméga majuscule (Ω)	[α] [BLEU] [EVAL]
Flèche vers la gauche (←)	[α] [BLEU] [FG]
Flèche vers le bas (↓)	[α] [BLEU] [FB]
Caractère grec rhô minuscule (ρ)	[α] [BLEU] [FD]
Caractère grec sigma minuscule (σ)	[α] [BLEU] [SIN]
Caractère grec tau minuscule (τ)	[α] [BLEU] [COS]
Caractère %	[α] [BLEU] [TAN]
Caractère tilde (˜)	[α] [BLEU] [√X]
Caractère grec oméga minuscule (ω)	[α] [BLEU] [Yx]
Caractère x surligné	[α] [BLEU] [1/x]
Caractère ±	[α] [BLEU] [+/-]
Caractère grec pi majuscule (Π)	[α] [BLEU] [EEX]
Caractère espagnol ¡	[α] [BLEU] [DEL]
Caractère espagnol ¿	[α] [BLEU] [<=]
Caractère @	[α] [BLEU] [ENTER]
Caractère &	[α] [ORANGE] [ENTER]
Caractère !	[α] [ORANGE] [DEL]
Caractère ?	[α] [ORANGE] [<=]
Caractère ¢	[α] [BLEU] [4]
Caractère ¥	[α] [BLEU] [5]
Caractère °	[α] [BLEU] [6]

Caractère π	[α] [BLEU] [1]
Caractère >	[α] [BLEU] [2]
Caractère ≥	[α] [BLEU] [3]
Accent aigu (sur e, i, u, o, a)	[α] [BLEU] [7]
Tilde (sur n) (ñ)	[α] [STO] [A] [BLEU] [8]
æ (sur e)	[α] [E] [A] [BLEU] [9]
å (sur a)	[α] [α] [A] [BLEU] [9]
ç (sur c)	[α] [C] [A] [BLEU] [9]
caractères spéciaux (sur d,p)	[α] [BLEU] [9]
caractère grec phi (ϕ)	[α] [EVAL] [A] [BLEU] [9]
accent grave (sur e, i, u, o, a)	[α] [ORANGE] [7]
accent circonflèxe (^)	[α] [ORANGE] [8]
tréma (¨)	[α] [ORANGE] [9]
Caractère $	[α] [ORANGE] [4]
Caractère £	[α] [ORANGE] [5]
Caractère o	[α] [ORANGE] [6]
Caractère ==	[α] [ORANGE] [1]
Caractère <	[α] [ORANGE] [2]
Caractère ≤	[α] [ORANGE] [3]

2
STRUCTURE DES REPERTOIRES



Il est important de respecter les noms et l'ordre des répertoires.

Pour créer ces répertoires, effectuez les opérations ci-après.

```
HOME
'ARITHM'
CRDIR
[VAR]
```

puis appuyez sur la touche se trouvant en dessous de ***ARITHM.***

```
'POLY'
CRDIR
[VAR]
```

puis appuyez sur la touche se trouvant en dessous de ***POLY.***

```
'RACINES'
CRDIR

ARITHM

'ALGO'
CRDIR

'DL'
CRDIR

'MATRICES'
CRDIR

[VAR]
```

puis appuyez sur la touche se trouvant en dessous de ***MATRICES.***

```
'ALG'
CRDIR

'APPLIM'
CRDIR

[VAR]
```

puis appuyez sur la touche se trouvant en dessous de ***ALG.***

```
'APPLIA'
CRDIR
HOME

'OUTILS'
CRDIR

'GEOMETRIE'
CRDIR

'EQUADIFF'
CRDIR

'PHYSIQUE'
CRDIR

'CHIMIE'
CRDIR

'APPLIC'
CRDIR
```

3
REGLAGES DE LA HP 48

Ces réglages permettent d'obtenir les résultats sous la forme que l'on désire. Vous pourrez par exemple exécuter la commande qui suit :

```
{ #80038400014FF2h #0h }
STOF
```

L'effet de cette commande est de modifier l'état des "flags". Ils peuvent être individuellement réglés à l'aide des commandes SF et CF (Set Flag et Clear Flag).

4
LES NOUVEAUTES DES HP 48g et gx

Sur les HP48 g et gx, `[ORANGE]` est devenu `[VIOLET]` et `[BLEU]` est devenu `[VERT]`. Mais ce ne sont pas les seules modifications !

Notons que la HP48g/gx a une vitesse d'horloge doublée par rapport à la HP48s/sx. Le gain de temps n'est environ que de 40%, car la gestion de la mémoire est plus prenante que sur les modèles précédents.

En effet, sur la HP48gx, seul le port 1 peut avoir sa mémoire fusionnée. Le deuxième emplacement est un emplacement à haute capacité : on peut y mettre jusqu'à 4 méga-octets. Ainsi, la HP48gx peut contenir jusqu'à 4224 Ko de mémoire.

Notons aussi que l'utilisateur de la HP48g/gx peut fabriquer lui-même des variables locales. Pour cela, reportez vous au manuel de référence.

Il y a eu aussi quelques modifications dans les adresses internes du calculateur. Nous en avons tenu compte dans la rédaction de cet ouvrage, en indiquant si nécessaire une deuxième version des programmes.

Note :
Dans les chapitres qui suivent, les temps de calculs indiqués sont ceux d'une HP48sx. Le temps serait environ réduit de moitié pour une HP48 g ou gx.

1

Programmes de base

Ce chapitre rassemble des programmes qui pourront servir à tout moment lors de l'utilisation du calculateur. Ils ne serviront pas uniquement pour les programmes mathématiques.

Pour cette raison nous vous conseillons de les mettre dans le répertoire principal (HOME) de votre HP 48.

Les programmes vous sont présentés dans l'ordre suivant :

- **LOOK** permet de regarder rapidement un texte.
- **FIND** permet de chercher un mot dans les fichiers.
- **PGM→** décompose un programme.
- **→PGM** recompose un programme.
- **ASS** pour fabriquer des programmes rapides.
- **SYS** rappelle un objet en mémoire.
- **DASS** pour modifier les programmes assemblés.
- **ERREUR** qui gère l'aide en ligne de tous les programmes de cet ouvarge.

L'utilisation du programme ERREUR est facultative. Il permet de connaitre la syntaxe d'utilisation des programmes de cet ouvrage.

1

Visualisation de fichiers texte *LOOK*

Ce programme permet de visualiser de manière simple les fichiers de texte.
On se déplace dans le texte avec les flèches ainsi qu'avec les touches [+] et [-].
Les flèches permettent de se déplacer lentement dans le texte tandis que [+] et [-] permettent de se déplacer plus vite.
La touche [+/-] permet de rechercher un mot dans le fichier.
La touche [NXT] permet de quitter la visualisation du fichier texte.

Ce programme est à enregistrer sous le nom LOOK.
Son checksum est #66E1h (659 octets).

Listing

```
«
IF DEPTH NOT
THEN  "LOOK Error:
Too Few Arguments"  DOERR
ELSE
IF DUP TYPE 2 ≠
THEN  "LOOK Error:
Bad Argument Type" DOERR
END
END 0 SWAP → M
« DO DUP M SWAP 20 * DUP 200 + SUB
CLLCD 1 DISP 7 FREEZE 0 WAIT
.1 - → MES
«  IF MES 25 SAME THEN 1 - END
IF MES 35 SAME THEN 1 + END
IF MES 34 SAME THEN 3 - END
IF MES 36 SAME THEN 3 + END
```

```
IF MES 26 SAME THEN DROP 0 DOERR END
IF MES 95 SAME THEN 9 + END
IF MES 85 SAME THEN 9 - END
IF MES 52 SAME THEN DUP
"Mot à rechercher" { "" α }
INPUT M SWAP 3 ROLLD SWAP 10 * 10
+ 99999 SUB
SWAP POS 10 / +  END
»
UNTIL 0 END
»
»
```

Lancement du programme :
Rappeler la chaîne de caractères dans la pile et lancer le programme par :

```
LOOK [ENTER].
```

Exemple d'utilisation :

```
'LOOK' [ENTER]
RCL
→STR
LOOK [ENTER]
```

Vous pouvez ainsi regarder le programme que vous venez de taper.

Remarque :
Pour saisir ... "LOOK Error:
Too Few Arguments" ...

Il faut taper sur une HP48 s ou sx :

```
[BLEU] [-] [NXT] [EVAL] [EVAL] [FH]
[SPC] [E] [ORANGE] [FD] [ORANGE] [FD]
[ORANGE] [EVAL] [ORANGE] [FD]
[BLEU] [.]
[COS] [ORANGE] [EVAL] [ORANGE] [EVAL]
```

```
[SPC] [F] [ORANGE] [E] [ORANGE] [Y^X]
[SPC] [A] [ORANGE] [FD] [ORANGE] [MTH]
[ORANGE] [TAN] [ORANGE] ['] [ORANGE]
[E] [ORANGE] [STO] [ORANGE] [COS] [FD]
```

Il faudra procéder de la même manière pour saisir "LOOK Error: Bad Argument Type" et dans chaque programme où il y aura du texte avec des sauts de lignes.

2
Recherche d'une chaîne *FIND*

Ce programme recherche un mot ou des lettres dans toutes les chaînes de caractères du répertoire.

Ce programme est à enregistrer sous le nom : FIND.

Son Checksum est : #CCF1h (297,5 octets)

Listing

```
  « IF DEPTH NOT THEN
"Entrez le mot clef à
rechercher."
DOERR END
CLLCD "FIND
Recherche de mot clef
→ Scanning
──────────

──────────"
```

```
1 DISP VARS → M V
« 1 V SIZE FOR I IFERR V I GET DUP 5
DISP RCL M POS
THEN DROP DROP
ELSE IF 0 ≠ THEN V I GET END END
NEXT
»
»
```

Lancement du programme :
Entrer la chaîne de caractère à rechercher puis taper

```
FIND
```

Exemple d'utilisation :

```
"BINET" [ENTER]
FIND
```

Cet exemple recherche les fichiers dans lesquels le mot BINET apparaît.

Explication :
Le programme recherche toutes les chaînes de caractères se trouvant dans le répertoire où l'on se trouve. Il donne le nom des fichiers qui contiennent le mot-clef recherché.

Remarque :
On peut ensuite rappeler ces fichiers en tapant **[BLEU] [STO]** puis les regarder avec LOOK (cf programme précédent)

Possibilités d'extension :
Il serait possible de taper un programme qui recherche dans tous les répertoires du calculateur le mot-clef désiré.

3
Décomposition d'un programme *PGM→*

Cet utilitaire décompose un programme.

Ce programme est à enregistrer sous le nom : PGM→
en tapant :

```
'PGM→' [STO]
```

après avoir saisi le programme.
Son checksum est : #92E5h (72 octets)

Listing:

```
« DUP IF TYPE 8 SAME THEN
# 54AFh SYSEVAL # 18DBFh SYSEVAL END »
```

Lancement du programme :
Rappeler un programme et taper

```
PGM→ [ENTER].
```

Exemple d'utilisation :

```
« 1000 1 BEEP » [ENTER]
PGM→ [ENTER]
```

Ceci décompose le programme « 1000 1 BEEP » comme le ferait avec une liste la commande LIST→.
Ce programme fait appel à deux commandes internes de la machines grâce à la fonction SYSEVAL. Attention ! N'utilisez pas vous-même cette commande en dehors des programmes situés dans ce livre , si vous n'en connaissez pas les effets.

4
Recomposition d'un programme →PGM

Cet utilitaire recompose un programme.

Ce programme est à enregistrer sous le nom : →PGM
en tapant après avoir saisi le programme :

```
'→PGM' [STO]
```

Son checksum est: #6E2Ch (74.5 octets)

Listing

```
« DUP DEPTH 1 - IF  <  THEN
# 18CEAh SYSEVAL # 5445h SYSEVAL END »
```

Lancement du programme :
De manière similaire à la commande →LIST, placer des éléments dans la pile et leur nombre comme dernier élément puis taper :

```
→PGM [ENTER]
```

Exemple d'utilisation :

```
« 1000 1 BEEP » [ENTER]
PGM→ [ENTER]

→PGM [ENTER]
```

Ceci décompose le programme « 1000 1 BEEP » puis recompose le programme.

Explication :
Ce programme fait appel à deux commandes internes de la machines grâce à la fonction SYSEVAL. Attention ! N'utilisez pas vous-même cette commande en dehors des programmes situés dans ce livre, si vous n'en connaissez pas les effets.

5
Compilation de chaînes *ASS*

Ce programme assemble les chaînes de caractères désignant un objet décompilé autrement dit, il le compile. Notez que ce programme n'utilise pas lui-même d'assembleur car il se sert de routines internes au calculateur concernant les objets graphiques.

Ce programme est à enregistrer sous le nom : ASS.

Son checksum est : #DA72h (112 octets).

Listing

```
« DUP SIZE 2 / .5 + IP "GROB 1 " SWAP +
" " + SWAP + OBJ→ # 4017h
SYSEVAL # 56B6h SYSEVAL DROP NEWOB »
```

Lancement du programme :

Entrer une chaîne de caractères compris entre 0 et 9 ou A et F. Cette chaîne doit correspondre à un objet !

Puis taper :

```
ASS [ENTER]
```

Exemple d'utilisation :

```
"47A20B2130" ASS
```

On obtient une liste vide.

Explication :
Ce programme compile n'importe quelle chaîne de caractères.
Il est donc important de vérifier que l'on a pas fait d'erreurs dans la saisie de la chaîne.

6
Rappel d'un objet en mémoire *SYS*

Ce programme rappelle l'objet en mémoire à l'adresse spécifiée sans l'évaluer s'il n'est pas un objet standard. Ceci vous permettra de ne plus perdre vos données, ce qui peut arriver en utilisant SYSEVAL.
Cette nouvelle version du programme SYS, qui fonctionne sur tous les différents types de HP48 nécessite le programme ASS précédent.

Ce programme est à enregistrer sous le nom : SYS.
Son checksum est : #514Eh (94 octets) pour la version HP48s/sx
Son checksum est : #C966h (97 octets) pour la version HP48g/gx

Version HP48 s et sx

Listing :

```
« IF DUP TYPE 10 == THEN #5A03h SYSEVAL
#C612h SYSEVAL  »
```

Ce programme permet à la HP48 s ou sx d'accéder aux adresses entre #70000h et #80000h. Reportez-vous au chapitre 15 pour en savoir plus.

Version HP48 g et gx

Listing :

```
« IF DUP TYPE 10 == THEN #5A03h SYSEVAL
#9E1Ah SYSEVAL DROP #715B1h SYSEVAL END
»
```

Lancement du programme :

Placer un entier binaire puis taper :

```
SYS [ENTER]
```

Exemple d'utilisation :

Taper :

```
#3223h SYS
```

renvoie `External`. Il s'agit de la commande SWAP mais non évaluée. Si vous tapez

```
2 [ENTER]
3 [ENTER]
#3223h SYS [ENTER]
EVAL [ENTER]
```

alors, vous obtiendrez le résultat attendu :

```
3
2
```

7
Décompilation d'un objet *DASS*

L'objet peut ensuite être recompilé avec ASS.

Ce programme est à enregistrer sous le nom : DASS.

Rentrez la chaîne suivante SANS ESPACE et SANS SAUT DE LIGNE.
Une fois la chaîne tapée, entrez DUP BYTES.

Le résultat doit être # 439Fh (ou # 17311d) et 166.
Puis taper :

```
ASS
```

(on utilise ASS pour compiler le programme)
Vous obtiendrez un External.

Son checksum est : #B366h (89 octets).

Listing

```
"D9D 20D 295 188 130 209 50F D55 02C 230
F6E 30C 1C1 632 230 CCD 205 600 08F 146
60C C8F B97 60D 814 313 016 917 414 313
1AE 215 F08 082 103 A62 808 219 39E EC0
808 217 0A6 214 C16 117 0CD 5DC 8D3 415
0B9 F06 B21 30"
```

Lancement du programme :
Placer un élément dans la pile puis taper

```
DASS
```

Exemple d'utilisation :
Saisir :

```
'DASS' RCL
DASS
```

Vous obtiendrez la chaîne de caractères que vous venez de taper.

Remarque :

pour comprendre le fonctionnement de ce programme, nous vous proposons de lire son listing.

Listing du programme DASS

```
02D9D       Entête de Programme
1592D       Efface la commande et
            vérifie DEPTH ≥ 1
03188       DUP Interne
05902       BYTES Interne (taille) →
            (Binaire Système)
055DF       ""
032C2       OVER Interne
03E6F       *2 Interne (Binaire Système)
61C1C       alloue taille
            (2:objet,1:Binaire Système)
03223       SWAP Interne
02DCC       Entête de Code
00065       Taille : 101 quartets
```

8F14660	GOSBVL	06641	Met TOS dans A
CC	A=A-1	A	A=taille de l'objet - 1
8FB9760	GOSBVL	0679B	Sauve les registres
D8	B=A	A	B=taille de l'objet - 1
143	A=DAT1	A	A=adresse de la chaine
130	D0=A		D0=adresse de la chaine
169	D0=D0+	10	saute entête et taille
174	D1=D1+	5	Se positionne sur l'objet
143	A=DAT1	A	A=adresse de l'objet
131	D1=A		D1=adresse de l'objet

A ce niveau, D0 pointe sur le premier caractère de l'objet de destination et D1 pointe sur l'objet à desassembler. De plus le registre B contient la taille de l'objet - 1.

```
      @1
AE2         C=0         B       Met C à zéro (champ B)
15F0        C=DAT1      1       Lit un quartet de l'objet
8082103     LAHEX       30      place 48  en décimal dans A
A62         C=C+A       B       Additionne 48 + quartet lu
8082193     LAHEX       39      place 57  en décimal dans A
9EEC0       ?A>=C       B       Est-ce que A≥C
            GOYES       @2      Si oui va en @2
8082170     LAHEX       07      place 7 dans A
A62         C=C+A       B       Additionne 55 + quartet lu
      @2
14C         DAT0=C      B       Met C.B  dans la chaine
161         D0=D0+      2       Saute 1 octet dans la chaine
170         D1=D1+      1       Saute un quartet dans l'objet
CD          B=B-1       A       décrémente B
5DC         GONC        @1      si B≥0 va en @1
8D34150     GOVLNG      05143   Récupération des registres
                                        Fin du code
```

```
60F9B SWAP et DROP Interne
```

A ce niveau, il ne reste plus que la chaine de caractère de l'objet désassemblé.

```
0312B Fin d'objet
```

Le fonctionnement du programme est le suivant :

- on met le nombre de quartets de l'objet dans B. C'est ce que l'on décomptera pour savoir si le travail est terminé.
- On se positionne par D0 dans la chaine de caractères et par D1 dans l'objet.
- On lit un quartet de l'objet.

- Si celui-ci a un code inférieur ou égal à 9, cela voudra dire que l'on calcule dans C : 48+nombre. Dans ce cas, on saute au cas @2. Sinon, on calcule 48+7+nombre=55+nombre.

La raison est simple : il y a une "rupture" alphanumérique entre 9 et A, qui se suivent dans le décompte héxadécimal. Ainsi 48+0=48 code "0", 49 code "1", ..., 57 code "9" et par la suite 55+10=65 code"A", etc...

NB : TOS signifie Top of Stack, c'est-à-dire l'adresse de premier objet dans la pile.

Le champ B est un champ à deux quartets.
Reportez-vous au chapitre sur l'assembleur pour plus de renseignements.

8
Gestion des erreurs
ERREUR

Ce programme gère l'aide en ligne de tous les programmes mathématiques qui vont suivre. Son but est d'aider l'utilisateur qui ne se souvient plus le jour J, comment il doit utiliser les programmes correspondants.

Ce programme est à enregistrer sous le nom : ERREUR

Le checksum est #B086h (782 octets).

Listing :

```
« { "Entrez le polynôme et
la valeur"
"Entrez param, opérateur
et les deux polynômes"
```

```
"Entrez les 2 polynômes"
"Entrez la constante et
le polynôme"
"Entrez les 2 polynômes
et l'ordre"
"Entrez le polynôme"
"Entrez un entier"
"Entrez les 2 DL
et l'ordre des DL"
"Entrez un DL"
"Entrez DL, puissance
et l'ordre"
"Entrez l'ordre des DL"
"Entrez l'exposant et
l'ordre du DL"
"Entrez le DL et
l'ordre"
"Entrez l'ordre du DL"
"Entrez la matrice"
"Entrez le dénominateur
et la matrice"
"Entrez les 2 matrices"
"Entrez la matrice et
la constante"
"Entrez le vecteur sol
et la matrice"
"Entrez les 2 matrices
et l'opérateur"
"Pas assez d'arguments"
"Entrez Y' fn de X et Y
xmin, xmax, et y(xmin)"
}
SWAP GET SWAP IF DEPTH 2 - ≥ THEN DROP
ELSE DOERR END »
```

Utilisation et remarque :

Ce programme sera utilisé par les programmes mathématiques des chapitres suivants. Il prend comme premier argument la taille minimum nécessaire de la pile, et comme deuxième argument le numéro

du message d'erreur à afficher.
Si vous désirez ne pas utiliser ce programme, remplacez le tout simplement par le programme suivant :

```
« DROP2 »
```

Vous n'aurez alors plus la possibilité d'avoir une aide en ligne, en utilisant chaque programme sans arguments.

9 EXEMPLE

Créons le programme « **SWAP DROP** ».

1ère solution : nous pouvons le taper tel quel : dans ce cas précis c'est facile.

2ème solution : nous pouvons le taper en RPL accéléré.
Cependant ce mode de programmation présente un danger car il n'y a pas de protection si les paramètres ne sont pas les bons.
Ici , nous allons placer nous mêmes une protection :
Pour cela nous avons l'adresse internal **18A8Dh et 60F9Bh** *(cf chapitre 17)*

18A8Dh	Vérifie DEPTH ≥ 2
60F9Bh	SWAP et DROP Interne.

Saisir alors :

```
#18A8Dh     SYS
#60F9Bh     SYS
2 →PGM
```

Nous aurions aussi pu taper *:*

```
"D9D20D8A81B9F06B2130"   [ENTER]
ASS
```

2
Arithmétique

Il nous a semblé important de favoriser l'arithmétique dans les calculs. En effet, quasiment tous les calculs qui nous sont donnés à effectuer sont dans $\mathbb{Q}$ *(corps des nombres rationnels).*

Nous avons décidé d'écrire les rationnels comme un couple de réels c'est-à-dire sous forme "complexe". Par exemple, 1/3 sera représenté par (1,3). Pour faciliter la lecture de ces nombres, la librairie d'affichage qui se trouve dans le chapitre des applications, à la fin de cet ouvrage, définit un nouveau mode d'affichage des nombres "complexes". Ce mode d'affichage sera automatiquement activé grâce au programme KEYQ de ce chapitre.

Il faudra cependant prendre garde à ne pas confondre les différents types d'opérations. C'est dans ce but que KEYQ redéfini le clavier.

Remarque:

De plus, nous avons cherché à améliorer la vitesse d'exécution des programmes de ce chapitre ainsi qu'à réduire la place qu'ils occupent sur le calculateur. Ainsi, pour chaque programme RPL de ce chapitre, suivra un programme à base d'adresses SYSEVAL (Internals). Vous pourrez utiliser l'une des deux versions au choix.
Avec la version à base d'internals, nous gagnons environ 50% de temps et 20% d'espace mémoire utilisé.

Les programmes sont classés de la manière suivante :

Opérations :

- **DECOMP** Décomposition en facteurs premiers
- **EDIVI** *Sous-programme de DECOMP*
- **PGCD** PGCD de deux entiers
- **PPCM** PPCM de deux entiers
- **R→Q** Transformation de fraction
- **Q→R** Transformation de nombres en fraction
- **QPLUS** Addition de deux fractions
- **QMOINS** Différence de deux fractions
- **QMULT** Produit de deux fractions
- **QDIV** Quotient de deux fractions
- **QSIMPL** Réduction au même dénominateur
- **QPUIS** Exponentiation d'une fraction
- **QNEG** Opposé d'une fraction rationnelle
- **QTEST** Détermination du plus grand nombre
- **QSQ** Carré d'une fraction
- **QINV** Inverse d'une fraction

Redéfinition du clavier de la HP 48 :
Le but est d'automatiser les opérations dans ce répertoire.

- **KEYQ** Redéfinition du clavier de la HP 48

1

Décomposition en facteurs premiers *DECOMP*

Ces deux programmes ont été faits suite au travail de Monsieur Guy Toublanc.

α - Sous-programme EDIVI

Ce sous-programme est utilisé par DECOMP.
Ce programme est à enregistrer sous le nom : EDIVI.

Listing

ATTENTION : Il faut rentrer ce listing sans espaces ni sauts de ligne puis le compiler avec ASS. Avant de le compiler, taper DUP BYTES : le résultat doit être # B2ABh et 309.

C'est-à-dire que l'on tape `[BLEU] [-] [α] [α] D9D2078BF1BB` etc...

```
"D9D 207 8BF 1BB 691 473 B1B 969 1DB BF1
CCD 201 010 070 801 001 351 748 FB9 760
7C6 010 211 097 8E1 AF2 302 AF5 0D9 08B
F80 CFA 7C9 7C8 0E4 63A 073 50B 657 C40
754 071 407 A30 793 072 307 130 7A2 073
207 520 7B1 011 2B1 059 D40 614 713 717
915 BF0 1A1 1A1 1A1 180 DF1 10A C39 0DB
0B9 1B4 754 FB1 05C FA1 0A4 F4F 0B9 590
CBE 0D5 6E2 891 C00 A94 118 976 50A 901
59F 208 F2D 760 E71 421 648 08C BB6 91B
213 0"
```

Lancement du programme EDIVI :
Il s'agit d'un sous-programme de DECOMP que vous n'avez pas à utiliser seul, le programme DECOMP l'utilise et lance son exécution de façon transparente pour l'utilisateur.

β - programme principal

Ce programme est à enregistrer sous le nom : DECOMP.

Son checksum est : #B3EAh (250.5 octets)

Listing

```
« DUP DUP #0h == SWAP #1h == +
IF THEN 1 →LIST ELSE { } SWAP
WHILE DUP EDIVI DUP DUP 1 ≠ *
REPEAT DO ROT OVER 1 →LIST  +
SWAP ROT OVER / SWAP UNTIL DUP2
DUP2 / * - B→R END DROP END
DROP DUP #1h == « DROP LIST→ SWAP
R→B SWAP →LIST »
« 1 →LIST + » IFTE END »
```

Lancement du programme :

On tape un entier binaire puis on éxécute :

```
DECOMP
```

Exemples d'utilisation :

```
#125d [ENTER]
DECOMP [ENTER]
```

renvoie

```
{ 5  5  #5d  }
```

Ceci signifie que 125=5*5*5

```
#123456789d [ENVOI]
DECOMP [ENVOI]
```

renvoie

```
{ 3  3  3607  #3803d  }
```

donc 123456789=3*3*3607*3803
et 3607 et 3803 sont des nombres premiers.

Remarque :

A titre d'information, nous vous fournissons le listing détaillé du programme EDIVI. Il est assez compliqué et les adresses des sauts à l'intérieur du programme sont symbolisées par @.

Listing du sous-programme EDIVI

Le sous-programme EDIVI prend un entier binaire au niveau 1 et renvoie un diviseur de cet entier, ou 0 si cet entier est premier.

```
02D9D   Entête Programme
1FB87   DUP
196BB   B→R
1B374   SQRT
1969B   R→B
1FBBD   SWAP
02DCC   Entête de Code
00103   Taille : 257 quartets
```

7080	GOSUB	@1	A.W=entier binaire du niveau1
100	R0=A		R0=n=entier à décomposer
135	D1=C		Récupère D1
174	D1=D1+	5	D1 pointe sur le niveau 2
8FB9760	GOSBVL	#0679B	Sauve registres B,D,D0,D1
7C60	GOSUB	@1	A.W=entier du niveau2=√n
102	R2=A		R2=racine de l'entier à décomposer
110	A=R0		A=n
978E1	?A=0	W	n est-il nul ?
	GOYES	@2	si oui, renvoyer 1 comme résultat
AF2	C=0	W	
302	LCHEX	#2	C.W=2
AF5	B=C	W	B.W=2

La boucle suivante cherche le premier quartet non nul de A et le

place dans P et C.S. Ceci va permettre de calculer avec le champ WP et de ne pas manipuler plus de quartets que nécessaire.
Remarque : ici, on a P=0, donc le premier P=P-1 donnera P=15.

```
      @3
0D          P=P-1                   quartet précédent
908BF       ?A=0        P           A est-il nul ?
            GOYES       @3          si oui, recommencer
80CF        C=P         15          C.S=P
A7C         A=A-1       W           A=n-1
97C80       ?A#0        W           si n-1=0, renvoyer 1
            GOYES       @4          sinon, chercher un diviseur
```

Si on arrive ici, on a A.W=0

```
      @2
E4          A=A+1       A           A.W=1
63A0        GOTO        @5          renvoyer A.W
      @4
7350        GOSUB       @6          teste si B | n (B=2)
B65         B=B+1       B           B.W=3
7C40        GOSUB       @6          teste si B | n (B=3)
7540        GOSUB       @7          B=B+2 et teste si B | n (B=5)
7140        GOSUB       @7          B=B+2 et teste si B | n (B=7)
```

Quand on arrive ici, B est tel que 2|B-1, 3|B-1 et 5|B-2. Comme 30 est multiple de 2, 3 et 5 (c'est leur ppcm), B+30 vérifiera les mêmes conditions. D'après ces conditions, les B+k avec k dans {1,2,3,5,7,8,9,11,13,14,15,17,18,19,21,23,25,26,27,28,29} sont divisibles par 2, soit par 3, soit par 5, donc ils ne divisent pas n. Il suffit donc de tester les B+k avec k dans {4,6,10,12,16,22,24,30} pour trouver les diviseurs de n compris entre B+1 et B+30. C'est ce qui est fait dans les lignes qui suivent.

```
      @10
7A30        GOSUB       @8          B=B+4 et teste si B|n
7930        GOSUB       @7          B=B+2 et teste si B|n
7230        GOSUB       @8          B=B+4 et teste si B|n
7130        GOSUB       @7          B=B+2 et teste si B|n
7A20        GOSUB       @8          B=B+4 et teste si B|n
7320        GOSUB       @9          B=B+6 et teste si B|n
```

```
7520        GOSUB       @7          B=B+2 et teste si B | n
7B10        GOSUB       @9          B=B+6 et teste si B | n
112         A=R         2           A=√n
B10         A=A-B       WP          si B≤√n
59D         GONC        @10         recommencer
                                    (B a augmente de 30)
406         GOC         @11         sinon n est premier
```

Remarque : si on a passer le GONC, c'est que CARRY=1, et donc le GOC se comporte comme un GOTO. L'avantage est qu'il se code sur trois quartets contre quatre pour GOTO.

Ici, on renvoie dans A.W l'entier binaire du niveau 1.

```
        @1
147         C=DAT1      A           C=adresse de l'entier
137         CD1EX                   D1 pointe sur l'entier
                                    binaire
179         D1=D1+      10          saute le prologue et la
                                    longueur
15BF        A=DAT1      16          A=entier
01          RTN                     fin du sous-programme
```

On a maintenant le sous-programme de test. Suivant qu'on y entre par @6, @7, @8 ou @9, on augmente B de 0, 2, 4 ou 6, puis on teste si B | n.

Remarque : quand on arrive ici par @7, @8 ou @9, ce sous-programme a déjà été exécuté au moins une fois, donc on a P=8. Donc la valeur de C.S n'intervient pas dans le B=B+C. Comme on a $B \leq \sqrt{n} < 2^{32}$, le champ WP avec P=8 est suffisant pour stocker B+C.

```
        @9
A11         B=B+C       WP          B=B+2
        @8
A11         B=B+C       WP          B=B+2
        @7
A11         B=B+C       WP          B=B+2
        @6
80DF        P=C         15          P=nb de quartets
                                    nécessaires pour contenir n
110         A=R0                    A=n
```

Les lignes qui suivent servent à calculer le reste de la division de A par B. Supposons que A s'écrive a(p)a(p-1)...a(0) et que B s'écrive b(q)b(q-1)...b(0), où les a(i) et les b(j) sont des chiffres héxadécimaux. On va retrancher $B*16^{p-q}$ a A autant de fois que possible, puis $B*16^{p-q-1}$, etc...
Ici, p est la valeur de P, et p-q va être calculé dans D.S.

```
AC3          D=0          S          Initialiser D.S à 0
      @13
90DB0        ?B#0         P          B.P <> 0 ?
             GOYES        @12        si oui, fini
B91          BSL          WP         décale B d'un quartet vers la
                                     gauche
B47          D=D+1        S          incrémenter D
54F          GONC         @13        recommencer (on a toujours
                                     CARRY=0 ici, car B est non
                                     nul, et la condition B.P<>0
                                     est toujours vérifiée avant
                                     D.S>15)

      @12
B10          A=A-B        WP         ! enlever a A autant
5CF          GONC         @12        ! de fois B que
A10          A=A+B        WP         ! possible
A4F          D=D-1        S          disivion terminée ?
4F0          GOC          @14        si oui, va en @14
B95          BSR          WP         décaler B d'un quartet vers
                                     la droite.
90CBE        ?A#0         P          si A.P est nul, on peut
                                     restreindre le champ pour
                                     gagner du temps.
             GOYES        @12
0D           P=P-1
56E          GONC         @12        le test A<>0 était négatif,
                                     donc CARRY=0
```

Ici, la division est terminée : A contient le reste de la division de n par B. Ce reste est inférieur à B, donc il est contenu dans A.WP pour P=8.

```
      @14
```

```
28          P=          8
91C00       ?A#0        WP          si le reste est non nul, B ne
                                    divise pas n
            RTNYES                  il faut continuer la
                                    recherche
A94         A=B         WP          A=diviseur de n
118         C=R0                    C=n
91650       ?C#A        WP          si A=C, n est premier
            GOYES       @5
      @11
A90         A=0         WP          renvoyer 0
      @5
159F        DAT1=A      16          remplace l'entier niveau2
                                    par le diviseur
20          P=          0           Remettre P à 0
8F2D760     GOSBVL      #067D2      Récupère les registres B, D,
                                    D0 et D1
E7          D=D+1             A     on fait D1=D1+5 au début,
                                    il faut indiquer qu'on a
                                    libéré un niveau (DROP)

142         A=DAT0      A           Fin du programme
164         D0=D0+      5
808C        PC=(A)

        196BB  B→R
        0312B  Fin d'objet
```

2

Calcul du PGCD de deux entiers *PGCD*

Ce programme calcule le PGCD de deux entiers.

Ce programme est à enregistrer sous le nom : PGCD.

Listing

ATTENTION : Il faut rentrer la chaîne suivante sans espaces ni sauts de ligne, puis il faut compiler la chaîne avec ASS. Faire ensuite DUP puis BYTES : le résultat doit être #2757h et 234.

```
"D9D 20F 1AA 19F 345 DBB F1F 1AA 19F 345
CCD 208 B00 014 713 717 915 371 001 351
748 FB9 760 147 137 179 157 797 A17 AFE
97A 96A E16 B10 120 101 822 81C 832 606
410 B65 101 822 81C 832 CD1 111 181 088
228 1E8 324 F11 89F A50 AFE B72 97A 606
EDF 969 C0A 74A 6D6 4FF 159 F8F 2D7 601
421 648 08C D53 45B 213 0"
```

La chaîne occupe 234 caractères. Son checksum est # 2757h. Une fois compilé le programme apparait sous la forme :

```
ABS External SWAP
ABS External Code
External
```

Lancement du programme :
Saisir deux nombres entiers, puis taper

```
PGCD
```

Exemple d'utilisation :

```
124 [ENTER]
638 [ENTER]
PGCD
```

Le calculateur *renvoie* : 2

Ce qui signifie que le Plus Grand Commun Diviseur de 124 et 638 est 2.

Temps d'éxecution : immédiat.

Annexe : Listing du programme PGCD.

```
02D9D           Entête de programme
1AA1F           ABS interne
543F9           R→B interne
1FBBD           SWAP interne
1AA1F           ABS interne
543F9           R→B interne
02DCC           Entête de code
000B8           taille 184 quartets
```

Quand le code est appelé, les niveaux 1 et 2 contiennent des entiers binaires.

```
147         C=DAT1      A       C1=adresse de l'objet 1
137         CD1EX               D1=adresse de l'objet 1
179         D1=D1+10            saute le prologue
1537        A=DAT1      W       A=entier 1
100         R0=A                R0=entier 1
135         D1=C                D1 pointe sur le niveau 1
174         D1=D1+5             D1 pointe sur le niveau 2
8FB9760     GOSBVL      0679B   sauve D0,D1,B,D
147         C=DAT1      A       C=adresse de l'objet 2
137         CD1EX               D1=adresse de l'objet 2
179         D1=D1+10            saute le prologue
1577        C=DAT1      W       C=entier 2
97A17       ?C=0        W       entier 2 = 0 ?
            GOYES       @1      si oui alors pgcd = entier 1
AFE         ACEX        W       A=entier 2, C=entier 1
97A96       ?C=0        W       entier 1 = 0 ?
            GOYES       @1      si oui alors pgcd = entier 2
AE1         B=0         B       Initialiser B à 0
```

Si p=2p' et q=2q', on a pgcd(p,q)=2*pgcd(p',q'). B va servir à compter combien de fois on peut ainsi mettre 2 en facteur.

```
6B10        GOTO        @2          aller en @2
     @5
120         AR0EX                   A=entier 1
101         R1=A                    R1=entier 1
822         SB=0                    vider SB
81C         ASRB                    A=(entier 1)/2
83260       ?SB=0                   l'entier 2 est-il pair ?
            GOYES       @3          si oui aller en @3
6410        GOTO        @4          si non aller en @4
     @3
```

Si on arrive ici, c'est qu'on a pu mettre 2 en facteur dans les deux entiers. On incrémente donc B et on continue à essayer de diviser par 2.

```
B65         B=B+1       B
     @2
101         R1=A                    R1=entier 2
822         SB=0                    vider SB
81C         ASRB                    A=(entier 2)/2, SB=reste
832CD       ?SB=0                   l'entier 2 est-il pair ?
            GOYES       @5          si oui tester l'entier 1
     @4
111         A=R1
118         C=R0
```

Quand on arrive ici, on sait que les entiers sont de la forme $2^n p$ et $2^n q$, avec p et q dans A et C, et n dans B. De plus p au moins est impair.

```
     @6
108         R0=C                    R0 = q
822         SB=0                    vider SB
81E         CSRB                    C=q/2 et SB=reste
8324F       ?SB=0                   q est-il pair ?
            GOYES       @8          si oui aller en @6
```

Si q=2q', comme p est impair, on a pgcd(p,q)=pgcd(p,q'). Donc on

peut diviser q par 2 tant que c'est possible.

```
118         C=R0                    C = q
```

Quand on arrive ici, p et q sont tous deux impairs.

```
9FA50       ?C>=A       W           est-ce que q≥p ?
            GOYES       @7          si oui, va en @7
AFE         ACEX        W           sinon échange p et q
      @7
```

Ici, C=max(p,q) et A=min(p,q).

```
B72         C=C-A       W           C=ABS(p-q)
97A60       ?C=0        W           est-ce que p=q ?
            GOYES       @8          si oui alors pgcd(p,q)=p=q est
                                    dans A
```

Comme pgcd(u,v)=pgcd(u-v,v), on remplace p par abs(p-q) et q par min(p,q) sans changer le pgcd. De plus, comme p et q etaient impairs, le nouveau p est pair et q est impair. Donc on peut continuer à diminuer p en le divisant par 2.

```
6EDF        GOTO        @6
```

A chaque parcours de la boucle, on remplace p et q par des nombres positifs qui ont le même pgcd, et tels que p+q diminue strictement. La boucle va donc nécessairement se terminer.

```
      @8
```

Quand on est ici, on sait que le pgcd cherché est 2^{B}*A.

```
969C0       ?B=0        B           Si B=0, termine
            GOYES       @1
A74         A=A+A       W           sinon, multiplie A par 2
A6D         B=B-1       B           et diminue B de 1
64FF        GOTO        @8
      @1
```

159D	DAT1=A	16	sauve le PGCD à la place de l'entier 2
8F2D760	GOSBVL	067D2	Récupère D,B,D1 et D0
142	A=DAT0	A	fin du programme
164	D0=DO+5		
808C	PC=(A)		

Remarque :
L'algorithme utilisé est dit "algorithme d'euclide binaire".

3
Calcul du PPCM de deux entiers *PPCM*

Ce programme calcule le P.P.C.M. de deux entiers.

Ce programme est à enregistrer sous le nom : PPCM.

Le checksum de ce programme est : #CD43h (43,5 octets)

Listing

```
« DUP2 PGCD 3 ROLLD * SWAP / ABS »
```

Lancement du programme :
Saisir deux nombres entiers. Puis taper

```
PPCM
```

Exemple d'utilisation :

```
124 [ENTER]
638 [ENTER]
PPCM
```

Le calculateur renvoie : 39556

Ce qui signifie que le Plus Petit Commun Multiple de 124 et 638 est 39556.

Temps d'éxecution : immédiat.

4

Transformation de fraction *Q→R*

Ce programme transforme une fraction rationnelle en son numérateur et son dénominateur.

Ce programme est à enregistrer sous le nom : Q→R.
Tapez :

```
'Q→R' STO
```

après avoir saisi le listing. Son checksum est : #2593h (75 octets)

Listing

```
« IF DUP TYPE THEN
C→R DUP2 PGCD SWAP OVER / ROT ROT /
ELSE 1 SWAP END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #E9AFh (150 octets)

```
"D9D 20D 295 12B F81 9FF 30D 9D2 09C 2A2
322 30B 213 030 040 D9D 20C 2D5 0CA 130
```

```
84E 204 005 743 444 322 302 C23 0EF 9A2
CAF 06E F9A 22C 230 997 A2B 3A1 602 9A2
322 300 29A 232 230 B21 30B 213 0"
```

Lancement du programme :
Saisir une fraction rationnelle (complexe) puis taper :

```
Q→R
```

Exemple d'utilisation :

```
(1,2) [ENTER]
 Q→R
```

qui représente 1/2

Et le calculateur renvoie :

```
2
1.
```

Temps d'éxecution : immédiat.

5
Transformation de deux nombres en une fraction *R→Q*

Ce programme transforme deux nombres en une fraction rationnelle (complexe).

Ce programme est à enregistrer sous le nom : R→Q.

Son checksum est : #2CB3h (95 octets)

Listing

```
« DUP2 PGCD SWAP OVER / ROT ROT / IF
DUP 0 < THEN NEG SWAP NEG SWAP END
IF DUP 1 == THEN DROP ELSE R→C END
»
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #FFCBh (130 octets)

```
"D9D 20D 488 1D8 A81 2BF 819 904 0D9 D20
2C2 307 2C5 032 230 837 A2C B91 607 B15
881 30C 2D5 084 E20 400 574 344 446 E15
881 30C 2D5 09C 2A2 79B 30C 5F2 6B2 130
B21 30"
```

Lancement du programme :
Saisir deux nombres (le dénominateur d'abord puis le numérateur) et taper :

```
R→Q
```

Exemple d'utilisation :

```
2 [ENTER]
1 [ENTER]
```

Puis taper

```
R→Q
```

Le calculateur renvoie :

```
(1,2)
```

soit 1/2 .

Temps d'éxecution : immédiat.

6
Addition de deux fractions *QPLUS*

Ce programme additionne deux fractions rationnelles.

Ce programme est à enregistrer sous le nom : QPLUS

Son checksum est : #E4DEh (78 octets)

Listing

```
« DUP2 IF TYPE 1 ≤ SWAP TYPE 1 ≤ AND
THEN QSIMPL + R→Q ELSE + END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #A68Eh (122 octets)

```
"D9D 20D 488 1D8 A81 961 262 C23 038 126
57B 30E F11 6E6 126 C12 163 812 657 B30
64B 30D A91 676 BA1 84E 206 015 359 4D4
05C 447 9A2 84E 203 025 D81 5B2 130"
```

Lancement du programme :
Saisir deux fractions rationnelles puis taper :

```
QPLUS
```

Exemple d'utilisation :

```
(1,2) [ENTER]
```

qui représente 1/2

```
(1,3) [ENTER]
```

qui représente 1/3
et taper :

```
QPLUS
```

Le calculateur renvoie :

```
(5,6)
```

soit 5/6 .
Car 5/6 = 1/2 + 1/3.

Temps d'éxecution : immédiat.

7
Différence de deux fractions *QMOINS*

Ce programme calcule la différence de deux fractions rationnelles.

Ce programme est à enregistrer sous le nom : QMOINS.

Son checksum est : #59B7h (79 octets)

Listing

```
« DUP2 IF TYPE 1 ≤ SWAP TYPE 1 ≤ AND
THEN QSIMPL - R→Q ELSE - END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #91DBh (122 octets)

```
"D9D 20D 488 1D8 A81 961 262 C23 038 126
57B 30E F11 6E6 126 C12 163 812 657 B30
64B 30D A91 690 DA1 84E 206 015 359 4D4
05C 418 9A2 84E 203 025 D81 5B2 130"
```

Lancement du programme :
Saisir deux fractions rationnelles puis taper :

```
QMOINS
```

Exemple d'utilisation :

```
(1,2) [ENTER]
```

qui représente 1/2

```
(1,3) [ENTER]
```

qui représente 1/3
et taper :

```
QMOINS
```

Le calculateur renvoie :

```
(1,6)
```

soit 1/6 .

Car 1/6 = 1/2 - 1/3.

Temps d'éxecution : immédiat.

8

Produit de deux fractions *QMULT*

Ce programme calcule le produit de deux fractions rationnelles.

Ce programme est à enregistrer sous le nom : QMULT.

Son checksum est : #685Ch (96.5 octets

Listing

```
« IF DUP2 TYPE 2 ≤ SWAP TYPE 2 ≤ AND
THEN Q→R ROT Q→R ROT * ROT ROT *
SWAP R→Q ELSE * END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #8CD9h (154 octets)

```
"D9D 20D 488 1D8 A81 961 262 C23 038 126
57B 30E F11 6E6 126 C12 163 812 657 B30
64B 30D A91 6EE DA1 84E 203 015 D82 559
230 84E 203 015 D82 559 230 CB9 A2C AF0
6CB 9A2 322 308 4E2 030 25D 815 B21 30"
```

Lancement du programme :

Saisir deux fractions rationnelles puis taper

```
QMULT
```

Exemple d'utilisation :

```
(1,2) [ENTER]
```

qui représente 1/2,

```
(1,3) [ENTER]
```

qui représente 1/3,
et taper :

```
QMULT
```

Le calculateur renvoie :

```
(1,6).
```

soit 1/6 .
Car 1/6 = 1/2 x 1/3.

Temps d'éxecution : immédiat.

9

Quotient de deux fractions *QDIV*

Ce programme divise une fraction par une autre fraction.

Ce programme est à enregistrer sous le nom : QDIV.

Son checksum est : #3E40h (95.5 octets)

Listing

```
« IF DUP2 TYPE 1 ≤ SWAP TYPE 1 ≤ AND
THEN Q→R ROT Q→R SWAP ROT * ROT
ROT * R→Q ELSE / END  »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #F780h (149 octets)

```
"D9D 20D 488 1D8 A81 961 262 C23 038 126
57B 30E F11 6E6 126 C12 163 812 657 B30
64B 30D A91 650 FA1 84E 203 015 D82 559
```

```
230 84E 203 015 D82 5CA F06 CB9 A2C AF0
6CB 9A2 84E 203 025 D81 5B2 130"
```

Lancement du programme :
Entrer les deux fractions rationnelles puis taper :

```
QDIV
```

Exemple d'utilisation :

```
(1,2) [ENTER]
```

qui représente 1/2,

```
3 [ENTER]
```

Taper :

```
QDIV
```

On obtient (1,6) car 1/2 /3 = 1/6.

Temps d'éxecution : immédiat.

10
Réduction au même dénominateur *QSIMPL*

Réduit au même dénominateur deux fractions rationnelles.

Ce programme est à enregistrer sous le nom : QSIMPL.

Son checksum est : #3B9Fh (163.5 octets)

Listing

```
« IF DUP TYPE 1 ≤ THEN Q→R SWAP ROT
```

```
Q→R SWAP ROT DUP2 PGCD SWAP OVER /
ROT DUP2 * 4 ROLL ROT SWAP / 5 ROLL
* 4 ROLL 4 ROLL * SWAP 3 PICK IF 0 <
THEN NEG ROT NEG ROT NEG ROT END END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #E9D1h (151 octets)

```
"D9D 20D 488 1D8 A81 84E 203 015 D82 559
230 84E 203 015 D82 52C 230 A32 168 4E2
040 057 434 44A 321 62C 230 EF9 A25 923
0CB 9A2 CAF 06E F9A 259 230 2C2 30C B9A
232 230 BBF 06C B9A 2CA F06 B21 30"
```

Lancement du programme :
Saisir deux fractions rationnelles puis taper :

```
QSIMPL
```

La HP renvoie le dénominateur commun puis les deux numérateurs.

Exemple d'utilisation :

```
(1,2)   [ENTER]
```

qui représente 1/2

```
(1,3)   [ENTER]
```

qui représente 1/3

```
QSIMPL
```

On obtient :

```
6
```

qui est le dénominateur commun,

```
3
```

qui est le numérateur du premier nombre,

```
2
```

qui est le numérateur du deuxième nombre car 1/2=3/6 et 1/3=2/6.

Temps d'éxecution : immédiat.

11
Exponentiation d'une fraction *QPUIS*

Ce programme élève une fraction rationnelle à une puissance entière.

Ce programme est à enregistrer sous le nom : QPUIS

Son checksum est : #1BA6h (131 octets)

Listing

```
« IF DUP2 TYPE 1 ≤ SWAP TYPE 1 ≤ AND
THEN SWAP OVER IF 0 < THEN SWAP NEG
SWAP Q→R ELSE Q→R SWAP END 3
PICK ^ SWAP ROT ^ R→Q ELSE ^ END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #4FEFh (168 octets)

```
"D9D 20D 6BB 1DB BF1 E71 262 C23 0E6 126
57B 30D A91 6D9 D20 322 30D 20B 1B2 130
```

```
84E 203 015 D82 5EF 116 837 A2C B91 6D9
D20 33F 060 29A 2CA F06 B21 30E F11 607
AA2 33F 060 7AA 288 130 9C2 A2E 2B3 0B4
916 72C 50B 213 0"
```

Lancement du programme :
Saisir la fraction rationnelle, puis l'exposant.
Taper ensuite :

```
QPUIS
```

Exemple d'utilisation :

```
(1,2) [ENTER]
3 [ENTER]
```

Taper ensuite :

```
QPUIS
```

On obtient :

```
(1,8)
```

car 1/2^3=1/8.

Temps d'éxecution : immédiat.

12
Fraction opposée
QNEG

Ce programme calcule l'opposé d'une fraction rationnelle.

Ce programme est à enregistrer sous le nom : QNEG

Son checksum est : #23B8h (51 octets)

Listing

```
« IF DUP TYPE 1 ≤ THEN NEG CONJ ELSE
NEG END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, exécutez DUP BYTES sur la chaine. Le résultat doit être : #4243h (45 octets)

```
"D9D 20D 488 12B A81 E71 26C B91 62B B15
599 A1B 213 0"
```

Lancement du programme :
Saisir la fraction rationnelle, puis taper :

```
QNEG
```

Exemple d'utilisation :

```
(1,2)    [ENTER]
QNEG
```

On obtient (-1,2).

Temps d'exécution : immédiat.

13
Recherche du plus grand nombre *QTEST*

Ce programme teste si un nombre est supérieur à l'autre.

Ce programme est à enregistrer sous le nom : QTEST.

Son checksum est : #10F8h (36.5 octets)

Listing

```
« QSIMPL > SWAP DROP »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #B35h (44 octets)

```
"D9D 208 4E2 060 153 594 D40 5C4 FDC E1B
9F0 6B2 130"
```

Lancement du programme :

Saisir un premier nombre rationnel, puis un deuxième.
Taper ensuite :

```
QTEST
```

Le programme indique par 1 (0 sinon) si le premier nombre est supérieur au second.

Exemple d'utilisation :

```
(1,2)   [ENTER]
```

qui représente 1/2,

```
(1,3)   [ENTER]
```

qui représente 1/3,
Taper ensuite :

```
QTEST
```

Le programme renvoie 1 , car 1/2 > 1/3.
Temps d'exécution : immédiat.

14
Carré d'une fraction rationnelle *QSQ*

Ce programme calcule le carré d'une fraction rationnelle.

Ce programme est à enregistrer sous le nom : QSQ

Son checksum est : #9EE1h (76.5 octets)

Listing

```
« IF DUP TYPE 1 ≤ THEN Q→R SQ SWAP
SQ IF DUP 1 == THEN DROP ELSE R→C
END END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #8B27h (98 octets)

```
"D9D 20D 488 12B A81 E71 26D A91 662 4B1
84E 203 015 D82 588 130 CB9 A27 472 69C
2A2 E2B 30B 491 688 130 CB9 A27 2C5 0B2
130"
```

Lancement du programme :
Saisir une fraction rationnelle, puis taper :

```
QSQ
```

Exemple d'utilisation :

```
(1,2) [ENTER]
```

qui représente 1/2,

```
QSQ
```

Le programme renvoie :

```
(1,4)
```

car 1/2^2=1/4

Temps d'exécution : immédiat.

15
Inverse d'une fraction
QINV

Ce programme calcule l'inverse d'une fraction rationnelle.

Ce programme est à enregistrer sous le nom : QINV

Son checksum est : #68F3h (100 octets)

Listing

```
« IF DUP TYPE 1 ≤ THEN Q→R DUP IF
0 < THEN NEG SWAP NEG SWAP END IF DUP
1 == THEN DROP ELSE R→C END END »
```

Version plus rapide

Ce listing est à saisir sans espaces ni sauts de ligne. Puis il est à assembler avec ASS. Avant l'assemblage, executez DUP BYTES sur la chaine. Le résultat doit être : #5AB8h (138 octets)

```
"D9D 20D 488 12B A81 E71 262 C23 0E6 126
57B 30D A91 687 2B1 84E 203 015 D82 588
130 837 A2C B91 6D9 D20 029 A23 223 002
9A2 322 30B 213 088 130 9C2 A2E 2B3 0B4
916 72C 50B 213 0"
```

Lancement du programme :
Saisir une fraction rationnelle puis taper :

```
QINV
```

Exemple d'utilisation :

```
(1,2)   [ENTER]
```

qui représente 1/2,

```
QINV
```

Le programme renvoie 2 car 1/(1/2) = 2.
Temps d'exécution : immédiat.

16
Redéfinition du clavier *KEYQ*

Ce programme redéfinit le clavier de la HP48.

Ce programme est à enregistrer sous le nom : KEYQ

Son checksum est : #428Bh (232 octets)

Listing

```
« { QPLUS 95 QMOINS 85 QMULT 75 QDIV
65 QINV 46 QTEST 83.5 QSQ 44.2 QPUIS
45 QNEG 52 } STOKEYS -62 SF -15 SF
-16 CF
»
```

Lancement du programme :
Taper

```
KEYQ
```

Explications :

Ci-dessous la liste des touches qui ont été redéfinies.

`[+]`	addition de deux objets
`[-]`	différence de deux ojbets
`[*]`	produit de deux objets
`[/]`	quotient de deux objets
`[1/X]`	inverse d'un objet
[α] `[ORANGE] [2]`	compare deux objets
x^2 `([ORANGE] [√X])`	carré d'un objet
$[Y^X]$	puissance d'un objet
`[+/-]`	opposé d'un objet

Cette nouvelle définition ne gène en rien le fonctionnement habituel de ces touches si ce n'est que les opérations sur les complexes sont remplacées par les opérations sur les rationnels.

Exemples d'utilisation :
Taper :

```
KEYQ
1       [ENTER]
(1,2)       [ENTER]
[+]
```

renvoie

```
(3,2)

'A'     [ENTER]
3 ENTER]
[-]
```

renvoie

```
'A-3'
```

(Fonctionnement habituel)

```
(6,5)   [ENTER]
2 [ENTER]
[Y^X]
```

renvoie

```
(36,25)
```

L'effet des touches se lance la première fois par le programme KEYQ, puis la définition est activée ou désactivée par appui sur les touches

[ORANGE] [α]

sur les HP48 s et sx

[VIOLET] [α]

sur les HP48 g et gx.

Un témoin USER ou 1USR (USR1) s'affiche en haut de l'écran dans le cas où la redéfinition est active. Dans le cas contraire, rien (NRL) ne s'affiche en haut de l'écran.

3

Calculs sur les polynômes

Convention :

Les polynômes sont représentés par des listes.

Par exemple { 1 0 0 2 } représente 1+2*x^3 (puisque 1+2*x^3 est équivalent à 1+0*x+0*x^2+2*x^3), ou encore { A B C } représente A+B*x+C*x^2.

Nous vous rappelons que l'expression "a+bx+cx^2+dx^3" est équivalente à l'expression "$a+bx+cx^2+dx^3$".

Les programmes de ce chapitre ont donc le gros avantage de pouvoir faire des calculs avec des polynômes à coefficients déterminés (2 ou 1/3) ou bien indéterminés ('A' par exemple).

Les programmes présentés dans ce chapitre devront être enregistrés dans le répertoire POLY.

Ils vous seront présententés dans l'ordre suivant :

- **PSIMPL** simplifie un polynôme.
- **VAL** calcule la valuation d'un polynôme.
- **EVALP** évalue un polynôme en un point.
- **OPER** effectue une opération entre deux polynômes.
- **PPLUS** additionne deux polynômes.

- **PMOINS** soustrait deux polynômes.
- **PCONST** multiplie un polynôme par une constante
- **PMULT** effectue la multiplication de deux polynômes.
- **PPUIS** élève un polynôme à une puissance.
- **PDIV** effectue la division euclidienne de deux polynômes.
- **PDIVC** effectue la division selon les puissances croissantes de deux polynômes.
- **PPGCD** calcule le plus grand commun diviseur de deux polynômes.
- **PPPCM** calcule le plus petit commun multiple de deux polynômes.
- **PCOMP** compose deux polynômes.
- **PTRANS** translate un polynôme.
- **PDER** dérive un polynôme.
- **PINT** intègre un polynôme.
- **P→L** convertit un polynôme à coefficients algèbriques en un polynôme à coefficients rééls.

De manière générale, les programmes concernant les polynômes commencent par un P et le reste de leut nom désigne l'opération qui est effectuée.

1
Simplification d'un polynôme *PSIMPL*

Simplifie un polynôme en retirant les 0 non significatifs.

Ce programme est à enregistrer sous le nom : PSIMPL.

Le checksum est : #762Dh (112 octets)

Listing

```
«  1 6 ERREUR IF DUP SIZE 0 ≠ THEN IF DUP
DUP SIZE GET ABS 0 SAME THEN
LIST→  SWAP DROP 1 -  →LIST PSIMPL END
END
»
```

Lancement du programme :
Saisir le polynôme puis taper

```
PSIMPL [ENTER]
```

Exemple d'utilisation :

```
{ 0 1 2 3 0 0 } [ENTER]
PSIMPL [ENTER]
```

renvoie :

```
{ 0 1 2 3 }
```

Temps d'exécution : 0,2 secondes.

2
Valuation d'un polynôme
VAL

Ce programme est à enregistrer sous le nom : VAL.

Le checksum est : #26E3h (135,5 octets)

Listing

```
« PSIMPL 1 → P V « IFERR WHILE P V GET 0
SAME REPEAT V 1 + 'V'
STO END V 1 - THEN DROP2 '∞' END »
»
```

Lancement du programme :
Saisir un polynôme puis taper :

```
VAL [ENTER]
```

Exemples d'utilisation :

```
{  1 2 3 4  }  [ENTER]
VAL [ENTER]
```

renvoie valuation : 0.

Temps d'exécution : 0,1 seconde.

```
{ 0 A B C } [ENTER]
VAL [ENTER]
```

renvoie valuation : 1.

Temps d'exécution : 0,2 seconde.

```
{ 0 0 0 0 1 } [ENTER]
VAL [ENTER]
```

renvoie valuation : 4.

Temps d'exécution : 0,3 seconde.

3
Valeur d'un polynôme
EVALP

Ce programme calcule la valeur d'un polynôme en un point précis .

Ce programme est à enregistrer sous le nom : EVALP.

Le checksum est : #9D4Eh (187 octets)

Listing :

```
«  2 1 ERREUR
→ A B « IF A SIZE 1 > THEN 0 1 A SIZE
FOR X A X GET B X 1 - QPUIS
QMULT QPLUS NEXT ELSE IFERR A 1 GET THEN
DROP2 0 END END »
»
```

Lancement du programme :

Entrer un polynôme et une valeur en laquelle on cherche à évaluer le polynôme puis taper :

```
EVALP [ENTER].
```

Exemple d'utilisation :

Calculer 1/2 + 1/3*x + 1/5*x^3 pour x=7/8

```
{ (1,2)  (1,3)  0  (1,5)  }  [ENTER]
(7,8)  [ENTER]
EVALP [ENTER]
```

renvoie

```
7109/7680
```

Temps d'exécution : 1,6 seconde.

```
{ 1 1 1 }  [ENTER]
'A'  [ENTER]
```

```
EVALP [ENTER]
```

renvoie

```
'1+A+A*A'
```

Temps d'exécution : 0,6 seconde.

4

Opérations entre deux polynômes *OPER*

Fonction :
Ce programme effectue n'importe quelle opération entre deux polynômes .

Ce programme est à enregistrer sous le nom : OPER.

Le checksum est : #439Ah (225,5 octets)

Listing

```
« 4 2 ERREUR 4 ROLL 4 ROLL 0 → d op L «
1 2 START WHILE DUP2 SIZE SWAP SIZE <
REPEAT 0 + END SWAP NEXT DUP SIZE 'L'
STO LIST→ DROP L 1 + ROLL LIST→
1 SWAP FOR I L I - 2 + ROLL d op
EVAL L I - 2 * I + ROLLD NEXT L
→LIST » »
```

Lancement du programme :

Ce programme attend 4 paramètres:

un paramètre d
un programme op (opérateur)
le premier polynôme p1
le second polynôme p2
Le programme revient à effectuer d op sur les monômes de p1 et p2.

Exemple d'utilisation :

```
2 [ENTER]
« * + »  [ENTER]
{ 1 2 3 } [ENTER]
{ 4 5 6 } [ENTER]
OPER
```

revient à effectuer

```
{ 1+4*2
2+5*2
3+6*2   }
```

Nous trouvons bien { 9 12 15 }.

Temps d'exécution : 0,3 seconde.

5
Addition de deux polynômes *PPLUS*

Ce programme est à enregistrer sous le nom : PPLUS.

Le checksum est : #EF35h (77,5 octets)

Listing

```
« 2 3 ERREUR 0  « DROP QPLUS »  4  ROLLD
4  ROLLD  OPER  »
```

Lancement du programme :
Entrer les deux polynômes dont on désire effectuer l'addition, puis taper:

```
PPLUS [ENTER].
```

Exemple d'utilisation :

```
{ (1,2) 2 3 } [ENTER]
{ 2 0 1 4 } [ENTER]
```

puis taper

```
PPLUS [ENTER]
```

On obtient :

```
{ (5,2) 2 4 4 }
```

Temps d'exécution : 0,8 seconde.

6

Soustraction de deux polynômes *PMOINS*

Ce programme est à enregistrer sous le nom : PMOINS.

Son checksum est : #29B9h (79,5 octets)

Listing

```
« 2 3 ERREUR
0  « DROP QMOINS »  4 ROLLD  4 ROLLD
OPER  »
```

Lancement du programme :
Entrer les deux polynômes dont on désire effectuer la soustraction, puis taper :

```
PMOINS [ENTER]
```

Exemple d'utilisation :

```
{ (1,2) 2 3 } [ENTER]
{ 2 0 1 4 } [ENTER]
PMOINS [ENTER]
```

On obtient :

```
{ (-3,2) 2 2 -4 }
```

Temps d'exécution : 0,8 seconde.

7
Produit d'un polynôme par une constante
PCONST

Ce programme calcule le produit d'un polynôme par une constante.

Ce programme est à enregistrer sous le nom : PCONST.

Son checksum est : #7ACBh (172 octets)

Listing

```
« 2 4 ERREUR →  K A  « IF A TYPE 1 ≤
THEN A 1 →LIST 'A' STO END K  « QMULT
SWAP DROP »  A A OPER IF DUP SIZE 1 ==
THEN LIST→ DROP END »
»
```

Lancement du programme :
Saisir la constante, puis le polynôme et taper

```
PCONST [ENTER]
```

Exemples d'utilisation :

```
3 [ENTER]
{ 1 2 3 } [ENTER]
```

taper

```
PCONST [ENTER]
```

renvoie

```
{ 3 6 9 }
```

Temps d'exécution : 0,6 seconde.

```
'C' [ENTER]
{ L M N } [ENTER]
PCONST
```

renvoie

```
{ 'L*C'  'M*C'  'N*C' }
```

Temps d'exécution : 0,5 seconde.

8

Multiplication de deux polynômes *PMULT*

Ce programme effectue le produit de deux polynômes.

Ce programme est à enregistrer sous le nom : PMULT.

Le checksum est : #15BBh (160 octets)

Listing

```
« 2 3 ERREUR →  A B  « { }  1 A SIZE FOR
I  { }  IF I 1 >  THEN 1  I 1 -  START
0 +  NEXT END  A I GET  B PCONST  +
PPLUS  NEXT  »  »
```

Lancement du programme :

Entrer les deux polynômes dont on désire effectuer la multiplication, puis taper :

```
PMULT [ENTER]
```

Exemples d'utilisation :

```
{ 1 2 3 } [ENTER]
{ 4 5 6 } [ENTER]
PMULT [ENTER]
```

renvoie

```
{ 4 13 28 27 18 }
```

Temps d'exécution : 4,3 secondes.

```
{ A 1 2 } [ENTER]
{ B 4 } [ENTER]
PMULT [ENTER]
```

renvoie

```
{ 'A*B' 'A*4+B' '4+2*B' 8 }
```

Temps d'exécution : 2,9 secondes.

```
{ (1,3)  (1,2)  (1,5) } [ENTER]
{ (7,3)  (1,4) } [ENTER]
PMULT [ENTER]
```

renvoie

```
{ (7,9)  (5,4)  (71,120)  (1,20) }
```

Temps d'exécution : 5,3 secondes.

Remarque : Tous les calculs effectués sont exacts et ne comportent aucune approximation. Car si l'on utilise des nombres réels, et si l'on cherche par la suite à obtenir la fraction rationnelle équivalente, cela devient impossible : les calculs avec les nombres rééls sont imprécis et mènent vite à des résultats faux.

9

Puissance d'un polynôme *PPUIS*

Calcule la puissance n-ième d'un polynôme.

Ce programme est à enregistrer sous le nom : PPUIS.

Son checksum est : #562Ah (267,5 octets)

Listing

```
« 2 1 ERREUR CASE DUP 0 == THEN DROP2 {
1 } END DUP 1 == THEN DROP END { } SWAP
WHILE DUP 1 > REPEAT IF DUP 2 MOD 0 ==
THEN SWAP 0 + SWAP 2 / ELSE SWAP 1 +
SWAP 1 - 2 / END END DROP SWAP DUP →P «
SWAP LIST→ DUP 2 + ROLL SWAP 1 SWAP
```

```
START DUP PMULT SWAP IF THEN P PMULT END
NEXT  » END »
```

Lancement du programme :
Saisir le polynôme puis la puissance à laquelle on veut l'élever. Puis taper :

```
PPUIS [ENTER].
```

Exemples d'utilisation :

• Calcul de (1+x)^3

```
{ 1 1 } [ENTER]
3 [ENTER]
```

Puis taper :

```
PPUIS [ENTER]
```

renvoie

```
 { 1 3 3 1 }
```

c'est à dire:

```
(1+x)^3 = 1 + 3*x + 3*x^2 + x^3
```

Temps d'exécution : 5,8 secondes.

• Calcul de (1+2*x+3*x^2+4*x^3)^4

```
{ 1 2 3 4 } [ENTER]
4 [ENTER]
```

Puis taper

```
PPUIS [ENTER]
```

renvoie

```
{ 1 8 36 120 310 648 1124 1608 1905 1840
1376 768 256 }
```

c'est-à-dire :
(1+2*x+3*x^2+4*x^3)^4=1+8*x+36*x^2+120*x^3+310*x^4+648*x^5 +1124*x^6+1608*x^7+1905*x^8+1840*x^9+1376*x^10+768*x^11+256* x^12.

Temps d'exécution : 28,3 secondes.

Explication : la méthode utilisée s'appelle méthode "des poids forts

vers les poids faibles". En effet, tout nombre entier m peut se calculer à partir de 0 par itération des deux opérations :

D: $n \rightarrow 2*n$

et E: $n \rightarrow 2*n+1$

Ainsi 13 = E 6 = E D 3 = E D E 1 = E D E E 0
Ceci revient à l'écriture binaire de m. En effet 13 = 1101 en base 2.
Il suffit ensuite d'appliquer D et E à notre polynôme.
D(A^n) = A^(2*n) et E(A^n) = A^(2*n+1).

10
Division de deux polynômes *PDIV*

Division euclidienne de deux polynômes.

Ce programme est à enregistrer sous le nom : PDIV.

Le checksum est : #C456h (317 octets)

Listing

```
« PSIMPL SWAP PSIMPL SWAP { } → A B DD
« WHILE A SIZE B SIZE ≥ REPEAT A SIZE B
SIZE → c d « A c GET B d GET QDIV
{ 0 1 } c d - PPUIS PCONST IF DUP TYPE 1
≤ THEN 1 →LIST END DUP DD PPLUS 'DD' STO
B PMULT A SWAP PMOINS PSIMPL 'A' STO
»  END DD A » »
```

Lancement du programme :

Saisir les deux polynômes dont on veut effectuer la division euclidienne puis taper :

```
PDIV [ENTER].
```

Le programme renvoie d'abord le quotient puis le reste .

Exemple d'utilisation :

```
{ 1 2 3 4 5 6 } [ENTER]
{ 7 8 9 } [ENTER]
PDIV [ENTER]
```

renvoie:

```
{ (872,2187)  (-10,243)  (-1,27)  (2,3)
}
```

qui est le quotient,

```
{ (-3917,2187)  (-1972,2187) }
```

qui est le reste.

car :

```
1+2*x+3*x^2+4*x^3+5*x^4+6*x^5=
(7+8*x+9*x^2)*(872/2187-10/243*x-
1/27*x^2+2/3*x^3) +
-3917/2187 - 1972/2187*x
```

Temps d'exécution : 35,2 secondes.

11

Division par puissances croissantes *PDIVC*

Ce programme effectue la division selon les puissances croissantes de deux polynômes.

Ce programme est à enregistrer sous le nom : PDIVC.

Le checksum est : #69A1h (579,5 octets)

Listing

```
« 3 5 ERREUR 0 → p1 p2 ordre coef
« IF p1 TYPE 5 == p2 TYPE 5 == AND THEN
p1 p2 WHILE DUP2 1 GET 0 SAME SWAP 1 GET
0 SAME AND REPEAT LIST→ 1 - →LIST
SWAP DROP SWAP LIST→ 1 - →LIST SWAP
DROP SWAP END 'p2' STO 'p1' STO END CASE
p2 TYPE 1 ≤ p2 TYPE 6 == OR THEN p2 QINV
p1 PCONST { 0 } END { } IF p1 TYPE 5 ≠
THEN p1 1 →LIST ELSE p1 END IF p2 1 GET
0 SAME THEN #305h DOERR END 'p1' STO p2
1 GET 'coef' STO 0 ordre START p1 1 GET
coef QDIV SWAP OVER + SWAP QNEG « QMULT
QPLUS » p1 p2 OPER LIST→ 1 - →LIST
SWAP DROP 'p1' STO NEXT p1 END » »
```

Lancement du programme :

Saisir les deux polynômes dont on veut effectuer la division selon les puissances croissantes, puis taper:

```
PDIVC [ENTER]
```

Le programme renvoie en premier le quotient puis le reste réduit.

Exemples d'utilisation :

Division selon les puissances croissantes de 1+2*x+3*x^2
par 4+5*x+6*x^2 à l'ordre 3.

```
{ 1 2 3 } [ENTER]
{ 4 5 6 } [ENTER]
3 [ENTER]
```

puis taper :

```
PDIVC [ENTER]
```

renvoie:

```
{ (1,4)  (3,16)  (9,64)  (-117,256) }
{ (369,256)  (351,128) }
```

donc (1+2*x+3*x^2)/(4+5*x+6*x^2)=1/4+3/16*x+9/64*x^2
-117/256*x^3 + (365/256+351/128*x)*x^4/(4+5*x+6*x^2)

Temps d'exécution : 5,3 secondes.

Autre exemple:

```
{ A 2 3 } [ENTER]
{ 1 2 } [ENTER]
2 [ENTER]
```

puis taper

[α] [α] **[FG] [D] [CST] [√X] [C] [ENTER]**

renvoie

```
{ A  '2-2*A'  '-1+4*A' }
      { '2-8*A' }
```

Temps d'exécution : 2,1 secondes.

Remarque :

Ce programme renvoie un reste réduit de la division c'est-à-dire le reste de la division divisé par X^(ordre+1).

12
PGCD de deux polynômes
PPGCD

Calcule le Plus Grand Commun Diviseur de deux polynômes.

Ce programme est à enregistrer sous le nom : PPGCD.

Le checksum est : #7C53h (86 octets)

Listing

```
« 2 3 ERREUR WHILE DUP PSIMPL { } ≠
REPEAT SWAP OVER PDIV SWAP DROP END DROP
»
```

Lancement du programme :
Saisir les deux polynômes puis taper :

```
PPGCD [ENTER].
```

Exemple d'utilisation :

```
{ 1 3 3 1 } [ENTER]
{ 1 1 } [ENTER]
PPGCD [ENTER]
```

renvoie

```
{ 1 1 }
```

Temps d'exécution : 15,8 secondes.

13
PPCM de deux polynômes *PPPCM*

Calcule le P.P.C.M. de deux polynômes.

Ce programme est à enregistrer sous le nom : PPPCM.

Son checksum est : #EBFCh (49 octets)

Listing

```
«  DUP2 PPGCD PDIV DROP PMULT »
```

Lancement du programme :
Saisir les deux polynômes dont on veut calculer le PPCM puis taper

```
PPPCM [ENTER]
```

Exemple d'utilisation :

```
{ 1 3 3 1 } [ENTER]
{ 1 1 } [ENTER]
PPPCM [ENTER]
```

renvoie

```
{ 1 3 3 1 }
```

Temps d'exécution : 22,6 secondes.

14
Composée de deux polynômes *PCOMP*

Ce programme calcule la composée de deux polynômes.

Ce programme est à enregistrer sous le nom : PCOMP.

Son checksum est : #B616h (184 octets)

Listing

```
« 2 3 ERREUR { 1 } → A B C
« { } B 1 GET + 2 B SIZE FOR I B I GET
SWAP « QMULT QPLUS » SWAP A C PMULT DUP
'C' STO OPER NEXT » »
```

Lancement du programme :

Saisir les deux polynômes g puis f pour calculer la composée fog. Puis taper:

```
PCOMP [ENTER].
```

Exemple d'utilisation :

```
{ A 1 } [ENTER]
{ 0 0 1 } [ENTER]
```

Puis taper

```
PCOMP [ENTER]
```

renvoie

```
{ 'A*A' 'A+A' 1 }
```

Temps d'exécution : 4,6 secondes.

```
{ 1 2 3 } [ENTER]
{ (1,6) (1,8) (1,10) }
[ENTER]
```

Puis taper

```
PCOMP [ENTER]
```

renvoie

```
{ (47,120) (13,20) (11,8) (6,5)
(9,10) }
```

Car

1/6 + 1/8*(1+2*x+3*x^2) + 1/10*(1+2*x+3*x^2)^2
=
47/120 + 13/20*x + 11/6*x^2 + 6/5*x^3 + 9/10*x^4

Temps d'exécution : 10,1 secondes.

15
Translation d'un polynôme *PTRANS*

Ce programme translate un polynôme.

Ce programme est à enregistrer sous le nom : PTRANS.

Son checksum est : D5BBh (181 octets)

Listing

```
« 2 1 ERREUR OVER SIZE → A n
« IF n 1 ≠ THEN 1 n 1 - FOR I n 1 -
I FOR J DUP J GET OVER J 1 + GET A
QMULT QPLUS J SWAP PUT -1 STEP NEXT END
» »
```

Lancement du programme :

Saisir le polynôme P(x) puis la valeur a.
Le programme calcule P(x+a). Pour cela taper

```
PTRANS [ENTER]
```

Exemple d'utilisation :

Calcul de P(x+1/2) ou P(x)=1+2*x+3*x^2+4*x^3

```
{ 1 2 3 4 }
(1,2)
PTRANS [ENTER]
```

renvoie

```
{ (13,4) 8 9 4 }
```

Temps d'exécution : 2,0 secondes.

16
Dérivée d'un polynôme
PDER

Ce programmme dérive un polynôme.

Ce programme est à enregistrer sous le nom : PDER

Son checksum est : # 8754h (165,5 octets)

Listing

```
« 1 6 ERREUR 0 → n « LIST→ DUP 1 -
'n' STO IF DUP 1 > THEN 2 FOR I I 1 -
QMULT n ROLLD -1 STEP n →LIST SWAP
DROP ELSE n 2 + DROPN { } END » »
```

Lancement du programme :

Saisir le polynôme que l'on veut dériver.
Puis taper :

```
PDER [ENTER].
```

Exemple d'utilisation :

```
{ 1 1 1 1 } [ENTER]
PDER [ENTER]
```

renvoie

```
{ 1 2 3 }
```

C'est-à-dire d/dx (1+x+x^2+x^3)= 1+2*x+3*x^2.

Temps d'exécution : 0,4 secondes.

17
Intégration d'un polynôme
PINT

Ce programme intègre un polynôme.

Ce programme est à enregistrer sous le nom : PINT.

Son checksum est : #9B4Ah (127 octets)

Listing

```
« 1 6 ERREUR 0 →  n « 0 +  LIST→  DUP
'n' STO 1 FOR I  I QDIV  n ROLLD  -1
STEP
n  →LIST  0 SWAP  + PSIMPL » »
```

Lancement du programme :

Saisir le polynôme que l'on veut intégrer puis taper :

```
PINT [ENTER].
```

Attention ! La constante d'intégration est prise nulle.

Exemple d'utilisation:

```
{ 1 2 3 4 } [ENVOI]
```

Taper

```
PINT [ENTER]
```

renvoie:

```
{ 0 1 1 1 1 }
```

Temps d'exécution : 0,7 seconde.

18
Conversion des coefficients *P→L*

Fonction :
Ce programme convertit un polynôme à coefficients rationnels en un polynôme à coefficients quelconques.

Ce programme est à enregistrer sous le nom : P→L.

Son checksum est : DA26h (226,5 octets)

Listing

```
« 0 0 → n p « LIST→ 'n' STO 1 n
START #5E652h SYSEVAL #54AFh SYSEVAL
#18DBFh SYSEVAL 'p' STO 1 p START IF
DUP TYPE 1 SAME THEN Q→R SWAP / END p
ROLLD NEXT p #18CEAh SYSEVAL #546Dh
SYSEVAL n ROLLD NEXT n →LIST »
»
```

Lancement du programme :

Saisir un polynôme puis taper :

```
P→L [ENTER].
```

Exemple d'utilisation :

```
{ (1,2) 3 'SIN((1,4))' } [ENTER]
P→L [ENTER]
```

renvoie:

```
{ '.5' '3' 'SIN(.25)' }
```

Temps d'exécution : 0,4 seconde.

4
Racines

Les programmes présentés dans ce chapitre seront enregistrés dans le répertoire RACINES.

Le but de ce chapitre est de trouver tous les zéros d'un polynôme quelconque à coefficients complexes.
Nous ne pourrons pas ainsi travailler avec les nombres rationnels comme dans le chapitre précédent. Néanmoins, si le polynôme est à coefficients entiers (ce qui est toujours possible si le polynôme est à coefficients rationnels), les racines rationnelles seront recherchées en priorité. Ces dernières seront données sous forme exacte.

Les programmes de ce chapitre vous sont présentés dans cet ordre :

- **DIVIS** recherche tous les diviseurs d'un nombre donné.
- **RACP** trouve toutes les racines de n'importe quel polynôme.

- **QPLUS, QMOINS, QMULT, QDIV, QINV, QNEG** qui transforment les opérations du chapitre précédent en opérations sur les nombres complexes.

- **EPS** définit la précision de calcul des racines non rationnelles.

1
Décomposition en diviseurs *DIVIS*

Ce programme décompose un nombre en ses diviseurs.

Ce programme est à enregistrer sous le nom : DIVIS.

Le checksum est : #895Dh (142,5 octets)

Listing

```
« 1 7 ERREUR ABS { } SWAP 1 OVER √ IP FOR
K DUP K / IF DUP FP THEN DROP ELSE ROT K
+ + SWAP END NEXT IF √ FP NOT THEN DUP
SIZE 2 SWAP SUB END »
```

Lancement du programme :
Saisir un nombre entier puis taper

```
DIVIS [ENTER]
```

Exemple d'utilisation :

```
126 [ENTER]
DIVIS [ENTER]
```

renvoie

```
{ 14 18 21 42 63 126 1 2 3 6 7 9 }
```

Temps d'exécution : 0,5 seconde.

```
255  [ENTER]
DIVIS  [ENTER]
```

renvoie

```
{ 17 51 85 255 1 3 5 15 }
```

Temps d'exécution : 0,5 seconde.

2
Racines d'un polynôme *RACP*

Ce programme trouve les racines d'un polynôme.

Ce programme est à enregistrer sous le nom : RACP.

Le checksum est : #D3C1h (1552 octets)

Listing

```
« 1 6 ERREUR CLLCD
"Recherche des Racines" 1 DISP PSIMPL
DUP SIZE 1 { } 0 0 0 0 → P n NB SOL z L1
L2 m « "Racine rationnelle" 2 DISP P DUP
SIZE GET RE IP ABS DIVIS 'L1' STO 1 L1
SIZE FOR I P DUP VAL 1 + GET RE IP ABS
DIVIS DUP LIST→ 'm' STO 1 m START NEG m
ROLLD NEXT m →LIST + 0 SWAP + 'L2' STO 1
L2 SIZE FOR J IF L1 I GET ABS 1 == L2 J
GET ABS 1 == OR L1 I GET L2 J GET DUP2
MAX 3 ROLLD MIN MOD 0 ≠ OR L2 J GET 0 ==
L1 I GET 1 ≠ AND NOT AND THEN IF P L2 J
GET L1 I GET / EVALP ABS EPS < THEN SOL
"'" L2 J GET →STR + IF L1 I GET 1 ≠ THEN
"/" + L1 I GET →STR + END "'" + STR→
DUP 4 DISP + 'SOL' STO NB 1 + 'NB' STO
END END NEXT NEXT IF n NB ≠ NB 1 ≠ AND
THEN P 1 NB 1 - FOR I SOL I GET →NUM NEG
{ 1 } + PDIV DROP NEXT RACP SOL + 'SOL'
STO SOL SIZE 1 + 'NB' STO ELSE IF n NB ≠
THEN "Racine quelconque" 3 DISP "" 2
DISP P DO DUP DUP 'P' STO 'z' EVALP SWAP
PDER DUP 'z' EVALP SWAP PDER 'z' EVALP P
SIZE 1 - DUP 1 - DUP SQ 3 PICK 3 PICK *
```

```
NEG → Q R S T U V W « z WHILE DUP 'z'
STO 4 DISP Q EVAL DUP ABS EPS > REPEAT R
EVAL S EVAL → a b c « b V b SQ * W a c *
* + √ DUP2 DUP2 + ABS 3 ROLLD - ABS ≥ 2 *
1 - * + DUP IF ABS 0 == KEY IF 1 == THEN
DROP 1 ELSE 0 END OR THEN DROP RAND 100
* 50 - RAND 100 * 50 - R→C ELSE T NEG a
* SWAP / z + END » END DROP » SOL z EPS
LOG NEG FIX →STR STD STR→ + 'SOL' STO P
z NEG 1 2 →LIST PDIV DROP UNTIL DUP SIZE
1 ≤ END DROP END END SOL » »
```

Lancement du programme :
Avant d'utiliser ce programme, il vous faut d'abord saisir les petits sous-programmes qui suivent.

Saisir un polynôme puis taper

```
RACP [ENTER]
```

Exemples d'utilisation :

```
{ 1 -3 3 -1 } [ENTER]
RACP [ENTER]
```

renvoie

```
{ 1 1 1 }
```

En effet $1-3x+3x^2-x^3=(x-1)(x-1)(x-1)$.

Temps d'exécution : 24,2 secondes.

```
{ -6 23 -29 12 } [ENTER]
RACP [ENTER]
```

renvoie

```
{ '3/4' 1 '2/3' }
```

En effet $-6+23x-29x^2+12x^3=(x-3/4)(x-1)(x-2/3)$.

Temps d'exécution : 31,2 secondes.

```
{ 0 1 0 1 } [ENTER]
RACP [ENTER]
```

renvoie

```
{ (0,1) (0,-1) 0 }.
```

En effet $x+x^3=(x-i)(x+i)x$.

Temps d'exécution : 28,1 secondes.

```
{ -2 0 1 } [ENTER]
RACP [ENTER]
```

renvoie

```
{ 1.41421356 -1.41421356 }
```

(Calculs effectués avec $eps=10^{-8}$)

Temps d'exécution : 14,4 secondes.

Remarque :
Les racines rationnelles seront données sous forme exacte si le polynôme est à coefficients entiers.

Facilité d'utilisation :
Si par hasard le processus de recherche des racines non rationnelles venait à diverger , vous pouvez par un appui sur *n'importe quelle touche* provoquer la recherche des racines à partir d'une valeur différente (qui sera choisie de façon aléatoire).

Le programme RACP travaille sur des polynômes à coefficients quelconques. C'est pourquoi dans ce répertoire, nous devons adapter les opérations. Il suffit d'ajouter les programmes du paragraphe suivant dans *ce* répertoire (le répertoire où sont placés les programmes relatifs aux racines). Vous n'aurez pas à utiliser les petits programmes qui suivent vous-même.

3
Petits sous-programmes

QPLUS

Ce sous-programme additionne deux nombres.

Ce programme est à enregistrer sous le nom : QPLUS

Listing

```
« + »
```

QMOINS

Ce programme soustrait un nombre à un nombre.

Ce programme est à enregistrer sous le nom : QMOINS.

Listing

```
« - »
```

QMULT

Ce programme multiplie un nombre par un autre nombre.

Ce programme est à enregistrer sous le nom : QMULT.

Listing

```
« * »
```

QDIV

Ce programme divise un nombre par un autre nombre.

Ce programme est à enregistrer sous le nom : QDIV

Listing

```
« / »
```

QINV

Ce programme renvoie l'inverse d'un nombre.

Ce programme est à enregistrer sous le nom : QINV.

Listing

```
« INV »
```

QNEG

Ce programme calcule l'opposé d'un nombre.

Ce programme est à enregistrer sous le nom : QNEG.

Listing

```
« NEG »
```

EPS

Cette donnée représente l'incertitude de calcul.

Ce programme est à enregistrer sous le nom : EPS.

Listing

```
1E-8
```

Remarque :

Cette valeur donne la précision avec laquelle les racines irrationnelles des polynômes seront calculées. Vous pouvez changer cette valeur en fonction de vos besoins.

5
Algorithmes

Les trois algorithmes qui vous sont proposés sont :

- **BEZOUT** L'algorithme de Bezout.
- **NEWTON** L'algorithme de Newton.
- **TRI** Un algorithme de tri par insertion.

Ils sont à placer dans le répertoire **ALGO**.

Ces trois programmes sont indépendants du reste de l'ouvrage.

1
Algorithme de Bezout
BEZOUT

Ce programme est à enregistrer sous le nom : BEZOUT.

Son checksum est : #5ECEh (430,5 octets)

Listing

```
« IF DEPTH 2 < THEN "Entrez deux
nombres" DOERR END DUP2 2 →LIST 0 0 0 {
1 0 } { 0 1 } 0 → a b r u v d x y q «
WHILE r 2 GET 0 > REPEAT r LIST→ DROP /
FLOOR 'q' STO x LIST→ DROP DUP 3 ROLLD q
* - 2 →LIST 'x' STO y LIST→ DROP DUP 3
ROLLD q * - 2 →LIST 'y' STO r LIST→
DROP DUP 3 ROLLD q * - 2 →LIST 'r' STO
END x 1 GET →STR "*" + a →STR + "+" + y
1 GET →STR + "*" + b →STR + "=" + r 1
GET →STR + » »
```

Lancement du programme :
Saisir deux nombres puis taper

```
BEZOUT [ENTER]
```

Exemples d'utilisation :

```
5 [ENTER]
7 [ENTER]
BEZOUT [ENTER]
```

renvoie

```
"3*5-2*7=1"
```

Temps d'exécution : 0,8 seconde.

```
17 [ENTER]
21 [ENTER]
BEZOUT [ENTER]
```

renvoie

```
"5*17-4*21=1"
```

Temps d'exécution : 0,8 seconde.

```
101 [ENTER]
```

```
3 [ENTER]
BEZOUT [ENTER]
```

renvoie

```
"-1*101+3*34=1"
```

Temps d'exécution : 0,7 seconde.

```
259 [ENTER]
128 [ENTER]
BEZOUT [ENTER]
```

renvoie

```
"43*259-87*128=1"
```

Temps d'exécution : 0,8 seconde.

Remarque :
L'algorithme de Bezout revient à trouver des entiers u et v tels que si a et b sont premiers entre eux : a*u+b*v=1

2
Algorithme de Newton
NEWTON

Résolution de f(x)=0 par l'algorithme de Newton.

Ce programme est à enregistrer sous le nom : NEWTON.

Son checksum est : #CCD6h (653,5 octets)

Listing

```
« 3 FIX "Entre la fonction de X"
```

```
{ "'" V } INPUT "« → X " SWAP + STR→
"Quel est le point de
départ :" { "" V } INPUT IF DUP "" SAME
THEN RAND RAND i * + →STR SWAP DROP END
STR→ →NUM → f x0 « « → x «
'(f(x+.000001)-f(x))*1000000'
→NUM '(f(x)-f(x-.000001))*1000000' →NUM
+ 2 / » » → der « DO x0 DUP DUP 1 DISP
'f(x0)' →NUM x0 der EVAL / - 'x0' STO
UNTIL x0 - ABS .000001 < END x0 » DUP RE
SWAP IM → re im « IF re ABS 1E-10 < THEN
0 're' STO END IF im ABS 1E-10 < THEN 0
'im' STO END re im R→C » » STD »
```

Lancement du programme :
Taper

```
NEWTON [ENTER]
```

Exemples d'utilisation :

• Taper :

```
NEWTON [ENTER]
```

Fonction :

```
'X^2+1' [ENTER]
```

Point de départ :

```
[ENTER]
```

c'est-à-dire départ aléatoire.
La réponse est communiquée:

```
(0,1)
```

• Taper :

```
NEWTON [ENTER]
```

Fonction :

```
'COS(X)-SIN(X)' [ENTER]
```

Point de départ

```
1 [ENTER]
```

la réponse donnée est :

```
.785398163398.
```

• Taper :

```
NEWTON [ENTER]
```

Fonction :

```
'LN(X)-1'  [ENTER]
```

Point de départ

```
[ENTER]
```

c'est-à-dire départ aléatoire.

La réponse donnée est :

```
2.71828182846
```

Remarque :
Ce programme remplace la fonction interne ROOT qui ne fonctionne qu'avec des réels. On peut choisir le point de départ de la recherche. Si on ne répond rien ([ENTER]) alors le point de départ sera aléatoire.

3
Tri par insertion de liste
TRI

Ce programme permet de ranger des nombres dans l'ordre croissant ou des chaînes de caractères dans l'ordre alphabétique.

Ce programme est à enregistrer sous le nom : TRI.

Son checksum est : #9813h (337,5 octets)

Listing

```
« IF DEPTH 0 == THEN "Entrez une liste"
DOERR END DUP SIZE SWAP { 0 } SWAP + 0 →
n T j « 2 n FOR k T 1 T k 1 + GET PUT
'T' STO k 1 + 'j' STO WHILE T j 1 - GET
T j GET > REPEAT T j 1 - T j GET T j 1 -
GET 4 ROLLD PUT j 3 ROLL PUT 'T' STO j 1
- 'j' STO END NEXT T 2 n 1 + SUB » »
```

Lancement du programme :
Entrer une liste puis taper

```
TRI [ENTER].
```

Exemples d'utilisation :

```
{ -2 1 -5 0 3 7 -10 2 } [ENTER]
```

taper

```
TRI [ENTER]
```

renvoie

```
{ -10 -5 -2 0 1 2 3 7 }
```

Temps d'exécution : 2,9 secondes.

```
{ "ANTOINE" "ROBERT" "PHILIPPE"
"AUGUSTE" "NESTOR" } [ENTER]
```

Taper

```
TRI [ENTER]
```

renvoie

```
{ "ANTOINE" "AUGUSTE" "NESTOR"
"PHILIPPE" "ROBERT" }
```

Temps d'exécution : 1,5 seconde.

Remarque :
Ce programme peut aussi bien trier un liste de réels qu'une liste de chaînes de caractères.

6

Développements limités à coefficients algébriques

Dans notre propos, "DL" signifie développement limité.

Les DL sont représentés par des listes, et les rationnels par un couple complexe (On ne traitera donc pas ici les DL à coefficients complexes.Dans ce dernier cas il faudra séparer partie réelle et partie imaginaire).

Le programme TAYLOR utilise tous les autres programmes du répertoire concernants les développements limités.

Les programmes de ce chapitre sont à enregistrer dans le répertoire DL.

Ils vous seront présentés dans l'ordre suivant :

- **TAYLOR** Calcul général des DL
- **DPLUS** Additionne deux DL
- **DMOINS** Soustrait deux DL
- **DMULT** Multiplie deux DL
- **DDIV** Divise deux DL
- **VAL** Calcule la valuation d'un DL
- **DPUIS** Exponentiation d'un DL
- **DCOMP** Composée de deux DL
- **DSIN** DL de la fonction sinus
- **DCOS** DL de la fonction cosinus
- **DSINH** DL de la fonction sinus hyperbolique

- **DCOSH** DL de la fonction cosinus hyperbolique
- **DEXP** DL de la fonction exponentielle
- **DLN** DL de la fonction logarithme népérien en 1
- **DASIN** DL de la fonction arcsinus
- **DASINH** DL de la fonction arcsinus hyperbolique
- **DATAN** DL de la fonction arctangente
- **DATANH** DL de la fonction arctangente hyperbolique
- **DTAN** DL de la fonction tangente
- **DTANH** DL de la fonction tangente hyperbolique
- **D1PLUS** DL de la fonction $(1+X)^A$
- **DNEG** Opposé d'un DL
- **DSIMPL** Simplification d'un DL

Calcul de développements limités *TAYLOR*

Ce programme calcule les développements limités.

Ce programme est à enregistrer sous le nom : TAYLOR.

Son checksum est : #3733h (1132,5 octets).

Listing

```
« ROT IF DUP TYPE 9 == THEN # 54AFh
SYSEVAL # 18DBFh SYSEVAL  ELSE # 202h
DOERR END   →LIST  ""  0
→ var ordre instr item type
« { }  1 instr SIZE FOR I instr I GET
'item' STO item TYPE 'type' STO CASE
type 0 SAME THEN item + END
type 1 SAME THEN item + END
type 7 SAME THEN "Noms locaux interdits"
```

```
SWAP DROP DOERR END
type 6 SAME THEN IF item var SAME THEN
{ 0 1 } 1 ordre 1 - START 0 + NEXT 1
→LIST ELSE item END + END
type 18 SAME THEN item →STR 'item' STO
CASE item "i" SAME THEN
"Caractère i interdit" SWAP DROP DOERR
END
item "*" SAME THEN ordre + 'DMULT' + END
item "/" SAME THEN ordre + 'DDIV' + END
item "+" SAME THEN ordre + 'DPLUS' + END
item "-" SAME THEN ordre + 'DMOINS' +
END
item "^" SAME THEN ordre + 'DPUIS' + END
item "!" SAME THEN { ! } + END
item "π" SAME THEN 'π' + END
item "√" SAME THEN (1,2) + ordre +
'DPUIS' + END
item "SQ" SAME THEN 2 + ordre + 'DPUIS'
+ END
ordre + "{ D" item + STR→ + ordre +
'DCOMP' +
END END END NEXT EVAL  »
»
```

Lancement du programme :

Saisir l'expression dont on cherche le développement limité, la variable par rapport à laquelle on calcule ce développement puis l'ordre du développement limité, enfin taper :

```
TAYLOR [ENTER].
```

Exemple d'utilisation :

```
'TANH(SIN(X))' [ENTER]
'X' [ENTER]
4 [ENTER]
```

```
TAYLOR [ENTER]
```

renvoie :

```
{ 0 1 0 (-1,2) 0 }
```

c'est-à-dire :

```
tanh(sin(x))=x-x^3/2 + o(x^4)
```

Temps d'exécution : 30 secondes.

```
'3*SIN(X)/(2+COS(X))' [ENTER]
'X' [ENTER]
5 [ENTER]
TAYLOR [ENTER]
```

renvoie :

```
{ 0 1 0 0 0 (-1,180) }
```

Temps d'exécution : 27.3 secondes.
Alors que le programme interne de la machine TAYLR n'arrive pas à terminer le calcul...

```
'(8*SIN(X/2)-SIN(X))/3' [ENTER]
'X' [ENTER]
5 [ENTER]
TAYLOR [ENTER]
```

renvoie :

```
{ 0 1 0 0 0 (-1,480) }
```

Temps d'exécution : 54,8 secondes.

Temps d'exécution sur une HP48gx : 36,4 secondes

Remarque :
La syntaxe à utiliser est la même que pour le programme TAYLR déjà intégré au calculateur.

Possibilités d'extension ultérieure :
La procédure qui ralentit le programme est le programme DCOMP. C'est donc celui-ci qu'il faut chercher à améliorer en priorité. De plus les Développements limités se font en 0 uniquement. C'est le cas le plus fréquent, mais on pourrait améliorer les programmes en leur faisant traiter le cas général.

2
Addition de deux DL
DPLUS

Addition de deux DL

Ce programme est à enregistrer sous le nom : DPLUS.

Son checksum est : #2D9Ah (418,5 octets)

Listing

```
« 3 8 ERREUR → p1 p2 ordre
« CASE p1 TYPE 5 ≠ p2 TYPE 5 == AND
THEN p2 1 p2 1 GET p1 QPLUS PUT END
p1 TYPE 5 == p2 TYPE 5 ≠ AND
THEN p1 1 p1 1 GET p2 QPLUS PUT END
p1 TYPE 5 == p2 TYPE 5 == AND
THEN p1 LIST→ DROP p2 LIST→ DROP
1 ordre + 1 FOR I I 1 + ROLL QPLUS
2 I 1 - * 1 + ROLLD -1 STEP ordre 1
+ 2 FOR I I ROLLD -1 STEP ordre 1 +
→LIST END p1 p2 QPLUS END » »
```

Lancement du programme :
Saisir les deux DL et l'ordre du développement limité. Puis taper

```
DPLUS [ENTER]
```

Exemple d'utilisation :

```
{ 0 1 0 2 } [ENTER]
{ 1 2 0 0 } ENTER
3 [ENTER]
```

Puis taper

```
DPLUS [ENTER]
```

renvoie

```
{ 1 3 0 2 }
```

Temps d'exécution : 0,8 seconde.

3
Différence de deux DL
DMOINS

Ce programme calcule la différence de deux DL.

Ce programme est à enregistrer sous le nom : DMOINS.

Son checksum est : #CDE8h (319,5 octets)

Listing

```
« 3 8 ERREUR → p1 p2 ordre
« CASE
p2 TYPE 0 == THEN p2 NEG 'p2' STO END
p2 TYPE 1 == THEN p2 QNEG 'p2' STO END
p2 TYPE 5 == THEN p2 LIST→ 1 SWAP
START QNEG p2 SIZE ROLL NEXT p2 SIZE
→LIST 'p2' STO END
```

```
p2 TYPE 7 == THEN p2 NEG 'p2' STO END
END
p1 p2 ordre DPLUS » »
```

Lancement du programme :
Saisir les deux DL, l'ordre des DL et taper

```
DMOINS [ENTER]
```

Exemple d'utilisation :

```
{ 0 1 0 2 } [ENTER]
{ 1 2 0 0 } ENTER
3 [ENTER]
DMOINS [ENTER]
```

renvoie

```
{ -1 -1 0 2 }
```

Temps d'exécution : 1 seconde.

4

Produit de deux DL
DMULT

Multiplication de deux DL

Ce programme est à enregistrer sous le nom : DMULT.

Son checksum est : #76C4h (563,5 octets)

Listing

```
« 3 8 ERREUR → p1 p2 ordre
« IF p1 TYPE 5 == p2 TYPE 1 ≤ AND THEN p2
p1 'p2' STO 'p1' STO END
IF p1 TYPE 5 == p2 TYPE 6 == AND THEN p2
p1 'p2' STO 'p1' STO END
IF p1 TYPE 1 ≤ p2 TYPE 1 ≤ AND THEN p1 p2
QMULT END
IF p1 TYPE 1 ≤ p1 TYPE 6 == OR p2 TYPE 5
== AND THEN p2 1 ordre 1 + SUB LIST→ 1
SWAP START p1 QMULT ordre 1 + ROLL NEXT
ordre 1 + →LIST END
IF p1 TYPE 5 == p2 TYPE 5 == AND THEN {
} 0 ordre FOR I 0 0 I FOR J p1 J 1 + GET
p2
I J - 1 + GET QMULT QPLUS NEXT + NEXT
END
» »
```

Lancement du programme :
Saisir les deux DL, l'ordre des DL et taper

```
DMULT [ENTER]
```

Exemple d'utilisation :

```
{ 0 1 0 2 } [ENTER]
{ 1 2 0 0 } ENTER
3 [ENTER]
DMULT [ENTER]
```

renvoie

```
{ 0 1 2 2 }
```

Temps d'exécution : 2,5 secondes.

5
Quotient de deux DL
DDIV

Division d'un DL par un autre DL.

Ce programme est à enregistrer sous le nom : DDIV.

Son checksum est : #6119h (119 octets)

Listing

```
« 3 8 ERREUR → ordre
« ordre ARITHM POLY PDIVC ARITHM DL DROP
1 ordre 1 + SUB » »
```

Lancement du programme :

Saisir les deux DL, l'ordre des DL et taper

```
DDIV [ENTER]
```

Exemple d'utilisation :

```
{ 0 1 0 2 } [ENTER]
{ 1 2 0 0 } ENTER
3 [ENTER]
DDIV [ENTER]
```

renvoie

```
{ 0 1 -2 6 }
```

Temps d'exécution : 5,4 secondes.

Remarque :

Ce programme utilise le programme PDIVC (division selon les puissantes croissantes) du répertoire POLY

6
Valuation d'un DL
VAL

Valuation d'un DL.

Ce programme est à enregistrer sous le nom : VAL.

Son checksum est : #E7E7h (95 octets)

Listing

```
 « 1 9 ERREUR 1 + 0 WHILE  OVER 1 GET 0
SAME  REPEAT 1 + SWAP 2 999 SUB SWAP END
SWAP DROP »
```

Lancement du programme :

Saisir un DL puis taper

```
VAL [ENTER]
```

Exemple d'utilisation :

```
{ 0 1 2 0 1 0 }

VAL [ENTER]
```

renvoie

```
1
```

Temps d'exécution : 0,1 seconde.

7
DL de $(1+X)^A$ en zéro
D1PLUS

Développement limité de (1+X)^A en 0

Ce programme est à enregistrer sous le nom : D1PLUS.

Son checksum est : #4EF2h (159,5 octets)

Listing

```
« 1 12 ERREUR → a ordre « { 1 } 1 ordre
FOR I 1 0 I 1 - FOR J a J QMOINS QMULT
NEXT I FACT QDIV + NEXT » »
```

Lancement du programme :
Saisir l'exposant et l'ordre du développement limité puis taper

```
D1PLUS [ENTER]
```

Exemple d'utilisation :
DL de √(1+X) à l'ordre 7:

```
(1,2) [ENTER]
7 [ENTER]
D1PLUS [ENTER]
```

renvoie

```
{ 1 (1,2) (-1,8) (1,16) (-5,128) (7,256)
(-21,1024) (33,2048) }
```

Temps d'exécution : 8,5 secondes.

8
Simplification d'un DL
DSIMPL

Ce programme simplifie un DL

Ce programme est à enregistrer sous le nom : DSIMPL.

Son checksum est : #6AA3h (109,5 octets)

Listing

```
« 1 9 ERREUR IF DUP SIZE 0 ≠ THEN IF DUP
DUP SIZE GET 0 == THEN LIST→  SWAP DROP
1 - →LIST DSIMPL END END »
```

Lancement du programme :
Saisir un développement limité puis taper

```
DSIMPL [ENTER].
```

Exemple d'utilisation :

```
{ 0 1 2 0 0 }  [ENTER]
DSIMPL [ENTER]
```

renvoie

```
{ 0 1 2 }
```

Temps d'exécution : 0,3 seconde.

Remarque :

Ce programme enlève les 0 à la fin des DL.

9
Composée de deux DL
DCOMP

Calcul de la composée de deux DL

Ce programme est à enregistrer sous le nom : DCOMP.

Son checksum est : #A9FFh (398 octets)

Listing

```
« 3 8 ERREUR → p1 p2 ordre
« IF p1 DSIMPL { 0 1 } SAME THEN p2 1
ordre 1 + SUB ELSE p2 p1 1 0 PUT { 1 } 1
ordre FOR J 0 + NEXT → B A C
« { }  B 1 GET + 1 ordre FOR I 0 + NEXT
2 ordre FOR I A C ordre DMULT DUP 'C'
STO B I GET ordre DMULT ordre DPLUS NEXT
B ordre 1 + GET C ordre DMULT ordre
DPLUS » END » »
```

Lancement du programme :

Saisir les deux DL puis l'ordre de développement.
Taper ensuite

```
DCOMP [ENTER]
```

Exemple d'utilisation :

DL à l'ordre 3 en x de:

$$1 + y + y^2 + o(y^3)$$

où $y=x+x^2+o(x^3)$

```
{ 0 1 1 0 }   [ENTER]
```

c'est-à-dire x+x^2

```
{ 1 1 1 0 }   [ENTER]
```

c'est-à-dire 1+y+y^2

```
3 [ENTER]
```

Taper ensuite

```
DCOMP [ENTER]
```

renvoie

```
{ 1 1 2 2 }
```

c'est-à-dire

$$1 + x + 2x^2 + 2x^3+o(x^3)$$

Temps d'exécution : 14,2 secondes.

10
Exponentiation d'un DL *DPUIS*

Elévation d'un DL à une puissance rationnelle

Ce programme est à enregistrer sous le nom : DPUIS.

Son checksum est : #B9EBh (239,5 octets)

Listing

```
 « 3 10 ERREUR 0 → p b ordre a
« p 1 GET DUP 'a' STO IF 0 SAME THEN
"IMPOSSIBLE" DOERR END p a ordre DDIV 1
0 PUT b ordre D1PLUS ordre DCOMP a b
QPUIS ordre DMULT » »
```

Lancement du programme :
Entrer le développement limité, la puissance et l'ordre du développement .
Puis taper

```
DPUIS [ENTER]
```

Exemple d'utilisation :

```
{ 1 1 0 0 } [ENTER]
4 [ENTER]
3 [ENTER]
DPUIS [ENTER]
```

renvoie :

```
{ 1 4 6 4 }
```

soit: 1 + 4x + 6x^2 + 4x^3 + o(x^3)

Temps d'exécution : 17,6 secondes.

Autre exemple :

```
{ 4 3 2 1 }
5
3
DPUIS
```

renvoit :

```
{ 1024 3840 8320 13280 }
```

Temps d'exécution : 19,6 secondes.

11
DL de la fonction sinus en zéro *DSIN*

Développement limité de SIN en 0

Ce programme est à enregistrer sous le nom : DSIN.

Son checksum est : #4BAAh (151,5 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 } -1 1 ordre
FOR I -1 QMULT SWAP OVER I FACT QDIV + 0
+ SWAP 2 STEP DROP 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement puis taper

```
DSIN [ENTER]
```

Exemple d'utilisation :
Calcul du DL à l'ordre 7 de SIN

```
7 [ENTER]
DSIN [ENTER]
```

renvoie :

```
{ 0 1 0 (-1,6) 0 (1,120) 0 (-1,5040) }
```

car SIN(x)=x -1/6*x^3 + 1/120*x^5 - 1/5040*x^7 + o(x^7)

Temps d'exécution : 0,9 seconde.

12
DL de la fonction cosinus en zéro *DCOS*

Développement limité de COS en 0

Ce programme est à enregistrer sous le nom : DCOS.

Son checksum est : #6A7Bh (151,5 octets)

Listing

```
« 1 14 ERREUR → ordre « { } -1 0 ordre
FOR I -1 QMULT DUP I FACT
QDIV ROT SWAP + 0 + SWAP 2 STEP DROP 1
ordre 1 + SUB » »
```

Lancement du programme :
Saisir l'ordre du développement puis taper
`DCOS [ENTER].`

Exemple d'utilisation :
Calcul du DL à l'ordre 7 de COS

```
7 [ENTER]
DCOS [ENTER]
```

renvoie :

```
{ 1 0 (-1,2) 0 (1,24) 0 (-1,720) 0 }
```

car COS(x)=1 -1/2*x^2 + 1/24*x^4 -1/720*x^6 + o(x^7)

Temps d'exécution : 0,9 seconde.

13
DL de sinus hyperbolique en zéro *DSINH*

Développement limité de SINH (fonction sinus hyperbolique) en 0

Ce programme est à enregistrer sous le nom : DSINH.

Son checksum est : #B8F7h (141,5 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 } 1 1 ordre
FOR I SWAP OVER I FACT QDIV + 0 + SWAP 2
STEP DROP 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement puis taper

```
DSINH [ENTER].
```

Exemple d'utilisation :
DL à l'ordre 7 de SINH

```
7 [ENTER]
DSINH [ENTER]
```

renvoie

```
{ 0 1 0 (1,6) 0 (1,120) 0 (1,5040) }
```

Temps d'exécution : 0,6 seconde.

14
DL de cosinus hyperbolique en zéro *DCOSH*

Développement limité de COSH en 0

Ce programme est à enregistrer sous le nom : DCOSH.

Son checksum est : #F11Bh (141,5 octets)

Listing

```
« 1 14 ERREUR → ordre « { } 1 0 ordre
FOR I DUP I FACT QDIV ROT SWAP + 0 +
SWAP 2 STEP DROP 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DCOSH [ENTER]
```

Exemple d'utilisation :
DL à l'ordre 7 de COSH

```
7 [ENTER]
DCOSH [ENTER]
```

renvoie

```
{ 1 0 (1,2) 0 (1,24) 0 (1,720) 0 }
```

Temps d'exécution : 0,6 seconde.

15
DL de la fonction exponentielle en zéro
DEXP

Développement limité de EXP en 0

Ce programme est à enregistrer sous le nom : DEXP.

Son checksum est : #D040h (102 octets)

Listing

```
« 1 14 ERREUR → ordre « { 1 } 1 ordre
FOR I I FACT QINV + NEXT »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DEXP [ENTER]
```

Exemple d'utilisation :
DL à l'ordre 7 de EXP

```
7 [ENTER]
DEXP [ENTER]
```

renvoie

```
{ 1 1 (1,2) (1,6) (1,24) (1,120) (1,720)
(1,5040) }
```

Temps d'exécution : 0,5 seconde.

16
DL de logarithme népérien en un *DLN*

Développement limité de LN en 1

Ce programme est à enregistrer sous le nom : DLN.

Son checksum est : #379Bh (122 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 } -1 1 ordre
FOR I -1 QMULT SWAP OVER I QDIV + SWAP
NEXT DROP »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DLN [ENTER].
```

Exemple d'utilisation :
DL à l'ordre 7 de LN.

```
7 [ENTER]
DLN [ENTER]
```

renvoie

```
{ 0 1 (-1,2) (1,3) (-1,4) (1,5) (-1,6)
(1,7) }
```

Temps d'exécution : 1,3 seconde.

17
DL de arc sinus en zéro
DASIN

Développement limité de ASIN en 0

Ce programme est à enregistrer sous le nom : DASIN.

Son checksum est : #9139h (179 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 1 0 } 1 3
ordre FOR I I 1 - QDIV I 2 - QMULT SWAP
OVER I QDIV + 0 + SWAP 2 STEP DROP 1
ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DASIN [ENTER]
```

Exemple d'utilisation :
DL à l'ordre 7 de ASIN

```
7 [ENTER]
DASIN [ENTER]
```

renvoie

```
{ 0 1 0 (1,6) 0 (3,40) 0 (5,112) }
```

Temps d'exécution : 1,2 seconde.

18
DL de arc sinus hyperbolique *DASINH*

Développement limité de ASINH en 0

Ce programme est à enregistrer sous le nom : DASINH.

Son checksum est : #0E5Fh (180 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 1 0 } 1 3
ordre FOR I 1 I - QDIV I 2 - QMULT SWAP
OVER I QDIV + 0 + SWAP 2 STEP DROP 1
ordre 1 + SUB »
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DASINH [ENTER]
```

Exemple d'utilisation :
DL à l'ordre 7 de ASINH

```
7 [ENTER]
DASINH [ENTER]
```

renvoie

```
{ 0 1 0 (-1,6) 0 (3,40) 0 (-5,112) }
```

Temps d'exécution : 1,2 seconde.

19
DL de arc tangente *DATAN*

Développement limité de ATAN en 0

Ce programme est à enregistrer sous le nom : DATAN.

Son checksum est : #FAFAh (146,5 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 } -1 1 ordre
FOR I QNEG SWAP OVER I QDIV + 0 + SWAP 2
STEP DROP 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DATAN [ENTER]
```

Exemple d'utilisation :
DL à l'ordre 7 de ATAN

```
7 [ENTER]
DATAN [ENTER]
```

renvoie

```
{ 0 1 0 (-1,3) 0 (1,5) 0 (-1,7) }
```

Temps d'exécution : 0,6 seconde.

20
DL de arc tangente hyperbolique *DATANH*

Développement limité de ATANH en 0

Ce programme est à enregistrer sous le nom : DATANH.

Son checksum est : #2659h (140 octets)

Listing

```
« 1 14 ERREUR → ordre « { 0 } 1 1 ordre
FOR I SWAP OVER I QDIV + 0 + SWAP 2 STEP
DROP 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DATANH [ENTER].
```

Exemple d'utilisation :
DL à l'ordre 7 de ATANH

```
7 [ENTER]
DATANH [ENTER]
```

renvoie

```
{ 0 1 0 (1,3) 0 (1,5) 0 (1,7) }
```

Temps d'exécution : 0,6 seconde.

21
DL de tangente en zéro
DTAN

Développement limité de TAN en 0

Ce programme est à enregistrer sous le nom : DTAN.

Son checksum est : #4E06h (244,5 octets)

Listing

```
« 1 14 ERREUR → ordre « 1 1 ordre 2 / IP
FOR N 0 1 N FOR K K 1 + PICK N K - 3 +
PICK QMULT QPLUS NEXT 1 2 N * 1 + R→C
QMULT NEXT { } 1 ordre 2 / IP 1 + START
+ 0 SWAP + NEXT 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DTAN [ENTER].
```

Exemple d'utilisation :
DL à l'ordre 7 de TAN

```
7 [ENTER]
DTAN
```

renvoie

```
{ 0 1 0 (1,3) 0 (2,15) 0 (17,315) }
```

Temps d'exécution : 2,5 secondes.

22
DL de tangente hyperbolique en zéro
DTANH

Développement limité de TANH en 0

Ce programme est à enregistrer sous le nom : DTANH.

Son checksum est : #2E9Fh (245,5 octets)

Listing

```
« 1 14 ERREUR → ordre « 1 1 ordre 2 / IP
FOR N 0 1 N FOR K K 1 + PICK N K - 3 +
PICK QMULT QPLUS NEXT -1 2 N * 1 + R→C
QMULT NEXT { } 1 ordre 2 / IP 1 + START
+ 0 SWAP + NEXT 1 ordre 1 + SUB »
»
```

Lancement du programme :
Saisir l'ordre du développement limité puis taper

```
DTANH [ENTER].
```

Exemple d'utilisation :
DL à l'ordre 7 de TANH

```
7 [ENTER]
```

renvoie

```
{ 0 1 0 (-1,3) 0 (2,15) 0 (-17,315) }
```

Temps d'exécution : 2,6 secondes.

23
Opposé d'un développement limité *DNEG*

Opposé d'un développement limité

Ce programme est à enregistrer sous le nom : DNEG.

Son checksum est : #04FBh (145 octets)

Listing

```
« 1 13 ERREUR → p ordre « p 1 ordre 1 +
SUB LIST→ DROP 0 ordre START QNEG ordre
1 + ROLL NEXT ordre 1 + →LIST »
»
```

Lancement du programme :
Saisir le développement limité et son ordre puis taper

```
DNEG [ENTER]
```

Exemple d'utilisation :

```
{ 1 2 1 } [ENTER]
2 [ENTER]
```

taper

```
DNEG [ENTER]
```

renvoie

```
{ -1 -2 -1 }
```

Temps d'exécution : 0,1 seconde.

7

Calculs matriciels

Ce chapitre traite des matrices rationnelles (écrites comme des matrices complexes).

Les programmes relatifs à ces matrices doivent être placés dans le répertoire MATRICES.

Ces matrices s'écrivent de façon usuelle avec des crochets [[ ]]

REPERTOIRE MATRICES

• **TR**	Trace d'une matrice rationnelle
• **CM→M**	Tranforme une matrice entière et son dénominateur en une matrice rationnelle
• **MSIMPL**	Tranforme une matrice rationnelle en une matrice entière et son dénominateur
• **MMULT**	Multiplication de deux matrices
• **MPLUS**	Addition de deux matrices
• **MMOINS**	Différence de deux matrices
• **LEVERRIER**	Algorithme de Leverrier
• **MINV**	Inversion d'une matrices
• **MCOMAT**	Comatrice d'une matrice
• **MPOLC**	Polynôme caractéristique
• **PENT**	Transforme un polynôme rationnel en un polynôme à coefficients entiers. (pour avoir ensuite avec RACP les racines exactes)

- **MDET** Déterminant d'une matrice
- **MRESOL** Résout AX=B (où A est inversible)
- **MCONST** Produit d'une matrice par une constante
- **MPUIS** Elevation d'une matrice rationnelle à une puissance

1
Trace d'une matrice
TR

Trace d'une matrice

Ce programme est à enregistrer sous le nom : TR.

Son checksum est : #1543h (117 octets)

Listing

```
« 1 15 ERREUR → M « 0 1 M SIZE 1 GET FOR
I 'M(I,I)' EVAL QPLUS NEXT » »
```

Lancement du programme :
Saisir la matrice puis taper

```
TR [ENTER].
```

Exemple d'utilisation :

```
[[ (1,4) (-1,4) (3,8) (1,8)  ]
[ (3,4) (-3,8) (5,4) (1,1)  ]
[ (1,4) (-7,4) (0,1) (-23,8)]
[ (3,4) (0,1)  (9,4) (-1,1) ]]
TR [ENTER]
```

renvoie

```
(-9,8)
```

soit -9/8

Temps d'exécution : 1 seconde.

2
Transformation en matrice rationnelle *CM→M*

Transforme une matrice entière et son dénominateur en une matrice rationnelle

Ce programme est à enregistrer sous le nom : CM→M.

Son checksum est : #B9AAh (250 octets)

Listing

```
« 2 16 ERREUR OVER IF 1 == THEN SWAP
DROP ELSE DUP SIZE LIST→ IF 2 == THEN *
END 1 + ROT → n c « ARRY→ n ROLLD 1 n 1
-
START c DUP2 PGCD ROT OVER / ROT ROT /
IF DUP 0 < THEN NEG SWAP NEG SWAP END
R→C n ROLLD NEXT →ARRY DUP MSIMPL SWAP
IF 1 == THEN SWAP END DROP » END
»
```

Lancement du programme :

Saisir le dénominateur puis la matrice à coefficients entiers.

Taper ensuite :

```
CM→M [ENTER].
```

Exemple d'utilisation :

```
9 [ENTER]

[[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 9 ]]
```

Taper ensuite

```
CM→M [ENTER]
```

renvoie

```
[ (1,9) (2,9) (1,3) ]
[ (4,9) (5,9) (2,3) ]
[ (7,9) (8,9) (1,1) ]]
```

Temps d'exécution : 0,7 seconde.

3
Transformation en une matrice entière et son dénominateur *MSIMPL*

Transforme une matrice rationnelle en une matrice entière et son dénominateur

Ce programme est à enregistrer sous le nom : MSIMPL.

Son checksum est : #E68Ch (259 octets)

Listing

```
« 1 15 ERREUR IF DUP TYPE 3 == THEN 1
SWAP ELSE DUP SIZE DUP LIST→ IF 2 ==
THEN * END DUP 1 + → L n m « ARRY→ DROP
n DUPN 1 n START Q→R DROP n ROLLD NEXT 1
n 1 - START DUP2 PGCD / * NEXT 1 n START
SWAP OVER SWAP C→R ROT ROT * SWAP / m
ROLLD NEXT m ROLLD L →ARRY » END
»
```

Lancement du programme :
Saisir la matrice à coefficients rationnels puis taper

```
MSIMPL [ENTER]
```

Exemple d'utilisation :

```
[[ (1,9) (2,9) (1,3) ]
 [ (4,9) (5,9) (2,3) ]
 [ (7,9) (8,9) (1,1) ]]
```

Taper

```
MSIMPL [ENTER]
```

qui renvoie :

```
9
[[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 9 ]]
```

Temps d'exécution : 1,1 seconde.

4
Multiplication de matrices *MMULT*

Multiplication de deux matrices rationnelles

Ce programme est à enregistrer sous le nom : MMULT.

Son checksum est : #652Fh (86 octets)

Listing

```
« 2 17 ERREUR MSIMPL ROT MSIMPL ROT *
ROT ROT * SWAP CM→M »
```

Lancement du programme :
Saisir les deux matrices que l'on veut multiplier puis taper

```
MMULT [ENTER]
```

Exemple d'utilisation :
Calculons le produit des deux matrices suivantes :

```
[[ (1,9) (2,9) (1,3) ]
 [ (4,9) (5,9) (2,3) ]
 [ (7,9) (8,9) (1,1) ]]
```

et

```
[[ 1 2 3 ]
 [ 4 5 6 ]
 [ 7 8 9 ]]
```

puis taper

```
MMULT [ENTER]
```

qui renvoie :

```
[[ (10,3) (4,1)  (14,3) ]
 [ (22,3) (9,1)  (32,3) ]
 [ (34,3) (14,1) (50,3) ]]
```

Temps d'exécution : 3,1 secondes.

5
Somme de deux matrices *MPLUS*

somme de deux matrices rationnelles

Ce programme est à enregistrer sous le nom : MPLUS

Son checksum est : #D141h (108,5 octets)

Listing

```
« 2 17 ERREUR MSIMPL 3 ROLL MSIMPL 3 ROLL
3 PICK * SWAP 4 PICK * + 3 ROLLD * SWAP
CM→M  »
»
```

Lancement du programme :

Saisir les deux matrices que l'on veut additionner puis taper
MPLUS [ENTER].

Exemple d'utilisation :
Calculons la somme des deux matrices suivantes :

```
[[ (1,9) (2,9) (1,3) ]
 [ (4,9) (5,9) (2,3) ]
 [ (7,9) (8,9) (1,1) ]]
```

et

```
[[ 1 2 3 ]
 [ 4 5 6 ]
 [ 7 8 9 ]]
```

Puis taper
MPLUS [ENTER]

qui renvoie

```
[[ (10,9) (20,9) (10,3) ]
 [ (40,9) (50,9) (20,3) ]
 [ (70,9) (80,9) (10,1) ]].
```

Temps d'exécution : 3,2 secondes.

6
Différence de deux matrices *MMOINS*

Différence de deux matrices rationnelles

Ce programme est à enregistrer sous le nom : MMOINS.

Son checksum est : #F331h (112 octets)

Listing

```
« 2 17 ERREUR MSIMPL 3 ROLL MSIMPL 3 ROLL
3 PICK * SWAP 4 PICK * SWAP - 3 ROLLD *
SWAP CM→M »
»
```

Lancement du programme :

Saisir les deux matrices que l'on veut soustraire puis taper
`MMOINS [ENTER].`

Exemple d'utilisation :
Calculons la différence des deux matrices suivantes :

```
[[ (1,9) (2,9) (1,3) ]
[ (4,9) (5,9) (2,3) ]
[ (7,9) (8,9) (1,1) ]]
```

et

```
[[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 9 ]]
```

Puis taper :
`MMOINS [ENTER]`

qui renvoie

```
[[ (-8,9)   (-16,9) (-8,3)  ]
 [ (-32,9)  (-40,9) (-16,3) ]
 [ (-56,9)  (-64,9) (-8,1)  ]].
```

Temps d'exécution : 3,2 secondes.

7
Produit d'une matrice par une constante
MCONST

Multiplie une matrice par une constante

Ce programme est à enregistrer sous le nom : MCONST.

Son checksum est : #5934h (91,5 octets)

Listing

```
« 2 18 ERREUR SWAP MSIMPL 3 ROLL Q→R 3
ROLL * 3 ROLLD * SWAP CM→M »
```

Lancement du programme :
Saisir la matrice et le coefficient puis taper

```
MCONST [ENTER].
```

Exemple d'utilisation :

```
[[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 9 ]]
```

```
(1,9)

MCONST [ENTER]
```

renvoie :

```
[[ (1,9) (2,9) (1,3) ]
[ (4,9) (5,9) (2,3) ]
[ (7,9) (8,9) (1,1) ]].
```

Temps d'exécution : 2 secondes.

8

Algorithme de Leverrier
LEVERRIER

Algorithme de Leverrier

Ce programme est à enregistrer sous le nom : LEVERRIER.
Le programme *Leverrier* est utilisé par des programmes qui suivent.

Son checksum est : #FC29h (320,5 octets)

Listing

```
« MSIMPL DUP SIZE 1 GET 1 + → d m n « m
IDN DUP n 2 - 1 SWAP FOR K m * DUP TR
NEG K / 3 ROLLD SWAP DUP 4 PICK * ROT +
NEXT SWAP DROP DUP m * { 1 1 } GET NEG
SWAP n ROLLD 1 IF d 1 ≠ THEN 2 n START d
* n ROLL OVER QDIV SWAP NEXT d / END n
ROLLD 1 n ROLLD 2 n FOR I I ROLL NEXT n
→LIST ROT ROT SWAP CM→M »
»
```

Quelques explications sur l'algorithme de Leverrier :
L'algorithme de Leverrier calcule le polynôme caractèristique et la comatrice d'une matrice carrée nxn A.

Soit P le polynôme caractéristique de cette matrice. On a alors :
$P(x)=\det(xI-A)=x^n + a_1 x^{n-1} + ... + a_n$

De plus, les coefficients de la comatrice de (xI-A) sont des polynômes en x de degré inférieur ou égal à n-1. La comatrice de (xI-A) s'écrit donc sous la forme : $x^{n-1}B_0 + ... + B_{n-1}$. B_i étant des matrices qui vérifient par conséquent l'égalité :

$$(xI-A)(x^{n-1}B_0 + ... + B_{n-1}) = P(x)\, I$$

Par identification des coefficients, et unicité de l'écriture polynômiale développée, il vient :

$B_0 = I$
$B_1 = a_1 I + AB_0$
...
$B_{n-1} = a_{n-1} I + AB_{n-2}$
$0 = a_n I + AB_{n-1}$

Par linéarité de la fonction trace, nous obtenons aussi :

$\mathrm{tr}\, B_1 = na_1 + \mathrm{tr}(AB_0)$
...
$\mathrm{tr}\, B_{n-1} = na_{n-1} + \mathrm{tr}(AB_{n-2})$
$0 = na_n + \mathrm{tr}(AB_{n-1})$

Si on trigonalise la matrice (xI-A), nous obtenons que I est semblable à la matrice triangulaire supérieure dont les coefficient diagonaux sont égaux à $x-x_i$

Nous avons les relations :

$$\mathrm{tr}((xI-A)^{-1}) = \sum 1/(x-x_i)$$

$$P(x) = \prod (x-x_i)$$
$$P'(x) = \sum_i \prod_{j\neq i} (x-x_j)$$

Donc $tr((xI-A)^{-1}) = P'(x)/P(x)$

$^{t}comat(xI-A) = det(xI-A) * (xI-A)^{-1}$

donc $tr(P(x)\ (xI-A)^{-1}) = tr(comat(xI-A)) = x_{n-1}\ trB_0 + ... + trB_{n-1}$

$= P'(x)$ par conséquent

L'unicité du développement polynômial s'écrit :

$tr\ B_1 = (n-1)\ a_1$
...
$tr\ B_{n-1} = (n(n-1))\ a_{n-1}$

soit finalement :

$-a_1 = tr\ AB_0$
...
$-na_n = tr\ AB_{n-1}$

Conclusion : il suffit de calculer la suite B_0, (a_1,B_1), ..., (a_{n-1},B_{n-1}), ..., a_n. On a les formules : $a_k = -tr(AB_{k-1})/k$
$B_k = a_k\ I + AB_{k-1}$

Le déterminant de A est $(-1)^n\ a_n$.
La comatrice de A est $(-1)^{n-1}\ B_{n-1}$.

9
Inverse d'une matrice
MINV

Calcule l'inverse d'une matrice de rationnels

Ce programme est à enregistrer sous le nom : MINV.

Son checksum est : #8379h (127 octets)

Listing

```
« 1 15 ERREUR LEVERRIER SWAP 1 GET IF
DUP 0 SAME THEN "NON INVERSIBLE" DOERR
END QINV QNEG MCONST »
```

Lancement du programme :
Saisir la matrice dont on veut connaitre l'inverse puis taper

```
MINV [ENTER]
```

Exemple d'utilisation :

```
[[ 1 2 3 ]
[ 0 1 0 ]
[ 3 0 5 ]]
```

puis taper

```
MINV [ENTER]
```

qui renvoie

```
[[ (-5,4)  (5,2)  (3,4) ]
[  (0,1)  (1,1)  (0,1) ]
[  (3,4) (-3,2) (-1,4) ]]
```

Temps d'exécution : 4,4 secondes.

10
Comatrice
MCOMAT

Comatrice d'une matrice rationnelle

Ce programme est à enregistrer sous le nom : MCOMAT.

Son checksum est : #CD57h (95 octets)

Listing

```
« 1 15 ERREUR LEVERRIER SWAP DROP TRN
DUP SIZE 1 GET 1 - -1 SWAP ^ MCONST »
```

Lancement du programme :
Saisir la matrice dont on cherche la comatrice puis taper

```
MCOMAT [ENTER]
```

Exemple d'utilisation:

```
[[ 1 2 3 ]
 [ 0 1 0 ]
 [ 3 0 5 ]]
```

Puis taper

```
MCOMAT [ENTER]
```

qui renvoie :

```
[[ 5   0  -3 ]
 [ -10 -4  6 ]
 [ -3  0   1 ]]
```

Temps d'exécution : 2,7 secondes.

11

Polynôme caractéristique *MPOLC*

Polynôme caractèristique

Ce programme est à enregistrer sous le nom : MPOLC.

Son checksum est : #C065h (57 octets)

Listing

```
« 1 15 ERREUR LEVERRIER DROP »
```

Lancement du programme :
Saisir la matrice dont on cherche le polynôme caractèristique puis taper

```
MPOLC [ENTER]
```

Exemple d'utilisation :

```
[[ 1 2 3 ]
[ 0 1 0 ]
[ 3 0 5 ]]

MPOLC [ENTER]
```

renvoie :

```
{ 4 2 -7 1 }
```

c'est-à-dire 4 + 2x - 7x^2 + x^3

Temps d'exécution : 2,4 secondes.

12
Transformation en polynôme à coefficients entiers
PENT

Transforme un polynôme rationnel en polynôme à coefficients entiers (pour utiliser à profit RACP)

Ce programme est à enregistrer sous le nom : PENT.

Son checksum est : #54EDh (179 octets)

Listing

```
« 1 15 ERREUR LIST→ → n « n DUPN 1 1 n
START SWAP Q→R 3 ROLLD PPCM SWAP n 1 +
ROLLD NEXT n 1 + ROLLD n DROPN 1 n START
SWAP OVER QMULT n 1 + ROLLD NEXT DROP n
→LIST » »
```

Lancement du programme :
Saisir un polynôme rationnel puis taper

```
PENT [ENTER]
```

Exemple d'utilisation :

```
{ (1,2) (1,6) (1,8) (2,3) }
PENT [ENTER]
```

renvoie

```
{ 12 4 3 16 }.
```

Temps d'exécution : 1 seconde.

13
Déterminant d'une matrice *MDET*

Déterminant d'une matrice rationnelle

Ce programme est à enregistrer sous le nom : MDET.

Son checksum est : #6ABEh (83 octets)

Listing

```
« 1 15 ERREUR MPOLC DUP SIZE 1 - -1 SWAP
^ SWAP 1 GET QMULT »
```

Lancement du programme :
Saisir la matrice dont on cherche le déterminant puis taper

```
MDET [ENTER]
```

Exemple d'utilisation :

```
[[ 1 2 3 ]
[ 0 1 0 ]
[ 3 0 5 ]]
```

puis taper

```
MDET [ENTER]
```

renvoie

```
-4
```

Temps d'exécution : 2,6 secondes.

14
Résolution d'un système de Cramer *MRESOL*

Résolution d'un système de Kramer

Le programme est à enregistrer sous le nom : MRESOL.

Son checksum est : #AA6Eh (61,5 octets)

Listing

```
« 2 19 ERREUR MINV TRN MMULT »
```

Lancement du programme :
Saisir le vecteur solution et la matrice représentant le système de Kramer puis taper

```
MRESOL [ENTER]
```

Exemple d'utilisation :
Résolvons le système

$$\begin{cases} x + y + z = 6 \\ 2x + z = 5 \\ -2x + 3y = 4 \end{cases}$$

```
[[ 6 5 4 ]]
```
le vecteur solution

```
[[ 1 1 1 ]
[ 2 0 1 ]
[-2 3 0 ]]
```
la matrice du système

```
MRESOL [ENTER]
```
renvoie
```
[[ 1 2 3 ]]
```

La solution est x=1 , y=2 , z=3

Temps d'exécution : 3,1 secondes.

15
Exponentiation d'une matrice *MPUIS*

Elevation d'une matrice à une puissance

A enregistrer sous le nom : MPUIS.

Son checksum est : #4495h (265 octets)

Listing

```
« 2 18 ERREUR OVER SIZE 1 GET IDN → A n
B « IF n 0 < THEN A MINV 'A' STO n NEG
'n' STO END WHILE n REPEAT IF n 2 MOD
THEN A B MMULT 'B' STO END A DUP MMULT
'A' STO n 2 / IP 'n' STO END B » »
```

Lancement du programme :
Saisir la matrice et la puissance entière puis taper

```
MPUIS [ENTER]
```

Exemple d'utilisation :

```
[[ 1 0 0 ]
 [ 0 2 0 ]
 [ 0 0 3 ]]

3
```

Puis taper

```
MPUIS [ENTER]
```

qui renvoie :

```
[[ 1 0 0 ]
 [ 0 8 0 ]
 [ 0 0 27]]
```

Temps d'exécution : 1,4 seconde.

8

Matrices algébriques

Cette partie traite des matrices algébriques (pouvant contenir des éléments non évalués).
Les programmes relatifs à ces matrices se trouvent dans le répertoire ALG.
Ces matrices ne peuvent pas s'écrire avec des crochets. Nous les écrirons donc avec des listes {{ }}
La notation {{ A B } { C D }} représente la matrice :

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

LE REPERTOIRE *ALG*

- **A→M** Transforme une matrice classique en une matrice algébrique
- **M→A** Transforme une matrice algébrique en une matrice classique (Array)
- **MSIZE** Dimensions d'une matrice
- **MOPER** Opérations sur les matrices
- **MPLUS** Somme de deux matrices
- **MMOINS** Différence de deux matrices
- **MCONST** Produit d'une matrice par une constante
- **MMULT** Produit de deux matrices
- **MTR** Trace d'une matrice carrée
- **MTRN** Transposée d'une matrice

- **LEVERRIER** Algorithme de Leverrier
- **MINV** Inverse d'une matrice
- **MCOMAT** Comatrice d'une matrice
- **MPOLC** Polynôme caractéristique
- **MDET** Déterminant d'une matrice
- **MCOLCT** Rassemble les termes d'une matrice
- **MRESOL** Résolution exacte d'un système de Cramer
- **MAT→** Eclate la matrice (ARRY→)
- **→MAT** Construit une matrice (→ARRY)

7

Transformation en matrice algébrique *A→M*

Tranforme une matrice usuelle en une matrice algébrique

Ce programme est à enregistrer sous le nom : A→M

Son checksum est : #BE29h (130 octets)

Listing

```
« 1 15 ERREUR DUP SIZE LIST→ DROP → p n
« ARRY→ DROP 1 p FOR I n →LIST p I - n
* I + ROLLD NEXT p →LIST » »
```

Lancement du programme :

Saisir une matrice classique puis taper

```
A→M [ENTER]
```

Exemple d'utilisation :

```
[[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 9 ]]
```

A→M renvoie

```
{{ 1 2 3 }
{ 4 5 6 }
{ 7 8 9 }}
```

Temps d'exécution : 0,2 seconde.

2
Taille d'une matrice algébrique *MSIZE*

Taille d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MSIZE

Son checksum est : #CC04h (57 octets)

Listing

```
« 1 15 ERREUR DUP SIZE SWAP 1 GET SIZE »
```

Lancement du programme :

Saisir la matrice algébrique puis taper

```
MSIZE [ENTER]
```

Exemple d'utilisation :

```
{{ 1  2  3  4 }
{ 5  6  7  8 }
{ 9 10 11 12 }}
MSIZE [ENTER]
```

renvoie

```
3
4
```

Temps d'exécution : immédiat.

3
Transformation en matrice usuelle *M→A*

Transforme la matrice en une matrice usuelle

Ce programme est à enregistrer sous le nom : M→A

Son checksum est : #7B7Fh (109 octets)

Listing

```
« 1 15 ERREUR DUP MSIZE → p n « LIST→ 1
- 1 SWAP START + NEXT LIST→ DROP p n 2
→LIST →ARRY » »
```

Lancement du programme :

Saisir la matrice algébrique à coefficients numériques puis taper

```
M→A [ENTER]
```

Exemple d'utilisation

```
{{ 1 2 3 }
{ 4 5 6 }
{ 7 8 9 }}
```

taper

```
M→A [ENTER]
```

renvoie

```
[[ 1 2 3 ]
 [ 4 5 6 ]
 [ 7 8 9 ]]
```

Temps d'exécution : 0,2 seconde.

4

Opérations sur les matrices algébriques *MOPER*

Opérations sur les matrices algébriques

Ce programme est à enregistrer sous le nom : MOPER

Son checksum est : #4822h (256,5 octets)

Listing

```
« 3 20 ERREUR SWAP DUP MSIZE DUP2 * → A
oper B p n c « B LIST→ 1 - 1 SWAP START
+ NEXT LIST→ DROP A LIST→ 1 - 1 SWAP
START + NEXT LIST→ DROP A LIST→ 1 - 1
SWAP START + NEXT LIST→ DROP 1 c START c
1 + ROLL oper EVAL c ROLLD NEXT 1 p FOR
I n →LIST p I - n * I + ROLLD NEXT p
→LIST » »
```

Lancement du programme :
Saisir les deux matrices, puis l'opération à effectuer sous forme de programme . Enfin taper :

```
MOPER [ENTER]
```

Exemple d'utilisation :

```
Matrice_1 coefficients_aij
Matrice_2 coefficients_bij
« operation »
```

Les coefficients de la matrice résultat a pour coefficients :

```
aij bij « oper ».
```

Remarque :
Ce programme est utilisé par les programmes qui suivent.

5
Somme de deux matrices algébriques *MPLUS*

Addition d'une matrice algébrique à une autre matrice algébrique

Ce programme est à enregistrer sous le nom : MPLUS

Son checksum est : #52E9h (43 octets)

Listing

```
« « + » MOPER »
```

Lancement du programme :
Saisir les deux matrices que l'on veut additionner puis taper

```
MPLUS [ENTER]
```

Exemple d'utilisation :

```
{{ A B C }
{ D E F }
{ G H I }}

{{ J K L }
{ M N O }
{ P Q R }}
```

puis taper

```
MPLUS [ENTER]
```

renvoie

```
{{ 'A+J' 'B+K' 'C+L' }
{ 'D+M' 'E+N' 'F+O' }
{ 'G+P' 'H+Q' 'I+R' }}
```

Temps d'exécution : 0,9 seconde.

6
Différence de deux matrices *MMOINS*

Soustraction d'une matrice algébrique à une autre matrice algébrique

Ce programme est à enregistrer sous le nom : MMOINS

Son checksum est : #EF1Dh (44 octets)

Listing

```
« « - » MOPER »
```

Lancement du programme :
Saisir les deux matrices que l'on veut soustraire puis taper

```
MMOINS [ENTER]
```

Exemple d'utilisation :

```
{{ A B C }
{ D E F }
{ G H I }}

{{ J K L }
{ M N O }
{ P Q R }}
```

puis taper

```
MMOINS [ENTER]
```

renvoie

```
{{ 'A-J' 'B-K' 'C-L' }
{ 'D-M' 'E-N' 'F-O' }
{ 'G-P' 'H-Q' 'I-R' }}
```

Temps d'exécution : 0,9 seconde.

7
Produit par une constante *MCONST*

Produit d'une matrice par une constante

Ce programme est à enregistrer sous le nom : MCONST

Son checksum est : #7ECDh (233 octets)

Listing

```
« 2 18 ERREUR OVER MSIZE DUP2 * 3 ROLLD
2 →LIST → nb taille « 1 nb 1 - START
DUP NEXT 1 taille 1 GET FOR I taille 2
GET →LIST nb taille 2 GET I * - 1 +
ROLLD NEXT taille 1 GET →LIST « * »
MOPER »
»
```

Lancement du programme :
Saisir la matrice rationnelle puis le coefficient par lequel on veut la multiplier. Taper ensuite

```
MCONST [ENTER]
```

Exemple d'utilisation :

```
{{ A B C }
{ D E F }
{ G H I }} [ENTER]

'Z' [ENTER]

MCONST [ENTER]
```

renvoie

```
{{ 'A*Z' 'B*Z' 'C*Z' }
{ 'D*Z' 'E*Z' 'F*Z' }
{ 'G*Z' 'H*Z' 'I*Z' }}
```

Temps d'exécution : 1,3 seconde.

8
Produit de matrices rationnelles *MMULT*

Produit de deux matrices rationnelles

Ce programme est à enregistrer sous le nom : MMULT

Son checksum est : #7271h (202,5 octets)

Listing

```
« 2 17 ERREUR DUP2 DUP SIZE SWAP 1 GET
SIZE ROT SIZE → A B q p n « 1 n FOR I 1
p FOR J 0 1 q FOR K A I GET K GET B K
GET J GET * + NEXT NEXT p →LIST NEXT n
→LIST » »
```

Lancement du programme :
Saisir les deux matrices que l'on veut multiplier puis taper

```
MMULT [ENTER]
```

Exemple d'utilisation :

```
{{ A 2 3 }
{ 4 B 6 }
{ 7 8 C }}

{{ E F G }
{ 0 1 0 }
{ H I J }}
```

puis taper

```
MMULT [ENTER]
```

renvoie

```
{{ 'A*E+3*H' 'A*F+2+3*I' 'A*G+3*J' }
{ '4*E+6*H' '4*F+B+6*I' '4*G+6*J' }
{ '7*E+C*H' '7*F+8+C*I' '7*G+C*J' }}
```

Temps d'exécution : 3,8 seconde.

9
Trace d'une matrice algébrique *MTR*

Trace d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MTR

Son checksum est : #7D47h (94,5 octets)

Listing

```
« 1 15 ERREUR → M « 0 1 M SIZE FOR I M I
GET I GET + NEXT » »
```

Lancement du programme :

Saisir la matrice algébrique puis taper

```
MTR [ENTER]
```

Exemple d'utilisation :

```
{{ A B C }
{ D E F }
{ G H I }}

MTR [ENTER]
```

renvoie

```
'A+E+I'
```

Temps d'exécution : 0,2 seoonde.

10
Transposée d'une matrice algébrique *MTRN*

Transposée d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MTRN

Son checksum est : #2EA3h (138,5 octets)

Listing

```
« 1 15 ERREUR DUP MSIZE → M n p « 1 p
FOR I 1 n FOR J M J GET I GET NEXT n
→LIST NEXT p →LIST » »
```

Lancement du programme :
Saisir la matrice puis taper

```
MTRN [ENTER]
```

Exemple d'utilisation :

```
{{ A B C D }
{ E F G H }
{ I J K L }}

MTRN [ENTER]
```

renvoie

```
{{ A E I }
{ B F J }
{ C G K }
{ D H L }}
```

Temps d'exécution : 0,7 seconde.

11
Rassemblement des termes *MCOLCT*

Rassemble les termes d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MCOLCT

Son checksum est : #CAEFh (218,5 octets)

Listing

```
« 1 15 ERREUR DUP DUP SIZE SWAP 1 GET
SIZE → n p « LIST→ 1 - 1 SWAP DUP2 IF ≤
THEN START + NEXT ELSE DROP2 END LIST→
DROP 1 n p * START COLCT n p * ROLLD
NEXT 1 n FOR I p →LIST p n I - * I +
ROLLD NEXT n →LIST » »
```

Lancement du programme :
Entrez une matrice puis taper

```
MCOLCT [ENTER]
```

Exemple d'utilisation :

```
{{ 'A+A'  'C-C+1' 3 }
{ 'A-B-C+2*B' 3 2 }
{ 1 '2-3' 0 }}

MCOLCT [ENTER]
```

renvoie

```
{{ '2*A' 1 3 }
{ 'A+B-C' 3 2 }
{ 1 -1 0 }}
```

Temps d'exécution : 2,8 secondes.

12
Algorithme de Leverrier
LEVERRIER

Algorithme de Leverrier

Ce programme est à enregistrer sous le nom : LEVERRIER

Son checksum est : #94BBh (317 octets)

Listing

```
« DUP SIZE → m n « n IDN A→M DUP 1 n 1
- FOR K m MMULT DUP MTR NEG K / 3 ROLLD
```

```
SWAP DUP 4 PICK MCONST ROT MPLUS NEXT
SWAP DROP DUP → m1 « 0 1 n FOR I m1 1
GET I GET m I GET 1 GET * + NEXT » NEG
SWAP n 1 + ROLLD 1 n 1 + ROLLD COLCT 2 n
1 + FOR I I ROLL COLCT NEXT n 1 + →LIST
SWAP MCOLCT » »
```

Lancement du programme :
Ce programme est utilisé par les programmes qui suivent.
Pour avoir des explications sur ce programme, consultez son homologue dans le chapitre précédent.

13
Inverse d'une matrice algébrique *MINV*

Inverse d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MINV

Son checksum est : #0014h (75,5 octets)

Listing

```
« 1 15 ERREUR LEVERRIER SWAP 1 GET NEG
INV MCONST »
```

Lancement du programme :
Saisir la matrice que l'on veut inverser puis taper
`MINV [ENTER]`

Exemple d'utilisation :

```
{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }}
```

taper

```
MINV [ENTER]
```

renvoie

```
{{ 0 0   1  }
{ 1 0 '-A' }
{ 0 1   0  }}
```

Temps d'exécution : 19,8 secondes.

14
Comatrice d'une matrice algébrique *MCOMAT*

Comatrice d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MCOMAT

Son checksum est : #1AB7h (95 octets)

Listing

```
« 1 15 ERREUR LEVERRIER SWAP DROP MTRN
DUP SIZE 1 - -1 SWAP ^ MCONST
»
```

Lancement du programme :
Saisir la matrice dont on cherche la comatrice puis taper

```
MCOMAT [ENTER]
```

Exemple d'utilisation :

```
{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }}
```

puis taper

```
MCOMAT [ENTER]
```

renvoie

```
{{ 0   1  0 }
{ 0   0  1 }
{ 1 '-A' 0 }}
```

Temps d'exécution : 20,3 secondes.

15
Polynôme caractéristique *MPOLC*

Polynôme caractéristique d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MPOLC.

Son checksum est : #C065h (57 octets)

Listing

```
« 1 15 ERREUR LEVERRIER DROP »
```

Lancement du programme :

Saisir la matrice dont on cherche le polynôme caractèristique puis taper

```
MPOLC [ENTER]
```

Exemple d'utilisation :

```
{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }}
```

taper

```
MPOLC [ENTER]
```

renvoie

```
{ -1 0 '-A' 1 }
```

c'est-à-dire -1 -a x^2 + x^3

Temps d'exécution : 18,9 secondes.

Remarque : une fois le polynôme caractèristique calculé, vous pouvez lancer le calcul des valeurs propres de la matrice. Pour cela, il suffit d'utiliser le programme **RACP** du répertoire **POLY**. Les racines du polynôme sont les valeurs propres de la matrice.

16
Déterminant
MDET

Calcule le déterminant d'une matrice algébrique

Ce programme est à enregistrer sous le nom : MDET.

Son checksum est : #4FEDh (54,5 octets)

Listing

```
« MPOLC DUP SIZE 1 - -1 SWAP ^ SWAP 1
GET * »
```

Lancement du programme :

Saisir la matrice dont on cherche le déterminant puis taper

```
MDET [ENTER]
```

Exemple d'utilisation :

```
{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }}
```

puis taper

```
MDET [ENTER]
```

renvoie

```
1
```

Temps d'exécution : 18,9 secondes.

17
Résolution d'un système de Cramer *MRESOL*

Résolution exacte d'un système de Cramer

Ce programme est à enregistrer sous le nom : MRESOL

Son checksum est : #E258h (66,5 octets)

Listing

```
« 2 19 ERREUR MINV MTRN MMULT »
```

Lancement du programme :

Saisir le vecteur solution comme une matrice à une seule ligne et la matrice représentant le système de Cramer puis taper :

```
MRESOL [ENTER]
```

Exemple d'utilisation :
Pour résoudre le système :

```
a*x + y   = a*c+d
z  = e
x  = c.
```

```
{{ 'A*C+D' E C }} [ENTER]

{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }} [ENTER]
```

taper

```
MRESOL [ENTER]
```

renvoie

```
{{ C D E }}
```

soit x=c , y=d , z=e.

Temps d'exécution : 21,6 secondes.

18
Décomposition d'une matrice
MAT→

Décompose une matrice algébrique

Ce programme est à enregistrer sous le nom : MAT→

Son checksum est : #D39Ch (133 octets)

Listing

```
« 1 15 ERREUR DUP DUP SIZE SWAP 1 GET
SIZE → n p « {} SWAP LIST→ 1 SWAP START
+ NEXT LIST→ DROP n p 2 →LIST DROP n p
2 →LIST » »
```

Lancement du programme :
Saisir une matrice puis taper

```
MAT→ [ENTER]
```

Exemple d'utilisation :

```
{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }} [ENTER]

MAT→ [ENTER]
```

renvoie

```
'A'
1
0
0
0
1
1
0
0
{ 3 3 }
```

Taille de la matrice

Temps d'exécution : 0,2 seconde.

19
Recomposition d'une matrice →*MAT*

Recompose une matrice

Ce programme est à enregistrer sous le nom : →MAT

Son checksum est : #D530h (156 octets)

Listing

```
« LIST→ DROP → n p « IF DEPTH n p * <
THEN "Pas assez d'arguments" DOERR END 1
n FOR I p →LIST n I - p * I + ROLLD NEXT
n →LIST » »
```

Lancement du programme :
Entrez les éléments de la matrice et sa taille, comme on le fait avec →ARRY puis taper :

```
→MAT [ENTER]
```

Exemple d'utilisation :

```
'A'     1     0     0     0
1       1     0     0
{ 3 3 } Taille de la matrice

→MAT [ENTER]
```

renvoie

```
{{ A 1 0 }
{ 0 0 1 }
{ 1 0 0 }}
```

Temps d'exécution : 0,2 seconde.

9

Applications des matrices

Les programmes présentés dans ce chapitre sont à placer dans la répertoires APPLIM (/ARITHM/MATRICES/APPLIM) et APPLIA (/ARITHM/MATRICES/ALG/APPLIA).

CONTENU DU REPERTOIRE APPLIM

- **JORDAN** Jordanisation d'une matrice nxn
- **GETL** Renvoie la nième ligne d'une matrice
- **SIMP** Renvoie le pgcd des éléments d'une matrice et la matrice divisée par ce pgcd
- **FLB?** Teste si une famille de vecteurs est libre
- **SYSMIN** Prend une matrice et renvoie une liste maximale de lignes indépendantes
- **L→ARR** Transforme une liste de vecteurs en matrice
- **BKER** Fabrique une famille de vecteurs libres
- **→MATR** Fabrique une matrice d'après une formule générique donnée

CONTENU DU REPERTOIRE APPLIA

- **GAUSS** Algorithme de Gauss
- **RSYST** Résolution symbolique d'un système quelconque
- **ESPPE** Equation des espaces propres d'une matrice

1

Jordanisation des matrices *JORDAN*

Jordanisation des matrices nxn à coefficients algébriques.

Ce programme est à enregistrer sous le nom : JORDAN.
(Dans le sous-répertoire APPLIM)
Son checksum est : #92C9h (850 octets)

Listing

```
« LIST→ DUP 2 + ROLL MSIMPL SIMP DUP
SIZE DUP 1 GET 0 0 0 {} 0 0 0 → D C A S
N P M K B F U G « 1 SWAP START 0 {} DUP
A DUP IDN 6 ROLL CLLCD "λ=" OVER + 1 DISP
D QMULT C / OVER * ROT SWAP - SWAP 0 'P'
STO 0 'K' STO {} 'F' STO WHILE OVER *
ROT OVER SYSMIN N OVER SIZE - DUP 6
ROLLD 8 PICK - DUP REPEAT N BKER 4 ROLLD
4 ROLLD 'P' INCR 2 DISP END DROP2 DROP2
'U' STO 'M' STO DO SWAP 4 PICK - K - IF
DUP THEN DUP 'K' STO+ {} 'G' STO 'M'
OVER P * STO- 1 1 ROT START 0 DO DROP
GETI DUP F + UNTIL FLB? END {} U ROT D P
1 - ^ * 1 P START ROT OVER SWAP + ROT
ROT OVER SWAP * D / C * NEXT DROP2 P N 2
→LIST L→ARR SIMP SWAP DROP 'G' STO+
NEXT G 'F' STO+ END DROP2 'P' DECR 3
DISP UNTIL M NOT END 'B' F STO+ P 2 * 2
+ DROPN NEXT B S L→ARR TRN DUP "P" →TAG
SWAP DUP MINV D A C * CM→M MMULT SWAP
MMULT "P-1AP" →TAG » »
```

Remarque : ce programme utilise GETL, SIMP, FLB?, SYSMIN, L→ARR et BKER.

Lancement du programme :
Saisir la matrice nxn que l'on veut jordaniser ainsi que des valeurs propres (sans multiplicité). (On les trouve grâce au programme RACP). Tapez ensuite : **JORDAN [ENTER]**

Exemples d'utilisation :

Cherchons une réduite de Jordan de la matrice

$$A = \begin{vmatrix} 1 & 1 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{vmatrix}$$

```
[[ 1 1 3 ][ 0 2 1 ][ 0 0 3 ]]  [ENTER]
{ 1 2 3 } [ENVOI]   (valeurs propres)
```

puis taper

```
JORDAN [ENTER]
```

renvoie

```
P: [[ 2 1 1 ][ 1 1 0 ][ 1 0 0 ]]
   matrice de passage
P-1AP: [[ 3 0 0 ][ 0 2 0 ][ 0 0 1 ]]
   réduite de Jordan
```

Temps d'exécution : 33,6 secondes.

Cherchons une réduite de Jordan de la matrice

$$A = \begin{vmatrix} -4 & 0 & -2 \\ 0 & 1 & 0 \\ 5 & 1 & 3 \end{vmatrix}$$

```
[[ -4 0 -2 ][ 0 1 0 ][ 5 1 3 ]] [ENTER]
{ 1 -2 }
```

puis taper

```
JORDAN [ENTER]
```

NB : Les valeurs propres nous sont données par le programme RACP.

JORDAN renvoie

```
P: [[ 1 2 -4 ][ 0 0 3 ][ -1 -5 9 ]]
P-1AP: [[ -2 0 0 ][ 0 1 1 ][ 0 0 1 ]]
```

Temps d'exécution : 35,5 secondes.

Cherchons une réduite de Jordan de la matrice

$$A = \begin{vmatrix} 7 & 1 & -2 \\ -1 & 5 & 2 \\ 0 & 0 & 6 \end{vmatrix}$$

```
[[ 7 1 -2 ][ -1 5 2 ][ 0 0 6 ]] [ENTER]
{ 6 }
```

puis taper

```
JORDAN [ENTER]
```

renvoie

```
P: [[ -5 1 1 ][ 1 -1 0 ][ -2 0 0 ]]
P-1AP: [[ 6 0 0 ][ 0 6 1 ][ 0 0 6 ]]
```

Temps d'exécution : 20,8 secondes.

Cherchons une réduite de Jordan de la matrice

$$A = \begin{vmatrix} 8 & -1 & -5 \\ -2 & 3 & 1 \\ 4 & -1 & -1 \end{vmatrix}$$

```
[[ 8 -1 -5 ][ -2 3 1 ][ 4 -1 -1 ]]
{ 4 2 }[ENTER]
```

puis taper

```
JORDAN [ENTER]
```

renvoie

```
P: [[ 1 -9 5 ][ 1 -9 -1 ][ 1 -9 -2 ]]
P-1AP: [[ 2 0 0 ][ 0 4 1 ][ 0 0 4 ]]
```

Temps d'exécution : 36,9 secondes.

Remarque :

JORDAN travaille avec des matrices à coefficients rationnels. La taille de la matrice à jordaniser peut être quelconque en théorie. Cependant lors de calculs avec de grosses matrices, on arrive à des dépassements de capacité.

2
Renvoi d'une ligne d'une matrice GETL

Prend une matrice et un entier n, et renvoie la nième ligne de la matrice.

Ce programme est à enregistrer sous le nom : GETL
(Dans le sous-répertoire APPLIM)

Son checksum est : #96B9h (93,5 octets)

Listing

Ce programme est à assembler avec ASS. Tapez la chaine suivante sans espaces ni sauts de lignes. Faites DUP BYTES et vérifiez qu'il est bien égal à : #A15Dh (175 octets). Si c'est le cas, compilez la chaîne à l'aide de la commande ASS.

```
"D9D 20D 8A8 12B F81 4EB 46D 9D2 0AE C81
E0E 302 C23 0BB 491 9A5 30F A45 08C 636
399 162 AC8 103 826 E90 160 831 62C E30
FED 303 223 0BD 370 083 16E 855 3E9 016
409 264 337 085 230 881 30D EE3 24B 2A2
244 304 929 1B2 130 B21 30"
```

3
pgcd des éléments d'une matrice SIMP

Prend une matrice et renvoie le PGCD de ses éléments ainsi que la matrice divisée par ce pgcd.

Ce programme est à enregistrer sous le nom : SIMP.
(Dans le sous-répertoire APPLIM)

Son checksum est : #49D4h (78,5 octets)

Listing

```
« DUP ARRY→ IF LIST→ 2 == THEN * END 1
- 1 SWAP START PGCD NEXT DUP 1 IFTE SWAP
OVER / »
```

Remarque : ce programme est un sous programme de JORDAN.

Exemple d'utilisation :

Prenons la matrice suivante :

```
[[ 5 15 35 ][ 25 25 25 ][ 75 85 35 ]]
```

SIMP renvoie :

```
5
```
(pgcd des éléments)
```
[[ 1 3 7 ][ 5 5 5 ][ 15 17 7 ]]
```
(matrice / 5)

Temps d'éxécution : 0,5 seconde.

4
Test de vecteurs libres FLB?

Teste si une famille de vecteurs est libre.

Ce programme est à enregistrer sous le nom : FLB?.
(Dans le sous-répertoire APPLIM)

Son checksum est : #5A06h (215,5 octets)

Listing

```
« DUP SIZE 0 →F P N « IF P 1 ≠ THEN 1 P
FOR K F K GET ARRY→ LIST→ IF 1 == THEN
1 SWAP END DROP 'N' STO+ NEXT N F 1 GET
SIZE + →ARRY DUP TRN * SIMP SWAP DROP
DET .5 > ELSE F LIST→ DROP ABS END » »
```

Exemple d'utilisation :

Considérons la famille de vecteurs suivante :

```
{ [1 1 3 ] [ 1 0 2 ] [ 2 2 6 ] }
```

FLB? [ENTER] renvoie :

```
0
```

ce qui signifie que la famille est liée.

Si la famille est libre, le programme renvoie 1.

Temps d'exécution : 1,2 seconde.

5
Liste maximale de lignes non liées SYSMIN

Prend une matrice et renvoie une liste maximale de lignes indépendantes.

Ce programme est à enregistrer sous le nom : SYSMIN.
(Dans le sous-répertoire APPLIM)

Son checksum est : #731Ch (213 octets)

Listing

```
« 0 → M P « IF M ABS THEN WHILE M 'P'
INCR GETL DUP ABS NOT REPEAT DROP END
SIMP SWAP DROP 1 →LIST P 1 + M SIZE 1
GET FOR K DUP M K GETL SIMP SWAP DROP +
DUP IF FLB? THEN SWAP END DROP NEXT ELSE
{} END » »
```

Exemple d'utilisation :
Considérons la matrice suivante :

```
[[ 1 2 3 ][ 4 5 6 ][7 8 9]]
```

SYSMIN renvoie :

```
{ [1 2 3] [4 5 6] }
```

Temps d'exécution : 2,7 secondes.

6
Transforme une liste de vecteurs L→ARR

Prend une liste de vecteurs et de matrices et une taille et reconstruit une matrice.

Ce programme est à enregistrer sous le nom : L→ARR.
(Dans le sous-répertoire APPLIM)

Son checksum est : #44DFh (78,5 octets)

Listing

```
« → L N « 1 L SIZE FOR K L K GET ARRY→
DROP NEXT N →ARRY » »
```

Exemple d'utilisation :

```
{ [1 2 3] [4 5 6] [7 8 9] }
{ 3 3 }        taille de la matrice
```

Temps d'exécution : 0,2 seconde.

7
Transforme une liste de vecteurs BKER

Prend deux lignes de vecteurs B et S, et deux entiers k et n. Il construit une famille de k vecteurs indépendants F dans le noyau du système représenté par S, tel que B union F soit libre. n est la taille des vecteurs.

Ce programme est à enregistrer sous le nom : BKER.
(Dans le sous-répertoire APPLIM)

Son checksum est : #846Bh (458,5 octets)

Listing

```
« IF OVER THEN 3 PICK SIZE 0 → B S K N P
L « IF P 1 > THEN S 2 P FOR K DUP K GET
SWAP 1 1 K 1 - START GETI DUP DUP DOT 5
ROLL SWAP OVER * ROT ROT OVER DOT * -
SIMP SWAP DROP ROT ROT NEXT ROT PUT NEXT
'S' STO END N 1 →LIST 0 CON {} DO OVER
'L' INCR 1 PUT IF P THEN 1 P FOR I S I
GET DUP2 DOT OVER * ROT ROT DUP DOT * -
SIMP SWAP DROP NEXT END IF B OVER * DUP
FLB? THEN 'B' STO + 'K' DECR 4 DISP ELSE
DROP2 END UNTIL K NOT END SWAP DROP B »
ELSE DROP2 DROP {} SWAP END »
```

Exemple d'utilisation :

```
{}
{ [-1 0 0] [0 -1 0] }
1
3
```

BKER renvoie :

```
{ [0 0 1 1] }
{ [0 0 1 1] }
```

Temps d'exécution : 4,5 secondes.

8

Fabrique une matrice →MATR

Ce programme remplit une matrice. Il prend un programme au niveau 2 et la taille de la matrice au niveau 1 (liste d'un ou de deux entiers). Le programme renvoie la matrice correspondante.

Ce programme est à enregistrer sous le nom : →MATR
(Dans le sous-répertoire APPLIM)

Son checksum est : #B0C0h (152 octets)

Listing

```
« DUP LIST→ IF 1 == THEN ROT 1 ROT FOR I
I OVER EVAL ROT ROT NEXT ELSE → N « 1
SWAP FOR I 1 N FOR J I J 4 PICK EVAL ROT
ROT NEXT NEXT » SWAP END DROP →ARRY »
```

Exemple d'utilisation :

```
« + »
{ 5 5 }
```

→MATR renvoie :

```
[[ 2 3 4 5 6  ]
[ 3 4 5 6 7  ]
[ 4 5 6 7 8  ]
[ 5 6 7 8 9  ]
[ 6 7 8 9 10 ]]
```

Temps d'exécution : 0,7 seconde.

Remarque : la matrice a pour coordonnées i+j.

9
Algorithme de Gauss
GAUSS

Algorithme de Gauss, calcul du rang

Ce programme est à enregistrer sous le nom : GAUSS.
(Dans le sous-répertoire APPLIA)

Son checksum est : #1334h (833,5 octets)

Listing

```
« 1 15 ERREUR DUP MSIZE 0 0 {} →M n p
rang piv M3 « M MAT→ →ARRY 'M3' STO 1 n
FOR I 1 p FOR J 1 n I - p J - MIN 1 +
FOR K 1 K FOR L 1 K FOR P 'M3(I+L-1,J+P-
1)' →NUM NEXT NEXT K DUP 2 →LIST →ARRY
DET IF ABS 1E-6 > K rang > AND THEN K
'rang' STO END NEXT NEXT NEXT 1 n FOR I
0 1 n I FOR J IF M J GET I GET ABS 1E-8
> THEN DROP2 J 0 END -1 STEP NOT IF THEN
'piv' STO IF I 1 > THEN M 1 I 1 - SUB
LIST→ DROP END M piv GET I n FOR J IF J
piv ≠ THEN M J GET M piv GET + LIST→ DROP
M piv GET I GET QINV M J GET I GET QMULT
1 p FOR K SWAP OVER QMULT p K - 3 + ROLL
SWAP QMOINS p K - 2 * 1 + K + ROLLD NEXT
DROP p →LIST END NEXT n →LIST 'M' STO
ELSE DROP END NEXT M rang » »
```

Ce programme calcule le rang de la matrice et sa réduite de Gauss.

Lancement du programme :
Saisir une matrice puis taper

```
GAUSS [ENTER]
```

Exemple d'utilisation :

```
{{ 1 2 3 }
{ 4 5 6 }
{ 7 8 9 }}

GAUSS [ENTER]
```

renvoie

```
{{ 1  2  3 }
{ 0 -3 -6 }
{ 0  0  0 }}        c'est-à-dire la réduite de Gauss

2                   c'est-à-dire le rang de la matrice
```

Temps d'exécution : 11 secondes.

10
Résolution symbolique d'un système *RSYST*

Résolution symbolique d'un système

Ce programme est à enregistrer sous le nom : RSYST
(Dans le sous-répertoire APPLIA)

Son checksum est : #8761h (582,5 octets)

Listing

```
« 2 19 ERREUR SWAP → B « MTRN MAT→ → n
« B LIST→ DROP n 1 n 1 GET 1 + PUT DUP
'n' STO →MAT MTRN GAUSS n 1 GET 0 0
```

```
→ A rg n J piv
« { } IF rg 1 ≥ THEN 1 rg FOR I DO J 1 +
'J' STO UNTIL A I GET J GET ABS 1E-8 >
END IF J n == THEN DROP "Impossible"
DOERR END "'X" J →STR + "'" + STR→ A I
GET J GET 'piv' STO A I GET n GET piv
QDIV IF J 1 + n < THEN J 1 + n 1 - FOR K
A I GET K GET QNEG piv QDIV "'X" K →STR
+ "'" + STR→ * + NEXT END = + NEXT END »
» » »
```

Lancement du programme :

Saisir le vecteur solution comme une matrice àune ligne puis la matrice représentant le système puis taper :

```
RSYST [ENTER].
```

Exemple d'utilisation :

Résolvons le système :

$$\begin{cases} x + 2y + 3z = 6 \\ 4x + 5y + 6z = 15 \\ 7x + 8y + 9z = 24. \end{cases}$$

```
{ 6 15 24 } [ENTER]

{{ 1 2 3 }
{ 4 5 6 }
{ 7 8 9 }} [ENTER]

RSYST [ENTER]
```

renvoie :

```
{ 'X1=6-2*X2-3*X3'  'X2=3-2*X3' }
```

Temps d'exécution : 19,7 secondes.

11
Equation des espaces propres *ESPPE*

Equation des espaces propres

Ce programme est à enregistrer sous le nom : ESPPE.
(Dans le sous-répertoire APPLIA)

Son checksum est : #BD39h (138,5 octets)

Listing

```
« 2 18 ERREUR SWAP DUP SIZE ROT → n L «
n IDN A→M L MCONST MMOINS { } 1 n START
0 + NEXT SWAP RSYST »
```

Lancement du programme :

Saisir la matrice représentant l'application et une valeur propre lui correspondant puis taper

```
ESPPE [ENTER]
```

Exemple d'utilisation :

Recherchons l'équation de l'espace propre de la matrice

```
{{ 1 2 3 }
{ 4 5 6 }
{ 7 8 9 }}
```

associé à la valeur propre 0

```
{{ 1 2 3 }
{ 4 5 6 }
{ 7 8 9 }}
0
ESPPE [ENTER]
```

renvoie

```
{  'X1=-2*X2-3*X3'  'X2=-2*X3' }
```

qui est l'équation de l'espace propre.

C'est un espace de dimension 1.

Temps d'exécution : 21,8 secondes.

10
Géométrie et Trigonométrie

Ce chapitre contient les programmes suivants :

Dans la répertoire GEOMETRIE :

- **ENVELOPPES** Traceur d'enveloppes de droites
- **DEVELOPPEE** Traceur de développée
- **CONICS** Décomposition des coniques

Dans le répertoire OUTILS :

- **TRIGO** Linéarise une expression trigonométrique

1
Enveloppe de droites *ENVELOPPES*

Ce programme trace une enveloppe de droites

α
Programme principal

Ce programme est à enregistrer sous le nom : ENVELOPPES.

Son checksum est : #5E6Eh (598,5 octets)

Listing

```
« ERASE DRAX { #0 #0 } PVIEW tmin 'T'
STO DO '(-
DB(T)*C(T)+B(T)*DC(T))/(A(T)*DB(T)-
B(T)*DA(T))' →NUM
'(C(T)*DA(T)+DC(T)*A(T))/(DA(T)*B(T)-
DB(T)*A(T))' →NUM R→C IFERR 'OLD' RCL
THEN DROP DUP DUP 'OLD' STO ELSE SWAP
DUP 'OLD' STO SWAP END LINE T h + 'T'
STO UNTIL T tmax > END  'OLD'  PURGE 7
FREEZE
»
```

Lancement du programme :

Après avoir enregistré les données des programmes qui suivent, taper :

```
ENVELOPPEES [ENTER]
```

Remarque :

Ce programme trace l'enveloppe des droites

```
a(t)*x + b(t)*y + c(t) = 0
```

Il faut rentrer les fonctions a,b,c et da,db,dc (leurs dérivées respectives), tmin et tmax représentent la variation de la variable, PPAR les dimensions de l'écran et h le pas de variation.

β
Annexes

A
Coefficient de x dans l'équation

Ce programme est à enregistrer sous le nom : A

Listing

```
« → T  'COS(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

B
Coefficient de y dans l'équation

Ce programme est à enregistrer sous le nom : B.

Listing

```
« → T  'SIN(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

C
Coefficient constant dans l'équation

Ce programme est à enregistrer sous le nom : C.

Listing

```
« → T  '-COS(2*T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DA
Dérivée de A

Ce programme est à enregistrer sous le nom : DA.

Listing

```
« → T  '-SIN(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DB
Dérivée de B

Ce programme est à enrtegistrer sous le nom : DB.

Listing

```
« → T  'COS(T)'  »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DC
Dérivée de C

Ce programme est à enregistrer sous le nom : DC

Listing

```
« → T  '2*SIN(2*T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

---

tmin
valeur de t de départ

Ce programme est à enregistrer sous le nom : tmin

Listing

```
0
```

Remarque :
Ceci n'est qu'un exemple.

tmax
valeur de t d'arrivée

Ce programme est à enregistrer sous le nom : tmax

Listing

```
6.5
```

Remarque : ceci n'est qu'un exemple.

PPAR
Paramètres d'écran

Ce programme est à enregistrer sous le nom : PPAR.

Listing

```
{ (-2,-2) (2,2) X 0 (0,0) FUNCTION Y }
```

Remarque :
Ceci n'est qu'un exemple.

h
pas de variation de t

Ce programme est à enregistrer sous le nom h.

Listing

```
0.1
```

Remarque :

Ceci n'est qu'un exemple.

Conclusion : on a volontairement non-automatisé la dérivation des fonctions A et B, laissant le faire manuellement par l'utilisateur sur son calculateur ou de tête.

2
Développée d'une courbe paramétrée *DEVELOPPEE*

Développée d'une courbe paramétrée

α
Programme principal

Ce programme est à enregistrer sous le nom : DEVELOPPEE.

Son checksum est : #B765h (626 octets)

Listing

```
« ERASE DRAX { #0 #0 } PVIEW tmin 'T'
STO DO
'X(T)-
DY(T)*(DX(T)^2+DY(T)^2)/(DX(T)*DDY(T)-
DY(T)*DDX(T))'
→NUM
'Y(T)+DX(T)*(DX(T)^2+DY(T)^2)/(DX(T)*DDY
(T)-DY(T)*DDX(T))'
→NUM R→C IFERR 'OLD' RCL THEN DROP DUP
DUP 'OLD' STO ELSE SWAP DUP 'OLD' STO
SWAP END LINE T h + 'T' STO UNTIL T tmax
> END 'OLD' PURGE 7 FREEZE »
```

Lancement du programme :
Après avoir complétés les programmes de l'annexe (voir le paragraphe β qui suit), taper :

```
DEVELOPPEE [ENTER].
```

Exemple d'utilisation :
Les données ci-dessous permettent d'obtenir une astroïde.

Remarque :
Ce programme permet d'obtenir une développée de la droite :

```
x = X(t)
y = Y(t)
```

β
Annexes

Ces dix petits programmes sont indispensables pour utiliser le programme DEVELOPPEE.

X
Fonction de x dans l'équation de la droite

Ce programme est à enregistrer sous le nom : X.

Listing

```
« → T  '3*COS(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

Y
Fonction de y dans l'équation de la droite

Ce programme est à enregistrer sous le nom : Y

Listing

```
« → T  '2*SIN(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DX
Dérivée première de X

Ce programme est à enregistrer sous le nom : DX.

Listing

```
« → T  '-3*SIN(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DY
Dérivée première de Y

Ce programme est à enregistrer sous le nom : DY.

Listing

```
« → T  '2*COS(T)' »
```

Remarque :
Ceci n'est qu'un exemple.

Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DDX
Dérivée seconde de X

Ce programme est à enregistrer sous le nom : DDX.

Listing

```
« → T  '-3*COS(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

DDY
Dérivée seconde de Y

Ce programme est à enregistrer sous le nom : DDY.

Listing

```
« → T  '-2*SIN(T)' »
```

Remarque :
Ceci n'est qu'un exemple.
Il faut présenter cette variable sous la forme d'une fonction comme ci-dessus.

---

tmin
valeur t de départ

Ce programme est à enregistrer sous le nom : tmin.

Listing

```
0
```

Remarque :
Ceci n'est qu'un exemple.

---

tmax
Valeur t d'arrivée

Ce programme est à enregistrer sous le nom : tmax.

Listing

```
6.29
```

Remarque :
Ceci n'est qu'un exemple.

PPAR
Paramètres d'écran

Ce programme est à enregistrer sous le nom : PPAR.

Listing

```
{ (-2,-3) (2,3) X 0 (0,0) FUNCTION Y }
```

Remarque :
Ceci n'est qu'un exemple.

---

h
pas de variation de t

Ce programme est à enregistrer sous le nom : h.

Listing

```
0.1
```

Remarque :
Ceci n'est qu'un exemple.

3
Décomposition des coniques *Conics*

Ce programme est décomposé en plusieurs sous-programmes. Le principal a pour nom CONICS. Les autres programmes de cette section sont néanmoins indispensables pour faire fonctionner le programme CONICS.

α
Conics

Ce programme est à enregistrer sous : CONICS

Son checksum est : #2848h (276,5 octets)

Listing

```
« -22 SF CLLCD "  ETUDES DE CONIQUES" 1
DISP 'EQ' STO CENTRE TEST CST IF 4 A * B
```

```
* C C * - 0 ≠ THEN ANGLE SITU PARAM ELSE
PARAB END { BS AS FF F E B D C A } PURGE
DO UNTIL KEY END DROP CLLCD CONIC ERASE
AUTO DRAX DRAW -22 CF »
```

Sous programmes : les sous-programmes de CONICS sont CLEAN, TEST, CENTRE, ANGLE, SITU, PARAM, PARAB, CST, DEGENERE, DEGY et DEGE2.

β
Clean

Nettoie le répertoire des variables crées par le programme

Ce programme est à enregistrer sous : CLEAN

Son checksum est : #9A52h (140 octets)

Listing

```
« { FF X0 Y0 F E B D C A c b a ALFA X0
Y0 PPAR P SY J I AS BS } PURGE »
```

Remarque : ce programme est un sous-programme de CONICS.

γ
Test

Ce programme est à enregistrer sous : TEST

Son checksum est : #B2A1h (162 octets)

Listing

```
« { A B C D E F } →L « 1 L SIZE FOR I L
I GET EVAL ABS IFERR DUP →NUM THEN DROP
ELSE IF 1 < THEN 0 L I GET STO END END
DROP NEXT » »
```

Remarque : ce programme est un sous-programme de CONICS.

γ
Centre

Ce programme calcule les variables de l'équation à résoudre.

Ce programme est à enregistrer sous : CENTRE

Son checksum est : #2116h (825 octets)

Listing

```
« 'X' PURGE 'Y' PURGE EQ 'X' ∂ DUP DUP
'X' ∂ 2 / 'A' STO 'Y' ∂ 'C' STO 0 'X' STO
0 'Y' STO EVAL 'D' STO 'X' PURGE 'Y'
PURGE EQ 'Y' ∂ DUP 'Y' ∂ 2 / 'B' STO 0
'X' STO 0 'Y' STO EVAL 'E' STO EQ EVAL
'F' STO 'X' PURGE 'Y' PURGE IF A 0 ≠ THEN
IF B A * C 4 / - 0 > B C 4 / A /- F * E
SQ 4 / - 0 > AND THEN "Pas de solution"
DOERR END END IF A 0 == B 0 == C 0 ==
AND AND THEN DEGENERE 0 DOERR END DEGE2
'C*C-4*A*B' EVAL IF 0 == THEN "PARABOLE"
1 DISP ELSE '(-2*B*D+C*E)/(4*A*B-C^2)'
EVAL '(2*E*A-C*D)/(C^2-4*A*B)' EVAL DUP2
'Y0' STO 'X0' STO 'i' * + →NUM 2 RND
→STR "Centre=" SWAP + 2 DISP END »
```

δ
Angle

Ce programme est à enregistrer sous : ANGLE

Son checksum est : #484Bh (152 octets)

Listing

```
« DEG A B - IF 0 ≠ THEN A B - INV C *
ELSE MAXR →NUM C SIGN * END ATAN 2 /
EVAL DUP 'ALFA' STO →STR "α=" SWAP + "°"
+ 3 DISP RAD »
```

Remarque : ce programme est un sous-programme de CONICS.

ε
Situ

Ce programme est à enregistrer sous : SITU

Son checksum est : #C0C2h (476,5 octets)

Listing

```
« DEG
'A*COS(ALFA)^2+B*SIN(ALFA)^2+C*COS(ALFA)
*SIN(ALFA)' EVAL 0 FF - / DUP SIGN 'AS'
STO INV ABS √ 'a' STO
'A*SIN(ALFA)^2+B*COS(ALFA)^2-
C*SIN(ALFA)*COS(ALFA)' EVAL 0 FF - / DUP
SIGN 'BS' STO INV ABS √ 'b' STO RAD IF a
b * AS * BS * 0 < THEN "Hyperbole" 1
DISP ELSE "Ellipse" 1 DISP IF a b < THEN
90 ALFA + 'ALFA' STO "α=" ALFA →STR + 3
```

```
DISP SITU END END  »
```

Remarque : ce programme est un sous-programme de CONICS.

ζ
Param

Ce programme est à enregistrer sous : PARAM

Son checksum est : #72BEh (272 octets)

Listing

```
« IF a b * AS * BS * 0 < THEN a SQ b SQ
+ √ 'c' STO ELSE a SQ b SQ - √ 'c' STO
END "a=" a →STR + 4 DISP "b=" b →STR +
5 DISP "c=" c →STR + 6 DISP "e=" c a /
→STR + 7 DISP IF c a / 0 == THEN
"Cercle" 1 DISP END »
```

Remarque : ce programme est un sous-programme de CONICS.

η
Parab

Ce programme est à enregistrer sous : PARAB

Son checksum est : #C5CFh (666,5 octets)

Listing

```
« "Nouvelle base=" 2 DISP B √ A ABS B ABS
+ √ INV * A √ A ABS B ABS + √ INV * R→C
DUP 'I' STO →STR "I=" SWAP + 3 DISP I RE
'i' * I IM - →NUM 'J' STO "J=" J →STR +
4 DISP E I RE * D I IM * - A I IM SQ * B
```

```
I RE SQ * + C I RE * I IM - INV * 2 NEG
INV * 'SY' STO D I RE * E I IM * + A I
IM SQ * B I RE SQ * + C I IM * I RE * -
INV * 2 NEG INV * 'P' STO F A I IM SQ *
B I RE SQ * + C I RE * I IM * - INV * SY
SQ - 2 INV * P INV * 'SX' STO
"dans (S,I,J) avec" 5 DISP "SX=" SX 3
RND →STR + " " + "SY=" SY 3 RND →STR +
+ 6 DISP "P=" P →STR + 7 DISP »
```

Remarque : ce programme est un sous-programme de CONICS.

ι

CST

Ce programme est à enregistrer sous : CST

Son checksum est : #D911h (135,5 octets)

Listing

```
« 'A*X0^2+B*Y0^2+C*X0*Y0+D*X0+E*Y0+F'
EVAL 'FF' STO »
```

Remarque : ce programme est un sous-programme de CONICS.

κ

Degenere

Ce programme est à enregistrer sous : DEGENERE

Son checksum est : #F691h (420 octets)

Listing

```
« IF D 0 ≠ E 0 ≠ OR THEN "Droite" 1 DISP
"Pente = " 2 DISP D NEG E INV * DUP IF
MAXR == THEN "+∞" SWAP DROP ELSE IF DUP
MAXR NEG == THEN "-∞" SWAP DROP END END
3 DISP DO UNTIL KEY END DROP ERASE DRAX
IF E 0 ≠ THEN CONIC ELSE 'X' PURGE 'X=-
F/D" EVAL 'EQ' STO FUNCTION END DRAW
ELSE IF F 0 == THEN "Tout le plan est" 2
DISP "solution" 3 DISP DO UNTIL KEY END
DROP ELSE "Pas de solution" 2 DISP DO
UNTIL KEY END DROP END END »
```

Remarque : ce programme est un sous-programme de CONICS.

λ Degx

Ce programme est à enregistrer sous : DEGX

Son checksum est : #D18Eh (243 octets)

Listing

```
« D SQ 4 A * F * - → R « IF R 0 < THEN
"Pas de solution" DOERR ELSE
"Deux Droites" 1 DISP "X = " D NEG R √ +
A / 2 / →STR + 2 DISP "X = " D NEG R √ -
A / 2 / →STR + 3 DISP DO UNTIL KEY END
DROP 0 DOERR END » »
```

Remarque : ce programme est un sous-programme de CONICS.

μ
Degy

Ce programme est à enregistrer sous : DEGY

Son checksum est : #8C35h (243 octets)

Listing

```
« E SQ 4 B * F * - → R « IF R 0 < THEN
"Pas de solution" DOERR ELSE
"Deux Droites" 1 DISP "X = " E NEG R √ +
B / 2 / →STR + 2 DISP "X = " E NEG R √ -
B / 2 / →STR + 3 DISP DO UNTIL KEY END
DROP 0 DOERR END » »
```

Remarque : ce programme est un sous-programme de CONICS.

ν
Dege2

Ce programme est à enregistrer sous : DEGE2

Son checksum est : #6403h (810,5 octets)

Listing

```
« IF C D * A E * 2 * - SQ D SQ A F * 4 *
- C SQ A B * 4 * - * == THEN
"Deux Droites" 1 DISP IF A 0 == C 0 == D
0 == AND AND THEN DEGY END IF B 0 == C 0
== E 0 == AND AND THEN DEGX END IF A 0
== B D SQ * E C * D * - F C SQ * + 0 ==
AND THEN C →STR " Y + " + D →STR + " =
```

```
0" + 2 DISP C D * →STR " X + " + B D *
→STR + " Y + " + F C * →STR + " =0" + 4
DISP DO UNTIL KEY END DROP 0 DOERR END
IF A 0 ≠ THEN C SQ 4 A * B * - → DD « IF
DD 0 < THEN "Pas de Solution" DOERR END
"(" 2 A * →STR + ") X + " + DUP 2 DISP 5
DISP "(" C DD √ - →STR + ") Y + " + 3
DISP "(" C DD √ + →STR + ") Y + " + 6
DISP D D SQ A F * 4 * - √ DUP2 + →STR "
= 0" + 4 DISP - →STR " = 0" + 7 DISP DO
UNTIL KEY END DROP 0 DOERR » END END »
```

Remarque : ce programme est un sous-programme de CONICS.

Exemple d'utilisation du programme CONICS :

Saisissez `'X^2+Y^2-4'`
puis effectuez

```
CONICS
```

Le programme répond :

```
Cercle
Centre=0
α=0°
a=2
b=2
c=0
e=0
```

puis trace la courbe.

Remarque :
a,b,c et e représentent les paramètres habituels des coniques. L'utilisateur ne se trouvera pas désorienté.

Exemple d'utilisation du programme CONICS :

Saisissez 'X^2-Y^2-4'
puis effectuez

```
CONICS
```

Le programme répond :

```
Hyperbole
Centre=0
α=0°
a=2
b=2
c=2.82842712475
e=1.41421356238
```

puis trace la courbe.

Exemples d'utilisation du programme CONICS :

Saisissez '(X+Y+1)*(X-Y+1)'
puis effectuez

```
CONICS
```

Le programme répond :

```
Deux Droites
(2) X +
(-2) Y +
2 = 0
(2) X +
(2) Y +
2 = 0
```

Remarque : Le traitement des cas dégénérés (droites, pas de solution ...) prend beaucoup de place, mais le résultat est appréciable. Car après tout, il n'est pas toujours simple de reconnaître une équation de conique dégénérée !

4
Trigonométrie
Trigo

Le programme ci-dessous est à enregistrer dans le répertoire OUTILS.

Linéarisation

Ce programme est à enregistrer sous : TRIGO.

Son checksum est : #3968h (1305 octets)

Listing

```
« CLLCD "      LINEARISATION

1. COS(n*x)
2. SIN(n*x)
3. COS^n(x)
4. SIN^n(x)" 1 DISP DO UNTIL KEY END IP
0 → k n
« IF k 82 == THEN CLLCD "COS(n*x)
n= ?"  { "" } INPUT STR→ 'n'
STO n 'X' * COS 0 0 n 2 / FOR K 0 K FOR
J n 2 K * COMB K J COMB * -1 J K + ^ * n
2 K * - J 2 * + 'COS(X)' SWAP ^ * + NEXT
NEXT COLCT = END IF k 83 == THEN CLLCD
"SIN(n*x)
n= ?" { "" } INPUT STR→ 'n' STO
n 'X' * SIN 0 0 n 1 - 2 / FOR K 0 K FOR
J n 2 K * 1 + COMB K J COMB * -1 J K + ^
* 'COS(X)' n 2 K * - 1 - 2 J * + ^ * +
NEXT NEXT COLCT 'SIN(X)' * = END IF k 84
== THEN CLLCD "COS^n(x)
```

```
n= ?"  { "" } INPUT STR→ 'n' STO
'COS(X)' n ^ END IF k 72 == THEN CLLCD
"SIN^n(x)
n= ?"  { "" } INPUT STR→ 'n' STO
'SIN(X)' n ^ END IF k 84 == k 72 == OR n
2 MOD NOT AND THEN 0 0 n 2 / FOR K IF K
n 2 / ≠ THEN n K COMB ELSE n n 2 / COMB 2
/
END n 2 K * - 'X' * IF k 84 == THEN COS
ELSE COS -1 K 1 + ^ * END * + NEXT COLCT
2 n 1 - ^ / = END IF k 84 == k 72 == OR
n 2 MOD AND THEN 0 0 n 2 / FOR K n K
COMB n 2 K * - 'X' * IF k 84 == THEN COS
ELSE
SIN -1 K ^ * END * + NEXT COLCT 2 n 1 -
^ / = END » »
```

Lancement du programme :

Taper :

```
TRIGO [ENTER].
```

Exemples d'utilisation :

Le calculteur répond que

```
SIN(8*X)=(80*COS(X)^3-192*COS(X)^5+128*COS(X)^7-
8*COS(X))*SIN(X)
```

ou encore

```
COS(X)^7=(21*COS(3*X)+7*COS(5*X)+COS(7*X)+35*COS
(X))/64
```

Temps d'exécution : moins de 10 secondes en général.

11
Physique

Ce chapitre utilise les possibilités graphiques et calculatoires de votre HP48 en les appliquant principalement à l'électricité.

Les programmes seront présentés dans l'ordre suivant :

• **BODE**	Trace et étudie des fonctions de transfert
• **ETUDE**	Sous programmes de BODE
• **PRECIS**	
• **SRND**	
• **ATTENDS**	
• **G0**	
• **SEE**	
• **GIBBS**	Calcule l'enthalpie

1
Fonctions de transfert *BODE*

Ce programme trace et étudie des fonctions de transfert.

Ce programme est à enregistrer sous le nom : BODE.

Son checksum est : #0B9Eh (419,5 octets)

Listing

```
« "Entrez H en fonction
de ω" { "'" ALG V } INPUT STR→ { j 'i' }
| 'H' STO H ABS 'G' STO H ARG 'PHI' STO
G { ω 'ALOG(ω)' } | LOG 20 * 'EQ' STO -
20 31 XRNG -400 40 YRNG 'ω' INDEP ERASE
DRAX DRAW ATTENDS ERASE PHI { ω
'ALOG(ω)' } | 'EQ' STO -3.15 3.15 YRNG
LABEL DRAX DRAW ATTENDS G LOG 20 * 'G'
STO ETUDE PRECIS »
```

Exemple d'utilisation :

Si l'on rentre comme fonction :

```
'1/(j*10*ω-j*1E5/ω)'
```

le programme exécute les éléme nts suiv ants :

```
* Trace G
* Trace Φ
* Annonce les éléments suivants :
     - Filtre passe bande
     - Pente des asymtotes
          en ω→ 0 : 20 dB/decade
          en ω→ ∞ : -20 dB/decade
     - G0 : -63,9 dB
     - Fréquence de coupure
     ω1: 38,4 Hz et ω2: 261 Hz
     - Facteur de qualité : .219
     - Largeur de bande : 222 Hz
```

Remarque : Lors des tracés, vous pouvez les interrompre en pressant la touche [ON]. Le petit symbole :-) s'affiche en bas de l'écran.

2
Etudie les fonctions de transfert *ETUDE*

Ce programme étudie des fonctions de transfert.

Ce programme est à enregistrer sous le nom : ETUDE.

Son checksum est : #A2AFh (1899,5 octets)

Listing

```
« TEXT CLLCD " ETUDE DES DIAGRAMMES "
1 DISP "   *** PATIENTEZ ***" 5 DISP
" *** ETUDE DES DIAGRAMMES ***" G 'ω'
PURGE 'ω' ∂ RE 'DER' STO -20 ALOG 'ω'
STO DER →NUM 10 LN * -20 ALOG * RE 30
ALOG 'ω' STO DER →NUM 10 LN * 30 ALOG *
RE 2 →LIST 'L' STO MAXR →NUM 'ω0' STO
IF 'L(1)' EVAL ABS .0001 < THEN 0 'L(1)'
STO END IF 'L(2)' EVAL ABS .0001 < THEN
0 'L(2)' STO END 'L(1)' EVAL SRND 'L(2)'
EVAL SRND 2 →LIST 'L' STO "-FILTRE " IF
'L(1)' EVAL 0 == 'L(2)' EVAL 0 ≠ AND THEN
"PASSE BAS-" + -20 ALOG 'ω' STO G →NUM
'G0' STO END IF 'L(1)' EVAL 0 ≠ 'L(2)'
EVAL 0 == AND THEN "PASSE HAUT-" + 31
ALOG 'ω' STO G →NUM 'G0' STO END IF
'L(1)' EVAL 0 > 'L(2)' EVAL 0 < AND THEN
"PASSE BANDE-" + DER 'ω' 1 ROOT 'ω0' STO
ω0 EVAL 'ω' STO G →NUM 'G0' STO END IF
'L(1)' EVAL 0 < 'L(2)' EVAL 0 > AND THEN
"COUPE BANDE-" + DER 'ω' 1 ROOT 'ω0' STO
ω0 EVAL 'ω' STO G →NUM 'G0' STO END DUP
```

```
7 DISP "-PENTES DES ASYMPTOTES"
"   EN ω→0 : " 'L(1)' EVAL →STR +
" db/decade" + "   EN ω→∞ : " 'L(2)'
EVAL →STR + " dB/decade" + "-G0 : " G0
SRND →STR + " dB" +
"-FREQUENCES DE COUPURE" "ω1: " G RE { ω
'ALOG(ω)' } | GO 2 √ LOG 20 * - = DUP
'ω' 0 ROOT ALOG 'ω1' STO 'ω' 30 ROOT
ALOG 'ω2' STO ω1 SRND →STR +
"Hz et ω2: " ω2 SRND →STR + " Hz" + +
"-Facteur de qualite: " IF ω2 ω1 - ABS
.000001 > THEN ω0 ω2 ω1 - / SRND →STR
ELSE "???" END + "-Largeur de la bande:
" IF ω2 ω1 - ABS .000001 > THEN ω2 ω1 -
SRND →STR + ELSE "???" + END " Hz" + 10
→LIST SEE 'ω' PURGE  »
```

Remarque : ce programme est utilisé par BODE.

3
Zoom sur la courbe
PRECIS

Sous programme de BODE.

Ce programme est à enregistrer sous le nom : PRECIS.

Son checksum est : #CB35h (624 octets)

Listing

```
« CLLCD 2 FIX "GROS PLAN SUR LA ZONE
```

```
DE COUPURE ..." 2 DISP 'PPAR' PURGE
ERASE G { ω 'ALOG(ω)' } | 'EQ' STO 'ω'
INDEP ω1 LOG 1 - ω2 LOG 1 + XRNG ω1 LOG
1 - ALOG 'ω' STO G →NUM RE DUP 'm' STO
'M' STO 1 131 FOR I 'ALOG(ALOG(ω1)-
1)+I*(ALOG(LOG(ω2)+1)/131)' →NUM 'ω' STO
G →NUM RE IF DUP m < THEN DUP 'm' STO
END IF DUP M > THEN DUP 'M' STO END DROP
NEXT m M { m M } PURGE YRNG { #0 #O }
PVIEW DRAX LABEL DRAW ATTENDS STD { PPAR
ω } PURGE »
```

Remarque : ce programme est un sous-programme de BODE

4
Arrondis
SRND

Sous programme de BODE.

Ce programme est à enregistrer sous le nom : SRND.

Son checksum est : #FF5Eh (56,5 octets)

Listing

```
« 2 SCI →STR STD IFERR STR→ THEN DROP
9.99E499 END »
```

Remarque : ce programme est un sous-programme de BODE

5
Préviens de la fin du calcul *ATTENDS*

Sous programme de BODE.

Ce programme est à enregistrer sous le nom : ATTENDS.

Son checksum est : #145Eh (169 octets)

Listing

```
« PICT { #0 #3Bh } GROB 9 5
04001800C90018000400 REPL ":-)" 7 DISP
DO UNTIL KEY END DROP "" 7 DISP PICT
{ #0 #3Bh } GROB 9 5
00000000000000000000   REPL »
```

Remarque : ce programme est un sous-programme de BODE

6
G0 *G0*

Sous programme de BODE.

Ce programme est à enregistrer sous le nom : G0.

Son checksum est : #0CF8h (17 octets)

Listing

```
0
```

Remarque : ce programme est un sous-programme de BODE

7
Affichage
SEE

Sous programme de BODE.

Ce programme est à enregistrer sous le nom : SEE.

Son checksum est : #AED5h (558 octets)

Listing

```
« → 1 « ERASE { #0 #0 } PVIEW { #0 #0 }
{ #82h •0 } LINE { #82h #0 } { #82h #3Fh
} LINE { #82h #3Fh } { #0 #3Fh } LINE {
#0 #3Fh } { #0 #0 } LINE 1 10 FOR I PICT
{ #2h } #3h 6 I 1 - * + IF I 1 == THEN 1
- END + 1 I GET 1 →GROB REPL NEXT 1 7
FOR I { #1h } #1h I * + { #81h } #1h I *
+ TLINE NEXT DO UNTIL KEY END DROP TEXT
» »
```

Remarque : ce programme est un sous-programme de BODE

8
Calcul de l'enthalpie *GIBBS*

Ce programme calcule ΔG.

Ce programme est à enregistrer sous le nom : GIBBS.

Son checksum est : #8198h (335,5 octets)

Listing

```
« { { "ΔH(T)" « 'ΔH(T)' SWAP = DEFINE »
} { "ΔG0"  « 'ΔG0' STO » } { "T0" « 'T0'
STO » } "" { "CALC"  « T0 T '-ΔH(T)'
EVAL T 2 ^ / EXPAN EXPAN EXPAN COLCT
COLCT T ∫ EVAL ΔG0 T0 / + T * EXPAN EXPAN
COLCT 'ΔG' SWAP =  » } { "EXIT"  « 2
TMENU » } } TMENU  »
```

Remarque : ce programme calcule approximativement mais de manière formelle ΔG(T) connaissant ΔH(T) et ΔG0 et T0, avec ΔG0=ΔG(T0). La formule utilisée est dite de "Gibbs Helmoltz".

Utilisation de programme :
Executez **GIBBS** sans paramètres. Entrez alors les valeurs des 3 expressions se trouvant dans le menu. Par exemple, rentrez **'700000-25000*T'** puis appuyez sur la touche **[DH(T)]** pour signaler que c'est la fonction que vous venez de rentrer qui correspond à DH(T).

12
Chimie

Les programmes de ce chapitre seront placés dans le répertoire CHIMIE. Ils seront présentés dans l'ordre suivant :

- **PH** calcule le pH d'une solution donnée.
- **AF** ajoute un acide fort à la solution
- **BF** ajoute une base fort à la solution
- **A1** ajoute un acide à une acidité à la solution
- **B1** ajoute une base faible à la solution
- **A2** ajoute un acide à deux acidités à la solution
- **A3** ajoute un acide à trois acidités à la solution
- **AM** ajoute un acide aminé à la solution
- **CALC** lance le calcul du pH

- **MASS** calcule la masse moléculaire d'une molécule
- **MDATA** données sur les masses des atomes

1
Saisie d'une solution
pH

Ce programme lance la saisie d'une solution.

Ce programme est à enregistrer sous le nom : PH.

Son checksum est : #B0C8h (111,5 octets)

Listing

```
« 'H' PURGE { AF BF A1 B1 A2 A3 AM CALC
} TMENU 'H-1E14/H' »
```

Remarque :

Ce programme crée un menu temporaire qui permet d'appeler les programmes correspondant à chaque catégorie de produits que l'on peut mettre en solution. Ce programme peut être **étendu** : il suffit de créer un sous programme pour rajouter un produit, sur le modèle de ceux qui sont fournis ici. Il faut ensuite rajouter le nom de ce programme dans la liste du programme PH.

2
Acide fort
AF

Ce programme permet de mettre un acide fort en solution.

Ce programme est à enregistrer sous le nom : AF.

Son checksum est : #9C4Fh (70 octets)

Listing

```
« "AF: Quelle est la concentration" { ""
} INPUT STR→ - »
```

Exemple d'utilisation :
appuyer sur AF, puis répondre
`1E-2 [ENTER]`

3
Base forte
BF

Ce programme permet de mettre une base forte en solution.

Ce programme est à enregistrer sous le nom : BF.

Son checksum est : #3EABh (70 octets)

Listing

```
« "BF: Quelle est la concentration" { ""
} INPUT STR→ + »
```

Exemple d'utilisation :
appuyer sur BF, puis répondre
`1E-2 [ENTER]`

4
Acide faible à une acidité
A1

Ce programme permet de mettre un acide faible en solution.

Ce programme est à enregistrer sous le nom : A1.

Son checksum est : #5A2Dh (130 octets)

Listing

```
« "A1: Quelle est la concentration" { ""
} INPUT STR→ "A1: Quel est le ka" { "" }
INPUT STR→ 'H' SWAP / 1 + / - »
```

Exemple d'utilisation :

appuyer sur A1, puis répondre

```
1E-2 [ENTER]
1E-10 [ENTER]
```

Note :
C'est le Ka qu'il faut saisir et non le pKa.

5
Base faible
B1

Ce programme permet de mettre une base faible en solution.

Ce programme est à enregistrer sous le nom : B1.

Son checksum est : #B32Dh (127,5 octets)

Listing

```
« "B1: Quelle est la concentration" { ""
} INPUT STR→ "B1: Quel est le ka" { "" }
INPUT STR→ 'H' / 1 + / + »
```

Exemple d'utilisation :
appuyer sur B1, puis répondre

```
1E-2 [ENTER]
1E-9 [ENTER]
```

Note :
C'est le Ka qu'il faut saisir et non le pKa ou le pKb.

6
Diacide
A2

Ce programme permet de mettre un acide faible en solution.

Ce programme est à enregistrer sous le nom : A2.

Son checksum est : #EC07h (257,5 octets)

Listing

```
« "A2: Quelle est la concentration" { ""
} INPUT STR→ "A2: Quel est le 1e ka" {
"" } INPUT STR→ "A2: Quel est le 2e ka"
{ "" } INPUT STR→ DUP2 * 'H' SQ SWAP /
OVER 'H' SWAP / + 1 + 2 / 3 ROLLD 'H' /
1 + SWAP 'H' SWAP / + INV SWAP INV + * -
»
```

Exemple d'utilisation :
appuyer sur A2, puis répondre

```
0.02 [ENTER]
1E-5 [ENTER]
1E-8 [ENTER]
```

Note :
Ce sont les Ka qu'il faut saisir et non les pKa.

7
Triacide
A3

Ce programme permet de mettre un acide faible en solution.

Ce programme est à enregistrer sous le nom : A3.

Son checksum est : #8E48h (453,5 octets)

Listing

```
« "A3: Quelle est la concentration" { ""
} INPUT STR→ "A3: Quel est le 1e ka" {
"" } INPUT STR→ "A3: Quel est le 2e ka"
{ "" } INPUT STR→ "A3: Quel est le 3e
ka" { "" } INPUT STR→ 3 DUPN 3 DUPN DUP2
* 'H' SQ SWAP / OVER 'H' SWAP / + 1 + 4
ROLLD * * 'H' 3 ^ SWAP / + 3 / 4 ROLLD
'H' / 1 + OVER 'H' SWAP / + 3 ROLLD *
'H' SQ SWAP / + 2 / 5 ROLLD 5 ROLLD OVER
* 'H' SQ / 3 ROLLD 'H' / 1 + SWAP 'H'
SWAP / + + INV SWAP INV + SWAP INV + * -
»
```

Exemple d'utilisation :

appuyer sur A3, puis répondre

```
0.04 [ENTER]
1E-1 [ENTER]
1E-6 [ENTER]
1E-17 [ENTER]
```

Note :
Ce sont les Ka qu'il faut saisir et non les pKa.

Rappel : pour saisir une puissance de dix non entière, il ne faut pas faire `1E1.4` par exemple, mais `'10^(1.4)'`.

8
Acide aminé
AM

Ce programme permet de mettre un acide faible en solution.

Ce programme est à enregistrer sous le nom : AM.

Son checksum est : #9D33h (270 octets)

Listing

```
« "AM: Quelle est la concentration" { ""
} INPUT STR→ "AM: Quel est le 1e ka" {
"" } INPUT STR→ "AM: Quel est le 2e ka"
{ "" } INPUT STR→ DUP2 DUP2 * 'H' SQ
SWAP / SWAP DROP SWAP 'H' SWAP / + 1 +
NEG 3 ROLLD DUP 'H' / 1 + 3 ROLLD * 'H'
SQ / + INV SWAP INV + * + »
```

Exemple d'utilisation :
appuyer sur AM, puis répondre

```
0.05 [ENTER]
1E-5 [ENTER]
1E-12 [ENTER]
```

Note :
Ce sont les Ka qu'il faut saisir et non les pKa.

Rappel : pour saisir une puissance de dix non entière, il ne faut pas faire `1E1.4` par exemple, mais `'10^(1.4)'`.

9
Calcul du pH
CALC

Ce programme permet de calculer le pH de la solution.

Ce programme est à enregistrer sous le nom : CALC.

Son checksum est : #954Eh (346 octets)

Listing

```
« DUP 'EQ' STO 0 'F1' STO 14 'F2' STO 1
15 FOR I F1 F2 + 2 / NEG ALOG 'H' STO EQ
→NUM I 15 / 100 * IP →STR " %" + 1 DISP
IF 0 > F1 F2 + 2 / SWAP THEN 'F1' STO
ELSE 'F2' STO END NEXT F1 F2 + 2 / 100 *
IP 100 / "pH" →TAG { F1 F2 EQ H } PURGE
»
```

Exemples d'utilisation :

Saisir la solution suivante :
* acide fort concentration 0.0001
* base forte concentration 0.0252
* monoacide concentration 0.001 et Ka=1E-10
* monobase concentration 0.01 et Ka=1E-9
* diacide concentration 0.02 et Ka1=1E-5 et Ka2=1E-8
* triacide concentration 0.04 et Ka1=1E-1 et Ka2=1E-6 et Ka3=1E-7

lancer CALC, et le programme répond :

```
pH : 2.44
```

Saisir la solution suivante :
* acide fort concentration 0.0001
* base forte concentration 0.0252
* monoacide concentration 0.001 et Ka=1E-10
* monobase concentration 0.01 et Ka=1E-9
* diacide concentration 0.02 et Ka1=1E-5 et Ka2=1E-8

* triacide concentration 0.04 et Ka1=1E-1 et Ka2=1E-6 et Ka3=1E-7
* acide aminé concentration 0,05 et Ka1=1E-5 et Ka2=1E-12
lancer CALC, et le programme répond :

```
pH : 6,10
```

10
Calcul de la masse moléculaire *MASS*

Ce programme permet de calculer la masse molaire d'une molécule.

Ce programme est à enregistrer sous le nom : MASS.

Son checksum est : #D1AAh (386,5 octets)

Listing

```
« → c « 0 WHILE c "" ≠ REPEAT c 1 1 SUB
c 2 999 SUB 'c' STO IF c NUM DUP 65 <
SWAP 0 ≠ AND THEN "" WHILE c 1 1 SUB NUM
DUP 65 < SWAP 0 ≠ AND REPEAT c 1 1 SUB +
c 2 999 SUB 'c' STO END STR→ ELSE 1 END
SWAP IF c 1 1 SUB NUM 97 ≥ THEN c 1 1 SUB
+ c 2 999 SUB 'c' STO END STR→ MDATA
SWAP POS 1 + MDATA SWAP GET * + END » »
```

Exemples d'utilisation :

Saisir la formule suivante :

```
"CH3COOH"
MASS [ENTER]
```

renvoie : `60`
Saisir la formule suivante :

```
"C6H13OH"
MASS [ENTER]
```

renvoie : `102`

Remarque :
Ceci signifie que la molécule $C_6H_{13}OH$ a pour masse 102 g/mol.

11
Masses atomiques
MDATA

Ce fichier contient les masses molaires atomiques.

Ce programme est à enregistrer sous le nom : MDATA.

Son checksum est : #F7E2h (66,5 octets)

Listing

```
{ C 12 H 1 O 16 N 14 }
```

Remarque :
Vous pouvez rajouter les masses molaires atomiques des atomes que vous désirez dans cette liste.

Remarque :
Vous pouvez aussi ajouter des molécules dont le nom contient plusieurs lettres : dans ce cas, la deuxième lettre devra être minuscule :

```
{ C 12 H 1 O 16 N 14 Cl 35.5 }
```

Exemple d'utilisation :

Saisir la formule suivante :

```
"C2H5Cl"
MASS [ENTER]
```

renvoie : `64.5`

13
Equations différentielles

Dans ce chapitre, nous allons résoudre graphiquement les équations différentielles du 1er et du 2ème ordre scalaires. Pour ces programmes, tout est automatisé. Il vous faut juste saisir dans la variable ITER, la précision du tracé. Les programmes sont :

- **DIFF1** Equation du 1er ordre scalaire
- **DIFF2** Equation du 2eme ordre scalaire
- **TRACER** Trace une suite de points
- **ITER** Précision des tracés

Ils utilisent TRACER et ITER comme des sous-programmes.

1
Tracé d'un ensemble de points *TRACER*

Ce programme trace un ensemble de points.

Ce programme est à enregistrer sous le nom : TRACER.

Son checksum est : #85FAh (94 octets)

Listing

```
« ERASE { #0 #0 } PVIEW DRAX LIST→ 1 - 1
SWAP START OVER
SWAP LINE NEXT DROP { } PVIEW »
```

Remarque :

Ce programme est un sous-programme de DIFF1 et de DIFF2.

2
Précision des tracés
ITER

Précision des tracés

Ce programme est à enregistrer sous le nom : ITER.

Listing

```
100
```

Remarque :
Plus le nombre est grand, plus le tracé sera précis (et lent).

3
Premier ordre
DIFF1

Equation différentielle du premier ordre

Ce programme est à enregistrer sous le nom : DIFF1.

Son checksum est : #4E26h (700 octets)

Listing

```
« IF DEPTH 4 < THEN
"Entrez y' fn de X et Y
xmin , xmax et y(xmin)" DOERR END 0 0 {
} 0 → f xmin xmax y0 ymin ymax sol h
« CLLCD "RESOLUTION DE dY/dX=" 1 DISP f
2 DISP "x0= " xmin →STR + ", y0= " + y0
→STR + 4 DISP y0 'ymin' STO y0 'ymax'
STO xmin y0 R→C sol + 'sol' STO y0 'Y'
STO xmax xmin - ITER / 'h' STO 1 ITER
FOR I xmin I xmax xmin - * ITER / + 'X'
STO f →NUM h * Y + 'Y' STO X Y R→C sol
+ 'sol' STO IF Y ymin < THEN Y 'ymin'
STO END IF Y ymax > THEN Y 'ymax' STO
END NEXT { X Y } PURGE xmin ymin R→C
PMIN xmax ymax R→C PMAX sol TRACER »
»
```

Lancement du programme :
Après avoir réglé ITER, rentrer l'expression y' fonction de X et de Y, les valeurs minimum et maximum de X et la valeur de Y correspondant à la valeur minimum de X . Taper ensuite

```
DIFF1 [ENTER]
```

Remarque :
Le programme DIFF1 s'utilise de la manière suivante :

Equation y'=f(x,y)
x minimum
x maximum
y(x minimum)

Ce programme calculera automatiquement les coordonnées qui devront être celles de l'écran à chaque fois.

Exemple d'utilisation :

```
'COS(X)'    Equation fonction de x et de y (éventuellement)
0
6.28
0

DIFF1 [ENTER]
```

renvoie une sinusoïde.

4
Second ordre
DIFF2

Equation différentielle du deuxième ordre.

Ce programme est à enregistrer sous le nom : DIFF2.

Son checksum est : #08A1h (821,5 octets)

Listing

```
« IF DEPTH 4 < THEN
"→ y''(X,Y,YP),xmin ,
xmax,y(xmin),y'(xmin) " DOERR END
0 0 { } 0 → f xmin xmax y0 yp0 ymin ymax
sol h « CLLCD "RESOLUTION DE d2Y/dX2=" 1
DISP f 2 DISP "x0 = " xmin →STR +
", y0= " + y0 →STR + 3 DISP "y'0= " yp0
→STR + 4 DISP y0 'ymin' STO y0 'ymax'
```

```
STO xmin y0 R→C sol + 'sol' STO y0 'Y'
STO xmax xmin - ITER / 'h' STO yp0 'YP'
STO 1 ITER FOR I xmin I xmax xmin - *
ITER / + 'X' STO h YP * f →NUM h SQ
2 / * + Y + 'Y' STO X Y R→C sol + 'sol'
STO IF Y ymin < THEN Y 'ymin' STO END
IF Y ymax > THEN Y 'ymax' STO END YP f
→NUM h * + 'YP' STO NEXT { X Y YP }
PURGE xmin ymin R→C PMIN xmax ymax R→C
PMAX sol TRACER »
»
```

Lancement du programme :

Après avoir réglé ITER, rentrer l'expression y'' fonction de X et de Y et YP (dérivée de Y), les valeurs minimum et maximum de X et la valeur de Y correspondant à la valeur minimum de X et la valeur de Y' en ce même point . Taper ensuite :

```
DIFF2 [ENTER].
```

Remarque :

Le programme DIFF2 s'utilise de la manière suivante.

Equation y''=f(x,y,y')
x minimum
x maximum
y(x minimum)
y'(x minimum)

***y'* est notée YP dans les équations.**

Exemple d'utilisation :

```
'-Y'        Equation fonction de X, de Y et YP
0
```

```
6.28
1
0
DIFF2 [ENTER]
```

renvoie une cosinusoïde.

14
Sujets corrigés

Ce chapitre présente des sujets corrigés proposés aux candidats lors de différentes épreuves. Les programmes détaillés dans ce livre sont ici exploités en *"situation réelle"*.

1
Mines-Ponts 1981
Epreuve pratique de Mathématiques

Epreuve pratique de Mathématiques 1981

ECOLE NATIONALE DES PONTS ET CHAUSSEES
ECOLES NATIONALES SUPERIEURES DE L'AERONAUTIQUE
DE TECHNIQUES AVANCEES, DES TELECOMMUNICATIONS
DES MINES DE PARIS, DES MINES DE SAINT-ETIENNE
DE LA METALLURGIE ET DE L'INDUSTRIE DES MINES DE NANCY
DES TELECOMMUNICATIONS DE BRETAGNE
ECOLE POLYTECHNIQUE (OPTION T.A.)

CONCOURS D'ADMISSION 1981

EPREUVE PRATIQUE DE MATHEMATIQUES

OPTIONS M,P' ET T.A.

(durée 2 heures)

PREAMBULE

N.B - *Les différentes questions du problème sont, dans une large mesure, indépendantes les unes des autres.*

1°) Par convention, pour tout réel A, $\sqrt[3]{A}$ désigne l'unique réel B vérifiant $B^3=A$; $\sqrt[3]{A}$ est donc défini même pour A négatif et dans ce dernier cas, $\sqrt[3]{A} = -\sqrt[3]{|A|}$. Il en résulte que $f_3(x)=\sqrt[3]{(\sin^2(x).tgx)}$ est défini pour tout x appartenant à l'intervalle $]-\pi/2,\pi/2[$ et que f_3 admet un développement limité à n'importe quel ordre au voisinage de 0.

2°) A titre d'information, l'origine du problème est résumée ci-dessous.
Au dix-huitième siècle (et donc avant l'usage des séries), des mathématiciens (HUYGHENS, SNELLIUS, GRUNBERGER) entreprirent de calculer des valeurs décimales approchées de π par des séries trigonométriques élémentaires : il s'agissait d'améliorer la double inégalité classique sin x < x < tg x valable pour 0 < x < π/2 en introduisant des fonctions
a - qui s'expriment simplement à l'aide des fonctions trigonométriques usuelles.
b - peu différentes de x pour x voisin de zéro.

Les fonctions f_1 et f_4 du texte sont celles de Snellius, f_2 et f_3 sont celles de Huyghens.

1°) Soit a et b deux réels positifs ou nuls. On pose :

$m(a,b)=(2a+b)/3$, $g(a,b)=\sqrt[3]{(a^2b)}$

En calculant $[m^3(a,b)-g^3(a,b)]$, trouver le signe de $[m(a,b)-g(a,b)]$ lorsque $a \neq b$. Donner une condition nécessaire et suffisante très simple pour que $m(a,b)=g(a,b)$.

2°) Dans cette question et dans toute la suite du problème, on désigne par x un réel appartenant à l'intervalle $]-\pi/2,\pi/2[$; on pose:

$f_1(x)=(3\sin(x))/(2+\cos(x))$ $\quad f_2(x)=1/3\,(8\sin(x/2)-\sin(x))$

$f_3(x)=\sqrt[3]{(\sin^2(x)\tg(x))}$ $\quad f_4(x)=1/3\,(2\sin(x)+\tg(x))$

Calculer les développements limités, à l'ordre 5 inclus, de $f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$ au voisinage de 0.

En déduire l'existence d'un réel strictement positif n tel que l'on ait, pour tout x appartenant à $]0,n[$, l'inégalité:

$f_1(x) < f_2(x) < f_3(x) < f_4(x)$

3°) On suppose ici que $0 < x < \pi/2$. Quel est le signe de $f_4(x)-f_3(x)$?

4°) On pose:

$u(x) = 3(2+\cos(x)).[f_2(x)-f_1(x)]$
Montrer qu'il existe des réels finis $\alpha,\beta,\gamma,\partial$, que l'on calculera tels que:

$u(x)= \alpha.\sin(2x) + \beta.\sin(3x/2) + \gamma.\sin(x) + \partial.\sin(x/2)$

Calculer $u'(x)$ et vérifier que:

$u'(x) = P\,(\cos(x/2))$

où P est un polynôme de degré 4 en $\cos(x/2)=y$.

Expliciter P(y) et le décomposer en un produit de facteurs réels. En déduire le signe de $u'(x)$ et , pour $0<x<\pi/2$, celui de $f_2(x)-f_1(x)$.

5°) On pose :
$v(x)= x - f_2(x)$,

calculer $v'(x)$ et montrer que :
$v'(x)= Q(\cos(x/2))$

où Q est un polynôme. En déduire le signe de $v'(x)$ et pour $0<x<\pi/2$ celui de $v(x)$.

6°) On pose:
$w(x)= f_3(x)-x$

calculer $w'(x)$; montrer que son signe est celui d'une expression trigonométrique simple ; en déduire, pour $0<x<\pi/2$, le signe de $w(x)$.

7°) Dans cette question, la valeur de π est supposée inconnue. Les valeurs des lignes trigonométriques de $\pi/4$ et $\pi/6$ sont seules connues.

Calculer des expressions simples par radicaux carrés de $\cos(\pi/12)$, $\sin(\pi/12)$, $\tg(\pi/12)$ (on remarquera que $\pi/12=\pi/4-\pi/6$ et on appliquera les formules de soustraction trigonométriques). On aura soin de rendre les dénominateurs rationnels.

Calculer ensuite une expression de $\sin \pi/24$ par radicaux carrés superposés.
En déduire une expression de :

$X = 12\, f_2(\pi/12)$
par radicaux carrés.

Calculer une valeur décimale approchée de X avec la précision permise par les instruments dont dispose le candidat. Celui-ci indiquera le matériel utilisé.

Calculer ensuite une expression par radicaux de :
$Y = 12\, f_3(\pi/12)$

Calculer une valeur approchée de Y, avec la précision indiquée plus haut.

Déduire des résultats précédents un encadrement de π.

Quelles remarques vous suggère le résultat (étant rappelé que Huyghens ne disposait pas des méthodes de calcul de l'analyse moderne, fondées essentiellement sur l'emploi des séries) ?

8°) On pose :

$$I = \int_0^{p/2} f_3(x)\,dx$$

Montrer que I a un sens. La calculer. Vérifier que :

$I > \pi^2/8$

Pouvait on aisément prévoir le résultat ?

Corrigé de l'épreuve pratique Mines 1981

L'utilisation du calculateur pour résoudre le problème est volontairement très détaillée.

1°) $m^3(a,b)-g^3(a,b)=(a-b)^2.(8a+b)/27$
donc $m(a,b)-g(a,b) \geq 0$ et $m(a,b)=g(a,b) <=> a=b$

2°) Pour cette question , nous allons utiliser le répertoire DL

NB : On peut aussi directement utiliser la commande TAYLOR qui se trouve dans le chapitre DL. Cependant La fonction TAYLR interne au calculateur ne permet pas de faire la calcul (trop long ou erreur).

Calcul du DL de f1
* Effectuons un DL à l'ordre 5 de sin

```
5
DSIN
```

renvoie

```
{ 0 1 0 (-1,6) 0 (1,120) }
```

* Effectuons un DL à l'ordre 5 de cos

```
5
DCOS
```

renvoie

```
{ 0 1 0 (-1,6) 0 (1,120) }
{ 1 0 (-1,2) 0 (1,24) 0 }
```

* Ajoutons 2 au DL de cos à l'ordre 5

```
2
5
DPLUS
{ 0 1 0 (-1,6) 0 (1,120) }
{ 3 0 (-1,2) 0 (1,24) 0 }
```

* Divisons à l'ordre 5 les 2 DL précédents.

```
5
DDIV
{ 0 (1,3) 0 0 0 (-1,540) }
```

* Multiplions par 3

```
3
5
DMULT
{ 0 1 0 0 0 (-1,180) }
```

D'où $f_1(x) = x - x^5/180 + o(x^5)$
Procédons de façon similaire pour le calcul du DL de f_2:
* Calculons un DL à l'ordre 5 de sin

```
5
DSIN
DUP
```

(NB : nous dupliquons la ligne pour l'utiliser plus tard)

* Calculons un DL à l'ordre 5 de sin(x/2)

```
{ 0 (1,2) 0 0 0 0 }
SWAP
5
DCOMP
```

renvoie

```
{ 0 (1,2) 0 (-1,48) 0 (1,3840) }
```

* Calculons un DL à l'ordre 5 de 8.sin(x/2)-sin(x)

```
8
5
DMULT
```

renvoie

```
{ 0 4 0 (-1,6) 0 (1,480) }
```

* Calculons le DL à l'ordre 5 de 8sin(x/2)-sin(x)

```
SWAP
5
DMOINS
```

renvoie

```
{ 0 3 0 0 0 (-1,160) }
```

* Calculons le DL à l'ordre 5 de f_2

```
(1,3)
5
DMULT
```

renvoie

```
{ 0 1 0 0 0 (-1,480) }
```

D'où $f_2(x)= x - x^5/480 + o(x^5)$

La résolution de f_3 et de f_4 se fait de la même manière.
Il faut cependant remarquer que
$\sqrt[3]{(\sin^2(x).\tg(x))} = \sin(x)/\sqrt[3]{\cos(x)}$
Finalement, calculs faits , on trouve :

$f_3(x)= x + x^5/45 + o(x^5)$

$f_4(x)= x + x^5/20 + o(x^5)$

Comme $-1/180 < -1/480 < 0 < 1/45 < 1/20$, il existe $n>0$ tel que pour tout $x \in]0,n[$ $f_1(x) < f_2(x) < x < f_3(x) < f_4(x)$

3°) On utilise le 1° avec $a=\sin(x)$ et $b=\tan(x)$.
Pour $x \in]0,\pi/2[$ $\sin x < \tan x$ donc :
$x \in]0,\pi/2[$ $f_4(x)-f_3(x) > 0$

4°) $u(x) = 3(2+\cos(x)).(f_2(x)-f_1(x))$
$u(x) = [(2+\cos(x)).(8\sin(x/2)-\sin(x))-3*3\sin(x)$
$=16\sin(x/2)-2\sin(x)+8\cos(x)\sin(x/2)-\sin(x)\cos(x)-9*\sin(x)$
$u(x)=-1/2\sin(2x)-11\sin(x)+16\sin(x/2)+8\cos(x)\sin(x/2)$

or $\sin(3x/2)=\sin(x+x/2)=2\sin(x/2)\cos(x)+\sin(x/2)$

Donc $u(x)=-1/2\sin(2x)-11\sin(x)+16\sin(x/2)+4(\sin(3x/2)-\sin(x/2))$

$u(x)=-1/2 \sin(2x) + 4 \sin(3x/2) -11 \sin(2x) + 12 \sin(x/2)$
$\alpha=-1/2$ $\beta=4$ $\gamma=-11$ $\partial=12$

$u'(x)=-\cos(2x)+6\cos(3x/2)-11\cos(x)+6\cos(x/2)$
$u'(x)=-[1-8\cos^2(x/2)+8\cos^4(x/2)]+6[-3\cos(x/2)+4\cos^3(x/2)]$
$-11[-1+2\cos^2(x/2)] + 6\cos(x/2)$

Nota Bene : On peut utiliser le programme TRIGO pour trouver les formules ci-dessus.

Finalement $u'(x)= 10 - 12y - 14y^2 + 24y^3 - 8y^4$

$P(y)=-8(y-1)^2[y-(1+\sqrt{6})/2][y-(1-\sqrt{6})/2]$

Donc $x \in]0,\pi/2[$, $\cos x \in]1/\sqrt{2},1[$ et $P(\cos x) > 0$
Donc u est croissante et $u(x) > 0$
Donc $x \in]0,\pi/2[$ $f_2(x)-f_1(x) > 0$

5°) $v(x)=x-f_2(x)$

$v'(x)=(2/3)\,(\cos(x/2)-1)^2$ Donc x E $]0,\pi/2[$ $v(x)>0$

6°) $w(x)=(\sin x)/(\cos x)^{1/3} - x$

$w'(x)= 1/3\,[2(\cos x)^{2/3} + (\cos x)^{-4/3} -3]$

En posant $t=(\cos x)^{2/3}$ $w'(x)=(1/3)\,(1/t^2)\,(t-1)^2\,(2t+1) > 0$

Donc x E $]0,\pi/2[$ $f_1(x) < f_2(x) < x < f_3(x)$

7°) $\cos(\pi/12)=\cos(\pi/4-\pi/6)=\cos(\pi/4)\cos(\pi/6)+\sin(\pi/4)\sin(\pi/6)$
$=\sqrt{2}/2*\sqrt{3}/2 + \sqrt{2}/2/2$

$\cos(\pi/12)=(\sqrt{6}+\sqrt{2})/4$

$\sin(\pi/12)=\sin(\pi/4)\cos(\pi/6)-\cos(\pi/4)\sin(\pi/6)$
$=\sqrt{2}/2*\sqrt{3}/2 - \sqrt{2}/2/2$

$\sin(\pi/12)=(\sqrt{6}-\sqrt{2})/4$

$\tan(\pi/12)=(\sqrt{6}-\sqrt{2})^2/4= 2-\sqrt{3}$

$1-2\sin^2(\pi/24)=\cos(\pi/12)$

$\sin(\pi/24)=\sqrt{(4-\sqrt{6}-\sqrt{2})}/\sqrt{8}$

$X=12\,f_2\,(\pi/12) =4\,(8\,\sin(\pi/24) - \sin(\pi/12))$
$=4\sqrt{8}\sqrt{(4-\sqrt{6}-\sqrt{2})} - \sqrt{6} + \sqrt{2}$

Numériquement:

On tape soit

4 8 √ * 4 6 √ - 2 √ - √ * 6 √ - 2 √ +

soit

`'4*√8*√(4-√6-√2)-√6+√2'` →NUM

donc 3.1415619 < X < 3.1415620

Matériel utilisé : HP 48sx.

$Y=12\, f_3(\pi/12)$
$=12 \sin(\pi/12)/\cos(\pi/12)1/3$
$=3(\sqrt{6}-\sqrt{2}) / 3\sqrt{(\sqrt{6}+\sqrt{2})} * 3\sqrt{4}$

Numériquement

'3*(√6-√2)/(√6+√2)^(1/3)*4^(1/3)' →NUM

donc 3.1419279 < Y < 3.1419280

D'après le 6°) 3.14156 < π < 3.14193

8°)

$$I = \int_0^{p/2} f_3(x)\ dx$$

Le problème est en π/2
Soit y<π/2

$$\int_0^{y} f_3(x)\ dx = \int_0^{y} \sin(x)/\sqrt[3]{\cos(x)}\ dx \longrightarrow 3/2$$

en faisant le changement de variable u = cos(x)

L'intégrale précédente tend vers 3/2 lorsque y tend vers π/2

Donc I a un sens et I=3/2 > π2/8 = 1.2337

$$x \in]0,\pi/2[\ f_3(x)>x \quad \text{donc} \quad \int_0^{\pi/2} f_3(x)dx < \int_0^{\pi/2} x\ dx = \pi^2/8$$

On pouvait donc aisément prévoir $I > \pi^2/8$

2
ESTP 91
Mathématiques

ECOLE SPECIALE DES TRAVAUX PUBLICS, DU BATIMENT ET DE L'INDUSTRIE
CONCOURS COMMUN D'ADMISSION
AUX ECOLES SUPERIEURES
DES TRAVAUX PUBLICS
DU BATIMENT
DE MECANIQUE - ELECTRICITE
DE TOPOGRAPHIE

SESSION 1991

DEUXIEME INTERROGATION DE MATHEMATIQUES

Durée : 4 heures

SUJET N°1

Pour tout entier $n \geq 0$ et tout réel $x > 0$, on pose

$$F_n(x) = \int_0^{\infty} \exp(-xt)\ (\sin t)^n\ dt$$

1) Montrer que cette intégrale généralisée converge.

2) Calculer $F_0(x)$ et $F_1(x)$

3) En écrivant

$$F_n(x)=\int_0^{\infty} \exp(-xt)\,(\sin t)^{n-1}\,d(-\cos t) \quad n\geq 2$$

et en utilisant l'intégration par partie, établir la relation
$F_n(x)=n(n-1)/(n^2+x^2)\,F_{n-2}(x)$.
En déduire la valeur de $F_5(x)$.

4) Pour k et n entiers tels que $2 \leq k \leq n$, on introduit les intégrales

$$I_{n,k}=\int_0^{\infty} x^{k-1}\,F_n(x)\,dx \qquad S_{n,k}=\int_0^{\infty} \sin^n(t)/t^k\,dt$$

On admet la convergence de ces intégrales, ainsi que la validité de l'interversion suivante des signes d'intégration:

$$\int_0^{\infty} x^{k-1}\,\left(\int_0^{\infty} e^{-xt}\,\sin^n t\,dt\right)dx$$

$$= \int_0^{\infty} \sin^n t\,\left(\int_0^{\infty} e^{-xt}\,x^{k-1}\,dx\right)dt$$

Exprimer $S_{n,k}$ au moyen de $I_{n,k}$

5) Déterminer les rationnels a , b et c tels que $S_{5,5} = a\,\pi$
et $S_{5,4}= b\,\ln(3) + c\,\ln(5)$

où ln désigne le logarithme népérien.

SOLUTION

1) x>0

soit Y>0 Z>0 Z>Y

$$\int_Y^Z | e^{-xt} \sin^n t | dt$$

$$\leq \int_Y^Z e^{-xt} |\sin^n t| dt \leq \int_Y^Z e^{-xt} dt$$

$$\leq -1/x [\exp(-xZ) - \exp(-xY)]$$

tend vers 0 quand Y et Z tendent vers +oo

Donc, d'après le critère de Cauchy, $F_n(x)$ converge.

2°)

$$F_0(x) = \int_0^\infty e^{-xt} dt = 1/x$$

$$F_1(x) = \int_0^\infty e^{-xt} \sin t \, dt$$

Intégrons par parties

$$F_1(x) = [-(e^{-xt} \sin t)/x]_0^\infty + \int_0^\infty e^{-xt} \cos t / x \, dt$$

$$= [-(e^{-xt} \cos t) /x^2]_0^\infty - \int_0^\infty e^{-xt} \sin t / x^2 \, dt$$

$= 1/x^2 - F_1(x)/x^2$

$F_1(x) = 1/x^2 \ / \ (1+1/x^2) = 1 \ / \ (1+x^2)$

3) d(-cos t)/dt= sin(t) donc pour n≥2

$$F_n(x) = \int_0^{\infty} e^{-xt} \sin^{n-1}t \, d(-\cos t)$$

Intégrons par parties

$$F_1(x) = [-e^{-xt}\sin^n t/x]_{0+}^{\infty} \int_0^{\infty} e^{-xt} n \sin^{n-1}t \cos t/x \, dt$$

$$= [-(e^{-xt} \sin^{n-1}t \, n \cos t) /x^2]_0^{\infty}$$
$$+ \int_0^{\infty} e^{-xt} \, n((n-1)\sin^{n-2}t \cos^2 t - \sin^n t) / x^2 \, dt$$

or $\cos^2(x)=1-\sin^2(x)$

donc $F_n(x)= n(n-1)/x^2 F_{n-2}(x) - n^2/x^2 F_n(x)$

donc $F_n(x)= n(n-1)/(n^2+x^2) F_{n-2}(x)$

Remarque : nous n'avons pas utilisé l'indice de l'énoncé.

$F_5(x)=5*4/(25+x^2) \ F_3(x)= 5*4*3*2/(25+x^2)/(9+x^2)*F_1(x)$

$F_5(x)= 120 / \{ (25+x^2)(9+x^2)(1+x^2) \}$

4)

$$I_{n,k} = \int_0^{\infty} \sin^n t \int_0^{\infty} e^{-xt} \, x^{k-1} \, dx \, dt$$

$k \geq 2$

$$J_k(t) = \int_0^{\infty} e^{-xt}\, x^{k-1}\, dx$$

Par intégration par parties, on obtient

$$J_k(t) = (k-1)/t\ J_{k-1}(t)$$

Par récurrence

$$J_k(t) = (k-1)!\ /\ t^{k-1} \quad J_1(t) = (k-1)!\ /\ t^k$$

$$I_{n,k} = (k-1)!\ S_{n,k}(t)$$

$$S_{n,k} = I_{n,k}\ /\ (k-1)!$$

5) $S_{5,5} = I_{5,5} / 4! = I_{5,5} / 24$

Remarque : Le calcul de 4! de fait en tapant : 4 FACT

$$S_{5,5} = \int_0^{\infty} x^4\ 120\ /\ (1+x^2)\ /\ (9+x^2)\ /\ (25+x^2)\ /\ 24$$

$$\frac{x^4}{(1+x^2)(9+x^2)(25+x^2)}$$

$$= \frac{ax+b}{1+x^2} + \frac{cx+d}{9+x^2} + \frac{ex+f}{25+x^2}$$

en multipliant par $1+x^2$ et en prenant x=i, on obtient

$1 / (9-1)(25-1) = ai+b$ donc $a=0$ $b=1/192$
en multipliant par $9+x^2$ et en prenant x=3i, on obtient

$81 / (1-9)(25-9) = ci+d$ donc $c=0$ $d=-81/128$

en multipliant par $25+x^2$ et en prenant x=5i, on obtient

625 / (1-25)(9-25) = ei+f donc e=0 f=625/384

en multipliant les coefficient par 120/24=5

$$\frac{x^4}{(1+x^2)(9+x^2)(25+x^2)} = \frac{5/192}{1+x^2} + \frac{-405/128}{9+x^2} + \frac{3125/384}{25+x^2}$$

comme d/dx (atan (x/a)) = 1/a 1/(1+x^2/a^2)
= a /(a^2+x^2)

$S_{5,5} = \pi$ (5/192/2 - 405/128/3/2 + 3125/384/5/2)

```
5 192 QDIV 2 QDIV
405 128 QDIV 3 QDIV 2 QDIV QMOINS
3125 384 QDIV 5 QDIV 2 QDIV QPLUS
```

donne :
115/384
donc a=115/384

$S_{5,4}=I_{5,4}/3!$

$$S_{5,4}=\int_0^{\infty} x^3 \; 120 \; / \; (1+x^2) \; / \; (9+x^2) \; / \; (25+x^2) \; / \; 24$$

$$\frac{x^3}{(1+x^2)(9+x^2)(25+x^2)} = \frac{ax+b}{1+x^2} + \frac{cx+d}{9+x^2} + \frac{ex+f}{25+x^2}$$

en multipliant par $1+x^2$ en en prenant x=i

on obtient -i/(9-1)(25-1) = ia+b

b=0 et a=-1/192

en multipliant par $9+x^2$ et en prenant x=3i

on obtient -27i/(1-9)(25-9)= 3ci+d

d=0 et c=9/128

```
27 8 QDIV 25 9 - QDIV 3 QDIV
```

donne 9/128

en multipliant par $25+x^2$ et en prenant x=5i

on obtient

```
-125 i/(1-25)(9-25)= 5ie+f
```

f=0 et e=-25/384
car

```
125 24 QDIV 9 25 - QDIV 5 QDIV
```

donne -25/384

Nota Bene :
$d/dx\ LN(A+X^2)=2X/(A+X^2)$

$$-5/16[Ln(1+x^2)]_0^N+135/32[Ln(9+x^2)]_0^N-125/32[Ln(25+x^2)]_0^N$$

$$=-5/16\ Ln(1+N^2) + 135/32\ Ln(9+N^2) - 125/32\ Ln(25+N^2) - 135/32\ Ln(9) + 125/32\ Ln(25)$$

$$= Ln(9+N^2)^{(135/32)}/(1+N^2)^{(5/16)}/(25+N^2)^{(125/32)} - 135/32\ Ln(9) + 125/32\ Ln(25)$$

Par équivalence $(9+N^2)^{(135/32)}/(1+N^2)^{(5/16)}/(25+N^2)^{(125/32)}$ tend vers 1 lorsque n tend vers $+\infty$
et donc par passage à la limite

$S_{5,4}$ = -135/32 Ln(9) + 125/32 Ln(25)
= -135/16 Ln(3) + 125/16 Ln(5)
Donc b=-135/16 et c=125/16

3
ULM 84

ENS ULM SEVRES 84 - Option M' - Epreuve pratique

NB: Les diverses parties de l'épreuve sont indépendantes.

I

Décrire sommairement l'outil de calcul utilisé. S'il s'agit d'une calculatrice de poche, on précisera en particulier les points suivants:

Marque - Modèle - nombre de chiffres affichés - nombre de registres de mémoire - caractère programmable ou non - imprimante - liste des principales et opérations fournies.

II

Le but de cet exercice est le calcul ex nihilo d'une table de logarithmes népériens des nombres de 1 à 10.

1°) Montrer qu'on peut se limiter au calcul des nombres 2,3,5,7

2°) Donner la liste des nombres entiers compris entre 1 et 100 n'admettant aucun diviseur premier distinct de 2,3,5,7.

3°) Soient a et b deux nombres entiers strictement positifs. Montrer que l'on peut calculer log(a/b) comme le produit de 2(a-b)(a+b) par une série de puissances L(t) en $t=(a-b)^2/(a+b)^2$. Donner l'expression L(t) et étudier sa convergence. Déterminer les restrictions éventuelles de la méthode.

4°) Par utilisation des résultats de 2° et 3°, calculer le logarithme de 4 nombres bien choisis de la forme a/b, a et b n'admettant pas de

diviseur premier distinct de 2,3,5 ou 7. On fera le calcul avec 13 décimales, en donnant le détail des opérations, et en indiquant comment dépasser la capacité de la machine.

5°) Par la résolution d'un système linéaire de 4 équations linéaires à 4 inconnues, déduire de 4° la valeur des logarithmes des nombres 2,3,5 et 7.

6°) Terminer le calcul des logarithmes des nombres entiers de 1 à 10. On donnera le résultat final sous forme d'une table avec 12 décimales.A titre de comparaison , on pourra donner la table obtenue par utilisation directe de la calculatrice - ou la lecture d'une table imprimée.

III

1°) Indiquer une méthode générale pour donner la liste de tous les diviseurs d'un nombre entier n, donné par sa décomposition en facteurs premiers. Quel est le nombre de ces diviseurs (y compris 1 et n) ?

2°) En application de ce qui précède, donner la liste explicite des diviseurs du nombre 675. En déduire la liste $Z = \{ a_1, ..., a_{12} \}$, rangée par ordre croissant, des nombres entiers n satisfaisant aux conditions suivantes :

$684 \leq n \leq 768$;

n est de la forme $2^a .b$ où $a \geq 0$ est entier, et b divise 675.

3°) On rappelle les symboles courants utilisés pour désigner les douze notes de musique de la gamme :

do do# ré ré# mi fa fa# sol sol# la la# si

En principe, une note de musique correspond à une fréquence sonore bien déterminée; pour décrire une gamme, on se donne une suite de 12 fréquences (mesurées en Hertz Hz) $f_1 < f_2 < ... < f_{11} < f_{12}$ avec f_{12}

$< 2\ f_1$, et l'on attribue la fréquence f_1 à do, la fréquence f_2 à do#, etc. Les notes "non altérées" sont do,ré,mi,fa,sol,la,si correspondant aux fréquences $f_1, f_3, f_5, f_6, f_8, f_{10}, f_{12}$. On pose $f_{13} = 2\ f_1$ (correspondant au do de l'octave supérieure)

La gamme de Zarlino s'obtient en choisissant une fréquence f et en posant $f_i = f_{a_i}$ où les a_i sont comme en 2°; la gamme de Pythagore s'obtient en choisissant la fréquence f_6 correspondant à la note fa, et en imposant aux rapports f_i/f_6 d'être des nombres rationnels de la forme $2^a.b$ avec a entier rationnel et b divisant 311. (Ceci détermine la gamme de façon unique.)

Pour chacune de ces deux gammes, on donnera sous forme de tableaux les nombres exprimés comme fractions :

- Les rapports f_i/f_1 pour $1 \leq i \leq 12$
- Les rapports f_{i+1}/f_i pour $1 \leq i \leq 12$ (correspondant aux "demi tons")
- Les rapports f_3/f_1, f_5/f_3, f_6/f_5, f_8/f_6, f_{10}/f_8, f_{12}/f_{10}, f_{13}/f_{12}

Quelles remarques vous suggèrent ces tableaux ?

4°) Si deux notes N_1 et N_2 correspondent respectivement aux fréquences f_1 et f_2, on mesure l'intervalle musical entre N_1 et N_2 par le nombre $I(N_1,N_2)=\log_2(f_2/f_1)$

(le logarithme est pris en base 2); l'unité d'intervalle est donc l'octave, ou intervalle séparant deux fréquences f et 2f.

La gamme tempérée de Rameau se compose de douze notes séparées par des intervalles égaux.

Donner sous forme de tableau les intervalles Z_1, ... ,Z_{12} entre le do et les notes successives do,do#, ... dans la gamme de Zarlino, ainsi que les nombres P_1, ..., P_{12} analogues pour la gamme de Pythagore, et les nombres R_1, ..., R_{12} pour la gamme de Rameau. On donnera aussi les écarts $Z_k - R_k$ et $P_k - R_k$ qui mesurent le "défaut de tempérament". Tous ces nombres sont exprimés sous forme décimale à 10^{-6} près.

5°) Sur le tableau précédent, vérifier que, à la précision donnée, tous les écarts Z_k-R_k et P_k-R_k sont des multiples entiers de celui d'entre eux qui a la plus faible valeur absolue. Déterminer deux nombres u et v tels que l'on puisse écrire des relations :

$$P_k\text{-}R_k = b_k\, u \qquad Z_k\text{-}R_k = c_k\, u + d_k\, v$$

pour k=1,...,12 où b_k,c_k et d_k sont des entiers que l'on déterminera explicitement. Donner la valeur numérique, à 10^{-10} près, de u et v et expliquer la remarque faite au début de ce numéro.

6°) On complète la gamme en insérant 5 nouvelles notes :

réb	*entre do et do#*
mib	*entre ré et ré#*
solb	*entre fa et fa#*
lab	*entre sol et sol#*
sib	*entre la et la#*

Dans la gamme de Pythagore étendue, toutes les fréquences sont de la forme $2^a.3^b.f_1$ avec a entier rationnel et $-6 \le b \le 10$ et f_1 la fréquence de do.

Donner sous forme de fractions les rapports f/f_1 pour ces 5 nouvelles notes, ainsi que les intervalles $\log_2(f/f_1)$ sous forme décimale, à 10^{-6} près.

7°) La gamme de Hölder se compose des 17 notes do,réb,do#,ré,..., et l'intervalle entre deux notes est toujours un multiple entier de 1/53 d'octave; les notes sont choisies de manière à être le plus proche possible des notes de la gamme de Pythagore étendue. On donnera le tableau des intervalles séparant deux notes successives dans cette gamme, en fonction de l'unité c=1/53 d'octave (appelée "comma de Hölder")

SOLUTION

PARTIE II

1°. Log1=0 , Log4=2Log2 , Log6= Log2 + Log3 , Log 8=3Log2
Log9=2Log3 , Log10 = Log2+Log5

On peut donc se contenter de calculer les logarithmes de 2,3,5 et 7.

2°. On peut imaginer un petit programme pour trouver ces nombres :

```
« → n « IF n 2 MOD 0 == THEN n 2 / DIVI
ELSE IF n 3 MOD 0 == THEN n 3 / DIVI
ELSE IF n 5 MOD 0 == THEN n 5 / DIVI
ELSE IF n 7 MOD 0 == THEN n 7 / DIVI
ELSE IF n 1 == THEN 1 ELSE 0 END
END END END END »
»
[ENTER]
'DIVI' [ENTER] [STO]

« { } 1 100 FOR I IF I DIVI THEN I + END
NEXT »
[ENTER]
'TEST' [ENTER] [STO]
```

Lancez TEST . On obtient la liste des nombres répondant à la question.
En faisant DUP SIZE , on sait qu'il y en a 46 :

```
1  2  3  4  5  6  7  8  9  10 12 14 15
16 18 20 21 24 25 27 28 30 32 35 36 40
42 45 48 49 50 54 56 60 63 64 70 72 75
80 81 84 90 96 98 100
```

3°. Si $-1 < x < 1$

$$\text{Log}\left(\frac{1+x}{1-x}\right) = \text{Log}(1+x) - \text{Log}(1-x) = 2x \sum_{n=0}^{\infty} x^{2n} / (2n+1)$$

or pour $x=(a-b)/(a+b)=1-2b/(a+b) < 1$ car a et b positifs

$x=-1+2a/(a+b) > -1$ pour la même raison

$$\text{Log}(a/b)=2(a-b)/(a+b)\ L(t)$$

$$\text{ou } L(t)=\sum_{n=0}^{\infty} t^n/(2n+1) \quad \text{et } t=(a+b)^2/(a-b)^2$$

La série converge pour $0 \leq t < 1$
Donc quels que soient a et b entiers positifs strictement.

4°. Pour que la convergence de la série soit le plus rapide possible, il faut que t soit le plus petit possible.
Or $t=(a-b)^2/(a+b)^2$

Parmi les nombres du 2°, les nombres qui conviennent le mieux sont :

a=81	a=64	a=50	a=49
b=80	b=63	b=49	b=48

Calculons la valeur du reste de la série :

$$R_n = \sum_{m=n+1}^{\infty} t^m/(2m+1) \leq \sum_{m=n+1}^{\infty} t^m/(2n+3)$$

$$R_n \leq t^{n+1}/(2n+3)(1-t)$$

a=81 b=80 t=1/1612 avec n=2 , $Rn \leq 8,3\ 10^{-15}$

a=64 b=63 t=1/1272 avec n=2 , $Rn \leq 3,5\ 10^{-14}$

a=50 b=49 t=1/992 avec n=3 , $Rn \leq 1,3\ 10^{-17}$

a=49 b=48 t=1/972 avec n=3 , $Rn \leq 1,5\ 10^{-17}$

On peut donc effectuer les calculs avec 13 décimales exactes

$$\text{Log}(81/80)=2/161\ L(1/1612)$$
$$L(1/1612)-R_2(1/1612)=1 + 1/1612/3 + 1/1614\ /5$$

En fractions rationnelles :

```
1
161
SQ
QDIV
3 QDIV
1 QPLUS
161 4 QPUIS
QINV
5 QDIV QPLUS

PUIS 2 QMULT 161 QDIV
```

Finalement Log(81/80)= 10078603223/811317126010 à 10^{-13} près

En effectuant la division à la calculatrice :

Log(81/80) = 0,0124225199985 à 10^{-13} près

De même pour les autres fractions:

Log(64/63) = 7804500526/495575541105 à 10^{-13} près

= 0,0157483569681 à 10^{-13} près

Log(50/49) = 2534840461/125470335295 à 10^{-13} près

= 0,0202027073176 à 10^{-13} près

Log(49/48) = 8746515924/424190993509 à 10^{-13} près

= 0,0206192872028 à 10^{-13} près

Pour dépasser la capacité de la machine, il suffit de faire les divisions à la main.

```
Log(81/80)=     -4 Log(2) + 4 Log(3) - Log(5)
Log(64/63)=      6 Log(2) - 2 Log(3) - Log(7)
Log(50/49)=        Log(2) + 2 Log(5) -2 Log(7)
Log(49/48)=     -4 Log(2) - Log(3) + 2 Log(7)
```

Il s'agit d'un système de quatre équations à quatre inconnues.

```
Log(2) =   10 Log(81/80) + 14 Log(64/63)
         + 5 Log(50/49) + 12 Log(49/48)
Log(3) =   16 Log(81/80) + 22 Log(64/63)
         + 8 Log(50/49) + 19 Log(49/48)
Log(5) =   23 Log(81/80) + 32 Log(64/63)
         + 12 Log(50/49) + 28 Log(49/48)
Log(7) = 28 Log(81/80) + 39 Log(64/63)
         + 14 Log(50/49) + 34 Log(49/48)
```

	Calcul "à la main"	*Calcul avec la HP-48*
Log(1) =	0	0
Log(2) =	0,693147180559	0,69314718056
Log(3) =	1,098612288667	1,09861228867
Log(4) =	1,386294361118	1,38629436112
Log(5) =	1,609437912433	1,60943791243
Log(6) =	1,791759469226	1,79175946923
Log(7) =	1,945910149054	1,94591014906
Log(8) =	2,079441541677	2,07944154168
Log(9) =	2,197224577334	2,19722457734
Log(10)=	2,302585092992	2,30258509299

PARTIE III

1°. si d_i sont les diviseurs de n de multiplicité respective a_i

$$n = \prod_{i=0}^{p} d_i^{a_i}$$

Les diviseurs de n sont les nombres $\prod_{i=0}^{p} d_i^{b_i}$ avec b_i entre 0 et a_i

Le nombre de diviseurs est donc $\prod_{i=0}^{p} (a_i+1)$

2°. Utilisons le programme DECOMP

```
#675h
DECOMP
```

renvoie

```
{ 3 3 3 5 #5 }
```

Donc $675 = 3^3 \times 5^2$

D'après la formule établie ci-dessus, il y a 12 diviseurs.
Il s'agit de { 1 3 5 9 15 25 27 45 75 135 225 675 }

Pour trouver ces diviseurs, vous pouvez utiliser le programme DIVIS du répertoire RACINES :

```
675 [ENTER]
DIVIS [ENTER]
```

renvoie cette liste.

Vous pouvez ensuite la trier avec le programme TRI.

Multiplions chacun de ces nombres par une puissance convenable de 2 pour qu'il soient compris entre 384 et 768 (strict)
Cette puissance de 2 est unique puisque 384 x 2 = 768

On obtient

```
Z = { 384 400 432 450 480 512 540 576
600 640 675 720 }
```

Pour trouver ces nombres, on peut faire un petit programme

```
« IF DUP 384 < THEN REPEAT 2 * DO DUP
384 ≥ END END » [ENTER]
'TEST2' [STO]
```

```
1 [ENTER]
TEST2 [ENTER]
```

renvoie 512.

3°.
Il suffit d'utiliser le programme QDIV

Ainsi

```
400 [ENTER]
384 [ENTER]
QDIV [ENTER]
```

renvoie 25/24

On peut de cette manière remplir les tableaux suivants

GAMME DE ZARLINO

note	i	f_i/f_1	f_{i+1}/f_i
do	1	1	25/24
do#	2	25/24	27/25
re	3	9/8	25/24
re#	4	75/64	16/15
mi	5	5/4	16/15
fa	6	4/3	135/128
fa#	7	45/32	16/15
sol	8	3/2	25/24
sol#	9	25/16	16/15
la	10	5/3	135/128
la#	11	225/128	16/15
si	12	15/8	16/15

$f_3/f_1 = 9/8$ $f_5/f_3 = 10/9$ $f_6/f_5 = 16/15$
$f_8/f_6 = 9/8$ $f_{10}/f_8 = 10/9$ $f_{12}/f_{10} = 9/8$
$f_{13}/f_{12} = 16/15$

GAMME DE PYTHAGORE

Il faut trouver les fréquences des notes "au dessus" de fa :
fa fa# sol sol# la la# si , soit 7 notes pour ces notes $2^a.b$ doit être compris entre 1 et 2 et les fréquences des notes "en dessous" de fa: do do# re re# mi , soit 5 notes
pour ces notes 2^a.b doit être compris entre 1/2 et 1
Une autre contrainte est que la plus grande fréquence soit inférieure au double de la plus petite fréquence.

```
      b=3x
•si x=0     on prend a=0      fa    f/f6=1
      ie 1
•si x=1     on prend a=-1     (do)  f/f6=1.5
      ie 3/2
•si x=2     on prend a=-3     sol   f/f6=1.125
      ie 9/8
•si x=3     on prend a=-4     (re)  f/f6=1.6875
            ie 27/16
•si x=4     on prend a=-6     la    f/f6=1.265625
      ie 81/64
•si x=5     on prend a=-7     (mi)  f/f6=1.8984375
      ie 243/128
•si x=6     on prend a=-9     si    f/f6=1.423828125
      ie 729/512
•si x=7     on prend a=-11    fa#   f/f6=1.06787109375
      ie 2187/2048
•si x=8     on prend a=-12    (do#) f/f6=1.60180664062
      ie 6561/4096
•si x=9     on prend a=-14    sol#  f/f6=1.20135498047
      ie 19683/16384
•si x=10    on prend a=-15    (re#) f/f6=1.8020324707
      ie 59049/32768
•si x=11    on prend a=-17    la    f/f6=1.35152435303
      ie 177147/131072
```

Les 5 notes correspondant aux notes en dessous de fa sont celles qui ont le plus grand coefficient f/f_6 , que l'on va diviser par 2 pour avoir leur valeur dans la gamme cherchée. (*)

GAMME DE PYTHAGORE

note	*i*	f_i/f_1	f_{i+1}/f_i
do	1	1	2187/2048
do#	2	2187/2048	256/243
re	3	9/8	2187/2048
re#	4	19683/6384	256/243
mi	5	81/64	256/243
fa	6	4/3	2187/2048
fa#	7	729/512	256/243
sol	8	3/2	2187/2048
sol#	9	6561/4096	256/243
la	10	27/16	2187/2048
la#	11	59049/32768	256/243
si	12	243/128	256/243

$f_3/f_1 = 9/8$ $f_5/f_3 = 9/8$ $f_6/f_5 = 256/243$
$f_8/f_6 = 9/8$ $f_{10}/f_8 = 9/8$ $f_{12}/f_{10} = 9/8$
$f_{13}/f_{12} = 256/243$

4°.
Le tableau se calcule à la machine, et on obtient à 10^{-6} près

i	Z_i	P_i	R_i	Z_i-R_i	P_i-R_i
1	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.058894	0.094738	0.083333	-0.024440	0.011404
3	0.169925	0.169925	0.166667	-0.003258	0.003258
4	0.228819	0.264663	0.250000	-0.021181	0.014663
5	0.321928	0.339850	0.333333	-0.011405	0.006517
6	0.415037	0.415037	0.416667	-0.001629	-0.001629
7	0.491853	0.509775	0.500000	-0.008147	0.009775
8	0.584963	0.584963	0.583333	0.001629	0.001629
9	0.643856	0.679700	0.666667	-0.022810	0.013033
10	0.736966	0.754888	0.750000	-0.013034	0.004888
11	0.813781	0.849625	0.833333	-0.019552	0.016292
12	0.906891	0.924813	0.916667	-0.009776	0.008146

5°.
Il semble que l'on obtienne numériquement

$P_k-R_k = b_k\ |P_6-R_6|$
$Z_k-R_k = r_k\ (Z_6-R_6) = c_k\ |P_6-R_6| + d_k\ |Z_6-P_6|$

$u=|P_6-R_6|=0.001629$
$v=|Z_6-P_6|=0$

k:	1	2	3	4	5	6	7	8	9	10	11	12
b_k:	0	7	2	9	4	-1	6	1	8	3	10	5
c_k:	0	-15	2	-13	-7	-1	-5	1	-14	-8	-12	-6

$P_k - R_k = Log_2(\ 2\text{^}a.3\text{^}x.4/3\ ) - (k-1)/12$
$= a + 2 + (x-1)\ Log2(3) - (k-1)/12$

En posant $u=Log_2(3)$ et $b_k = x-1$

$P_k-R_k = b_k\ u + 1/12\ (19x + 12a - k + 6)$

Finalement $P_k - R_k = b_k\ u$

On vérifie dans chaque cas que $19x + 12a - k + 6 = 0$

De même Z_k-R_k $= Log_2(\ 2\text{^}a.3\text{^}x.5\text{^}y/3.2\text{^}7) - (k-1)/12$
$= a - 7 + (x-1)\ Log_2(3) + y\ log_2(5) - (k-1)/12$

Si $v=7Log_2(3)+Log_2(5)-161/12$
$Z_k - R_k = (x-1-7y)\ u + y\ v + 1/12\ (12a + 19y + 28x - k - 102)$
Or $12a + 19y + 28x - k - 102 = 0$ dans chacun des cas

Donc $Z_k-R_k = c_k\ u + d_k\ v$

les valeurs de dk sont donc

k	1	2	3	4	5	6	7	8	9	10	11	12
d_k	0	2	0	2	1	0	1	0	2	1	2	1

et numériquement v=0.000001

6°. $f/f_1 = 2\text{^}a.3\text{^}b$ b entre -6 et 10
on veut que $1 \leq f/f_1 < 2$

Donc en procédant de manière similaire au 3°

note	*a*	*b*
do	0	0
réb	8	-5
do#	-11	7
ré	-3	2
mib	5	-3
ré#	-14	9
mi	-6	4
fa	2	-1
solb	10	-6
fa#	-9	6
sol	-1	1
lab	7	-4
sol#	-12	8
la	-4	3
sib	4	-2
la#	-15	10
si	-7	5

Pour réb	$f/f_1 = 256/243$	$\text{Log}_2(f/f_1) = 0.075187$
Pour mib	$f/f_1 = 32/27$	$\text{Log}_2(f/f_1) = 0.245112$
Pour solb	$f/f_1 = 1024/729$	$\text{Log}_2(f/f_1) = 0.490225$
Pour lab	$f/f_1 = 128/81$	$\text{Log}_2(f/f_1) = 0.660150$
Pour sib	$f/f_1 = 16/9$	$\text{Log}_2(f/f_1) = 0.830075$

7°.
Intervalle entre
la note et la suivante

do	*4 c*
réb	*1 c*
do#	*4 c*
ré	*4 c*
mib	*1 c*

ré#	*4 c*
mi	*4 c*
fa	*4 c*
solb	*1 c*
fa#	*4 c*
sol	*4 c*
lab	*1 c*
sol#	*4 c*
la	*4 c*
sib	*1 c*
la#	*4 c*
si	*4 c*

4
MINES-PONTS 85
Epreuve pratique (mathématiques)

Options M,P',TA
Epreuve Pratique

1°) Déterminer la constante A pour que le polynôme
$P_1(x) = A + x^4 (1-x)^4$ soit divisible par x^2+1

A étant choisi, calculer (et en donner des valeurs approchées à 10^{-6} près) les intégrales :

$$J_1 = \int_0^1 \frac{P1(x)}{x^2+1} dx \quad \text{et} \quad I = \int_0^1 \frac{a}{x^2+1} dx$$

2°)

On pose $K_1 = \int_0^1 x^4(1-x)^4\,dx$;

calculer K_1 et vérifier la double inégalité

$J_1 - K_1 < \pi < J_1 - K_1/2$

3°) On désigne par p et q des entiers naturels et on pose

$I_{p,q} = \int_0^1 x^p\,(1-x)^q\,dx$

Trouver, pour $q \geq 1$, une relation entre $I_{p,q}$ et $I_{p+1,q-1}$; en déduire une expression de $I_{p,q}$ au moyen de p!, de q!, et de (p+q+1)! Vérifier , sur cette expression, la valeur trouvée au **2°**

4°) Calculer une valeur approchée à 10^{-6} près de $K_2 = I_{8,8}$; démontrer ladouble inégalité

$$K_2/2 < \int_0^1 \frac{x^8\,(1-x)^8}{x^2+1}\,dx < K^2$$

5°) Montrer qu'il existe un nombre l_2 tel que le polynôme $P_2(x)=4+l_2x^8\,(1-x)^8$ soit divisible par x^2+1
l_2 étant choisi, calculer les coefficient du polynôme quotient et donner unevaleur approchée à 10^{-6} près de l'intégrale :

$$J_2 = \int_0^1 \frac{P_2(x)}{x^2+1}\,dx$$

Démontrer la double inégalité $J_2 - l_2\,K_2/2 < \pi < J_2 - l_2\,K_2$. Peut-on la vérifier à partir des valeurs trouvées de J_2, de K_2, de l_2 et de la valeur connue de π ?

6°) La méthode précédente peut être prolongée en déterminant un nombre l_3

tel que le polynôme $P_3(x) = 4 + l_3\, x^{12}(1-x)^{12}$ soit divisible par x^2+1 et en évaluant l'intégrale $K_3 = I_{12,12}$. On trouverait ainsi un nombre rationnel J_3 qui serait une valeur approchée de π. Donner le signe de $\pi - J_3$ et une évaluation de sa valeur absolue.

SOLUTION

1°)
x^2+1 divise $P_1(x)$ ssi $P_1(i)=0$ et $P_1(-i)=0$

ssi $A+(1-i)^4=0$ et $A+(1+i)^4=0$
ssi $A = 4$

$P_1(x)= 4 + x^4\,(1-x)^4$

Plaçons nous dans le répertoire POLYNOMES

Calculons $(1-x)^4$

```
{ 1 -1 } [ENTER]
4 [ENTER]
PPUIS [ENTER]
```

renvoie

```
{ 1 -4 6 -4 1 }
```

Donc x^4 (1-x)^4 + 4 = 4 + x^4 - 4 x^5 + 6 x^6 - 4 x^7 + x^8
soit

```
{ 4 0 0 0 1 -4 6 -4 1 }
```

Divisons ce polynôme par x^2+1 soit { 1 0 1 }

```
{ 4 0 0 0 1 -4 6 -4 1 }  [ENTER]
{ 1 0 1 } [ENTER]
PDIV [ENTER]
```

renvoie

```
{ 4 0 -4 0 5 -4 1 }
{ 0 0 }
```

Donc $P_1(x)/(x^2+1) = 4 - 4x^2 + 5x^4 - 4x^5 + x^6$

Donc $J_1 = \int_0^1 4 - 4x^2 + 5x^4 - 4x^5 + x^6 \, dx$

Intégrons ce polynôme

```
{ 4 0 -4 0 5 -4 1 }  [ENTER]
PINT [ENTER]
```

renvoie

```
{ 0 4 0 (-4,3) 0 1 (-2,3) (1,7) }
```

Evaluons J_1

```
{ 0 4 0 (-4,3) 0 1 (-2,3) (1,7) }
  1
EVALP [ENTER]
```

renvoie 22 / 7

$J_1 = 22/7 = 3.142857$ à 10^{-6} près

$I = \int_0^1 4 / (x^2+1) = 4 \, [\, \text{Atan } x \,]_0^1 = \pi$

$I = \pi = 3.141592$ à 10^{-6} près

2°)

$K_1 = \int_0^1 x^8 - 4x^7 + 6x^6 - 4x^5 + x^4 \, dx$

En saisissant:

```
{ 0 0 0 0 1 -4 6 -4 1 } [ENTER]
PINT [ENTER]
1 [ENTER]
EVALP [ENTER]
```

On obtient 1/630

Donc $K_1 = 1/630$

Pour $0 \leq x \leq 1 \quad 1 \leq 1 + x^2 \leq 2$
$1/2 \leq 1/(1+x^2) \leq 1$

$$1/2\, K_1 < \int_0^1 x^4 (1-x)^4 / (1+x^2)\, dx < K_1$$

$$1/2\, K_1 < J_1 - I < K_1$$

$$J_1 - K_1 < \pi < J_1 - K_1/2$$

3°)

Intégrons par parties

$$I_{p,q} = [\, x^{p+1}/(p+1)\, (1-x)^q \,]_0^1 + q/(p+1) \int_0^1 x^{p+1} (1-x)^{q-1}\, dx$$

$$I_{p,q} = q/(p+1)\, I_{p+1,q-1}$$

En itérant cette formule

$$I_{p,q} = q!\, /\, (p+1)...(p+q)\, I_{p+q,0}$$

et $$I_{p+q,0} = \int_0^1 x^{p+q}\, dx = 1/(p+q+1)$$

$I_{p,q} = p!\ q!\ /\ (p+q+1)!$

$K_1 = I_{4,4} = 4!\ 4!\ /\ 9! = 1/630$

On retrouve bien le résultat du 2°

4°)

$K_2 = I_{8,8} = 8!\ 8!/17! = 1\ /\ 218790$

Il suffit de faire

```
8 FACT
DUP
*
17 FACT
QDIV
```

On trouve 1 / 218790

$K_2 = 4.10^{-6}$ à 10^{-6} près

Comme précédemment

1/2 x^8 (1-x)^8 ≤ x^8 (1-x)^8 / (x2+1) ≤ x^8 (1-x)^8

$$K_2/2 \leq \int_0^1 x^8\,(1-x)^8\ /\ (x^2+1)\,dx \leq K_2$$

5°)

x^2+1 divise $P_2(x)$ ssi $P_2(i)=0$ et $P_2(-i)=0$
ssi $4 + l2\,(1-i)^8 = 0$ et $4 + l_2\,(1+i)^8 = 0$

ssi $4 + 16\,l_2 = 0$
ssi $l_2 = -1/4$

Calculons $P_2(x)/(x^2+1)$

```
{ 1 -1 } [ENTER]
8 [ENTER]
PPUIS [ENTER]
```

renvoie

```
{ 1 -8 28 -56 70 -56 28 -8 1 }
```

Soit multiplié par x^8

```
{ 0 0 0 0 0 0 0 0 1 -8 28 -56 70 -56 28
-8 1 }
```

Puis

```
(-1,4) [ENTER]
PCONST [ENTER]
```

on obtient

```
{ 0 0 0 0 0 0 0 0 (-1,4) 2 -7 14 (-35,2)
14 -7 2 (-1,4) }
```

Donc $P_2(x)$ est:

```
{ 4 0 0 0 0 0 0 0 (-1,4) 2 -7 14 (-35,2)
14 -7 2 (-1,4) }
```

```
{ 4 0 0 0 0 0 0 0 (-1,4) 2 -7 14 (-35,2)
14 -7 2 (-1,4) } [ENTER]
{ 1 0 1 } [ENTER]
PDIV [ENTER]
```

donne

```
{ 4 0 -4 0 4 0 -4 0 (15,4) 2 (-43,4) 12
(-27,4) 2 (-1,4) }
{ 0 0 }
```

Donc $P_2(x)/(x^2+1) = 4 - 4x^2 + 4x^4 - 4x^6 + 15/4\ x^8$
$+ 2x^9 - 43/4\ x^{10} + 12x^{11} - 27/4\ x^{12}$
$+ 2x^{13} - 1/4\ x^{14}$

Si on intègre

```
{ 4 0 -4 0 4 0 -4 0 (15,4) 2 (-43,4) 12
(-27,4) 2 (-1,4) } [ENTER]
PINT
```

puis évaluation en 1

```
1
EVALP
```

renvoie :

```
47171 / 15015
```

Soit numériquement 3.141592 à 10^{-6} près

$J_2 = I + l_2 \int_0^1 x^8 (1-x)^8 / (1+x^2)\, dx$

$I + l_2 K_2 < J_2 < I + l_2/2\ K_2$

Donc $J_2 - l_2 K_2/2 < \pi < J_2 - l_2 K_2$

$l2\ K_2/2 = K_2/8 < 10^{-6}$
on ne peut donc pas vérifier l'inégalité avec la précision choisie

6°)

On veut comme précédemment $P_3(i)=0$ et $P_3(-i)=0$
donc $4 - 64\ l_3 = 0$ donc $l_3 = 1/16$

On déduit comme précédemment que

$0 < \pi - J_3 < I_{12,12} / 16$
$I12,12 = 12!\ 12! / 25! = 1 / 67603900$

Le calcul s'effectue ainsi :

```
12 FACT
25 QDIV
24 QDIV
23 QDIV
22 QDIV
21 QDIV
```

etc...

```
13 QDIV
```

Donc finalement en faisant

```
16 QDIV
```

$0 < \pi - J_3 < 1 / 1081662400$

Donc $0 < \pi - J_3 < 9{,}3\ 10^{-10}$.

15
Trucs et astuces

1
REMPLACEMENT DES MESSAGES D'ERREURS DE LA HP48

La commande # 764Eh SYSEVAL avec dans la pile :

```
<n> Array of String
```

remplace les messages d'erreurs dont les numéros sont de la forme :

```
# nxx
```

par ceux contenus dans le tableau.

On rétablit les messages normaux par :

```
<n>  # 76AEh SYSEVAL
```

Attention !
Pour les messages :

```
# 2xx
```

il faut en plus un :

```
<2h> # 7709h SYSEVAL
```

sinon les commandes ne sont plus utilisables à partir de la ligne de commande.

2
MENUS SUR LA HP48

La commande MENU peut prendre comme argument un numéro de bibliothèque. Dans ce cas elle affiche la liste de ses commandes.

Par exemple :

```
1792 MENU
```

Il s'agit de la bibliothèque utilisée pour interpréter la ligne de commande.

En plus des noms et des chaînes de caractères, les libellés des menus utilisateurs peuvent être :

- des caractères
- des GROB 21x8
- des programmes commençant par l'adresse 40788h

Dans le cas d'un programme , celui-ci est évalué et doit renvoyer dans la pile un objet de l'un des types précédents.

Par exemple :

```
"FB920F7" ASS
{ GROB 21 8
000000CCCCCCCCCCCC333333333333CCCCCCCCCC
CC333333 }
« " CHOIX ? " DOERR »
"D9D2088704AFBC0A60D0C6150B2130" ASS { }
+ « TIME » +
# 40788h SYS « "MENU UTILISATEUR" 1 DISP
"TEST" » 2 →PRG
"COUCOU" 2 →LIST
4 →LIST
MENU
```

1ère touche : caractère *"rectangle gris"*
2ème touche : Affiche "CHOIX ?"
3ème touche : renvoieTIME
4ème touche : place COUCOU sur la ligne de commande

3
LES PORTS DE LA HP48*sx*

Si vous avez des cartes d'extensions sur votre HP48sx, et si le port est fermé, il devient impossible de rappeler certains fichiers, et en particulier les bibliothèques.

Ce problème a été résolu, en sachant que de manière générale le port 1 commence en mémoire à l'adresse # 80000h et le port 2 à l'adresse # C0000h .

Si vous avez une carte dans le port 1, par exemple, saisissez :

```
# 80000h
SYSEVAL
```

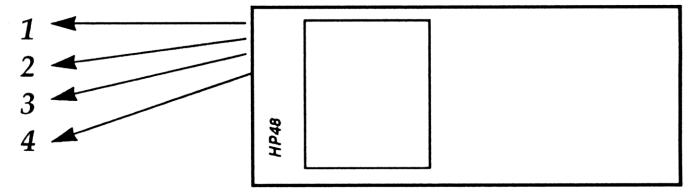
et le premier objet du port est rappelé.

Pour les objets suivants, il suffit de rajouter les tailles des programmes, multipliées par 2, au fur et à mesure.
Le programme ci-dessous rappelle tous les objets des ports :

```
« → p « IF p 1 SAME THEN # 80000h ELSE #
C0000h END
1 p PVARS DROP SIZE IF DUP 0 > THEN FOR
I DUP SYSEVAL DUP
BYTES SWAP DROP 2 * ROT + NEXT DROP ELSE
DROP2 "Port vide"
END »
»
```

ATTENTION : Si vous avez une carte fusionnée (MERGE) dans le port 2, et une carte non fusionnée dans le port 1, l'adresse du port 1 sera en #C0000h au lieu de #80000h. De plus une carte fusionnée n'est plus accessible par le moyen décrit ci-dessus.

4
LE PORT SERIE DE VOTRE HP48



Pour une connexion d'une HP48 à un IBM Compatible PC avec une prise série 9 Broches :

Relier :	*Protection*	*1 à la prise (partie métallique)*
	Transmet	*2 à 2*
	Reçoit	*3 à 3*
	Terre	*4 à 5*

avec une prise série 25 Broches :

Relier :	*Protection*	*1 à 1*
	Transmet	*2 à 3*
	Reçoit	*3 à 2*
	Terre	*4 à 7*

5
LES PORTS DE LA HP48gx

Comme la HP48sx, la HP48gx peut accueillir deux cartes d'extension. Le premier port de la HP48gx est semblable à ceux de la HP48sx : il contient des cartes de 32 ou de 128Ko. Cependant, le deuxième port a une structure différente car il permet d'adresser jusqu'à 4 méga-octets de mémoire (RAM ou ROM). Ce deuxième port correspond en fait à

32 ports du type de ceux de la HP48sx !

Comment accéder aux objets du port 1 ?

De manière similaire à ce qui se passe pour la HP48sx, le programme suivant rappelera tous les objets du port 1 :

```
« # C0000h 1 1 PVARS DROP SIZE IF DUP 0
> THEN FOR I DUP SYSEVAL DUP
BYTES SWAP DROP 2 * ROT + NEXT DROP ELSE
DROP2 "Port vide" END »
```

En effet, le port 1 se trouve à l'adresse #C0000h en mémoire.

Comment accéder aux objets du port d'extension de 4Mo ?

En fait, la HP48gx procède à un adressage virtuel sur 10 quartets. Ces objets qui lui servent d'adresse ont pour entête "02BAA".

Le petit programme suivant récupère les données sur le port n seule entrée du programme.

Il remplace le programme précédent, dans le cas où le port est le port n° 1.

Son checksum est : B355h (304 octets)

```
« → n « IF n 1 ≤ THEN #0 1 'n' STO ELSE
#7073Fh n 1 - 11 * + END « DASS 11 15
SUB » → D « D EVAL #C0000h 1 n PVARS
DROP SIZE FOR I DUP D EVAL 3 PICK +
"AAB20" SWAP + ASS #715B1h SYSEVAL DUP 4
ROLLD BYTES SWAP DROP 2 * + NEXT DROP2 »
» »
```

6
LA COMMANDE PKT DE VOTRE HP48

Voici quelques arguments que la commande PKT accepte :

```
"G" "F"      Finish
"G" "L"      Logout
"G" "D"      Remote Directory
"C"          Remote host
```

7
UN PROGRAMME PEEK SUR VOTRE HP48

```
"D9D 20B B69 1B9 691 CCD 200 300 014 713
706 179 147 137 15B F13 715 9F0 713 714
216 480 8CB 213 0"
```

Tapez la ligne ci-dessus SANS ESPACES NI SAUTS DE LIGNE
entrez :

```
DUP
```

puis :

```
BYTES
```

Le résultat doit être :

```
# FBE5h  ( ou 64485d )
78
```

Si ce n'est pas le cas, vérifiez la chaîne. Entrez :

```
DROP2
```

puis :

```
ASS
```

Apparaît alors :

```
B→R R→B Code
```

Saisissez :

```
'PEEK' STO
```

Par exemple :

```
#0 PEEK
```

doit renvoyer :

```
#8001FDAD801B9632h
```

8
LES REPERTOIRES CACHES DE LA HP48

La HP possède déjà un répertoire caché, et on peut aisément en créer d'autres.
Les répertoires cachés de la HP48 ont pour nom: ''
La difficulté vient donc de la création de ce nom vide.
L'adresse # 15781h fait appel à ce nom.

1
VISITE DU REPERTOIRE CACHE

Entrez

```
# 15781h SYSEVAL
```

Vous voyez apparaître **Alarms UserKeys UserKeys.CRC** dans le menu.
Vous voyez en haut de l'écran **HOME**, mais vous n'êtes plus dans **HOME**. La preuve : Appuyez sur [VAR].

Une autre preuve :
Entrez

```
PATH
```

{ HOME } apparaît

Vous ne voyez qu'un seul nom, et pourtant :
Entrez

```
SIZE
```

Oui, vous êtes dans un sous répertoire!
Entrez

```
PATH 2
```

GET : '' apparaît. C'est le nom du répertoire caché.
Il est très dangereux de modifier ou détruire les trois fichiers se trouvant dans ce répertoire.

2
CREER UN REPERTOIRE CACHE

Comme il en existe déjà un, on ne peut pas créer de répertoire caché dans le répertoire HOME.

Entrez

```
HOME.
```

Puis

```
'ESSAI' CRDIR
```

et

```
ESSAI
```

Nous allons créer dans ce sous répertoire, un répertoire caché

Entrez :

```
# 15777h SYSEVAL
```

puis :

```
CRDIR
```

votre répertoire CACHE est créé.
Pour y accéder, entrez :

```
# 15777h SYSEVAL EVAL
```

Les effets secondaires :

Tous les programmes qui se trouvaient déjà dans le répertoire où vous créez votre répertoire caché, deviennent invisibles. Ils ne sont pas détruits et ils reapparaitront dès que vous aurez fait dans le répertoire où vous avez créé le répertoire caché :

```
# 15777h SYSEVAL PURGE
```

9
LA COMMANDE WSLOG

La commande WSLOG est une sorte de mouchard qui indique les 4 dernières opérations du système qui ont été faites.

Chaque ligne est déterminée par un code, la date et l'heure de l'opération.

Code 0 : [ON]-[SPC] a été fait pour mettre le calculateur en mode "comma" puis [ON] a été refait pour réveiller le calculateur.

Code 1 : Le système a détecté que le voltage était faible au niveau du contact des piles.Le calculateur a donc mis le calculateur en mode "comma".

Code 2 : Une erreur Infra-rouge "hard" s'est produite.

Code 3 : L'execution est passée par l'adresse 0.

Code 4 : Le système de l'horloge a été corrompu.

Code 5 : Lors du rallumage, le calculateur a constaté des changements dans les données des cartes.

Code 6 : Inutilisé.

Code 7 : Le mot CMOS en RAM a été corrompu. (test de corruption potentiel de la RAM).

Code 8 : Une anomalie a été constatée mettant en jeu la configuration du matériel.

Code 9 : La liste d'alarme a été corrompue.

Code A : Inutilisé.

Code B : La carte d'extension a été retirée (ou remise).

Code C : Redémarrage "hard" (du à une décharge éléctrostatique ou au bouton de l'utilisateur).

Notez que ce bouton se situe sous la calculatrice, sous le pied en caoutchouc supérieur gauche.

Code D : Une adresse RPL attendue n'a pas été trouvée.
Code E : La table de configuration a été corrompue.

Code F : La carte de système RAM a été retirée.
Notez que le fait de saisir [ON]-[SPC] efface les données de WSLOG...
Il suffit ensuite d'entrer [ON] pour rallumer le calculateur.

10
FONCTIONNEMENT DU CALCULATEUR

Le microprocesseur qui équipe les HP 28 et 48 est un microprocesseur Saturn à 4 bits. Il utilise des adresses codées sur 5 quartets.

La mémoire de ces calculateurs est compliquée...
Par exemple, le constructeur annonce pour une HP48sx :

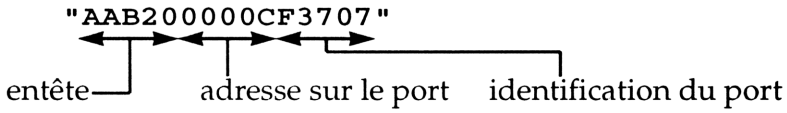
```
ROM          256k       524288
RAM           32k       65536
port 1       128k       262144
port 2       128k       262144

Total                   1114112
```

Or le microprocesseur permet un adressage de **00000** à **FFFFF** soit 1048575. Il reste 32k à loger !!! Pour ce faire, la HP48sx et les HP48 g et gx utilisent une **mémoire cachée**. Lorsque vous utilisez votre calculateur de façon standard, le mode d'adressage est le suivant sur une HP48 sx

```
00000.6FFFF      ROM
70000.7FFFF      RAM
80000.BFFFF      port 1
C0000.FFFFF      port 2
```

Cependant 32k de ROM (de **70000** à **7FFFF**) sont recouverts. C'est ce que l'on appelle la **ROM cachée**.

Pour les HP48 g et gx, elles possèdent 512 Ko de ROM chacune. Ceci prend donc tout l'espace de mémoire ! Mais comme précédemment, la ROM est cachée. Mais sur la HP48gx, le système de codage est différent :
L'adresse "interne" d'un objet est codée sur 10 quartets. La structure du codage est la suivante :



L'identification du port est : 00000 pour le port 1
F3707 pour le port 2
Pour chaque port suivant elle est de #Bh en plus de #7073Fh. (codé à l'envers)

Le scanner de la HP48 s et sx :

Comme vous le savez peut être, la HP48s/sx possède un scanner interne.
Pour y accéder, tapez ON-D simultanément puis backspace (⇐)
Vous êtes alors dans le *SCANNER*.
Attention, ce scanner permet de lire la mémoire, mais aussi d'y écrire.
Appuyez sur ON-C pour sortir du scanner.

Le Scanner fonctionne avec les touches suivantes:

+,-	*augmente et diminue l'adresse de*	*1h*
***,/**	*augmente et diminue l'adresse de*	*100h*
up,down	*augmente et diminue l'adresse de*	*1000h*
ENTER	*va à l'adresse # 100h*	
NEG	*va à l'adresse # F000Ah*	
EEX	*va à l'adresse # 80000h (port 1)*	
DEL	*va à l'adresse # C0000h (port 2)*	
INV	*va à l'adresse # F0AD4h*	

EVAL *effectue un GOSUB à l'adresse ou l'on est.*
0..9 | *poke ce code hexadécimal et*
A..F | *incrémente l'adresse de 1.*

Cependant, le mode d'adressage de la mémoire n'est pas le même pour le scanner que celui précédemment décrit.

Le mode d'adressage du scanner est le suivant :

```
00000.7FFFF   ROM
80000.BFFFF   port 1
C0000.EFFFF   début port 2
F0000.FFFFF   RAM
```

Il apparait ici qu'une partie du port 2 est invisible (remarque : on peut toujours intervertir les deux cartes à moins qu'elles soient fusionnées).

11
QUELLE MACHINE AVEZ-VOUS ?

Sur une HP48 s ou sx :
Placez vous dans le scanner, par ON-D puis (⇐).
Taper ensuite EVAL. Vous verrez alors apparaître un message du type :

```
Version HP48-D
Copyright HP 1989
```

La lettre (ici D) correspond à la version de votre 48s/sx.
Il y a 7 versions, de A à E puis I et J. La version A est la plus imparfaite et la version J est la plus récente.

Sur une HP48 g ou gx :

Saisissez au clavier la commande

```
VERSION
```

Vous verrez apparaitre aussi :

```
"Version HP48-M"
"Copyright HP 1993"
```

La lettre (M ici) correspond à la version de votre calculateur. K est la 1ère version de la HP48g et L la 1ère version de la HP48gx.

POUR EN SAVOIR PLUS

Quel que soit votre matériel HP (calculateur,imprimante) retournez le. Vous y trouverez un numéro.

Il est de la forme:

```
YY WW C NNNNN
```

Ajouter 1960 à **YY** : C'est l'année de fabrication de votre machine. **WW** est la semaine de fabrication de votre machine (WW est entre 1 et 52)

C est le pays de fabrication de votre machine.
Parmi les plus répandus:

```
A : USA
S : Singapour
```

NNNNN est la nnnnn_ième machine à avoir été fabriquée cette semaine là.

12
Questions-réponses

Question : Pourquoi mon écran se gèle-t-il un instant de temps en temps ?
Réponse : Le calculateur doit de temps en temps éxecuter un "garbage collection" pour libérer la mémoire non utilisée.

Il regarde la mémoire, et ce temps qu'il prend provoque un léger gel de l'écran.

Question
Pourquoi est-ce que (1/3)*3 est égal à 0.99999999999 ?
Réponse
C'est dû à la manière utilisée par le calculateur pour représenter les rationnels. Il les représente comme des réels. Le nombre de décimal étant fini, cela provoque une erreur d'arrondis. Les programmes du chapitre 2 sont là pour compenser ce défaut.

Question
Je veux que p soit une valeur numérique et non un symbole. Que dois-je faire ?
Réponse
Vous utiliser le mode de résultat symbolique. Pour changer ce mode, rendez-vous dans le menu [MODES], et presser la touche [SYM].

Question
Si j'additionne 34°F et 11°F, j'obtiens 504.67 °F. Pourquoi est-ce que je n'obtiens pas 47 °F ?
Réponse
Cela n'a pas de sens d'additionner deux températures : si l'on mélange deux bols d'eau, l'un à 20° et l'autre à 30°, on n'obtiendra pas un mélange à 50°.

Question
Ma carte d'éxtension RAM était branchée dans mon calculateur quand j'ai changé la pile de la carte. Je ne trouve plus rien dessus. Que s'est il passé ?
Réponse
Vous avez oublié d'allumer votre calculateur avant de changer les piles de la carte. Le calculateur n'alimente la carte que si celui-ci est allumé !

Question
Pourquoi est-ce que j'obtiens "Invalid Data Card" lorsque j'allume mon calculateur avec une carte d'éxtension mémoire ?

Réponse : Ce message apparait lorsque vous introduisez une nouvelle carte RAM dans votre calculateur. Cela signifie juste qu'il n'y a pas de données valides sur la carte. Si ce message apparait dans d'autres conditions, cela signfie par exemple que vous avez perdu vos données sur la carte.

Question : J'ai l'impression que ma HP48sx/gx met plus de temps à s'éteindre et à s'allumer. Que se passe-t-il ?

Réponse : Cela est dû d'habitude aux cartes d'extension et aux bibliothèques. Lorsque vous allumez votre calculateur, il vérifie sa RAM (plus il y en a, plus ça prend de temps) et vérifie si les bibliothèques doivent êtres initialisées.

Question : Pourquoi est-ce que le port Infra-Rouge ne fonctionne pas à plus de quelques centimètres ?

Réponse : La capacité de reception des diodes infra-rouges à été réduite pour éviter certains phénomènes lors des concours et des examens !

Question : Pourquoi est-ce que je ne pas accélerer ma HP48, alors que je pouvais le faire avec ma HP28s.

Réponse : C'est impossible car la HP48 va déjà au maximum de sa vitesse. Sa gestion de la mémoire et de son affichage prennent plus de temps.

13
RETOUR AU MICROPROCESSEUR

Registres saturniens

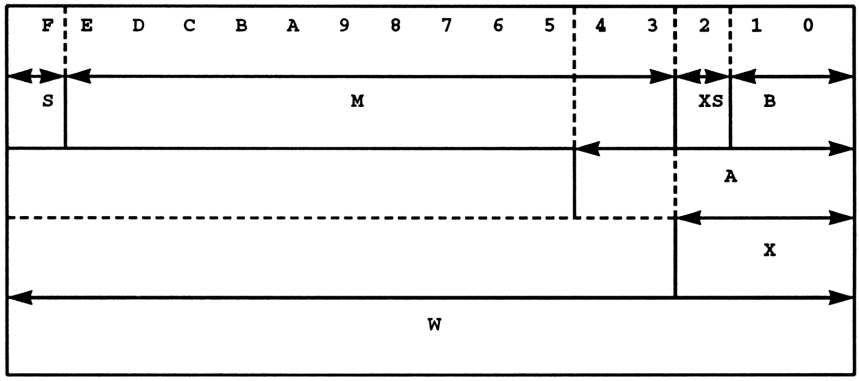
Les registres (mémoires internes) sont les suivants:

A	pointeur d'objet courant	sur 16 quartets
B	pointeur du Return Stack RSTK	sur 16 quartets
C	usage général	sur 16 quartets
D	Espace restant entre la pile et la Return Stack (D * 5 quartés libres)	
R0	*registres de sauvegarde de A,C sur 16 quartets*	
R1		
R2		
R3		
R4		
D0	pointeur sur la suite du programme effectué 5 quartets	
D1	pointeur sur les données de la pile 5 quartets	
PC	adresse de l'instruction en cours 1 quartet	
IN	gestion du clavier... 4 quartets	
OUT	3 quartets	
CARRY	bit de retenue 1 bit	
HST	bits MP SB SR XM du système 4 bits	
STATUS	16 drapeaux de 0 à F 16 bits	
P	numéro de champ 1 quartet	

Il y a aussi RSTK (return Stack) pile de 8 niveaux qui garde les adresses de retour lors d'appel à des sous programmes.

STRUCTURE DES REGISTRES A, B, C, D, R0, R1, R2, R3, R 4

Chaque registre est décomposé en champs.
Les dénominations correspondent aux cas suivants:



champ P : quartet n°P
champ WP : quartets de 0 à P

Attention : toutes les données sont inversées en mémoire:
12345 sera stocké 54321.

LES DIFFERENTS TYPES

En-tête	*Objet*	*Type*	*Exemple*
02911	Binaire Système	20	<1h>
02933	Nombre Réel	0	1.25
02955	Réel Long	21	Long Real
02977	Nombre Complexe	1	(1,1)
0299D	Complexe Long	22	Long Complex
029BF	Caractère	24	Character
029E8	Matrice Réelle	3	[[1][2]]
029E8	Matrice Complexe	4	[[(1,1)]]
02A0A	Matrice Liée	23	Linked Array
02A2C	Chaîne	2	"BONJOUR"
02A4E	Entier Binaire	10	# 123h

02A74	Liste	5	{ A B C }
02A96	Répertoire	15	DIR END
02AB8	Symbolique algébrique	9	'COS(X)'
02ADA	Unité	13	1_m
02AFC	Objet Signé	12	A : 1
02B1E	Graphique	11	GROB 1 1 00
02B40	Bibliothèque	16	Library 268:
02B62	Sauvegarde	17	Backup
02B88	Données de Bibliothèque	26	Library Data
02BAA	Adresse sur HP48gx	27	External
02D9D	Programme	8	« CLLCD »
02DCC	Code	25	Code
02E48	Nom Global	6	'ABC'
02E6D	Nom Local	7	'ABC'
02E92	Nom XLIB	14	XLIB 792 1

Chaque objet est identifié en mémoire par son en-tête.
Nous allons étudier la structure des objets dans le cas de la HP48.

BINAIRE SYSTEME **02911**

Structure :

```
<En-tête><Données>
```

Exemple pour créer le binaire système <12345h>

```
"1192054321" ASS
```

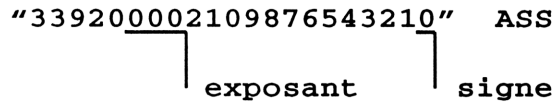
N'oubliez pas qu'il faut tout inverser.

NOMBRE REEL **02933**

Structure :

```
<En-tête><Données>
```

Les données sont sous la forme BCD 12.
Par exemple 1.23456789012

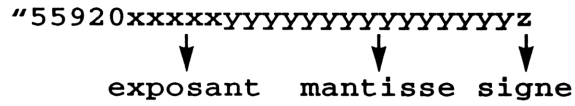


REEL LONG 02955

Structure :

```
<En-tête><Données>
```

Les données sont sous la forme BCD 15



NOMBRE COMPLEXE 02977

Structure :

```
<En-tête><Partie Réelle><Partie Immaginaire>
```

Les parties réelles et imaginaires sont sous la forme BCD 12

LONG COMPLEXE 0299D

Structure :

```
<En-tête><Partie Réelle><Partie Immaginaire>
```

Les parties réelles et imaginaires sont sous la forme BCD 15

CARACTERE 029BF

Structure :

```
<En-tête><Données>
```

Par exemple le caractère 7Fh = 127d

```
"FB920F7" ASS
```

MATRICE 029E8

Structure :

```
<En-tête><Taille><En-tête objet><nb de dim>
<dim 1>..<dim n>
<objet 1>...<objet m>
```

On peut créer des matrices de n'importe quoi:Il suffit de modifier l'en-tête d'objet.
Par exemple, les messages d'erreurs sont des Matrices de Chaînes. Ces matrices ne sont pas visibles sur la pile et ne peuvent pas être éditées par Matrix Editor.

MATRICE LIEE 02A0A

Ces matrices ne semblent pas être utilisées par la machine. Leur format est le suivant :

```
<En-tête><Taille><Type du
tableau><Nombre de
dimensions><Dim 1>...<Dim n><Offset 1>
...<Offset (dim 1)*...*(dim n)><Objet 1>
...<Objet n>
```

`<Offset i>` est un pointeur (relatif) sur un des objets situés après. Ceci permet de ne stocker qu'une fois des objets qui se trouvent à plusieurs endroits du tableau. Ainsi pour une matrice identité complexe d'ordre 5, on gagne (32-5)*25-64=611 quartets !

CHAINE 02A2C

Structure :

```
<En-tête><Taille><Données>
```

La taille est le nombre de quartets pris par les données+5
Par exemple pour la chaîne "ABC" sera codée

```
C2A20 B0000 142434
```

Bh = 11 d car 142434 représente 6 quartets + 5 = 11

LISTE 02A74

Structure :

```
<En-tête><objet 1>...<objet n><En-tête fin>
```

L'en-tête de fin est l'adresse 0312B pour la HP48

Exemple :

```
"47A20B2130" ASS
```

renvoie

```
{ }
```

REPERTOIRE 02A96

Structure :

```
<En-tête><Attachements><Decalage 1><00000>
<nom n><objet n><décalage n> .....
<nom 2><objet 2><décalage 2>
<nom 1><objet 1>
```

Attachements a une taille de 3 quartets. Il code le fait que le répertoire est on non le répertoire HOME.

<Décalage i> est le décalage du début de l'adresse du décalage jusqu'au début du nom de l'objet.

<nom i> est le nom de l'objet.
<objet i> est l'objet.

Si le répertoire est HOME : <Attachements> contient le nombre de bibliothèques attachées à HOME, en particulier la bibliothèque 2 et la bibliothèque 1792. Si le répertoire est un sous-répertoire : <Attachements> contient le nombre de bibliothèques attachées ou bien 7FF si aucune n'est attachée.

SYMBOLIQUE / ALGEBRIQUE 02AB8

Structure :

```
<En-tête> ... <Fin>

<Fin>
```

est

```
0312B
```

Exemple

```
"8BA2084E2010859C2A290DA1B2130"  ASS
```

renvoie

```
'X-1'
```

car

```
8BA20   En-tête
84E20 10 85 'X'
9C2A2 1
90DA1 -
B2130   Fin
```

UNITES 02ADA

Structure :

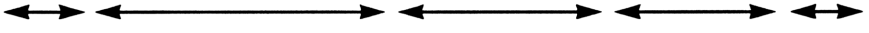
```
<En-tête><Nombre><unité 1>...<unité n>
<opération 1> .. <opération n>
<Fin>
```

<Fin> est 0312B

Exemple :

1_m/s est codé :

```
"ADA20 9C2A2 C2A20 70000 D6 C2A20 70000 37 86B01 68B01 B2130"
```



```
En-tête  1        "m"           "s"       Indicateurs  Fin
```

OBJETS SIGNES 02AFC

Structure :

```
<En-tête><Signature><objet>
```

GRAPHIQUE 02B1E

Structure :

```
<En-tête><Longueur><nb lignes><nb
colonnes><données>
```

<Taille> est le nombre de quartets des données + 15.

BIBLIOTHEQUE 02B40

Structure :

```
<En-tête><Nom bibliothèque>
<numéro de bibliothèque>
<Décalage de Hash Table>
<Décalage de Table des messages>
<Décalage de table de liaison>
<Décalage du config><Données>
<Checksum>
```

<Numéro de bibliothèque> Codé sur 3 quartets
<Données> = Hash Table, Table des messages, Table des liaisons, code de configuration et le reste
<Checksum> codé sur 4 quartets

BACKUP 02B62

Structure :

```
<En-tête><Taille><Nom><Données>
```

DONNEES DE BIBLIOTHEQUE 02B88

Structure :

```
<En-tête><Taille><Identification>
<Données><Fin>
```

PROGRAMME 02D9D

Structure :

```
<En-tête>...<Fin>
```

La fin est 0312B

Exemple

```
9D920   En-tête
E1632   «
858A1   CLLCD
93632   »
B2130   Fin

"9D920E1632858A193632B2130" ASS
```

renvoie

```
« CLLCD »
```

CODE 02DCC

Structure :

```
<En-tête><Taille><Codes>
```

Taille=Taille de <Codes> + 5

NOM GLOBAL 02E48

Structure :

```
<En-tête><nom>
```

NOM LOCAL 02E6D

Structure :

```
<En-tête><nom>
```

NOM XLIB 02E92

Structure :

```
<En-tête><numéro de bibliothèque><numéro
d'objet>
```

Par exemple :

sur un calculateur HP48 s/sx :

```
"29E20C01000"
 ASS
```

renvoie

```
MINEHUNT
```

si vous possédez HPSOLVE,

```
XLIB 268 0
```

dans le cas contraire.

sur un calculateur HP48 g/gx :

```
"29E20BA0570"
ASS
```

renvoie

```
MINEHUNT
```

car Minehunt est fourni en standard sur les HP48 g et gx.

Sur HP48GX uniquement

ADRESSE VIRTUELLE 02BAA

Structure :

```
<En-tête><Adresse de l'objet><Code port>
```

L'adresse de l'objet est celle qu'il possède ou possèderait s'il était dans le port n°1. C'est-à-dire, c'est une adresse comprise entre #C0000h et #FFFFFh.
Comme l'adresse de l'objet, le code du port est codé sur 5 quartets.
L'adresse codant le port 1 est : #0
L'adresse codant le port 2 est : #7073Fh
L'adresse codant le port n (n ≥2) est : #7073Fh + #Bh*(n-1)

Exemple :

Le premier objet du port 1, s'il existe, est appelé par

```
"AAB200000C00000" ASS
```

Le premier objet du port 2, s'il existe, est appelé par

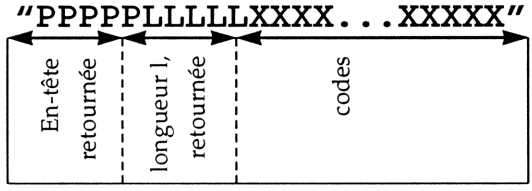
```
"AAB200000CF3707" ASS
```

N'oubliez pas qu'il faut inverser les adresses dans le codage.

16
Programmation en assembleur

INTRODUCTION

Pour programmer en assembleur sur HP48, on utilise des objets 'code assembleur' dont l'en-tête est 02DCC (l'objet apparait dans la pile sous la forme 'Code'). On les crée grâce à une chaîne de codes hexadécimaux que l'on 'compile' avec le programme 'ASS'.
La chaîne doit avoir le format suivant :



Par exemple, le programme suivant, qui n'a aucun effet (un programme assembleur se termine en général par ces trois instructions) :

```
142          A=DAT0  A
164          D0=D0+  5
808C         PC=(A)
```

s'obtient en tapant:

```
"CCD20F0000142164808C"
```

puis en exécutant 'ASS'
On obtient:

```
Code
```

Cet objet peut être rangé dans une variable ou évalué comme un programme RPL.

Ici, l'évaluation ne produit aucun effet.

Pour éviter de 'planter' la machine, un programme assembleur doit respecter certaines règles:

• Les registres ayant une signification particulière (A,B,D,D0,D1) doivent avoir un contenu valide à la sortie, et, s'il est modifié, le registre P doit être remis à 0.

Si on a besoin d'utiliser ces registres, il faut d'abord sauver leur contenu. Pour cela, on peut soit utiliser les registres de sauvegarde Rn, soit la pile RSTK, soit des routines de sauvegarde en ROM : on sauve les registres par GOSBVL 0679B sur HP48 (qui modifient C et D0), puis on les récupère à la fin par GOSBVL 067D2 sur HP48.

• Le programme doit se terminer en provoquant l'évaluation de l'objet RPL suivant. Ceci peut se faire en terminant le programme par les instructions:

`A=DAT0 A`	chargement de l'objet suivant dans A
`D0=D0+ 5`	D0 pointe sur l'objet dont le programme suivant devra lancer l'exécution.
`PC=(A)`	évaluation de l'objet suivant.

On peut aussi sauter à une routine en ROM qui s'en chargera. Par exemple, si à la fin d'un programme, on a obtenu dans A l'adresse d'un objet que l'on veut placer dans la pile, on peut terminer par 'GOVLNG 02911' .

Enfin, on peut terminer par un saut à un autre objet RPL. Par exemple, si on veut terminer un programme en plaçant 'π' dans la pile, on peut finir par:

```
LCHEX     1AABD
A=C       A
PC=(A)
```

où 1AABD est l'adresse de π.

Un programme a souvent besoin de prendre des arguments dans la pile. Ceci se fait par l'intermédiaire du registre D1. A l'adresse D1, on trouve l'adresse de l'objet du niveau 1, à l'adresse D1+5, celle de l'objet du niveau 2, etc..

Par exemple, un programme servant à échanger les objets des niveaux 1 et 3 (équivalent de ROT ROT SWAP) est par exemple:

```
143       A=DAT1   A      A=adr obj 1
179       D1=D1+   10     on pointe sur le
                           niveau 3
147       C=DAT1   A      C=adr obj 3
141       DAT1=A   A      on remplace par
                           l'objet 1
1C9       D1=D1-   10     on pointe sur le
                           niveau 1
145       DAT1=C   A      on remplace par
                           l'objet 3
142       A=DAT0   A
164       D0=D0+5           fin
808C      PC=(A)
```

Sur HP 48, on l'obtient par:

```
"CCD20120001431791471411C9145142164808C"
ASS
```

Attention, ce programme suppose qu'il y a au moins trois objets dans la pile. Il est dangereux de l'exécuter s'ils n'y sont pas.
Ceci étant dit,voyons à travers des examples comment on program-

me en assembleur. Et voici tout d'abord la liste des instructions de l'assembleur : elle vous permettront de comprendre la suite de notre propos.

LES MNEMONIQUES DE L'ASSEMBLEUR SATURN

Les lettres a,b,c utilisées dans les codes représentent les champs:

a	b	c	champ
0	8	0	P
1	9	1	WP
2	A	2	XS
3	B	3	X
4	C	4	S
5	D	5	M
6	E	6	B
7	F	7	W
-	-	F	A

La lettre r sert à la fois au codage des instructions et des registres. Quand elle apparait dans un code, on trouve le (ou les) registres correspondants dans le mnémonique grâce au tableau suivant:

r(code)	r	r1	r2
0,4,8,C	A	A	B
1,5,9,D	B	B	C
2,6,A,E	C	C	A
3,7,B	D	D	C

Par exemple, le code 818F94, représenté par 818crx, signifie 'B=B-5 A', car c=F donc le champ est A, r≥8 donc l'instruction est 'r=r-(x+1)', et r=9 donc le registre est B.
Codage d'un registre de sauvegarde:

n	registre
0,8	R0
1,9	R1
2,A	R2
3,B	R3
4,C	R4

Enfin, dans les explications, la notation r.c signifie 'champ c du registre r'.

Voici la liste des codes, et les mnémoniques correspondants.

00	RTNSXM	retour de sous-programme et XM=1
01	RTN	retour de sous-programme
02	RTNSC	retour de sous-programme et CARRY=1
03	RTNCC	retour de sous-programme et CARRY=0
04	SETHEX	mode hexadécimal
05	SETDEC	mode décimal
06	RSTK=C	empile C.A
07	C=RSTK	dépile C.A
08	CLRST	vide la pile
09	C=ST	place ST dans C.X
0A	ST=C	place C.X dans ST
0B	CSTEX	échange ST et C.X
0C	P=P+1	Incrémente P
0D	P=P-1	Décrémente P
0Ecr	0≤r≤3:	r1=r1&r2 'et' logique
	4≤r≤7:	r2=r2&r1
	8≤r≤B:	r1=r1!r2 'ou' logique
	c≤r≤F:	r2=r2!r1

0F	RTI		retour d'interruption
10n	0≤n≤4	Rn=A	sauve A.W dans Rn
	8≤n≤C	Rn=C	sauve C.W dans Rn
11n	0≤n≤4	A=Rn	récupère A.W
	8≤n≤C	C=Rn	récupère C.W
12n	0≤n≤4	ARnEX	échange A.W et Rn
	8≤n≤C	CRnEX	échange C.W et Rn
130	D0=A		place A.A dans D0
131	D1=A		
132	AD0EX		échange A.A et D0
133	AD1EX		
134	D0=C		place C.A dans D0
135	D1=C		
136	CD0EX		échange C.A et D0
137	CD1EX		
138	D0=AS		place les quartets 0 à 3 de A ds D0
139	D1=AS		
13A	AD0XS		échange les quartets 0 à 3 de A et D0
13B	AD1XS		
13C	D0=CS		place les quartets 0 à 3 de C dans D0
13D	D1=CS		
13E	CD0XS		échange les quartets 0 à 3 de C etD0
13F	CD1XS		
14x	DAT0=A c		écrit A.c à l'adresse contenue dans D0
	DAT1=A c		
	A=DAT0 c		place les quartets situés à l'adresse D0 dans A.c
	A=DAT1 c		
	DAT0=C c		écrit C.c à l'adresse contenue ds D0
	DAT1=C c		

	C=DAT0 c	place les quartets situés à l'adresse D0 dans C.c
	C=DAT1 c	
avec	*0≤x≤7 : c=champ A*	
	8≤x≤F : c=champ B	
15xy	*idem* avec:	
	0≤x≤7: c=champ codé par y (de type 'a')	
	8≤y≤F: c=quartets 0 à y	
16x	D0=D0+ x+1	ajoute x+1 à D0
17x	D1=D1+ x+1	
18x	D0=D0- x+1	soustrait x+1 à D0
1Cx	D1=D1- x+1	
19xy	D0= yx	place yx dans les quartets 0 à 1 de D0
1Axyzt	D0= tzyx	place tzyx dans les quartets 0 à 3 de D0
1Bxyztu	D0=utzyx	place utzyx dans D0
1Dxy	D1= yx	
1Exyzt	D1=tzyx	
1Fxyztu	D1=utzyx	
2x	P= x	place x dans P
3x...	LCHEX...	charge les x+1 quartets suivants dans les quartets P, P+1, ..., P+x de C
400	RTNC	retour de sous-programme si CARRY=1
4xy	GOC yx	saut si CARRY=1
500	RTNNC	retour de sous-programme si CARRY=0
5xy	GONCyx	saut si CARRY=0
6xyz	GOTO zyx	saut
7xyz	GOSUB zyx	saut à un sous-programme

800	OUT=CS	place C.0 dans OUTPUT
801	OUT=C	place C.X dans OUTPUT
802	A=IN	place INPUT dans A
803	C=IN	place INPUT dans C
804	UNCNFG	??? utilisés pour déplacer la RAM
805	CONFIG	??? sur la HP48
806	C=ID	???
807	SHUTDN	??? utilisé lorsqu'on éteint la machine
8080	INTON	autorise les interruptions
80810	RSI	???
8082x..	LAHEX ...	id LCHEX pour le registre A
8083	BUSCB	???
8084x	ABIT=0 x	met à 0 le bit x de A
8085x	ABIT=1 x	met à 1 le bit x de A
8086x	?ABIT=0 x	teste si le bit x de A est à 0
8087x	?ABIT=1 x	teste si le bit x de A est à 1
8088x	CBIT=0 x	
8089x	CBIT=1 x	
808Ax	?CBIT=0 x	
808Bx	?CBIT=1 x	
808C	PC=(A)	place les quartets situés à l'adresse contenue dans A dans le PC
808D	BUSCD	???
808E	PC=(C)	
808F	INTOFF	interdit les interruptions
809		C+P+1place C+P+1 dans C.A
80A	RESET	???
80B	BUSCC	???
80Cx	C=P x	place P dans le quartet x de C
80Dx	P=C x	place le quartet x de C dans P
80E	SREQ ?	???
80Fx	CPEX x	échange P et le quartet x de C
81r	0≤r≤3 rSLC	permutation circulaire à gauche d'un quartet de r.W
	4≤r≤7 rSRC	permutation circulaire à droite

				d'un quartet de r.W
	C≤r≤F	rSRB		décalage à droite d'un bit de r.W (bit sortant dans SB)
818crx	r=r±x+1	c	0≤r≤3 :	+ ajoute x+1 à r.c
			8≤r≤B :	- retranche x+1 à r.c
819cr	rSRB	c	0≤r≤3 :	décalage à droite d'un bit de r.c
81Ac0n		Rn=r	c	0≤n≤4 r=A sauve r.c dans Rn
81Ac1n		r=Rn	c	8≤n≤C r=C récupère r.c dans Rn
81Ac2n		rRnEX	c	échange r.c et Rn
81B2	PC=A			
81B3	PC=C			
81B4	A=PC			
81B5	C=PC			
81B6	APCEX			
81B7	CPCEX			
82x	HST&=not(x)			met à zéro les bits indiqués de HST (ex: x=3 -> XM=0 et SB=0)
831	?XM=0			
832	?SB=0			
834	?SR=0			
838	?MP=0			
84x	ST=0	x		met à 0 le bit x de ST
85x	ST=1	x		met à 1 le bit x de ST
86x	?ST=0	x		teste si le bit x de ST est à 0
87x	?ST=1	x		teste si le bit x de ST est à 1
88x	?P=x			
89x	?P#x			
8Ar	?	<t1>	A	tests
8Br	?	<t2>	A	tests
9ar	?	<t1>	a	tests
9br	?	<t2>	b	tests

r	0-3	4-7	8-B		C-F
<t1>	r1=r2	r1#r2	r=0		r#0
<t2>	r1>r2	r1<r2	r1≥r2	r1≤r2	

8Cxyzt	GOLONG	tzyx	saut
8Dxyztu	GOVLNG	utzyx	saut absolu:
		utzyx	est l'adresse où sauter
8Exyzt	GOSUBL	tzyx	saut à un sous-programme
8Fxyztu	GOSBVL	utzyx	saut à un sous-programme (saut absolu)

Aar	<i1>	a
Abr	<i2>	b
Bar	<i3>	a
Bbr	<i4>	b
Cr	<i1>	A
Dr	<i2>	A
Er	<i3>	A
Fr	<i4>	A

r	0-3	4-7	8-B	C-F
<i1>	r1=r1+r2	r1=r1+r1	r2=r2+r1	r=r-1
<i2>	r=0	r1=r2	r2=r1	r1r2EX
<i3>	r1=r1-r2	r=r+1	r2=r2-r1	r1=r2-r1
<i4>	rSL	rSR	r=-r	r=-r-1

rSL décale d'un quartet vers la gauche
rSR décale d'un quartet vers la droite
r1r2EX échange r1 et r2
r=-r-1 éffectue un 'non' logique

Les instructions de test (dont le nom commence par '?') doivent être

suivies des instructions GOYES ou RTNYES, correspondants à l'action à exécuter dans le cas où le test est positif. RTNYES a pour code 00, et GOYES est codé sur deux quartets par l'offset du saut à éffectuer.

De plus, pour un test positif, CARRY est mis à 1, et pour un test négatif, CARRY est mis à 0.
Les instructions de branchement GOYES, GOC, GONC, GOTO, GOLONG, GOSUB, et GOSUBL correspondent à des saut relatifs: l'adresse où sauter est calculée à partir de la valeur actuelle du PC et d'un décalage codé sur 2,3, ou 4 quartets.
Pour GOYES, GOC, GONC, GOTO, et „GOLONG, le décalage est la différence entre l'adresse à atteindre et l'adresse du premier quartet codant ce décalage.

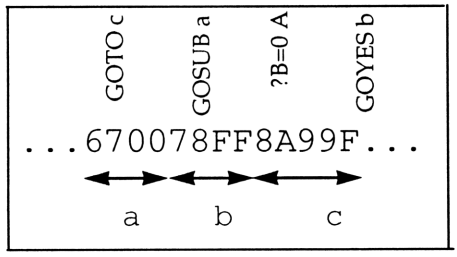
Pour GOSUB et GOSUBL, le décalage est la différence entre l'adresse à atteindre et l'adresse suivant l'instruction de saut.

Dans tous les cas, le décalage est une valeur signée, codée à l'envers.

Par exemple, la séquence:

```
a:      GOTO      c
b:      GOSUB     a
c:      ?B=0  A
        GOYES     b
```

se code :



En effet:

- 'GOTO' a pour code 6xyz, et l'adresse c se situe 7 quartets après l' adresse du x, donc le décalage se code '700'

- 'GOSUB' a pour code 7xyz, et l'adresse a se trouve 8 quartets devant le quartet suivant le gosub (1er code du '?B=0'), donc le décalage est de -8, ce qui donne '8FF' (sur une HP, taper '#0 8 -'. On obtient #FFFFFFFFFFFFFF8h en mode HEX. Il suffit alors de prendre les 3 derniers chiffres et de les retourner)
- '?B=0 A' se code 8A9 (cf tableaux précédents)
- 'GOYES b' est codé par le décalage xy entre l'adresse de x et l'adresse a: -7, ce qui donne '9F'. ('#0 7 -' donne #FFFFFFFFFFFFFF9h)

Remarque : cette séquence n'a aucun sens, ce n'est qu'un exemple de codage.

Programmation en assembleur Saturn par l'exemple

Afin de mieux comprendre comment programmer en assembleur, étudions quelques exemples. Avant de commencer, il faut savoir qu'il est très fréquent dans ce genre de programmation de "planter" sa machine et d'obtenir des "Memory lost". Il ne faut donc surtout pas essayer de programmer en assembleur si la machine contient des données importantes non sauvegardées. En particulier, si on utilise une carte d'extension de la RAM, il est préférable qu'elle ne soit pas fusionnée, et qu'elle soit protégée en écriture.

Commençons par un exemple très simple : le programme ADR. Ce programme renvoie l'adresse de l'objet de niveau 1.

```
02D9D       Objet programme

02A4E       Objet Entier binaire
0000A       Entier sur 5 quartets
00000       Entier : 0

06657       NEWOB Interne
1FBBD       SWAP
```

```
02DCC       Objet "Code"
00026       Taille

143     A=DAT1 A    A=adresse de l'objet de niveau 1
174     D1=D1+5     D1 pointe sur le niveau 2
147     C=DAT1 A    C=adresse de l'objet de niveau 2
                    (#0h)
136     CD0EX       D0 pointe sur #0h et son ancienne
                    valeur et sauvee dans C
169     D0=D0+10    saute le prologue et la longueur
140     DAT0=A A    remplace 00000 par l'adresse de
                    l'objet 1
134     D0=C        récupère D0
E7      D=D+1 A     un niveau de pile libéré
142     A=DAT0 A  |
164     D0=D0+A   |     fin de la routine
808C    PC=(A)    |

0312B       Fin de l'objet "Programme"
```

Pour comprendre ce programme, il faut savoir comment est gérée la pile RPL par le système. La pile est une suite d'adresses dont le bas est pointé par D1. Par `A=DAT1 A`, on obtient l'adresse de l'objet 1, par `D1=D1+5 / A=DAT1 A`, on obtient celle de l'objet 2, etc... La fin de la pile est marquée par l'adresse 00000. Le contenu D indique le nombre de niveaux de pile qu'on peut ajouter. Si D est non nul, on peut ajouter un niveau par D=D-1 A / D1=D1-5 / DAT1=A A, ou A contient l'adresse de l'objet à ajouter. Si on veut ajouter un niveau alors que D est nul, il faut appeler un sous-programme appelé "*Garbage Collector*" qui détruit les objets inutilisés et réorganise la mémoire pour trouver de la place.

Revenons à **ADR**. Ce programme fonctionne de la manière suivante : l'adresse cherchée est renvoyée dans un entier binaire. Comme le fait de créer un tel entier n'est pas très simple, on commence par en placer un dans la pile, que le programme n'aura qu'à modifier.
Mais placer cet objet dans la pile ne revient en fait qu'à y mettre son

adresse. Donc, si on le modifie, la modification apparait dans le programme lui-même. Pour éviter cela, on force la machine à créer une copie de cet entier binaire en effectuant un NEWOB. On peut utiliser le NEWOB interne car comme le programme met #0h dans la pile, on est sur qu'elle n'est pas vide.
Le SWAP suivant ne doit par contre pas être sous forme interne, car il faut vérifier qu'il y a bien un objet au niveau 2.
La routine assembleur lit alors l'adresse de l'objet 1 et la place dans l'entier binaire. Notez qu'au début on effectue un `D1=D1+5`, et à la fin `D=D+1 A`. Ces deux instructions ont pour effet un "DROP".

Pour entrer ce programme, il faut taper la chaîne suivante sans espaces ni sauts de ligne :

```
"D9D 20E 4A2 0A0 000 000 007 566 0DB BF1
CCD 206 200 014 317 414 713 616 914 013
4E7 142 164 808 CB2 130"
```

DUP BYTES doit donner : #6D85h (83 octets). Executez alors ASS.

Un des points les plus importants à retenir est qu'il faut en général utiliser la commande NEWOB avant de modifier un objet trouve dans la pile. En effect, éxaminons le programme **M10** suivant, dont le but est de multiplier par 10 le nombre réél du niveau 1 (en ajoutant 1 à son exposant) :

```
02DCC       Objet "Code"
0002A       Taille

143     A=DAT1 A    A=adresse objet 1
133     AD1EX       D1=adresse objet 1, A=pointeur sur
                    le niveau 1
174     D1=D1+5     D1 pointe sur le champ "exposant"
                    du réel
1573    C=DAT1 X    C.X=exposant
05      SETDEC      mode décimal
B36     C=C+1 X     exposant + 1
04      SETHEX      mode héxadécimal
1553    DAT1=C X    remettre l'exposant en place
```

```
131     D1=A          D1 pointe sur le niveau 1
142     A=DAT0 A      |
164     D0=D0+A       | fin de la routine
808C    PC=(A)        |
```

Outre le fait que ce programme ne vérifie pas son argument, il est incorrect. Si on essaye 456 **M10**, on obtient 4560, ce qui correct. Cependant, si on essaye 4 **M10**, on obtient 4, si on fait 456 DUP **M10**, on obtient 4560 aux niveaux 1 et 2, et si on fait 4 'X' STO X **M10**, on obtient cette fois 40, mais si on rappelle X, on voit qu'il est passé à 40 aussi ! Tout ceci serait évité simplement en ajoutant un NEWOB devant ce programme.

En effet, dans le premier cas, tout se passe bien car quand on tape 456, la HP crée un nouvel objet RAM correspondant au réel 456 et place son adresse dans la pile.

Dans le deuxième cas, la HP reconnait que 4 est un entier entre -9 et 9, et elle possède tous ces entiers en ROM. Elle ne crée donc pas de nouvel objet et se contente de mettre l'adresse du 4 en ROM dans la pile. Comme l'adresse à laquelle essaie d'écrire le `DAT1=C X` est en ROM, cette fonction n'a aucun effet et le 4 reste inchangé.

Dans le troisième cas, le DUP ne crée pas de nouvel objet; il ne fait que copier l'adresse du niveau 1. Il est donc normal que la modification apparaisse aussi au niveau 2.

Dans le dernier cas, le premier 4 placé dans la pile est en ROM, mais le STO en crée une copie qu'il place dans le répertoire en cours sous le nom X. Quand on rappelle X, c'est simplement l'adresse de cette copie qu'on place dans la pile. Donc c'est X que M10 modifie. Notez que lorsque nous stockons un objet dans un port, la HP lui associe un checksum (avec BYTES). Si on avait fait l'expérience en mettant 4 dans 0:X, après éxecution de M10, on aurait eu comme précédemment 40 dans la pile et dans 0:X. Mais le checksum ne serait plus correct. A la première vérification de ce checksum, par exemple en éteignant et rallumant la machine, nous aurions obtenu un "Invalid Data Card" et le contenu du port serait perdu !

Pour corriger ce programme, il faut lui rajouter un test des

arguments (evaluer ce programme sur une chaîne de caractères donnerait très vite un *memory lost*), et un NEWOB :

```
02D9D   Objet "Programme"
18AB2   Vérifie DEPTH ≥ 1
63B2D   Vérifie que l'objet est un réel
06657   NEWOB interne

02DCC   Objet "Code"
0002A   Taille

143     A=DAT1 A    A=adresse objet 1
133     AD1EX       D1=adresse objet 1, A=pointeur sur
                    le niveau 1
174     D1=D1+5     D1 pointe sur le champ "exposant"
                    du réel
1573    C=DAT1 X    C.X=exposant
05      SETDEC      mode décimal
B36     C=C+1 X     exposant + 1
04      SETHEX      mode héxadécimal
1553    DAT1=C X    remettre l'exposant en place
131     D1=A        D1 pointe sur le niveau 1
142     A=DAT0 A    |
164     D0=D0+A     | fin de la routine
808C    PC=(A)      |

0312B   Fin de l'objet "Programme"
```

Le programme final s'obtient en tapant sans espaces ni sauts de ligne la chaîne suivante :

```
"D9D 202 BA8 1D2 B36 756 60C CD2 0A2 000
143 133 174 157 305 B36 041 553 131 142
164 808 CB2 130"
```

DUP BYTES doit donner #00C0h (77 octets). Tapez alors **ASS** puis **'M10' STO**.

Voyons à présent un exemple un petit plus compliqué. Il s'agit de convertir un réel long en chaîne de caractères. Les réels longs ne sont pas normalement accessibles à l'utilisateur, mais la HP contient les routines nécessaires pour effectuer les opérations dessus (voir la liste des Internals dans les adresses 2Axxx). Ce programme va nous permettre de voir comment créer un objet RPL en assembleur.

Pour distinguer les réels longs des réels simples, le programme affiche un "X" (comme eXtended real) à la place du "E" de l'exposant, et affiche ce "X" même si l'exposant est nul. Les chaînes obtenues seront donc par exemple "3X" pour 3, "4X-2" pour 4*10^-2, "1.23X2" pour 123.

Voici le listing :

```
02D9D          Objet "Programme"
1884D          Met à 0 le numéro de l'ins-
truction en cours
18AB2          Vérifie DEPTH ≥ 1

02DCC          Objet "Code"
000D2          Taille

3404000        LCHEX 00040      taille maximale
                                nécessaire
                                (en quartets)
8FAE261        GOSBVL 162EA     commence à créer
                                un chaîne de
                                caractères
```

Ici D0 contient l'adresse du premier caractère de la chaîne crée.

```
143     A=DAT1 A        A=adresse du réel long
131     D1=A            D1 pointe sur le prologue
                        du réel long
179     D1=D1+10        on saute le prologue et
                        l'exposant
1537    A=DAT1 W        A=mantisse (signe dans le
                        champ S)
```

Mettre éventuellement le signe de la mantisse.

```
948F0   ?A=0 S      positif ?
        GOYES @1    sinon mettre "-"
31D2    LCHEX 2D    Code ASCII du "-"
14C     DAT0=C B    Le mettre dans la chaîne
161     D0=D0+2     caractère suivant
AC0     A=0 S       effacer le signe
```

Ecrire le premier chiffre.

```
@1
3103    LCHEX 30    Code ASCII du "0"
810     ASLC        amener le premier chiffre de la
810     ASLC        mantisse dans le quartet 0
A86     C=A  P      C.B contient maintenant le code de
                    ce chiffre
14C     DAT0=C B    ajouter ce caractère
161     D0=D0+2     et pointer sur le suivant
A80     A=0 P       effacer le chiffre écrit
810     ASLC        passer au suivant
97842   ?A=0 W      tous les chiffres suivants sont nuls ?
        GOYES @2    si oui, inutile de les afficher
```

S'il y en a d'autres, mettre une virgule.

```
31C2    LCHEX 2C    Code de ","
14C     DAT0=C B    ajouter une virgule
161     D0=D0+2     caractère suivant
```

Ecrire la suite de la mantisse.

```
3103    LCHEX 30    Code du "0"

@3
A86     C=A P       convertir le chiffre en ASCII
14C     DAT0=C B    l'ajouter à la chaîne
161     D0=D0+2     passer au caractère suivant
A80     A=0 P       effacer le chiffre
810     ASLC        passer au suivant
```

```
97CEE   ?A=0 W      tant qu'il reste des chiffres non nuls
        GOYES @3

Mettre un "X"

@2
3185    LCHEX 58    Code du "X"
14C     DAT0=C B    ajouter un "X"
161     D0=D0+2     caractère suivant
```

Si l'exposant est non nul, il faut l'ajouter.

```
1C4     D1=D1-5     D1 point sur l'exposant du réel long
143     A=DAT1 A    A=exposant
8A8F3   ?A=0 A      l'exposant est nul ?
        GOYES @4    si oui, ne pas l'afficher
```

Ecrire éventuellement le signe de l'exposant

```
3400005 LCHEX 50000 limite des exposants positifs
8B221   ?C>A  A     exposant positif ?
        GOYES @5    si oui, ne pas l'afficher
05      SETDEC      passer en mode décimal
F8      A=-A A      A=valeur absolue de l'exposant
04      SETHEX      revenir en mode hexadécimal
31D2    LCHEX 2D    Code du "-"
14C     DAT0=C B    ajouter un "-"
161     D0=D0+2     caractère suivant
```

Chercher le premier chiffre non nul.

```
@5
25      P=5         initialiser P à 5

@6
0D      P=P-1       décrementer P
908BF   ?A=0 P      tant que le chiffre est nul
        GOYES @6
```

Ecrire l'exposant.

```
@7

303       LCHEX 3     C.P=3
1500      DAT0=A P    écrire la moitié de poids faible du
                      code ASCII
160       D0=D0+1
1540      DAT0=C P    écrire la moitié de poids fort
160       DO=DO+1
0D        P=P-1       chiffre suivant
5CE       GONC @7     tant qu'ils ne sont pas tous écrits

20        P=0         remettre P à 0
```

Terminer la chaîne de caractères et la mettre dans la pile

```
8F17661       GOSBVL 16671    termine la chaîne
8DB6630       GOVLNG 0366F    remplace le réel long
                              par la chaîne

0312B         Fin de l'objet "Programme"
```

Pour créer la chaîne de caractères destinée à recevoir le résultat de la conversion, on utilise des sous-programmes de la ROM.

La routine **162EAh** fait deux choses : elle sauvegarde les registres D0, D1, D et B, elle réserve en mémoire une zone dont la taille (en quartets) est contenue dans C.A, y place le prologue "Chaîne de caracteres (02A2C), et place dans D0 l'adresse de l'emplacement où écrire le premier caractère de la chaîne. Cette routine modifie le contenu des registres A, B, C, D, R0 et R1. Après l'appel, R0 contient l'adresse du prologue de la chaîne crée, et R1 l'adresse à laquelle doit être stockée la taille de la chaîne.

La routine 16671h sert à terminer la création de la chaine. Quand on appelle cette routine, R0 doit contenir la valeur renvoyée par 162EA, et D0 doit pointer sur l'adresse qui suit le dernier caractè-

re de la chaîne. La routine calcule alors la longueur de la chaîne crée, la place derrière le pologue, et libère la mémoire reservée en trop. Les registres A, B, C, D, R1, D0 et D1 sont modifiés.

La routine **0366Fh** récupère les registres B, D, D0 et D1, remplace le premier niveau de la pile par l'objet dont l'adresse est dans R0, et lance l'évaluation de l'objet RPL suivant.

L'interêt de ces routines est qu'on a pas besoin de connaître à l'avance la taille exacte qu'aura l'objet : il n'y a besoin que d'une majoration.

Remarque : par cette méthode, on peut créer tous les objets qui ont le même format de base que les chaînes, c'est-à-dire ceux dont la longueur est stockée après le prologue (entiers binaires, matrices, matrices liées, objets graphiques, bibliothèques, backups, données de bibliothèques, codes). Il suffit de changer le prologue après l'appel de 162EAh, avec par exemple :

```
D0=D0-10
LCHEX <prologue>
DAT0=C A
D0=D0+10
```

Pour créer un objet qui n'a pas ce format, on peut appeler **162EAh**, faire D0=D0-10, construire l'objet, et appeler la routine **16691h** au lieu de **16671h**. Les paramètres sont les mêmes, sauf A qui doit contenir la même valeur que D0 (ceci s'obtient par AD0EX / D0=A).

Remarque : le programme ne vérifie pas que l'objet du niveau 1 est bien un réel long. Si ce n'est pas le cas, la chaine qu'il renvoie n'a bien sûr pas de sens, mais il n'y a pas ici de danger particulier puisqu'on ne fait que lire dans cet objet.

Pour entrer le programme, il faut taper la chaîne suivante sans espaces ni sauts de lignes :

```
"D9D 20D 488 12B A81 CCD 202 D00 034 040
008 FAE 261 143 131 179 153 794 8F0 31D
214 C16 1AC 031 038 108 10A 861 4C1 61A
808 109 784 231 C21 4C1 613 103 A86 14C
161 A80 810 97C EE3 185 14C 161 1C4 143
8A8 F33 400 005 8B2 210 5F8 043 1D2 14C
161 250 D90 8BF 303 150 016 015 401 600
D5C E20 8F1 766 18D F66 30B 213 0"
```

Vérifier qu'il n'y a pas eu de fautes de frappe en tapant **DUP BYTES**, qui doit donner #926Fh (240 octets), puis éxecuter **ASS 'LONGR' STO**.

Vous pouvez alors tester ce programme sur le réel long π, obtenu par **#2A458h SYSEVAL**, et sur les autres réels longs de la ROM, dont on trouve les adresses dans le chapitre *Internals*.

Pour finir, nous allons étudier un exemple qui n'a pas de réelle utilité dans le cadre d'une classe prépa, mais qui montre comment on peut accéder à l'affichage dans un programme assembleur. Ce programme fait tomber de la neige dans l'écran de votre HP.

Nous allons utiliser quelques internals très utiles :

#12635h renvoie au niveau 1 un GROB 131x56 : c'est l'objet graphique correspondant à tout ce qui se trouve au dessus de la barre de menu. Pour afficher nos flocons de neige, nous allons écrire dans ce GROB.

Remarque : quand on évalue l'internal #12635 l'objet renvoyé dans la pile est parfois affiché comme "External" qu lieu de "Graphic". Ceci est dû au fait que le système modifie le prologue de cet objet, et il n'est plus reconnu comme un objet RPL. Ce qui suit le prologue a cependant toujours le même format qu'un objet graphique normal. Nous n'aurons donc pas à nous soucier de cela. Il est également pos-

sible que le GROB renvoyé ait 64 lignes au lieu de 56.

`#61C1Ch` sert à réserver de la mémoire. Il prend au niveau 2 une chaîne de caractères vide ("") et au niveau 1 un "Binaire système" <n> . Il renvoie une chaîne de n quartets ne contenant que des 0.

Rappelons qu'un objet graphique a le format suivant :

<prologue><longueur L><nb de lignes><nb de colonnes><données>
5quartets 5quartets 5quartets 5quartets L-15 quartets

Les données sont organisées par lignes, la première étant celle du haut. Chaque ligne contient un nombre pair de quartets (34 pour 131 colonnes), le premier correpondant aux 4 pixels les plus à gauche. Un bit à 1 représente un pixel allumé, et un bit à 0 un pixel éteint.

Le programme commence par réserver de la mémoire à l'aide du syseval `61C1C`, afin d'y ranger les informations nécessaires pour repérer la position des flocons à déplacer : les cinq premiers quartets contiendront le nombre de flocons à déplacer, puis pour chaque flocon, on aura les informations suivantes :

5 quartets : offset du quartet contenant le flocon dans le grob
1 quartet : état du quartet avant qu'on affiche le flocon
2 quartets : nombre maximum de lignes qu'il reste à parcourir
1 quartet : masque du pixel où se trouve le flocon
2 quartets : colonne du flocon.

Une fois la mémoire réservée, on boucle sur la routine assembleur jusqu'à ce que l'utilisateur appuie sur une touche. La routine assembleur fait deux choses : elle commence par faire descendre tous les flocons d'une ligne, puis elle ajoute un nouveau flocon sous la ligne qui sépare la zone d'affichage de la pile de celle des indicateurs.

Voici le listing correspondant :

```
02D9D   Objet "Programme"
055DF   Chaine ""

11920   Objet "Binaire Système"
87200   <278h>

61C1C   Crée un chaîne de 278h quartets
        mise à 0
230C3   DO
12635   Renvoie l'objet graphique
        correspondant à l'affichage
        normal

02DCC   Objet "Code"
002A6   Taille

143     A=DAT1 A    A=adresse objet graphique
174     D1=D1+5     | drop
E7      D=D+1 A     |
8FB9760 GOSBVL 0679B sauve D0,D1,B,D
3441000 LCHEX 00014 C=20
C2      C=C+A A     saute l'entête, la taille et les
                    dimensions du GROB
D5      B=C A       B pointe sur le début des données
143     A=DAT1 A    A=adresse de la chaîne
818F09  A=A+10 A    saute l'entête et la taille
100     R0=A        R0=adresse buffer
131     D1=A        D1=adresse buffer

147     C=DAT1 A    C=nombre de flocons
D7      D=C A       D=nombre de flocons

174     D1=D1+5     D1 pointe sur le premier flocon
CF      D=D-1 A     décrémenter le nombre de flocons
560     GONC @1     s'il était nul
64B0    GOTO @2     aller en @2

@1
147     C=DAT1 A    C=offset du flocon dans le GROB
```

```
C9        C=C+B A      C=adresse du quartet contenant le
                       pixel
06        RSTK=C       sauver C
175       D1=D1+6      D1 pointe sur le nombre de lignes
                       restantes
14B       A=DAT1 B     A=nombre de lignes à parcourir
A6C       A=A-1 B      une de moins
560       GONC @3      s'il n'y en a plus
6531      GOTO @4      s'arrêter

@3
149       DAT1=A B     remettre en place
171       D1=D1+2      D1 pointe sur le masque du pixel
1574      C=DAT1 S     C.S=masque
DA        A=C A        A=adresse du quartet contenant le
                       pixel
3422000   LCHEX 00022  C=nombre de quartets par ligne
CA        A=A+C A      A=adresse du quartet de la ligne de
                       dessous
130       D0=A         D0=adresse du quartet de dessous
1524      A=DAT0 S     A.S=quartet de dessous
0E46      A=A&C A      tester le pixel du dessous
948D3     ?A=0 S       vide ?
          GOYES @5     si oui, aller en @5
```

Si le pixel du dessous n'est pas vide, il faut regarder si le flocon peut glisser à côté. Il faut donc tester les pixels voisins. On va choisir au hasard de quel côté faire le premier test. Pour cela, on va utiliser un des bits de l'horloge de la HP : le n° 3 (facile à tester et varie beaucoup)

```
1B83100 D0= 00138    D0=adresse de l'horloge
1524    A=DAT0 S     A.S=quartet de poids faible
A44     A=A+A S      bit 3 dans CARRY
471     GOC @6

7E31    GOSUB @7     teste le côté gauche
552     GONC @5      si vide aller en @5
75A1    GOSUB @8     teste le côté droit
5E1     GONC @5      si vide aller en @5
1C1     D1=D1-2
```

```
6AE0    GOTO @4      le flocon ne peut plus bouger

@6
7791    GOSUB @8     teste le côté droit
501     GONC @5      si vide, va en @5
7221    GOSUB @7     teste le côté gauche
590     GONC @5      si vide, va en @5
1C1     D1=D1-2
65D0    GOTO @4      le flocon ne peut plus bouger
```

Quand on est ici, le flocon doit tomber dans le pixel du quartet pointé par D0 repéré par le masque contenu dans C.S

```
@5
1524    A=DAT0 S     A=quartet où ajouter le flocon
AC8     B=A S        sauver le fond dans B.S
0E4E    A=A!C S      ajouter le flocon
1504    DAT0=A S     remettre le flocon en place
1C7     D1=D1-8      D1 pointe sur l'offset
07      C=RSTK       récupère la position précédente du
                     flocon
136     CD0EX        D0 pointe sur le quartet où effacer
                     le flocon et C sur sa nouvelle
                     position
E9      C=C-B A      C=offset de la nouvelle position
                     dans l'écran
145     DAT1=C A     nouvel offset
174     D1=D1+5      D1 pointe sur le quartet à remettre
                     en place
1574    C=DAT1 S     C=quartet à remettre en place pour
                     effacer
1544    DAT0=C S     effacer le flocon
AC4     A=B S        A.S=nouveau fond à sauvegarder
1514    DAT1=A S     le sauver
175     D1=D1+6      D1 pointe sur le flocon suivant

@11
CF      D=D-1 A      un flocon à traiter en moins
460     GOC @2       si fini, va en @2
625F    GOTO @1      flocon suivant
```

Ici, on a déplacé tous les flocons existants. Il faut maintenant en créer un nouveau sur la première ligne. On va encore se servir de l'horloge pour trouver sa colonne au hasard.

```
@2
1B83100   D0=HEX 00138   D0=adresse de l'horloge
D9        C=B A          C=adresse début de l'écran
06        RSTK=C         sauver C
D0        A=0 A          effacer A.A
14A       A=DATO B       A.A contient un nombre entre 0 et
                         255
3438000   LCHEX 0083     C=131=nombre de colonnes dans
                         l'écran
8F42F30   GOSBVL 03F24   diviser A.A par C.A. le reste est
                         dans A.
178       D1=D1+9        D1 pointe sur le champ de la
                         colonne
149       DAT1=A B       écrire le numéro de colonne dans le
                         buffer
303       LCHEX 3        C.P=masque des deux derniers bits
0E02      C=C&A P        C.P=numéro de colonne modulo 4
80D0      P=C 0          P=numéro de colonne modulo 4
338421    LCHEX 1248     suivant que P vaut 0, 1, 2 ou 3, le
                         quartet 3 de C contiendra 1, 2, 4 ou
                         8
80D3      P=C 3          P=masque correspondant à la
                         colonne
80CF      C=P 15         C.S=masque
20        P=0            remettre P à 0
1C0       D1=D1-1        D1 pointe sur le champ du masque
1554      DAT1=C S       sauver le masque
3182      LCHEX 28       C=40=nombre de lignes à parcourir
1C1       D1=D1-2        D1 pointe sur le champ du nombre
                         de lignes
14D       DAT1=C B       sauver le nombre de lignes
819F0     ASRB A         diviser A par 2
819F0     ASRB A         diviser A par 2
34EF100   LCHEX 001FE    C=510=15*34=offset du début de la
                         16ème ligne
CA        A=A+C A        A=offset du pixel dans l'écran
07        C=RSTK         C=adresse du début de l'écran
```

```
C2          C=C+A A     C=adresse du quartet contenant le
                        flocon
134         D0=C        D0=adresse du quartet contenant le
                        flocon
1524        A=DAT0 S    A.S=fond à sauvegarder
1C0         D1=D1-1     D1 pointe sur le champ du fond
1514        DAT1=A S    sauver le fond
1C4         D1=D1-5     D1 pointe sur le champ de l'offset
141         DAT1=A A    sauver l'offset
0E4E        A=A!C S     ajouter le flocon au fond
1504        DAT0=A S    le mettre à l'écran
110         A=R0        A=adresse du début du buffer
130         D0=A        D0=pointe sur le début du buffer
142         A=DAT0 A    A=ancien nombre de flocons
E4          A=A+1 A     un de plus
140         DAT0=A A    remettre en place
8D34150     GOVLNG 05143 fin de la routine (éxecute les
                        instructions GOSBVL #067D2 /
                        A=DAT0 A / D0=D0+5 / PC=(A))
```

On arrive ici si le flocon qu'on essaie de bouger est bloqué. Il faut l'enlever du buffer, et décaler tous les flocons suivants pour occuper l'emplacement libéré.

```
@4
110         A=R0        A=adresse du début du buffer
130         D0=A        D9 pointe sur le début du buffer
142         A=DAT0 A    A=nombre de flocons
CC          A=A-1 A     un de moins
140         DAT0=A A    remettre en place
1C5         D1=D1-6     D1 pointe sur le début des
                        informations sur ce flocon
137         CD1EX       copier cette adresse dans C
135         D1=C        sans modifier D1
134         D0=C        le mettre aussi dans D0
06          RSTK=C      sauver cette adresse
DB          C=D A       C=nombre de flocons non traités
CE          C=C-1 A     enlever 1
560         GONC @9     si au moins 1, va en @9
644F        GOTO @2     sinon c'était le dernier
```

```
@9
17A     D1=D1+11    D1 pointe sur le flocon suivant

@10
15BA    A=DAT1 11   A contient les informations sur le
                    flocon suivant
158A    DAT0=A 11   les déplacer en arrière
16A     D0=D0+11    pointer les suivantes
17A     D1=D1+11
CE      C=C-1 A     s'il reste des flocons, recommencer
5FE     GONC @10
07      C=RSTK      récuperer l'adresse à laquelle on
                    s'était arrêté
135     D1=C        la remettre dans D1
6C1F    GOTO @11    flocon suivant
```

Le sous programme suivant teste si le flocon peut tomber du côté gauche

```
@7
170     D1=D1+1     D1 pointe sur la colonne du flocon
14F     C=DAT1 B    C=colonne
1C0     D1=D1-1     D1 pointe sur le masque du pixel
A6E     C=C-1 B     décrementer la colonne
400     RTNC        si <0, on est au bord, donc pas
                    possible d'aller à gauche. revenir
                    avec CARRY=1
10C     R4=C        sauver la colonne
1574    C=DAT1 S    C.S=masque
1C7     D1=D1-8     D1 pointe sur l'offset
143     A=DAT1 A    A=offset du pixel à l'écran
177     D1=D1+8     D1 pointe sur le masque
C0      A=A+B A     A=adresse du pixel
833     SB=0        mettre SB à zéro pour récupérer le
                    bit qui va sortir après le décalage
81942   CSRB S      décaler le masque, SB=bit sortant
832A0   ?SB=0       s'il sort 1, le pixel de gauche
        GOYES @12   est dans un autre quartet
818427  C=C+8 S     nouveau masque : 1000
CC      A=A-1 A     quartet précédent
@12
```

```
130       D0=A          D0=adresse du quartet du pixel de
                        gauche
1524      A=DAT0 S      A.S=quartet contenant le pixel de
                        gauche
0E46      A=A&C S       tester si le pixel de gauche est libre
94C00     ?A#0 S        si non, retourner avec CARRY=1
          RTNYES
3122      LCHEX 22      C=34=nombre de quartets par ligne
CA        A=A+C A       A=adresse du quartet de dessous
130       D0=A          D0 pointe sur le quartet de dessous
1524      A=DAT0 S      A.S=quartet du pixel de dessous
0E46      A=A&C S       tester si le pixel est libre
94C00     ?A#0 S        si non, retourner avec CARRY=1
          RTNYES
1554      DAT1=C S      sauver le nouveau masque
170       D1=D1+1       D1 pointe sur la colonne
114       A=R4          A=nouvelle colonne
149       DAT1=A B      sauver la nouvelle colonne
1C0       D1=D1-1       D1 pointe sur le masque
03        RTNCC         revenir avec CARRY=0
```

La routine suivante teste si le flocon peut tomber du côté droit.

```
@8
170       D1=D1+1       D1 pointe sur la colonne du flocon
14B       A=DAT1 B      A=colonne
1C0       D1=D1-1       D1 pointe sur le masque du pixel
B64       A=A+1 B       incrémenter la colonne
3138      LCHEX 83      C=131=nombre de colonnes
9EE00     ?A>=C B       dernière colonne ?
          RTNYES        si oui, revenir avec CARRY=1
104       R4=A          sauver la nouvelle colonne
1574      C=DAT1 S      C.S=masque du pixel
1C7       D1=D1-8       D1 pointe sur l'offset
143       A=DAT1 A      A=offset
177       D1=D1+8       D1 pointe sur le masque
C0        A=A+B A       A=adresse de gauche de C.S
A46       C=C+C S       décalage à gauche de C.S
570       GONC @13      s'il sort un 1, on change de quartet
B46       C=C+1 S       masque=0001
E4        A=A+1 A       quartet suivant
```

```
@13
130       D0=A          D0=adresse du quartet du pixel de
                        droite
1524      A=DAT0 S      A.S=quartet contenant le pixel de
                        droite
0E46      A=A&C S       tester si le pixel est libre
94C00     ?A#0 S        si non, revenir avec CARRY=1
          RTNYES
3122      LCHEX 22      C=34=nombre de quartets par ligne
CA        A=A+C A       A=adresse du quartet de dessous
130       D0=A          D0 pointe sur le quartet de dessous
1524      A=DAT0 S      A.S=quartet de dessous
0E46      A=A&C S       tester si le pixel de dessous est libre
94C00     ?A#0 S        si non, revenir avec CARRY=1
          RTNYES
1554      DAT1=C S      sauver le nouveau masque
170       D1=D1+1       D1 pointe sur la colonne
114       A=R4          A=nouvelle colonne
149       DAT1=A B      sauver la nouvelle colonne
1C0       D1=D1-1       D1 pointe sur le masque
03        RTNCC         revenir avec CARRY=0

33920                   Objet "Réel"
8990000000000020 0.02
1A71F                   WAIT
230ED                   UNTIL
1A873                   KEY
236B9                   END
1FBF3                   DROP2
0312B                   fin du "Programme"
```

Remarque : quand on effectue un décalage à droite du type rSRB, le bit sortant est ajouté à SB (par un OU logique). Si on veut récupérer exactement ce bit, il faut donc mettre SB à 0 avant le décalage.
Remarque : la barre de menu est dans un objet graphique différent de celui qui reste de l'affichage. On peut obtenir son adresse par l'internal **#12645h**. Il est possible au cours d'un programme de supprimer la barre de menu pour n'avoir plus qu'un GROB de 64 lignes pour tout l'écran. Pour cela, il faut appeler au début du programme l'internal **#4E2CFh**. Après cela, le GROB renvoyé par **#12635h** fait 64

lignes et recouvre tout l'écran.

Pour rentrer le programme, tapez la chaîne suivante sans espaces ni sauts de ligne :

```
"D9D 20F D55 011 920 872 00C 1C1 63C 032
536 21C CD2 06A 200 143 174 E78 FB9 760
344 100 0C2 D51 438 18F 091 001 311 47D
717 4CF 560 64B 014 7C9 061 751 4BA 6C5
606 531 149 171 157 4DA 342 200 0CA 130
152 40E 469 48D 31B 831 001 524 A44 471
7E3 155 275 A15 E11 C16 AE0 779 150 172
215 901 C16 5D0 152 4AC 80E 4E1 504 1C7
071 36E 914 517 415 741 544 AC4 151 417
5CF 460 625 F1B 831 00D 906 D01 4A3 438
000 8F4 2F3 017 814 930 30E 028 0D0 338
421 80D 380 CF2 01C 015 543 182 1C1 14D
819 F08 19F 034 EF1 00C A07 C21 341 524
1C0 151 41C 414 10E 4E1 504 110 130 142
E41 408 D34 150 110 130 142 CC1 401 C51
371 351 340 6DB CE5 606 44F 17A 15B A15
8A1 6A1 7AC E5F E07 135 6C1 F17 014 F1C
0A6 E40 010 C15 741 C71 431 77C 082 281
942 832 A08 184 27C C13 015 240 E46 94C
003 122 CA1 301 524 0E4 694 C00 155 417
011 414 91C 003 170 14B 1C0 B64 313 89E
E00 104 157 41C 714 317 7C0 A46 570 B46
E41 301 524 0E4 694 C00 312 2CA 130 152
40E 469 4C0 015 541 701 141 491 C00 333
920 899 000 000 000 002 0F1 7A1 DE0 323
78A 19B 632 3FB F1B 213 0"
```

DUP BYTES doit donner #6906h (774 octets). Tapez alors **ASS 'NEIGE' STO**

Ces quelques exemples ont montré comment réaliser les opérations de base en assembleur : manipulation de pile, création d'objets et graphisme. La meilleure façon d'apprendre l'assembleur est encore d'expérimenter : à vous de modifier ces exemples et d'en créer de nouveaux.

17

Internals de la HP 48

On appelle Internal, une adresse interne au calculateur, que l'on peut utiliser avec un simple ***SYSEVAL*** *ou dans un programme.*

Nous avons listé ci-dessous les pricipales adresses internes de la HP48.Il en existe bien sûr beaucoup d'autres...
Les adresses ont été classées par familles, puis dans chaque famille par ordre d'adresse.

I

La programmation à partir d'adresses "Internals"

Nous pourrions situer cette programmation entre la programmation RPL simple et la programmation assembleur. Elle a deux avantages majeurs : elle est simple et rapide.

Une bonne partie de la ROM de votre calculateur HP a été écrite à partir d'adresses Internals : vous pourrez ainsi décompiler *facilement* sa mémoire.

Pour programmer, vous aurez besoin de trois outils fondamentaux fournis dans le chapitre 1 :

SYS	pour créer un objet internal
→PGM	pour assembler des internals
PGM→	pour désassembler des internals

Un autre outil très appréciable sera le programme **NEW** créant un nouvel affichage de votre calculateur. Il se trouve dans le chapitre suivant.

Le plus souvent une adresse **Internal** apparaît à l'écran comme *External* ou bien un bloc d'*External*. Le programme NEW évoqué ci-dessus remplace l'internal par son adresse.

Par exemple le SWAP interne est créé en faisant :

```
#3223h SYS
```

Il apparait à l'écran sous la forme :

```
External
```

Mais avec le nouvel affichage, nous voyons à l'écran :

```
$ 3223h
```

La seule difficulté quand on programme à base d'adresses Internal est de connaître justement ces adresses. Une partie vous est fournie dans ce chapitre.

Les programmes du chapitre 2 ont été réécrits à base d'adresses **Internal** pour aller plus vite à l'exécution. En voici un exemple :

Le programme Q→R
sous sa forme RPL classique :

```
« IF DUP TYPE THEN
C→R DUP2 PGCD SWAP OVER / ROT ROT /
ELSE 1 SWAP END »
```

Le même programme a été traduit en adresses **Internals** :

```
02D9D       Début de programme
1592D       vérifie DEPTH ≥ 1
18FB2       teste le type de l'argument
  03FF9          <1h>
       02D9D     Début de programme
       2A2C9     1
       03223     SWAP
       0312B     Fin de programme
  04003          <2h>
       02D9D     Début de programme
       05D2C     C→R interne
       031AC     DUP2 interne
       84E20     Nom global
       04        4 lettres
       50        P
       47        G
       43        C
       44        D
       03223     SWAP interne
       032C2     OVER interne
       2A9FE     / interne (réels)
       60FAC     ROT ROT interne
       2A9FE     / interne (réels)
       032C2     OVER
       2A799     prend TOS et teste > 0
       61A3B     si TOS=true alors Return
       2A920     NEG interne
       03223     SWAP interne
       2A920     NEG interne
       03323     SWAP
       0312B     Fin de programme
0312B       Fin de programme
```

Remarque : si vous comparez ce listing à la suite de codes fournie dans le chapitre 2, vous remarquerez sans doute que le codage est inversé : ainsi les adresses `02D9D / 0312B` seront codées `"D9D20B2130"`.

Vous avez donc vu dans le chapitre 2, comment fabriquer un tel objet à partir de la chaîne de carcatères correspondante et de la commande **ASS**. Voici une autre méthode, plus appropriée à la programmation en adresses **Internals**.

Voici comment procéder avec le programme précédent :

```
#1592Dh SYS
#18FB2h SYS
#03FF9h SYS

#2A2C9h SYS
#03223h SYS
2 →PGM

#04003h SYS

#05D2Ch SYS
#031ACh SYS
'PGCD'
#03223h SYS
#032C2h SYS
#2A9FEh SYS
#60FACh SYS
#2A9FEh SYS
#032C2h SYS
#2A799h SYS
#61A3Bh SYS
#2A920h SYS
#03223h SYS
#2A920h SYS
#03323h SYS
15 →PGM

6 →PGM
```

Voici une manière bien différente de créer des programmes ! Nous reconstruisons chaque bloc de programme à la main. De plus c'est en modifiant de cette manière une routine de la ROM de la HP, que nous avons créé le programme **NEW** du chapitre suivant.

Note : grâce à la commande →PGM, vous pouvez aisémenet inclure des bouts de code dans vos programmes, RPL ou à base d'internals.

A vous maintenant de vous lancer à l'aide de la liste qui suit.

II
Une liste d'adresses "Internals" classée par famille

Nous entendons par **TOS** (Top of Stack) le haut de la pile, c'est-à-dire son premier élément.
Pour les opérations, IL NE FAUT PAS CONFONDRE :

-1 interne	*qui soustrait 1*
-1	*le nombre -1*

Nous ne pouvons vous donner que ce conseil : essayez ces adresses, vous verrez rapidement leurs avantages.

Si vous cherchez la signification d'une adresse #X qui ne se trouve pas dans cette liste, procédez de la manière suivante :

- Executez `#Xh SYS`
- Décomposez le résultat rendu à l'aide de la commande `PGM→`
- Si les élements qui sont dedans ne vous sont pas connus, ou s'ils n'apparaissent pas dans la liste qui suit, reproduidez la même démarche sur cette nouvelle adresse.
- Sinon, en ayant compris les différents sous-commandes effectuées, vous pourrez deviner l'action de votre adresse.
- Vous pourrez ainsi compléter la longue liste qui suit !

Affichage et graphisme

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
01F6D	01F6D	CLLCD interne (Pile et haut de l'affichage)
01FA7	01FA7	CLLCD interne (Tout)
041A7	041A7	OFF interne
041ED	041ED	OFF interne
0E05B	0E05B	Affiche un graphique à la place actuelle
1158F	1158F	BLANK interne (2:Binaire Système,1:Binaire Système)
11679	11679	Affiche le graphique a la position
11CF3	11CF3	3 →GROB interne (1:Chaîne)
11D00	11D00	2 →GROB interne (1:Chaîne)
11F80	11F80	1 →GROB interne (1:Chaîne)
1200C	1200C	1 →GROB interne (1:Chaîne)
123C8	123C8	DISP interne (2:Chaîne,1:Binaire Système) taille 10
123E5	123E5	<4h> DISP interne (1:Chaîne) taille 10
123F5	123F5	<3h> DISP interne (1:Chaîne) taille 10
12405	12405	<2h> DISP interne (1:Chaîne) taille 10
12415	12415	<1h> DISP interne (1:Chaîne) taille 10
12429	12429	DISP interne (2:Chaîne,1:Binaire Système) taille 8
1245B	1245B	<0h> DISP interne (1:Chaîne) taille 8
1246B	1246B	<1h> DISP interne (1:Chaîne) taille 8
1247B	1247B	<2h> DISP interne (1:Chaîne) taille 8
1248B	1248B	<3h> DISP interne (1:Chaîne) taille 8
1249B	1249B	<4h> DISP interne (1:Chaîne) taille 8
124AB	124AB	<5h> DISP interne (1:Chaîne) taille 8
124BB	124BB	<6h> DISP interne (1:Chaîne) taille 8
124CB	124CB	<7h> DISP interne (1:Chaîne) taille 8
12635	12635	Rappelle le graphique de l'écran entier
12645	12645	Rappelle le graphique du menu (131x8)
12665	12665	Rappelle PICT
12770	12770	Coupe la chaîne pour l'affichage (1:Chaîne)
127A7	127A7	Coupe la chaîne au saut de ligne(LF) (1:Chaîne)
1314D	1314D	TEXT interne
137B6	137B6	Rappelle la position courante

		→ Ligne, Colonne:Binaire Système
140AB	140AB	DISP interne (2:Tout,1:Nombre Réel)
142FB	142FB	FREEZE interne (1:Nombre Réel)
1685C	1685C	2:Chaîne 1:Binaire Système → "Nombre: Chaîne"
1686A	1686A	2:Chaîne 1:Binaire Système → "Nombre: Chaîne"
1A584	1A584	DISP
1A5A4	1A5A4	FREEZE
1A858	1A858	CLLCD
1C8EA	1C8EA	REPL
1CA62	1CA62	SIZE interne (1:Graphique)
1E04A	1E04A	INDEP
1E07E	1E07E	PMIN
1E09E	1E09E	PMAX
1E0BE	1E0BE	AXES
1E0E8	1E0E8	CENTR
1E101	1E101	CENTR interne (1:Nombre Réel)
1E126	1E126	RES
1E150	1E150	*H
1E170	1E170	*W
1E190	1E190	DRAW
1E1AB	1E1AB	AUTO
1E1C6	1E1C6	DRAX
1E1E1	1E1E1	SCALE
1E201	1E201	PDIM
1E22B	1E22B	DEPND
1E25F	1E25F	ERASE
1E27A	1E27A	PX→C
1E29A	1E29A	C→PX
1E2BA	1E2BA	GRAPH
1E2D5	1E2D5	LABEL
1E2F0	1E2F0	PVIEW
1E31A	1E31A	PIXON
1E344	1E344	PIXOFF
1E36E	1E36E	PIX?
1E398	1E398	LINE
1E3C2	1E3C2	TLINE
1E3EC	1E3EC	BOX
1E416	1E416	BLANK

1E436	1E436	PICT
1E456	1E456	GOR
1E46A	1E46A	GOR interne (3:Graphique,2:Liste,1:Graphique)
1E488	1E488	GOR interne (3:Graphique,2:Nombre Complexe, 1:Graphique)
1E4A6	1E4A6	GOR interne (3:PICT,2:Liste,1:Graphique)
1E4C4	1E4C4	GOR interne (3:PICT,2:Nombre Complexe, 1:Graphique)
1E4E4	1E4E4	GXOR
1E572	1E572	LCD→
1E58D	1E58D	→LCD
1E5AD	1E5AD	→GROB
1E5D2	1E5D2	ARC
1E606	1E606	TEXT
1E621	1E621	XRNG
1E641	1E641	YRNG
1E661	1E661	FUNCTION
1E681	1E681	CONIC
1E6A1	1E6A1	POLAR
1E6C1	1E6C1	PARAMETRIC
1E6E1	1E6E1	TRUTH
1E701	1E701	SCATTER
1E721	1E721	HISTOGRAM
1E741	1E741	BAR
20133	20133	BARPLOT
20167	20167	HISTPLOT
2018C	2018C	SCATRPLOT
201B1	201B1	LINFIT
201D6	201D6	LOGFIT
201FB	201FB	EXPFIT
20220	20220	PWRFIT
2025E	2025E	BESTFIT
20CAD	20CAD	RCL interne (1:PICT)
20F8A	20F8A	PURGE interne (1:PICT)
391EE	391EE	FREEZE (tout l'écran)
393D3	393D3	FREEZE (haut d'écran)
393FD	393FD	FREEZE (Pile)
39451	39451	FREEZE (ligne de menu)

3A1FC	3A1FC	mise à jour du menu
47A1A	47A1A	XRNG interne (2:Nombre Réel,1:Nombre Réel)
47A42	47A42	YRNG interne (2:Nombre Réel,1:Nombre Réel)
47A6A	47A6A	INDEP interne (1:Nombre Réel)
47A8D	47A8D	DEPND interne (1:Nombre Réel)
47C5F	47C5F	ARC interne (4:Nombre complexe,3,2,1:Nombre reel)
491D5	491D5	AUTO interne
4A16C	48B9D	ΣLINE interne
4AC61	4AC61	CENTR interne (1:Nombre Complexe)
4AE3C	4AE3C	SCALE interne (2:Nombre Réel,1:Nombre Réel)
4AF77	4AF77	INDEP interne (1:Nom Global)
4AF8B	4AF8B	INDEP interne (1:Liste)
4AFB3	4AFB3	DEPND interne (1:Nom Global)
4AFC7	4AFC7	DEPND interne (1:Liste)
4AFEF	4AFEF	RES interne (1:Nombre Réel)
4B012	4B012	RES interne (1:Nombre Réel>0/Entier Binaire)
4B03A	4B03A	AXES interne (1:Nombre Complexe)
4B04E	4B04E	AXES interne (1:Liste)
4B09E	4B09E	PMIN interne (1:Nombre Complexe)
4B0C6	4B0C6	PMAX interne (1:Nombre Complexe)
4B206	4B206	PDIM interne (2:Nombre Complexe,1:Complex)
4B300	4B300	PDIM interne (2:Entier Binaire,1:Entier Binaire)
4B323	4B323	PDIM interne (2:Binaire Système,1:Binaire Système)
4B553	4B553	*H interne (1:Nombre Réel)
4B5AD	4B5AD	*W interne (1:Nombre Réel)
4B60C	4B60C	ERASE interne
4B6AC	4B65C	DRAW interne
4C607	4C486	DRAX interne
4D1AA	4D1AA	GRAPH interne
4E875	4E889	LABEL interne
4F011	4F009	PVIEW interne (1:Nombre Complexe)
4F02F	4F027	PVIEW interne (1:Liste)
4F0AC	4F0AC	PX→C interne (1:Liste)
4F179	4F179	C→PX interne (1:Nombre Complexe)
4F37C	4F37C	STO interne (2:Graphique,1:PICT)
4F3EF	4F3EF	PIXON interne (1:Nombre Complexe)
4F458	4F458	PIXON interne (1:Liste)
4F471	4F471	PIXOFF interne (1:Nombre Complexe)

4F48A	4F48A	PIXOFF interne (1:Liste)
4F4A3	4F4A3	PIX? interne (1:Nombre Complexe)
4F4BC	4F4BC	PIX? interne (1:Liste)
4F525	4F525	LINE interne (2:Liste,1:Liste)
4F539	4F539	TLINE interne (2:Liste,1:Liste)
4F584	4F584	LINE interne (2:Nombre Complexe, 1:Nombre Complexe)
4F598	4F598	TLINE interne (2:Nombre Complexe, 1:Nombre Complexe)
4F665	4F665	BOX interne (2:Liste,1:Liste)
4F688	4F688	BOX interne (2:Nombre Complexe, 1:Nombre Complexe)
4F6A1	4F6A1	BLANK interne (2:Entier Binaire,1:Entier Binaire)
4F6BA	4F6BA	GOR/GXOR interne (4:VRAI/FAUX,3:Graphique,2:Liste,1:Graphique)
4F6F6	4F6F6	GOR/GXOR interne (4:VRAI/FAUX,3:Graphique,2:Cplx,1:Graphique)
4F741	4F741	GOR/GXOR interne (4:VRAI/FAUX,3:PICT,2:Cplx/Liste,1:Graphique)
4F8D1	4F8D1	+ interne (2:Graphique,1:Graphique)
4F999	4F999	REPL interne (3:Graphique,2:Liste,1:Graphique)
4F9F3	4F9F3	REPL interne (3:Graphique, 2:Nombre Complexe,1:Graphique)
4FA2F	4FA2F	REPL interne (3:PICT,2:Liste/Nombre Complexe, 1:Graphique)
4FB74	4FB74	SUB interne (3:Graphique,2:Liste,1:Liste)
4FBC4	4FBC4	SUB interne (3:Graphique,2: Complexe,1:Complexe)
4FBF6	4FBF6	SUB interne (3:PICT,2/1:Liste ou Nombre Complexe)
4FC28	4FC28	NEG interne (1:Graphique)
4FC3C	4FC3C	NEG interne (1:PICT)
4FD2C	4FD2C	ARC interne (4:List,3:Entier Binaire,2,1:Nombre Réel)
503D4	503D4	LCD→ interne
50438	50438	→LCD interne (1:Graphique)
5046A	5046A	CLLCD interne
5048D	5048D	→GROB interne (2:Tout,1:Nombre Réel)
505B2	505B2	Graphic 0x0
53AAC	53AAC	Horloge affichée
53ABA	53ABA	Horloge non affichée

Chaînes de caractères

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX
04D3E	04D3E	annule une chaine (1:Chaîne) → ""
04D43	04D43	annule une chaîne (1:Chaîne) → ""
04D57	04D57	DROP et annule une chaîne (2:Chaîne,1:Tout) → ""
050ED	050ED	Premier caractère d'une chaîne (1:Chaîne) → Caractère
0518A	0518A	+ interne (2:Chaîne,1:Chaîne)
05193	05193	+ interne (2:Chaîne,1:Chaîne)
052EE	052EE	+ interne (2:Chaîne,1:Caractère)
052FA	052FA	+ interne (2:Liste,1:Tout)
055DF	055DF	""
05616	05616	SIZE interne (1:String) → Binaire Système
05622	05622	OVER et SIZE interne (2:Chaîne) → Binaire Système
05636	05636	SIZE interne (1:Chaîne) → Binaire Système
05902	05902	SIZE interne (1:Tout) → Binaire Système
05B15	05B15	Chaîne → Nom Global (1:Chaîne)
05BE9	05BE9	Nom Local ou Global → Chaîne (1:Nom Global/Nom Local)
05E81	05E81	→TAG interne (2:Tout,1:Chaîne)
05EC7	05EC9	OBJ→ interne (1:Signé)
0E029	0E029	Affiche une chaîne à la place actuelle
11CF3	11CF3	3 →GROB interne (1:Chaîne)
11D00	11D00	2 →GROB interne (1:Chaîne)
11F80	11F80	1 →GROB interne (1:Chaîne)
1200C	1200C	1 →GROB interne (1:Chaîne)
123C8	123C8	DISP interne (2:Chaîne,1:Binaire Système) taille 10
123E5	123E5	<4h> DISP interne (1:Chaîne) taille 10
123F5	123F5	<3h> DISP interne (1:Chaîne) taille 10
12405	12405	<2h> DISP interne (1:Chaîne) taille 10
12415	12415	<1h> DISP interne (1:Chaîne) taille 10
12429	12429	DISP interne (2:Chaîne,1:Binaire Système) taille 8
1245B	1245B	<0h> DISP interne (1:Chaîne) taille 8

1246B	1246B	<1h> DISP interne (1:Chaîne) taille 8
1247B	1247B	<2h> DISP interne (1:Chaîne) taille 8
1248B	1248B	<3h> DISP interne (1:Chaîne) taille 8
1249B	1249B	<4h> DISP interne (1:Chaîne) taille 8
124AB	124AB	<5h> DISP interne (1:Chaîne) taille 8
124BB	124BB	<6h> DISP interne (1:Chaîne) taille 8
124CB	124CB	<7h> DISP interne (1:Chaîne) taille 8
12770	12770	Coupe la chaîne pour l'affichage (1:Chaîne)
127A7	127A7	Coupe la chaîne au saut de ligne(LF) (1:Chaîne)
14088	14088	→STR interne (1:Tout)
140AB	140AB	DISP interne (2:Tout,1:Nombre Réel)
140F1	140F1	CHR interne (1:Nombre Réel)
1410F	1410F	NUM interne (1:Chaîne)
14137	14137	STR→ interne (1:Chaîne)
1420A	1420A	> interne (2:Chaîne,1:Chaîne)
142A6	142A6	< interne (2:Chaîne,1:Chaîne)
142BA	142BA	≥ interne (2:Chaîne,1:Chaîne)
142E2	142E2	≤ interne (2:Chaîne,1:Chaîne)
15978	15978	→STR interne (1:Tout)
15B13	15B13	→STR interne (1:Tout)
15B31	15B31	→STR interne (1:Tout)
15B3D	15B3D	→STR interne (1:Nombre Réel)
1605F	1605F	rajoute des " dans la chaîne (1:Chaîne)
160E5	160E5	→STR interne (1:Nom global)
162AC	162AC	→STR interne (1:Nombre Réel)
162B8	162B8	→STR interne (1:Nombre Réel)
1685C	1685C	2:Chaîne 1:Binaire Système → "Nombre: Chaîne"
1686A	1686A	2:Chaîne 1:Binaire Système → "Nombre: Chaîne"
18513	18513	STO interne (2:Tout,1:Nom Global)
18873	18873	AND interne (2:Chaîne,1:Chaîne)
18887	18887	OR interne (2:Chaîne,1:Chaîne)
1889B	1889B	XOR interne (2:Chaîne,1:Chaîne)
188D2	188D2	NOT interne (1:Chaîne)
188E6	188E6	AND interne (2:Chaîne,1:Chaîne)
188F5	188F5	OR interne (2:Chaîne,1:Chaîne)
18904	18904	XOR interne (2:Chaîne,1:Chaîne)
18961	18961	NOT interne (1:Chaîne)
1AB67	1AB67	+ (XLIB 2 68)

1AC93	1AC93	+ interne (2:Tout,1:Liste)
1ACA7	1ACA7	+ interne (2:Chaîne,1:Tout)
1ACBB	1ACBB	+ interne (2:Tout,1:Chaîne)
1ACD7	1ACD7	+
1AD09	1AD09	-
1C85C	1C85C	SUB
1C8BB	1C8BB	SUB interne (3:Chaîne,2:Nombre Réel,1:Nombre Réel)
1C9B8	1C9B8	SIZE
1CA26	1CA26	SIZE interne (1:Chaîne)
1CAB4	1CAB4	POS
1CAD7	1CAD7	POS (2:Chaîne,1:Chaîne)
1CB0B	1CB0B	→STR
1CB26	1CB26	STR→
1CB46	1CB46	NUM
1CB66	1CB66	CHR
1CF7B	1CF7B	OBJ→
225F5	225F5	→TAG interne (2:Tout,1:Chaîne)
238A4	238A4	STR→ (1:Chaîne) → Nom Global et VRAI/FAUX
2D816	2D816	RECN interne (1:Chaîne/Nom Global/Nom Local)
43395	43395	INPUT interne (2:Chaîne,1:Chaîne)
433CC	433CC	INPUT interne (2:Chaîne,1:Liste)
4FAF7	4FAF7	REPL interne (3:Chaîne,2:Nombre Réel,1:Chaîne)
63191	63191	ajoute chr(10) (Line Feed) (1:Chaîne)

Constantes

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03F8B	03F8B	<2933h>
03F95	03F95	<2977h>
03F9F	03F9F	<2A74h>
03FA9	03FA9	<2911h>
03FB3	03FB3	<2D9Dh>
03FBD	03FBD	<2AB8h>

03FC7	03FC7	<2A96h>
03FD1	03FD1	<2E6Dh>
03FDB	03FDB	<2955h>
03FE5	03FE5	<2ADAh>
03FF9	03FF9	<1h>
04003	04003	<2h>
0400D	0400D	<3h>
04017	04017	<4h>
04021	04021	<5h>
0402B	0402B	<6h>
04035	04035	<7h>
0403F	0403F	<8h>
04049	04049	<9h>
04053	04053	<Ah>
0405D	0405D	<Bh>
04067	04067	<Ch>
04071	04071	<Dh>
0407B	0407B	<Eh>
04085	04085	<Fh>
0408F	0408F	<10h>
04099	04099	<11h>
040A3	040A3	<12h>
040AD	040AD	<13h>
040B7	040B7	<14h>
040C1	040C1	<15h>
040CB	040CB	<16h>
040D5	040D5	<17h>
040DF	040DF	<18h>
040E9	040E9	<19h>
040F3	040F3	<1Ah>
040FD	040FD	<1Bh>
04107	04107	<1Ch>
04111	04111	<1Dh>
0411B	0411B	<1Eh>
04125	04125	<1Fh>
0412F	0412F	<20h>
04139	04139	<21h>
04143	04143	<22h>

0414D	0414D	<23h>
04157	04157	<24h>
04161	04161	<25h>
0416B	0416B	<26h>
04175	04175	<27h>
0417F	0417F	<28h>
04189	04189	<29h>
04193	04193	<2Ah>
0419D	0419D	<2Bh>
05176	05176	<FFFFFh>
055D5	055D5	# B02A740000502A2Ch
055F3	055F3	'' (Algébrique)
0DF01	0DF01	'Alarms' (Nom Global)
0DF28	0DF28	'Alarms' (Nom Global)
0E47A	0E47A	'M' (Nom Local)
0E483	0E483	'N' (Nom Local)
0E4A0	0E4A0	'M' (Nom Local)
0E4AE	0E4AE	'N' (Nom Local)
0E4C1	0E4C1	'M' (Nom Local)
0EFEE	0EFEE	8192
0F003	0F003	491520
0F018	0F018	29491200
0F02D	0F02D	707788800
0F042	0F042	4954521600
1439B	1439B	''halt' (Nom Local)
14483	14483	''nohalt' (Nom Local)
15442	15442	'': ''
1576C	1576C	'EQ' (Nom Global)
15777	15777	''
15781	15781	'' (Nom Global)
1613F	1613F	''''
19A72		'ALRMDAT' (Nom Global)
19B1F		'ALRMDAT' (Nom Global)
19DBE		'ALRMDAT' (Nom Global)
1A471	1A471	# 526260410h
1A48A		<7DAC5h>
1A494		<7DDDBh>
1A7CE	1A7CE	8192

1A9F9	1A9F9	# 8010h
1AABD	1AABD	π (XLIB 2 63)
1AB23	1AB23	e (XLIB 2 66)
1AB45	1AB45	i (XLIB 2 67)
1C87A	1C87A	<C55h>
1C889	1C889	<C22h>
1C898		<855h>
1C8A7		<822h>
1C8F4	1C8F4	<C5Ch>
1C903	1C903	<C2Ch>
1C912	1C912	<85Ch>
1C921	1C921	<82Ch>
1C930		<313h>
1C93F		<515h>
1CC03	1CC03	11
1CC1D	1CC1D	12
1CC37	1CC37	13
1CC51	1CC51	14
1CC6B	1CC6B	20
1CC85	1CC85	15
1CCA4	1CCA4	21
1CCB9	1CCB9	<4Fh>
1CCC3	1CCC3	22
1CCD8	1CCD8	<5Fh>
1CCE2	1CCE2	23
1CCF7	1CCF7	<6Fh>
1CD01	1CD01	24
1CD16	1CD16	<7Fh>
1CD20	1CD20	25
1CD3A	1CD3A	16
1CD54	1CD54	17
1CD69	1CD69	<AFh>
1CD73	1CD73	26
1CD8D	1CD8D	27
1CDF2	1CDF2	18
1CE07	1CE07	19
1E460	1E460	<C5Ch>
1E47E	1E47E	<C2Ch>

1E49C	1E49C	<85Ch>
1E4BA	1E4BA	<82Ch>
1F00E	1F00E	# 1234250h
1F024		<7D9DFh>
1F02E		<7D8EAh>
1F038		<7DDA4h>
1F96F		''num' (Nom Local)
1F97E	1F97E	''fcn' (Nom Local)
211B4	211B4	'CST' (Nom Global)
2164C	2164C	SWAP et FAUX interne
21660	21660	SWAP DROP et VRAI interne
225A4	225A4	'S' (Nom Global)
2372E	2372E	''stop' (Nom Local)
2373F	2373F	''noname' (Nom Local)
23754	23754	{ ''noname' ''stop' }
2387E	2387E	''ioinprogress' (Nom Local)
23908	23908	'st' (Nom Local)
23913	23913	'ofs' (Nom Local)
23920	23920	'tok' Nom Local)
2394B	2394B	'st' (Nom Local)
23956	23956	'ofs' (Nom Local)
23963	23963	'tok' (Nom Local)
24A2D	24A2D	'i' (Nom Local)
24A36	24A36	'j' (Nom Local)
24A5D	24A5D	'i' (Nom Local)
24A6B	24A6B	'j' (Nom Local)
24B0A	24B0A	'j' (Nom Local)
24B1D	24B1D	'i' (Nom Local)
24B30	24B30	'i' (Nom Local)
24BB6	24BB6	'j' (Nom Local)
24BD3	24BD3	'i' (Nom Local)
24BE1	24BE1	'i' (Nom Local)
25A0B	25A0B	'1(Nom Local)
25A16	25A16	''2' (Nom Local)
25A21	25A21	''3' (Nom Local)
25A3B	25A3B	''1' (Nom Local)
25A46	25A46	''2' (Nom Local)
25A51	25A51	''3' (Nom Local)

272CD	272CD	''ttt' (Nom Local)
272DC	272DC	''str' (Nom Local)
272EB	272EB	''ofs' (Nom Local)
272FA	272FA	''tok' (Nom Local)
27309	27309	''rbv' (Nom Local)
27318	27318	''idfflg' (Nom Local)
2732D	2732D	''tmpop' (Nom Local)
27340	27340	''tmppdat' (Nom Local)
27357	27357	''ploc' (Nom Local)
27368	27368	''bv' (Nom Local)
27375	27375	''unbound' (Nom Local)
2A2B4	2A2B4	0
2A2C9	2A2C9	1
2A2DE	2A2DE	2
2A2F3	2A2F3	3
2A308	2A308	4
2A31D	2A31D	5
2A332	2A332	6
2A347	2A347	7
2A35C	2A35C	8
2A371	2A371	9
2A386	2A386	-1
2A39B	2A39B	-2
2A3B0	2A3B0	-3
2A3C5	2A3C5	-4
2A3DA	2A3DA	-5
2A3EF	2A3EF	-6
2A404	2A404	-7
2A419	2A419	-8
2A42E	2A42E	-9
2A443	2A443	3.14159265359
2A458	2A458	3.14159265358979 (Réel Long)
2A472	2A472	9.99999999999E499
2A487	2A487	-9.99999999999E499
2A49C	2A49C	1.E-499
2A4B1	2A4B1	-1.E-499
2A4C6	2A4C6	0 (Réel Long)
2A4E0	2A4E0	1 (Réel Long)

2A4FA	2A4FA	2 (Réel Long)
2A514	2A514	3 (Réel Long)
2A52E	2A52E	4 (Réel Long)
2A548	2A548	5 (Réel Long)
2A562	2A562	.1 (Réel Long)
2A57C	2A57C	.5 (Réel Long)
2A596	2A596	10 (Réél Long)
2C1FD	2C1FD	'∑DAT' (Nom Global)
2C738		'∑PAR' (Nom Global)
2D3A0	2D3A0	"PKNO' (Nom Local)
2D3B1	2D3B1	"PACKET' (Nom Local)
2D3C6	2D3C6	"RETRY' (Nom Local)
2D3D9	2D3D9	"ERRMSG' (Nom Local)
2D3EE	2D3EE	"KP' (Nom Local)
2D3FB	2D3FB	"LNAME' (Nom Local)
2D40E	2D40E	"OBJ' (Nom Local)
2D41D	2D41D	"OPOS' (Nom Local)
2D42E	2D42E	"EXCHP' (Nom Local)
2D45A	2D45A	"KLIST' (Nom Local)
2D46D	2D46D	"KMODE' (Nom Local)
2D480	2D480	"KPTRN' (Nom Local)
2D493	2D493	"KRM' (Nom Local)
2D4A2	2D4A2	"MaxR' (Nom Local)
2E9D5	2E9D5	'IOPAR' (Nom Global)
2EA59	2EA59	'IOPAR' (Nom Global)
2F211	2F211	"KML' (Nom Local)
30794	30794	"HPHP48-M" (Selon le modèle)
31C37	31C37	"IWrap' (Nom Local)
31F87	31F87	'PRTPAR' (Nom Global)
31FB8	31FB8	'PRTPAR' (Nom Global)
34D30	34D30	" (Nom Local)
34DBB	34DBB	"symb' (Nom Global)
36BF6		'#a' (Nom Local)
36C01		'#b' (Nom Local)
36C2F		'#b' (Nom Local)
36CEF		'#b' (Nom Local)
36D18		'#b' (Nom Local)
38A3E	38A3E	"SavedUI' (Nom Local)

3FACF	3FACF	'SKEY' (Nom Global)
3FAE8	3FAE8	'SKEY' (Nom Local)
41125	41125	<FFFFBh>
4093B	4093B	'αENTER' (Nom Global)
409DF	409DF	'βENTER' (Nom Global)
41A43	41A43	'UserKeys' (Nom Global)
41A69	41A69	'UserKeys.CRC' (Nom Global)
41BD7	41BD7	'S' (Nom Global)
41BEA	41BEA	'S' (Nom Local)
4353E	4353E	'ALG' (Nom Global)
43555	43555	'ALG' (Nom Local)
4358A	4358A	'.' (Nom Global)
4359D	4359D	'.' (Nom Local)
435CE	435CE	'V' (Nom Global)
435E1	435E1	'V' (Nom Local)
47459	47459	'X' (Nom Global)
48D4B		'ALRMDAT' (Nom Global)
4A145		'X' (Nom Global)
4A19E		'X' (Nom Global)
4A1DE		'X' (Nom Global)
4A22D		'X' (Nom Global)
4A25E		'X' (Nom Global)
4AB1C	4AB1C	'X' (Nom Global)
4AB2A	4AB2A	(0,0)
4AB59	4AB59	'Y' (Nom Global)
4C944		''xmax' (Nom Local)
4C955		''N' (Nom Local)
4CF55	4CF55	''EnvOK' (Nom Local)
4CF68	4CF68	''EXITFCN' (Nom Local)
4D30D		''EnvOK' (Nom Local)
4D352		''EnvOK' (Nom Local)
4D36F		''EnvOK' (Nom Local)
4FF9D	4FF9D	''xe' (Nom Local)
4FFAA	4FFAA	''ye' (Nom Local)
4FFB7	4FFB7	''x' (Nom Local)
4FFC2	4FFC2	''y' (Nom Local)
4FFCD	4FFCD	''xc' (Nom Local)
4FFDA	4FFDA	''yc' (Nom Local)

4FFE7	4FFE7	"r2' (Nom Local)
50005	50005	"up' (Nom Local)
50012	50012	"exit' (Nom Local)
505B2	505B2	Graphic 0x0
50D3E	50D3E	"PlotEnv' (Nom Local)
50FCE	50FCE	'X' (Nom Global)
50FE6	50FE6	'Y' (Nom Global)
51288	51288	'PPAR' (Nom Global)
51436		's1' (Nom Global)
5190B		"PlotEnv' (Nom Local)
524AF	524AF	(0,0)
524F7	524F7	(1,0)
5251C	5251C	MAXR
5267F	5267F	(0,1)
526AE	526AE	(0,-1)
52D26	52D26	{ " " " " } (Noms Locaux)
5456F	5456F	"tcls' (Nom Local)
54580	54580	"fcls' (Nom Local)
5460E	5460E	"tcls' (Nom Local)
54624	54624	"fcls' (Nom Local)
5465D	5465D	"tcls' (Nom Local)
5466E	5466E	"fcls' (Nom Local)
54954	54954	Dérivation complète interne- (2:Symbolique,1:Symbolique)
549CC	549CC	{ "dvar' }
549DB	549DB	"dvar'(Nom Local)
54CDB	54CDB	MINR interne (1.E-499)
54D12	54D12	MAXR interne (9.99999999999E499)
54D35	54D35	π interne (3.14159265359)
54D58	54D58	i (0,1) Interne
54D7B	54D7B	e (2.71828182846) interne
54DD0	54DD0	"xSYMfcn' (Nom Local)
54DE7	54DE7	"xfcn' (Nom Local)
5566B	5566B	"oth' (Nom Local)
55783	55783	"scl' (Nom Local)
55792	55792	"xSYMfcn' (Nom Local)
557A9	557A9	"xfcn' (Nom Local)
55800	55800	"xSYMfcn' (Nom Local)

55817	55817	''xfcn' (Nom Local)
56859	56859	'IERR' (Nom Global)
56976	56976	''sumexpr' (Nom Local)
5698D	5698D	''sumvar' (Nom Local)
56F0B		''dv' (Nom Local)
5720B		''nm' (Nom Local)
57218		''op' (Nom Local)
578E2		''ni' (Nom Local)
578EF		''ns' (Nom Local)
5793F		'n0' (Nom Global)
5795D		's0' (Nom Global)
57EF3	57EF3	''*s' (Nom Local)
58149	58149	''+s' (Nom Local)
58DB6	58DB6	''fl' (Nom Local)
59115	59115	''nmls' (Nom Local)
592C0	592C0	''c' (Nom Local)
592CB	592CB	''b' (Nom Local)
592D6	592D6	''a' (Nom Local)
59304	59304	's1' (Nom Global)
59517	59517	''n' (Nom Local)
59522	59522	''prog' (Nom Local)
59646	59646	''n' (Nom Local)
5A60F	5A60F	{ ''piflag' }
5A614	5A614	''piflag' (Nom Local)
5A665	5A665	''d' (Nom Local)
5A670	5A670	''r' (Nom Local)
5A761	5A761	''d' (Nom Local)
5A76C	5A76C	''R' (Nom Local)
5A777	5A777	''est' (Nom Local)
5A786	5A786	''X' (Nom Local)
5A791	5A791	''T' (Nom Local)
5AAE5	5AAE5	''bnds' (Nom Local)
5D67D	5D67D	''which' (Nom Local)
5D690	5D690	''op1' (Nom Local)
5D69F	5D69F	''op2' (Nom Local)
5FDC1	5FDC1	''ct' (Nom Local)
5FDCE	5FDCE	''pp' (Nom Local)
6080B	6080B	''reg' (Nom Local)

6081A	6081A	"sur' (Nom Local)
60829	60829	"cts' (Nom Local)
60838	60838	"sun' (Nom Local)
60847	60847	"mlg' (Nom Local)
60856	60856	"ckd' (Nom Local)
60865	60865	"prd' (Nom Local)
60874	60874	"prp' (Nom Local)
60883	60883	"rhs' (Nom Local)
60BE9	60BE9	"patternls' (Nom Local)
60C04	60C04	"compos' (Nom Local)
60C19	60C19	"varls' (Nom Local)
60C4F	60C4F	"patternls' (Nom Local)
60C6A	60C6A	"compos' (Nom Local)
60CCA	60CCA	"compos' (Nom Local)
60D7F	60D7F	"varls' (Nom Local)
60E8C	60E8C	'&1' (Nom Local)
60E97	60E97	'&2' (Nom Local)
60EA2	60EA2	'&3' (Nom Local)
60EAD	60EAD	'&4' (Nom Local)
61D3A	61D3A	" (Nom Local)
634F7	634F7	VRAI et FAUX interne
6350B	6350B	FAUX et VRAI interne
63533	63533	<1h> et FAUX interne
63B5A	63B5A	* non évalué
64B12	64B12	<2Ch>
64B1C	64B1C	<2Dh>
64B26	64B26	<2Eh>
64B30	64B30	<2Fh>
64B3A	64B3A	<30h>
64B44	64B44	<31h>
64B4E	64B4E	<32h>
64B58	64B58	<33h>
64B62	64B62	<34h>
64B6C	64B6C	<35h>
64B76	64B76	<36h>
64B80	64B80	<37h>
64B8A	64B8A	<38h>
64B94	64B94	<39h>

64B9E	64B9E	<3Ah>
64BA8	64BA8	<3Bh>
64BB2	64BB2	<3Ch>
64BBC	64BBC	<3Dh>
64BC6	64BC6	<3Eh>
64BD0	64BD0	<3Fh>
64BDA	64BDA	<40h>
64BE4	64BE4	<41h>
64BEE	64BEE	<42h>
64BF8	64BF8	<43h>
64C02	64C02	<44h>
64C0C	64C0C	<45h>
64C16	64C16	<46h>
64C20	64C20	<4Ah>
64C2A	64C2A	<4Fh>
64C34	64C34	<50h>
64C3E	64C3E	<51h>
64C48	64C48	<52h>
64C52	64C52	<53h>
64C5C	64C5C	<54h>
64C66	64C66	<55h>
64C70	64C70	<56h>
64C7A	64C7A	<57h>
64C84	64C84	<5Bh>
64C8E	64C8E	<60h>
64C98	64C98	<61h>
64CA2	64CA2	<62h>
64CAC	64CAC	<64h>
64CB6	64CB6	<65h>
64CC0	64CC0	<6Fh>
64CCA	64CCA	<70h>
64CD4	64CD4	<71h>
64CDE	64CDE	<72h>
64CE8	64CE8	<73h>
64CF2	64CF2	<74h>
64CFC	64CFC	<75h>
64D06	64D06	<7Ah>
64D10	64D10	<80h>

64D1A	64D1A	<82h>
64D24	64D24	<83h>
64D2E	64D2E	<8Fh>
64D38	64D38	<91h>
64D42	64D42	<92h>
64D4C	64D4C	<9Ah>
64D56	64D56	<9Eh>
64D60	64D60	<9Fh>
64D6A	64D6A	<A0h>
64D74	64D74	<A1h>
64D7E	64D7E	<A2h>
64D88	64D88	<A5h>
64D92	64D92	<A6h>
64D9C	64D9C	<A7h>
64DA6	64DA6	<A9h>
64DB0	64DB0	<AAh>
64DBA	64DBA	<AEh>
64DC4	64DC4	<B1h>
64DCE	64DCE	<BBh>
64DD8	64DD8	<C0h>
64DE2	64DE2	<CCh>
64DEC	64DEC	<D0h>
64DF6	64DF6	<E1h>
64E00	64E00	<EAh>
64E0A	64E0A	<EEh>
64E14	64E14	<F0h>
64E1E	64E1E	<FDh>
64E28	64E28	<FFh>
64E32	64E32	<100h>
64E3C	64E3C	<102h>
64E46	64E46	<106h>
64E50	64E50	<107h>
64E5A	64E5A	<110h>
64E64	64E64	<111h>
64E6E	64E6E	<123h>
64E78	64E78	<124h>
64E82	64E82	<131h>
64E8C	64E8C	<132h>

64E96	64E96	<133h>
64EA0	64EA0	<134h>
64EAA	64EAA	<135h>
64EB4	64EB4	<136h>
64EBE	64EBE	<137h>
64EC8	64EC8	<138h>
64ED2	64ED2	<139h>
64EDC	64EDC	<13Ah>
64EE6	64EE6	<13Bh>
64EF0	64EF0	<13Dh>
64EFA	64EFA	<13Eh>
64F04	64F04	<151h>
64F0E	64F0E	<200h>
64F18	64F18	<205h>
64F22	64F22	<311h>
64F2C	64F2C	<411h>
64F36	64F36	<412h>
64F40	64F40	<444h>
64F4A	64F4A	<451h>
64F54	64F54	<452h>
64F5E	64F5E	<510h>
64F68	64F68	<511h>
64F72	64F72	<550h>
64F7C	64F7C	<610h>
64F86	64F86	<650h>
64F90	64F90	<700h>
64F9A	64F9A	<861h>
64FA4	64FA4	<862h>
64FAE	64FAE	<865h>
64FB8	64FB8	<86Eh>
64FC2	64FC2	<A03h>
64FCC	64FCC	<A11h>
64FD6	64FD6	<A12h>
64FE0	64FE0	<A1Ah>
64FEA	64FEA	<A21h>
64FF4	64FF4	<A22h>
64FFE	64FFE	<A2Ah>
65008	65008	<A61h>

65012	65012	<A62h>
6501C	6501C	<A65h>
65026	65026	<A6Eh>
65030	65030	<AA1h>
6503A	6503A	<AA2h>
65044	65044	<AAAh>
6504E	6504E	<C06h>
65058	65058	<C07h>
65062	65062	<C08h>
6506C	6506C	<C0Ah>
65076	65076	<C0Bh>
65080	65080	<DFFh>
6508A	6508A	<E00h>
65094	65094	<70000h>
6509E	6509E	<FFFFFh>
650A8	650A8	2.71828182846
650BD	650BD	.5
650D2	650D2	-.5
650E7	650E7	10
650FC	650FC	180
65111	65111	200
65126	65126	360
6513B	6513B	400
65150	65150	"]"
6515C	6515C	"["
6516A	6516A	"["
65176	65176	"{"
65182	65182	"}"
6518E	6518E	"#"
6519A	6519A	"_"
651A6	651A6	"$"
651B2	651B2	"&"
651BE	651BE	chr(27)
651CA	651CA	"»"
651D6	651D6	"«"
651E2	651E2	"E"
651EE	651EE	"<" (angle)
651FA	651FA	"Σ"

65206	65206	"|"
65212	65212	" " (14 espaces)
65238	65238	chr(10) (Line Feed)
65244	65244	"der"
65254	65254	" "
65260	65260	"UNKNOWN"
65278	65278	""""
65284	65284	"'"
65290	65290	","
6529C	6529C	"."
652A8	652A8	";"
652B4	652B4	"("
652C0	652C0	")"
652CC	652CC	"^"
652D8	652D8	"*"
652E4	652E4	"/"
652F0	652F0	"+"
652FC	652FC	"-"
65308	65308	"="
65314	65314	"√"
65320	65320	"der" (derivée)
6532C	6532C	"GROB"
6533E	6533E	"C$"
6534C	6534C	"0"
65358	65358	"1"
65364	65364	"2"
65370	65370	"3"
6537C	6537C	"4"
65388	65388	"5"
65394	65394	"6"
653A0	653A0	"7"
653AC	653AC	"8"
653B8	653B8	"9"
653C4		<726A5h>
653CE		<72704h>
653D8		<72DCFh>
653E2		<72F1Eh>
653EC		<736F9h>

653F6		<7232Ch>
65400		<7260Ah>
6540A		<72281h>
65414		<72FE6h>
6541E	6541E	Caractère chr(0)
65425	65425	Caractère chr(31) : (...)
6542C	6542C	Caractère "
65433	65433	Caractère #
6543A	6543A	Caractère *
65441	65441	Caractère +
65448	65448	Caractère ,
6544F	6544F	Caractère -
65456	65456	Caractère .
6545D	6545D	Caractère /
65464	65464	Caractère 0
6546B	6546B	Caractère 1
65472	65472	Caractère 2
65479	65479	Caractère 3
65480	65480	Caractère 4
65487	65487	Caractère 5
6548E	6548E	Caractère 6
65495	65495	Caractère 7
6549C	6549C	Caractère 8
654A3	654A3	Caractère 9
654AA	654AA	Caractère :
654B1	654B1	Caractère ;
654B8	654B8	Caractère <
654BF	654BF	Caractère =
654C6	654C6	Caractère >
654CD	654CD	Caractère A
654D4	654D4	Caractère B
654DB	654DB	Caractère C
654E2	654E2	Caractère D
654E9	654E9	Caractère E
654F0	654F0	Caractère F
654F7	654F7	Caractère G
654FE	654FE	Caractère H
65505	65505	Caractère I

6550C	6550C	Caractère J
65513	65513	Caractère K
6551A	6551A	Caractère L
65521	65521	Caractère M
65528	65528	Caractère N
6552F	6552F	Caractère O
65536	65536	Caractère P
6553D	6553D	Caractère Q
65544	65544	Caractère R
6554B	6554B	Caractère S
65552	65552	Caractère T
65559	65559	Caractère U
65560	65560	Caractère V
65567	65567	Caractère W
6556E	6556E	Caractère X
65575	65575	Caractère Y
6557C	6557C	Caractère Z
65583	65583	Caractère a
6558A	6558A	Caractère b
65591	65591	Caractère c
65598	65598	Caractère d
6559F	6559F	Caractère e
655A6	655A6	Caractère f
655AD	655AD	Caractère g
655B4	655B4	Caractère h
655BB	655BB	Caractère i
655C2	655C2	Caractère j
655C9	655C9	Caractère k
655D0	655D0	Caractère l
655D7	655D7	Caractère m
655DE	655DE	Caractère n
655E5	655E5	Caractère o
655EC	655EC	Caractère p
655F3	655F3	Caractère q
655FA	655FA	Caractère r
65601	65601	Caractère s
65608	65608	Caractère t
6560F	6560F	Caractère u

65616	65616	Caractère v
6561D	6561D	Caractère w
65624	65624	Caractère x
6562B	6562B	Caractère y
65632	65632	Caractère z
65639	65639	Caractère chr(141) : →
65640	65640	Caractère chr(171) : «
65647	65647	Caractère chr(187) : »
6564E	6564E	Caractère chr(128) : (angle)
65655	65655	Caractère chr(136) : der
6565C	6565C	Caractère chr(132) : ∫
65663	65663	Caractère (
6566A	6566A	Caractère chr(10) : (Line Feed)
65671	65671	Caractère chr(135) : π
65678	65678	Caractère)
6567F	6567F	Caractère chr(133) : Σ
65686	65686	Caractère (espace)
6568D	6568D	Caractère _
65694	65694	Caractère [
6569B	6569B	Caractère]
656A2	656A2	Caractère {
656A9	656A9	Caractère }
656B0	656B0	Caractère chr(137) : ≤
656B7	656B7	Caractère chr(138) : ≥
656BE	656BE	Caractère chr(139) : ≠
656C5	656C5	"R<<" (affichage angulaire)
656D5	656D5	"R<Z" (affichage angulaire)
656E5	656E5	"XYZ"
656F5	656F5	"«»"
65703	65703	"{}"
65711	65711	"[]"
6571F	6571F	""""
6572D	6572D	"::"
6573B	6573B	"()"
65749	65749	""""""
65757	65757	"ECHO"
65769	65769	"EXIT"
6577B	6577B	"Undefined"

65797	65797	"RAD"
657A7	657A7	"GRAD"
69A97		"Radix' (Nom Local)
69AAA		"KeysOK?' (Nom Local)
69AC1		"ExprLit' (Nom Local)
69AD8		"BuffW' (Nom Local)
69AEB		"BuffH' (Nom Local)
69AFE		"SaveBlank' (Nom Local)
69B19		"ManOp' (Nom Local)
69B2C		"nohalt' (Nom Local)
69B41		"AppMode' (Nom Local)
69B58		"NameGrob' (Nom Local)
69B71		"EXITFCN' (Nom Local)
69B88		"FontGauge' (Nom Local)
69BA3		"LE' (Nom Local)
69BB0		"LB' (Nom Local)
69BBD		"TE' (Nom Local)
69BCA		"FormEnvOK' (Nom Local)
69BE5		"prow' (Nom Local)
69BF6		"pcol' (Nom Local)
69C07		"cursy' (Nom Local)
69C1A		"cursx' (Nom Local)
69C2D		"ttt' (Nom Local)
69C3C		"source' (Nom Local)
69C51		"ofs' (Nom Local)
69C60		"tok' (Nom Local)
69C6F		"rbv' (Nom Local)
69C7E		"idfflg' (Nom Local)
69C93		"tmpop' (Nom Local)
69CA6		"tmppdat' (Nom Local)
69CBD		"ploc' (Nom Local)
69CCE		"bv' (Nom Local)
69CDB		"unbound' (Nom Local)

Listes

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX
0521F	0521F	+ interne (2:Liste,1:Liste)
052FA	052FA	+ interne (2:Liste,1:Tout)
05459	05459	→LIST interne (N..2:Tout,1:Binaire Système)
055E9	055E9	{ }
056B6	056B6	GET interne sur un objet quelconque
05902	05902	SIZE interne (1:Tout) → Binaire Système
05EC7	05EC9	OBJ→ interne (1:Signé)
05EEA	05EEC	change l'en-tête du premier élement de la liste en Nom Global
140AB	140AB	DISP interne (2:Tout,1:Nombre Réel)
15978	15978	→STR interne (1:Tout)
15B13	15B13	→STR interne (1:Tout)
15B31	15B31	→STR interne (1:Tout)
18513	18513	STO interne (2:Tout,1:Nom Global)
18706	18706	TVARS interne (1:Liste)
18EBA	18EBA	EVAL interne (1:Algebraic/Liste/Programme)
19529	19529	Liste de Binaire Système → Réels
1AB67	1AB67	+ (XLIB 2 68)
1AC93	1AC93	+ interne (2:Tout,1:Liste)
1ACA7	1ACA7	+ interne (2:Chaîne,1:Tout)
1ACBB	1ACBB	+ interne (2:Tout,1:Chaîne)
1ACD7	1ACD7	+ (XLIB 2 69)
1AD09	1AD09	- (XLIB 2 70)
1C783	1C783	→LIST (XLIB 2 152)
1C85C	1C85C	SUB (XLIB 2 156)
1C8CF	1C8CF	SUB interne (3:Liste,2:Nombre Réel,1:Nombre Réel)
1C95A	1C95A	LIST→ (XLIB 2 158)
1C973	1C973	explose l'objet (liste,programme) → (N..2:Tout,1:Nombre Réel)
1C9B8	1C9B8	SIZE (XLIB 2 160)
1CAF0	1CAF0	POS interne (2:List,1:Tout)

1D186	1D186	CON (XLIB 2 173)
1D1EA	1D1EA	CON interne (2:Liste,1:Nombre RéelComplexe)
1D221	1D221	CON interne (2:Liste,1:Nombre Complexe)
1D407	1D407	PUT (XLIB 2 176)
1D524	1D524	PUT interne (3:Liste,2:Nombre Réel/Liste,1:Tout)
1D5DF	1D5DF	PUTI (XLIB 2 177)
1D701	1D701	PUTI interne (3:List,2:Nombre Réel/Liste,1:Tout)
1D7C6	1D7C6	GET (XLIB 2 178)
1D898	1D898	GET interne (2:Liste,1:Nombre Réel/Liste)
1D8C7	1D8C7	GETI (XLIB 2 179)
1D9BC	1D9BC	GETI interne (2:Liste,1:Nombre Réel/Liste)
35CAE	35CAE	CON interne (2:Liste,1:Nombre Réel)
4FA7A	4FA7A	REPL interne (3:Liste,2:Nombre Reel,1:Liste)
62B88	62B88	LIST→ interne (1:Liste) → (N..1:Tout) (pas de compteur)
62C41	62C41	LIST→ interne (1:List) → (N..2:Tout, 1:Binaire Système)
631A5	631A5	-1 et →LIST interne (N..2:Tout,1:Binaire Système)
631B9	631B9	2 →LIST interne
631CD	631CD	3 →LIST interne
631E1	631E1	DUP et LIST→ interne
631F5	631F5	SWAP et LIST→ interne

Matrices et Vecteurs

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03562	03562	Nombre d'éléments d'une matrice (1:Matrice) → Binaire Système
0358F	0358F	En-tête des éléments d'une matrice (1:Matrice) → Binaire Système
035A9	035A9	SIZE interne (1:Matrice) → Liste de deux Binaires Système

03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX
052FA	052FA	+ interne (2:Liste,1:Tout)
056B6	056B6	GET interne sur un objet quelconque
05902	05902	SIZE interne (1:Tout) → Binaire Système
0764E	0764E	Stocke les messages d'erreurs (1:Matrice,2:Binaire Système)
140AB	140AB	DISP interne (2:Tout,1:Nombre Réel)
15978	15978	→STR interne (1:Tout)
15B13	15B13	→STR interne (1:Tout)
15B31	15B31	→STR interne (1:Tout)
18513	18513	STO interne (2:Tout,1:Nom Global)
194BB	194BB	verifie la Matrice Réele (1:Matrice)
1AB67	1AB67	+ (XLIB 2 68)
1AC93	1AC93	+ interne (2:Tout,1:Liste)
1ACA7	1ACA7	+ interne (2:Chaîne,1:Tout)
1ACBB	1ACBB	+ interne (2:Tout,1:Chaîne)
1ACD7	1ACD7	+ (XLIB 2 69)
1AD09	1AD09	- (XLIB 2 70)
1ADEE	1ADEE	* (XLIB 2 71)
1AF05	1AF05	/ (XLIB 2 72)
1B278	1B278	INV (XLIB 2 76)
1B426	1B426	SQ (XLIB 2 80)
1BFDE	1BFDE	DET (XLIB 2 120)
1BFFE	1BFFE	DOT (XLIB 2 121)
1C01E	1C01E	CROSS (XLIB 2 122)
1C03E	1C03E	RSD (XLIB 2 123)
1C9B8	1C9B8	SIZE (XLIB 2 160)
1CA4E	1CA4E	SIZE interne (1:Matrice)
1CDB1	1CDB1	TYPE interne (1:Matrice)
1D009	1D009	→ARRY (XLIB 2 170)
1D02C	1D02C	→ARRY interne (1:Nombre Réel)
1D040	1D040	→ARRY interne (1:Liste)
1D092	1D092	ARRY→ (XLIB 2 171)
1D0AB	1D0AB	ARRY→ interne (1:Matrice)
1D0DF	1D0DF	RDM (XLIB 2 172)
1D10C	1D10C	RDM interne (2:Matrice,1:Liste)
1D125	1D125	RDM interne (2:Nom Global,1:Liste)
1D152	1D152	RDM interne (2:Nom Local,1:Liste)

1D186	1D186	CON (XLIB 2 173)
1D2DC	1D2DC	IDN (XLIB 2 174)
1D313	1D313	IDN interne (1:Nombre Réel)
1D34A	1D34A	IDN interne (1:Nom Global)
1D36D	1D36D	IDN interne (1:Nom Local)
1D392	1D392	TRN (XLIB 2 175)
1D3BF	1D3BF	TRN interne (1:Nom Global)
1D3E2	1D3E2	TRN interne (1:Nom Local)
1D407	1D407	PUT (XLIB 2 176)
1D4DE	1D4DE	PUT interne (3:Matrice,2:Nombre Réel/Liste, 1:Nombre)
1D5DF	1D5DF	PUTI (XLIB 2 177)
1D6B6	1D6B6	PUTI interne (3:Matrice,2:Nombre Réel/Liste, 1:Nombre)
1D7C6	1D7C6	GET (XLIB 2 178)
1D86B	1D86B	GET interne (2:Matrice,1:Nombre Réel/Liste)
1D8C7	1D8C7	GETI (XLIB 2 179)
1D96C	1D96C	GETI interne (2:Matrice,1:Nombre Réel/Liste)
1DD06	1DD06	V→ (XLIB 2 180)
1DD29	1DD29	V→ interne (1:Nombre Complexe)
1DD3D	1DD3D	V→ interne (1:Matrice)
1DE66	1DE66	→V2 (XLIB 2 181)
1DE7F	1DE7F	→V2 interne (2:Nombre Réel,1:Nombre Réel)
1DEC2	1DEC2	→V3 (XLIB 2 182)
1DEDB	1DEDB	→V3 interne (3:Nombre Réel,2:Nombre Réel, 1:Nombre Réel)
2C32E	2C32E	Σ+ interne (1:Matrice)
2C684	2C675	COLΣ interne (2:Nombre Réel/Matrice,1: Réel)
35D35	35D35	IDN interne (1:Matrice)
35DEB	35DEB	NEG interne (1:Matrice)
35E2C	35E2C	RND interne (2:Matrice,1:Nombre Réel)
35EA9	35EA9	TRNC interne (2:Matrice,1:Nombre Réel)
35F30	35F30	CONJ interne (1:Matrice)
35F8F	35F8F	RE interne (1:Matrice)
35FEE	35FEE	IM interne (1:Matrice)
36039	36039	R→C interne (2:Matrice,1:Matrice)
360B6	360B6	C→R interne (1:Matrice)
36115	36115	+ interne (2:Matrice,1:Matrice)

36278	36278	- interne (2:Matrice,1:Matrice)
362DC	362DC	* interne (2/1:Nombre Réel/Nombre Complexe, 1/2:Matrice)
363CC	363DB	/ interne (2:Matrice,1:Nombre Réel/Complexe)
36435	36444	SQ interne (1:Matrice)
3643F	3644E	* interne (2:Matrice,1:Matrice)
365AC	365BB	RSD interne (3:Matrice,2:Matrice,1:Matrice)
366F6	36705	DOT interne (2:Matrice,1:Matrice)
36782	36791	CROSS interne (2:Matrice,1:Matrice)
368E5	368F4	CNRM interne (1:Matrice)
3690D	3690D	RNRM interne (1:Matrice)
369CB	369E9	ABS interne (1:Matrice)
36A2A	36A48	DET interne (1:Matrice)
36B0B	36A99	INV interne (1:Matrice)
36B60	36AC3	/ interne (2:Matrice,1:Matrice)
3811F	3811F	TRN interne (1:Matrice)

Nombres et Fonctions

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX
03DBC	03DBC	+ interne (2:Binaire Système,1:Binaire Système)
03DE0	03DE0	- interne (2:Binaire Système,1:Binaire Système)
03DEF	03DEF	+1 interne (1:Binaire Système)
03E0E	03E0E	-1 interne (1:Binaire Système)
03E2D	03E2D	+2 interne (1:Binaire Système)
03E4E	03E4E	-2 interne (1:Binaire Système)
03E6F	03E6F	*2 interne (1:Binaire Système)
03E8E	03E8E	/2 interne (1:Binaire Système)
03EB1	03EB1	AND interne (2:Binaire Système,1:Binaire Système)
03EC2	03EC2	* (2:Binaire Système,1:Binaire Système)
03EF7	03EF7	/ interne (2,1:Binaire Système) → (2:reste,1:quotient)
04DD7	04DD7	découpe (1:Binaire Système <abcde>) →

		2:<cde> 1:<ab>
052FA	052FA	+ interne (2:Liste,1:Tout)
05C27	05C27	R→C interne (2:Nombre Réel,1:Nombre Réel)
05C8A	05C72	Longs Réels → Long Complexe (2,1:Nombres Réels)
05D2C	05D2C	C→R interne (1:Nombre Complexe)
05DBC	05DBC	Long Complexe → Longs Réels (1:Long Complexe)
0F584	0F584	== interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F598	0F598	≠ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5AC	0F5AC	< interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5C0	0F5C0	> interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5D4	0F5D4	≤ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5E8	0F5E8	≥ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F6A2	0F6A2	+ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F774	0F774	- interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F792	0F792	* interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F823	0F823	/ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F878	0F873	^ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FB6F	0FB6F	MAX interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FB8D	0FB8D	MIN interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FBAB	0FBAB	% interne (2:Unité,1:Nombre Réel)
0FC3C	0FC3C	%CH interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FCCD	0FCCD	%T interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FD68	0FD68	RND interne (2:Unité,1:Nombre Réel)
0FD8B	0FD8B	TRNC interne (2:Unité,1:Nombre Réel)
140AB	140AB	DISP interne (2:Tout,1:Nombre Réel)
1415F	1415F	*1000 (1:Nombre Réel)
15978	15978	→STR interne (1:Tout)
15B13	15B13	→STR interne (1:Tout)
15B31	15B31	→STR interne (1:Tout)
15B3D	15B3D	→STR interne (1:Nombre Réel)
162AC	162AC	→STR interne (1:Nombre Réel)
162B8	162B8	→STR interne (1:Nombre Réel)
18513	18513	STO interne (2:Tout,1:Nom Global)
18CD7	18CD7	ABS interne (1:Nombre Réel) → Binaire Système
18CEA	18CEA	→Binaire Système (1:Nombre Réel)
18DBF	18DBF	→Nombre Réel (1:Binaire Système)
194F7	194F7	Deux Nombres Réels → Binaire Système
1950B	1950B	Deux Binaire Système → Nombres Réels

19529	19529	Liste de Binaire Système → Réels
1969B	1969B	R→B (XLIB 2 9)
196BB	196BB	B→R (XLIB 2 10)
1A5E4	1A5E4	→NUM (XLIB 2 53)
1A995	1A995	NEG (XLIB 2 60)
1AA1F	1AA1F	ABS (XLIB 2 61)
1AA6E	1AA6E	CONJ (XLIB 2 62)
1AADF	1AADF	MAXR (XLIB 2 64)
1AB01	1AB01	MINR (XLIB 2 65)
1AB67	1AB67	+ (XLIB 2 68)
1AC93	1AC93	+ interne (2:Tout,1:Liste)
1ACA7	1ACA7	+ interne (2:Chaîne,1:Tout)
1ACBB	1ACBB	+ interne (2:Tout,1:Chaîne)
1ACD7	1ACD7	+ (XLIB 2 69)
1AD09	1AD09	- (XLIB 2 70)
1ADEE	1ADEE	* (XLIB 2 71)
1AF05	1AF05	/ (XLIB 2 72)
1B02D	1B02D	^ (XLIB 2 73)
1B124	1B124	^ interne (2:Nombre Réel,1:Nombre Réel)
1B185	1B185	XROOT (XLIB 2 74)
1B1CA	1B1CA	XROOT (XLIB 2 75)
1B278	1B278	INV (XLIB 2 76)
1B30D	1B30D	ARG interne (1:Nombre Réel)
1B32A	1B32A	SIGN (XLIB 2 78)
1B374	1B374	SQRT (XLIB 2 79)
1B3F5	1B3F5	SQRT interne (1:Nombre Réel)
1B426	1B426	SQ (XLIB 2 80)
1B47B	1B47B	SQ interne (1:Nombre Réel)
1B48F	1B48F	SQ interne (1:Nombre Complexe)
1B4AC	1B4AC	SIN (XLIB 2 81)
1B505	1B505	COS (XLIB 2 82)
1B55E	1B55E	TAN (XLIB 2 83)
1B5B7	1B5B7	SINH (XLIB 2 84)
1B606	1B606	COSH (XLIB 2 85)
1B655	1B655	TANH (XLIB 2 86)
1B6A4	1B6A4	ASIN (XLIB 2 87)
1B6EA	1B6EA	ASIN interne (1:Nombre Réel)
1B72F	1B72F	ACOS (XLIB 2 88)

1B775	1B775	ACOS interne (1:Nombre Réel)
1B79C	1B79C	ATAN (XLIB 2 89)
1B7EB	1B7EB	ASINH (XLIB 2 90)
1B830	1B830	ACOSH (XLIB 2 91)
1B86C	1B86C	ACOSH interne (1:Nombre Réel)
1B8A2	1B8A2	ATANH (XLIB 2 92)
1B8DE	1B8DE	ATANH interne (1:Nombre Réel)
1B905	1B905	EXP (XLIB 2 93)
1B94F	1B94F	LN (XLIB 2 94)
1B995	1B995	LN interne (1:Nombre Réel)
1B9C6	1B9C6	LOG (XLIB 2 95)
1BA0C	1BA0C	LOG interne (1:Nombre Réel)
1BA3D	1BA3D	ALOG (XLIB 2 96)
1BA8C	1BA8C	LNP1 (XLIB 2 97)
1BAC2	1BAC2	EXPM (XLIB 2 98)
1BB41	1BB41	FACT (XLIB 2 100)
1BB6D	1BB6D	IP (XLIB 2 101)
1BBA3	1BBA3	FP (XLIB 2 102)
1BBD9	1BBD9	FLOOR (XLIB 2 103)
1BC0F	1BC0F	CEIL (XLIB 2 104)
1BC45	1BC45	XPON (XLIB 2 105)
1BC71	1BC71	MAX (XLIB 2 106)
1BCE3	1BCE3	MIN (XLIB 2 107)
1BD55	1BD55	RND (XLIB 2 108)
1BDD1	1BDD1	TRNC (XLIB 2 109)
1BE4D	1BE4D	MOD (XLIB 2 110)
1BE9C	1BE9C	MANT (XLIB 2 111)
1C79E	1C79E	R→C (XLIB 2 153)
1C7CA	1C7CA	RE (XLIB 2 154)
1C819	1C819	IM (XLIB 2 155)
1C98E	1C98E	C→R (XLIB 2 159)
1CA85	1CA85	SIZE interne (1:Entier Binaire)
1E783	1E783	AND (XLIB 2 229)
1E7DD	1E7DD	AND interne (2:Nombre Réel,1:Nombre Réel)
1E809	1E809	OR (XLIB 2 230)
1E863	1E863	OR interne (2:Nombre Réel,1:Nombre Réel)
1E88F	1E88F	NOT (XLIB 2 231)
1E8D9	1E8D9	NOT interne (1:Nombre Réel)

1E8F6	1E8F6	XOR (XLIB 2 232)
1E946	1E946	XOR interne (2:Nombre Réel,1:Nombre Réel)
1EA6C	1EA6C	== interne (2:Nombre Réel,1:Nombre Complexe)
1EA76	1EA76	== interne (2:Nombre Complexe,1:Nombre Réel)
1EB8D	1EB8D	≠ interne (2:Nombre Réel,1:Nombre Complexe)
1EB97	1EB97	≠ interne (2:Nombre Complexe,1:Nombre Réel)
1ED7E	1ED7E	≤ interne (2:Nombre Réel,1:Nombre Réel)
1ED9B	1ED9B	≥ (XLIB 2 238)
1EE1D	1EE1D	≥ interne (2:Nombre Réel,1:Nombre Réel)
1F9C4	1F9C4	→Q (XLIB 2 263)
1F9E9	1F9E9	→Qπ (XLIB 2 264)
1EC5D	1EC5D	> (XLIB 2 236)
1ECFC	1ECFC	≤ (XLIB 2 237)
22618	22618	→TAG interne (2:Tout,1:Nombre Réel)
2A5B0	2A5B0	Long Réel→Réel interne (1:Long Nombre Réel)
2A5C1	2A5C1	Réel → Long Réel interne (1:Nombre Réel)
2A70E	2A70E	MIN interne (2:Nombre Réel,1:Nombre Réel)
2A871	2A871	< interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A87F	2A87F	> interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A88A	2A88A	> interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A895	2A895	≥ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8A0	2A8A0	≥ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8AB	2A8AB	≤ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8B6	2A8B6	≤ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8C1	2A8C1	== interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8CC	2A8CC	≠ interne (2:Nombre Réel,2:Nombre Réel) → VRAI/FAUX
2A8D7	2A8D7	SIGN interne (1:Nombre Réel)
2A8F0	2A8F0	ABS interne (1:Long Réel)
2A900	2A900	ABS interne (1:Nombre Réel)

2A920	2A920	NEG interne (1:Nombre Réel)
2A930	2A930	MANT interne (1:Nombre Réel)
2A943	2A943	+ interne (2:Long Réel,1:Long Réel)
2A94F	2A94F	- interne (2:Long Réel,1:Long Réel)
2A974	2A974	+ interne (2:Nombre Réel,1:Nombre Réel)
2A981	2A981	- interne (2:Nombre Réel,1:Nombre Réel)
2A99A	2A99A	* interne (2:Long Réel,1:Long Réel)
2A9BC	2A9BC	* interne (2:Nombre Réel,1:Nombre Réel)
2A9C9	2A9C9	% interne (2:Nombre Réel,1:Nombre Réel)
2A9E8	2A9E8	/ interne (2:Long Réel,1:Long Réel)
2A9FE	2A9FE	/ interne (2:Nombre Réel,1:Nombre Réel)
2AA0B	2AA0B	%T interne (2:Nombre Réel,1:Nombre Réel)
2AA30	2AA30	%CH interne (2:Nombre Réel,1:Nombre Réel)
2AA5F	2AA5F	^ interne (2:Long Réel,1:Long Réel)
2AA70	2AA70	^ interne (2:Nombre Réel,1:Nombre Réel)
2AA81	2AA81	XROOT interne (2:Nombre Réel,1:Nombre Réel)
2AA92	2AA92	INV interne (1:Long Réel)
2AAAF	2AAAF	INV interne (1:Nombre Réel)
2AAEA	2AAEA	√ interne (1:Long Réel)
2AB09	2AB09	√ interne (1:Réel)
2AB1C	2AB1C	EXP interne (1:Long Réel)
2AB2F	2AB2F	EXP interne (1:Nombre Réel)
2AB42	2AB42	EXPM interne (1:Nombre Réel)
2AB5B	2AB5B	LN interne (1:Long Réel)
2AB6E	2AB6E	LN interne (1:Réel >0)
2AB81	2AB81	LOG interne (1:Réel >0)
2AB94	2AB94	LNP1 interne (1:Long Réel)
2ABA7	2ABA7	LNP1 interne (1:Nombre Réel)
2ABBA	2ABBA	ALOG interne (1:Nombre Réel)
2ABDC	2ABDC	MOD interne (2:Nombre Réel,1:Nombre Réel)
2ABEF	2ABEF	SIN interne (1:Nombre Réel)
2AC06	2AC06	SIN interne (1:Long Réel)
2AC17	2AC17	TAN interne (1:Long Réel)
2AC40	2AC40	COS interne (1:Nombre Réel)
2AC57	2AC57	COS interne (1:Long Réel)
2AC91	2AC91	TAN interne (1:Nombre Réel)
2ACC1	2ACC1	ASIN interne (1:Nombre Réel ≤ 1)
2ACF1	2ACF1	ACOS interne (1:Nombre Réel ≤ 1)

2AD21	2AD21	ATAN interne (1:Nombre Réel)
2ADAE	2ADAE	SINH interne (1:Nombre Réel)
2ADDA	2ADDA	COSH interne (1:Nombre Réel)
2ADED	2ADED	TANH interne (1:Nombre Réel)
2AE00	2AE00	ASINH interne (1:Nombre Réel)
2AE39	2AE39	XPON interne (1:Nombre Réel)
2AE62	2AE62	COMB interne (2:Nombre Réel,1:Nombre Réel)
2AE75	2AE75	PERM interne (2:Nombre Réel,1:Nombre Réel)
2AF4D	2AF4D	FP interne (1:Nombre Réel)
2AF60	2AF60	IP interne (1:Nombre Réel)
2AF73	2AF73	CEIL interne (1:Nombre Réel)
2AF86	2AF86	FLOOR interne (1:Nombre Réel)
2AF99	2AF99	FLOOR interne (1:Long Réel)
2AFC2	2AFC2	RAND interne
2B044	2B044	RDZ interne (1:Nombre Réel)
2B0C4	2B0C4	! interne (1:Nombre Réel)
2B529	2B529	RND interne (2:Nombre Réel,1:Nombre Réel)
2B53D	2B53D	TRNC interne (2:Nombre Réel,1:Nombre Réel)
2C09F	2C09F	UTPN interne (3:Nombre Réel,2: Réel,1:Nombre Réel)
2C149	2C12E	UTPC interne (2:Nombre Réel,1:Nombre Réel)
2C174	2C15E	UTPF interne (3:Nombre Réel,2:Réel,1:Nombre Réel)
2C19A	2C189	UTPT interne (2:Nombre Réel,1:Nombre Réel)
2C684	2C675	COLΣ interne (2:Nombre Réel/Matrice,1:Réel)
2A6F5	2A6F5	MAX interne (2:Nombre Réel,1:Nombre Réel)
2A81F	2A81F	< interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2EC11	2EC11	ABS(IP(Nombre Réel)) → Binaire Système
35EC2	35EC2	RND interne (2:Nombre Complexe,1:Nombre Réel)
35F17	35F17	TRNC interne (2:Nombre Complexe,1:Nombre Réel)
362DC	362DC	* interne (2/1:Nombre Réel/Complexe,1/2:Matrice)
363CC	363DB	/ interne (2:Matrice,1:Nombre Réel Complexe)
4F3D1	4F3D1	Entier Binaire → Binaire Système (1,2:Entier Binaire)
50262	50262	+1 interne (1:real)
5198F	5198F	IM interne (1:Nombre Réel)
519A3	519A3	RE interne (1:Nombre Complexe)
519B7	519B7	IM interne (1:Nombre Complexe)
51A07	51A07	Long Réel→Complexe (1:Long Réel,2:Long Réel)
51A37	51A37	Nombre Réel→Complexe (change type)

51B70	51B70	NEG interne (1:Nombre Complexe)
51BB2	51BB2	CONJ interne (1:Nombre Complexe)
51BD0	51BD0	+ interne (2:Nombre Complexe,1:Nombre Réel)
51BF8	51BF8	+ interne (2:Nombre Réel,1:Nombre Complexe)
51C16	51C16	+ interne (2:Nombre Complexe,1:Nombre Complexe)
51CD4	51CD4	- interne (2:Nombre Réel,1:Nombre Complexe)
51CE8	51CE8	- interne (2:Nombre Complexe,1:Nombre Réel)
51CFC	51CFC	- interne (2:Nombre Complexe,1:Nombre Réel)
51D4C	51D4C	* interne (2:Nombre Complexe,1:Nombre Réel)
51D60	51D60	* interne (2:Nombre Réel,1:Nombre Complexe)
51D88	51D88	* interne (2:Nombre Complexe,1:Nombre Complexe)
51E19	51E19	/ interne (2:Nombre Réel,1:Nombre Complexe)
51E64	51E64	/ interne (2:Nombre Complexe,1:Nombre Réel)
51EC8	51EC8	/ interne (2:Nombre Complexe,1:Nombre Complexe)
51EFA	51EFA	INV interne (1:Nombre Complexe)
52062	52062	ABS interne (1:Nombre Complexe)
52099	52099	ARG interne (1:Nombre Complexe)
520CB	520CB	SIGN interne (1:Nombre Complexe)
52107	52107	√ interne (1:Nombre Complexe)
52193	52193	EXP interne (1:Nombre Complexe)
521E3	521E3	LN interne (1:Nombre Complexe)
522BF	522BF	LOG interne (1:Nombre Complexe)
52305	52305	ALOG interne (1:Nombre Complexe)
52342	52342	^ interne (2:Nombre Réel,1:Nombre Complexe)
52360	52360	^ interne (2:Nombre Complexe,1:Nombre Réel)
52374	52374	^ interne (2:Nombre Complexe,1:Nombre Complexe)
52530	52530	SIN interne (1:Nombre Complexe)
52571	52571	COS interne (1:Nombre Complexe)
525B7	525B7	TAN interne (1:Nombre Complexe)
5262F	5262F	SINH interne (1:Nombre Complexe)
52648	52648	COSH interne (1:Nombre Complexe)
5265C	5265C	TANH interne (1:Nombre Complexe)
52675	52675	ATAN interne (1:Nombre Complexe)
527EB	527EB	ATANH interne (1:Nombre Complexe)
52804	52804	ASIN interne (1:Nombre Complexe)
5281D	5281D	ASINH interne (1:Nombre Complexe)
52836	52836	ACOSH interne (1:Nombre Complexe)
52863	52863	ACOS interne (1:Nombre Complexe)

53D04	53D04	AND interne (2:Entier Binaire,1:Entier Binaire)
53D15	53D15	OR interne (2:Entier Binaire,1:Entier Binaire)
53D26	53D26	XOR interne (2:Entier Binaire,1:Entier Binaire)
53D4E	53D4E	NOT interne (1:Entier Binaire)
53EA0	53EA0	+ interne (2:Entier Binaire,1:Entier Binaire)
53EB0	53EB0	- interne (2:Entier Binaire,1:Entier Binaire)
53EC3	53EC3	NEG interne (1:Entier Binaire)
53ED3	53ED3	* interne (2:Entier Binaire,1:Entier Binaire)
53F05	53F05	/ interne (2:Entier Binaire,1:Entier Binaire)
5429F	5429F	/ interne (2:Nombre Réel,1:Entier Binaire)
542BD	542BD	/ interne (2:Entier Binaire,1:Nombre Réel)
542D1	542D1	* interne (2:Nombre Réel,1:Entier Binaire)
542EA	542EA	* interne (2:Entier Binaire,1:Nombre Réel)
542FE	542FE	- interne (2:Nombre Réel,1:Entier Binaire)
5431C	5431C	- interne (2:Entier Binaire,1:Nombre Réel)
54330	54330	+ interne (2:Nombre Réel,1:Entier Binaire)
54349	54349	+ interne (2:Entier Binaire,1:Nombre Réel)
5435D	5435D	B→R interne
543F9	543F9	R→B interne
54422	54422	Nombre Réel → Entier Binaire
54EA0	54EA0	RE interne (1:Symbolique)
54EB9	54EB9	IM interne (1:Symbolique)
54ED2	54ED2	NOT interne (1:Symbolique)
54EEB	54EEB	NEG interne (1:Symbolique)
54F04	54F04	ABS interne (1:Symbolique)
54F1D	54F1D	CONJ interne (1:Symbolique)
54F36	54F36	INV interne (1:Symbolique)
54F4F	54F4F	ARG interne (1:Symbolique)
54F68	54F68	SIGN interne (1:Symbolique)
54F81	54F81	√ interne (1:Symbolique)
54F9A	54F9A	SQ interne (1:Symbolique)
54FB3	54FB3	SIN interne (1:Symbolique)
54FCC	54FCC	COS interne (1:Symbolique)
54FE5	54FE5	TAN interne (1:Symbolique)
54FFE	54FFE	SINH interne (1:Symbolique)
55017	55017	COSH interne (1:Symbolique)
55030	55030	TANH interne (1:Symbolique)
55049	55049	ASIN interne (1:Symbolique)

55062	55062	ACOS interne (1:Symbolique)
5507B	5507B	ATAN interne (1:Symbolique)
55094	55094	ASINH interne (1:Symbolique)
550AD	550AD	ACOSH interne (1:Symbolique)
550C6	550C6	ATANH interne (1:Symbolique)
550DF	550DF	EXP interne (1:Symbolique)
550F8	550F8	LN interne (1:Symbolique)
55111	55111	LOG interne (1:Symbolique)
5512A	5512A	ALOG interne (1:Symbolique)
55143	55143	LNP1 interne (1:Symbolique)
5515C	5515C	EXPM interne (1:Symbolique)
55175	55175	! interne (1:Symbolique)
5518E	5518E	IP interne (1:Symbolique)
551A7	551A7	FP interne (1:Symbilic)
551C0	551C0	FLOOR interne (1:Symbolique)
551D9	551D9	CEIL interne (1:Symbolique)
551F2	551F2	XPON interne (1:Symbolique)
5520B	5520B	MANT interne (1:Symbolique)
55224	55224	D→R interne (1:Symbolique)
5523D	5523D	R→D interne (1:Symbolique)
55256	55256	UBASE interne (1:Symbolique)
5526F	5526F	UVAL interne (1:Symbolique)
55D1E	55D1E	COMB interne (2:Symbolique,1:Nombre Réel)
55D37	55D37	COMB interne (2:Nombre Réel,1:Symbolique)
55D50	55D50	COMB interne (2:Symbolique,1:Symbolique)
55D69	55D69	PERM interne (2:Symbolique,1:Nombre Réel)
55D82	55D82	PERM interne (2:Nombre Réel,1:Symbolique)
55D9B	55D9B	PERM interne (2:Symbolique,1:Symbolique)
55DB4	55DB4	RND interne (2:Symbolique,1:Nombre Réel)
55DCD	55DCD	RND interne (2:Réel/Complexe/Matrice/Unité,1:Symbolique)
55DE6	55DE6	RND interne (2:Symbolique,1:Symbolique)
55DFF	55DFF	TRNC interne (2:Symbolique,1:Nombre Réel)
55E18	55E18	TRNC interne (2:Nombre Réel/Complexe/Matrice/Unité,1:Symb)
55E31	55E31	TRNC interne (2:Symbolique,1:Symbolique)
55E4A	55E4A	MAX interne (2:Symbolique,1:Nombre Réel/Unité)
55E63	55E63	MAX interne (2:Nombre Réel/Unité,1:Symbolique)

55E7C	55E7C	MAX interne (2:Symbolique,1:Symbolique)
55E95	55E95	MIN interne (2:Symbolique,1:Nombre Réel/Unité)
55EAE	55EAE	MIN interne (2:Nombre Réel/Unité,1:Symbolique)
55EC7	55EC7	MIN interne (2:Symbolique,1:Symbolique)
55EE0	55EE0	^ interne (2:Symbolique,1:Réel/Complexe/Unité)
55EF9	55EF9	^ interne (2:Réel/Complexe/Unité,1:Symbolique)
55F12	55F12	^ interne (2:Symbolique,1:Symbolique)
55F2B	55F2B	+ interne (2:Symbolique,1:Réel/Complexe/Unité)
55F44	55F44	+ interne (2:Réel/Complexe/Unité,1:Symbolique)
55F5D	55F5D	+ interne (2:Symbolique,1:Symbolique)
55F76	55F76	- interne (2:Symbolique,1:Réel/Complexe/Unité)
55F8F	55F8F	- interne (2:Réel/Complexe/Unité,1:Symbolique)
55FA8	55FA8	- interne (2:Symbolique,1:Symbolique)
55FC1	55FC1	* interne (2:Symbolique,1: Réel/Complexe/Unité)
55FDA	55FDA	* interne (2:Réel/Complexe/ Unité,1:Symbolique)
55FF3	55FF3	* interne (2:Symbolique,1:Symbolique)
5600C	5600C	/ interne (2:Symbolique,1:Réel/Complexe/Unité)
56025	56025	/ interne (2:Réel/Complexe/Unité,1:Symbolique)
5603E	5603E	/ interne (2:Symbolique,1:Symbolique)
56057	56057	MOD interne (2:Symbolique,1:Nombre Réel)
56070	56070	MOD interne (2:Nombre Réel,1:Symbolique)
56089	56089	MOD interne (2:Symbolique,1:Symbolique)
6256A	6256A	+3 interne (1:Binaire Système)
6257A	6257A	+4 interne (1:Binaire Système)
6258A	6258A	+5 interne (1:Binaire Système)
6259A	6259A	+6 interne (1:Binaire Système)
625AA	625AA	+7 interne (1:Binaire Système)
625BA	625BA	+8 interne (1:Binaire Système)
625CA	625CA	+9 interne (1:Binaire Système)
625DA	625DA	+10 interne (1:Binaire Système)
625EA	625EA	+12 interne (1:Binaire Système)
625FA	625FA	-3 interne (1:Binaire Système)
6260A	6260A	-4 interne (1:Binaire Système)
6261A	6261A	-5 interne (1:Binaire Système)
6262A	6262A	-6 interne (1:Binaire Système)
6264E	6264E	*10 interne (1:Binaire Système)
62674	62674	*8 interne (1:Binaire Système)
62691	62691	*6 interne (1:Binaire Système)

626F7	626F7	DUP et +2 interne (1:Binaire Système)
62BF1	62BF1	*10 interne (1:Nombre Réel)
62CE1	62CE1	R→SB et DUP interne
62E7B	62E7B	Réel→Binaire Système et SWAP interne (1:Réel)
62E8F	62E8F	Réel→Long Réel et SWAP interne (1:Nombre Réel)
632A9	632A9	SWAP et R→C interne (2:Nombre Réel,1:Réel)
63B5A	63B5A	* non évalué
63B96	63B96	Système Binaire→Long Réel interne (1:Binaire Système)

Opérations sur la pile

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
0314C	0314C	DEPTH interne (1:Binaire Système)
03188	03188	DUP interne
031AC	031AC	DUP2 interne
031D9	031D9	DUPN interne (N..2:Tout,1:Binaire Système)
03223	03223	SWAP interne
03244	03244	DROP interne
03258	03258	DROP2 interne
0326E	0326E	DROPN interne (N..2:Tout,1:Binaire Système)
03295	03295	ROT interne
032C2	032C2	OVER interne
032E2	032E2	PICK interne (N..2:Tout,1:Binaire Système)
03325	03325	ROLL interne (N..2:Tout,1:Binaire Système)
0339E	0339E	ROLLD interne (N..2:Tout,1:Binaire Système)
04D57	04D57	DROP et annule une chaîne (2:Chaîne,1:Tout) → ""
05622	05622	OVER et SIZE interne (2:Chaîne) → Binaire Système
188AF	188AF	si SIZE(TOS) = SIZE(TOS-1) (Chaîne), NEWOB et sinon SWAP
18A5B	18A5B	Sauve le dernier RPL et verifie DEPTH ≥ 3
18A68	18A68	Verifie DEPTH ≥ 3
18A80	18A80	Sauve le dernier RPL et verifie DEPTH ≥ 2

18A8D	18A8D	Verifie DEPTH ≥ 2
18AA5	18AA5	Sauve le dernier RPL et verifie DEPTH ≥ 1
18AB2	18AB2	Verifie DEPTH ≥ 1
18B6D	18B6D	Sauve le dernier RPL et verifie DEPTH ≥ 5
18B7A	18B7A	Verifie DEPTH ≥ 5
18B92	18B92	Sauve le dernier RPL et verifie DEPTH ≥ 4
18B9F	18B9F	Verifie DEPTH ≥ 4
18ECE	18ECE	Sauve le dernier RPL, vérifie DEPTH ≥ 1 et contrôle les arguments
18EDF	18EDF	Sauve le dernier RPL, vérifie DEPTH ≥ 2 et contrôle les arguments
18EF0	18EF0	Sauve le dernier RPL, vérifie DEPTH ≥ 3 et contrôle les arguments
18F01	18F01	Sauve le dernier RPL, vérifie DEPTH ≥ 4 et contrôle les arguments
18F12	18F12	Sauve le dernier RPL, vérifie DEPTH ≥ 5 et contrôle les arguments
1F047	1F047	DROP2 et 0 interne
1FB87	1FB87	DUP (XLIB 2 269)
1FBA2	1FBA2	DUP2 (XLIB 2 270)
1FBBD	1FBBD	SWAP (XLIB 2 271)
1FBD8	1FBD8	DROP (XLIB 2 272)
1FBF3	1FBF3	DROP2 (XLIB 2 273)
1FC0E	1FC0E	ROT (XLIB 2 274)
1FC29	1FC29	OVER (XLIB 2 275)
1FC44	1FC44	DEPTH (XLIB 2 276)
1FC64	1FC64	DROPN (XLIB 2 277)
1FC7F	1FC7F	DUPN (XLIB 2 278)
1FC9A	1FC9A	PICK (XLIB 2 279)
1FCB5	1FCB5	ROLL (XLIB 2 280)
1FCD0	1FCD0	ROLLD (XLIB 2 281)
1FCEB	1FCEB	CLEAR (XLIB 2 282)
2164C	2164C	SWAP et FAUX interne
21660	21660	SWAP DROP et VRAI interne
60EE7	60EE7	SWAP des niveaux 2 et 3 interne
60F21	60F21	DROP niveau 3 interne
60F4B	60F4B	DROP3 interne
60F54	60F54	DROP7 interne

60F66	60F66	DROP6 interne
60F72	60F72	DROP5 interne
60F7E	60F7E	DROP4 interne
60F83	60F83	DROP4 interne
60F9B	60F9B	SWAP et DROP interne
60FAC	60FAC	3 ROLL interne
60FBB	60FBB	4 ROLL interne
60FD8	60FD8	5 ROLL interne
61002	61002	6 ROLL interne
6103C	6103C	8 ROLL interne
6106B	6106B	7 ROLL interne
6109E	6109E	4 ROLLD interne
610C4	610C4	5 ROLLD interne
610FA	610FA	6 ROLLD interne
6112A	6112A	SWAP DROP SWAP DROP interne
6113C	6113C	SWAP DROP SWAP DROP SWAP DROP interne
611A3	611A3	+1 et PICK interne (N..2:Tout,1:Binaire Système)
611BE	611BE	+2 et PICK (1:Binaire Système)
611D2	611D2	+3 et PICK (1:Binaire Système)
611E1	611E1	+4 et PICK (1:Binaire Système)
611FE	611FE	3 PICK interne
6121C	6121C	4 PICK interne
6123A	6123A	5 PICK interne
6125E	6125E	6 PICK interne
61282	61282	7 PICK interne
612A9	612A9	8 PICK interne
612CC	612CC	- et ROLL interne (N..3:Tout,2:Binaire Système,1:Binaire Système)
612DE	612DE	+ et ROLL interne (N..3:Tout,2:Binaire Système,1:Binaire Système)
612F3	612F3	+1 et ROLL interne (N..2:Tout,1:Binaire Système)
61305	61305	DUP +2 et ROLL interne (N..2:Tout,1:Binaire Système)
61318	61318	+2 et ROLL interne (N..2:Tout,1:Binaire Système)
6132C	6132C	- et ROLLD interne (N..3:Tout,2:Binaire Système,1:Binaire Système)
6133E	6133E	+ et ROLLD interne (N..3:Tout,2:Binaire Système,1:Binaire Système)
61353	61353	+1 et ROLLD interne (N..2:Tout,1:Binaire Système)

61365	61365	+2 et ROLLD interne (N..2:Tout,1:Binaire Système)
61380	61380	SWAP et OVER interne
626F7	626F7	DUP et +2 interne (1:Binaire Système)
6270C	6270C	DROP et SWAP interne
62726	62726	DROP SWAP et DROP interne
62747	62747	SWAP et DUP interne
62775	62775	ROT et DUP interne
62794	62794	SWAP interne (2,1:Binaire Système)
627A7	627A7	DROP et DUP interne
627BB	627BB	DUP et SIZE interne (1:Chaîne) → (2:Chaîne,1:Binaire Système)
627D5	627D5	+ et DUP interne (2,1:Binaire Système)
62809	62809	+1 et DUP interne (1:Binaire Système)
6281A	6281A	-1 et DUP interne (1:Binaire Système)
62830	62830	SWAP DROP et DUP interne
6284B	6284B	SWAP DROP et SWAP interne
62864	62864	DROP niveau 4 interne
62880	62880	DROP niveau 5 interne
62B0B	62B0B	DROP2 et FAUX interne
62BC4	62BC4	7 ROLLD interne (7:...,1:Tout)
62C69	62C69	NEWOB et SWAP interne
62C7D	62C7D	ROT et DUP2 interne (3:Tout,2:Tout,1:Tout)
62CA5	62CA5	ROT et OVER interne
62CB9	62CB9	DUP DUP interne
62CCD	62CCD	OVER et DUP interne
62CE1	62CE1	R→SB et DUP interne
62D09	62D09	4 ROLLD et DUP interne
62D31	62D31	OVER et SWAP interne
62D45	62D45	ROLL et SWAP interne (N..2:Tout,1:Binaire Système)
62D59	62D59	"" et SWAP interne
62DE5	62DE5	4 PICK + et SWAP interne (1:Binaire Système)
62DFE	62DFE	+ et SWAP interne (1:Binaire Système)
62E26	62E26	+1 et SWAP interne (2:Tout,1:Binaire Système)
62E3A	62E3A	<0h> et SWAP
62E4E	62E4E	-1 <1h> et SWAP interne (1:Binaire Système)
62E67	62E67	<1h> et SWAP
62E7B	62E7B	Réel→Binaire Système et SWAP interne (1: Réel)
62E8F	62E8F	Réel→Long Réel et SWAP interne (1:Nombre Réel)

62ECB	62ECB	4 ROLL et SWAP interne
62EDF	62EDF	OVER et SWAP interne
62EF3	62EF3	4 PICK et SWAP interne
62F75	62F75	DROPN et DROP interne (1:Binaire Système)
62F89	62F89	ROLL et DROP interne (N..2:Tout,1:Binaire Système)
62FB1	62FB1	DUP et ROT interne
62FC5	62FC5	DROP et ROT interne
62FD9	62FD9	-1 et ROT interne (1:Binaire Système)
63029	63029	DROP et OVER interne
63051	63051	+ et OVER interne (1:Binaire Système)
63079	63079	<0h> et OVER
630DD	630DD	DUP et PICK interne (N..2:Tout,1:Binaire Système)
630F1	630F1	DUP et ROLL interne (N..2:Tout,1:Binaire Système)
63119	63119	8 ROLLD interne
6312D	6312D	10 ROLLD interne
632A9	632A9	SWAP et R→C interne (2:Nombre Réel,1:Réel)
63411	63411	DUP et → compteur de boucle Binaire Système
63425	63425	SWAP et → compteur de boucle Binaire Système
63439	63439	OVER et → compteur de boucle Binaire Système
6344D	6344D	SWAP et NEXT (boucle interne) interne
63466	63466	DROP et NEXT (boucle interne) interne
6347F	6347F	DUP et FOR 0 to (TOS)-1 (1:Binaire Système) interne
6365F	6365F	OVER et Si TOS-1 < TOS (Binaire Système) → VRAI/FAUX interne
636DC	636DC	OVER et Si TOS-1 > TOS (Binaire Système) → VRAI/FAUX interne
637B8	637B8	ROT et +1 (Binaire Système) interne
637F4	637F4	DROP et -1 (Binaire Système) interne
6386C	6386C	SWAP et DUP2 interne
63A6F	63A6F	DUP et Si TOS est { } → VRAI/FAUX interne
63A88	63A88	DUP et <0h> interne
63A9C	63A9C	DUP et <1h> interne
63C2C	63C2C	SWAP et 4 ROLL interne
63C40	63C40	DUP2 et 5 ROLL interne
63C68	63C68	OVER et OVER interne
63C7C	63C7C	SWAP et 4 PICK interne
63C90	63C90	OVER et 5 PICK interne
63F6A	63F6A	+ et SWAP interne (2:Chaîne,1:Chaîne)

63FA6	63FA6	DROP et DROPN interne
63FBA	63FBA	4 PICK et 4 PICK interne

Gestion générale du Système

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03130	03130	Retour RPL
03A81	03A81	VRAI
03AC0	03AC0	FAUX
03AF2	03AF2	NOT interne (1:VRAI/FAUX)
03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX
041A7	041A7	OFF interne
041ED	041ED	OFF interne
04CE6	04CE6	Rappelle le contenu de l'adresse 70673 sur 5 quartets
04D0E	04D0E	Stocke dans l'adresse 70673 (1:Binaire Système)
04D33	04D33	nettoie les 5 quartets de l'adresse 70673
04DD7	04DD7	découpe (1:Binaire Système <abcde>) → 2:<cde> 1:<ab>
04ED1	04ED1	Effectue le dernier message d'erreur
04FAA	04FAA	Erreur "Power lost"
04FB6	04FB6	Erreur "Insufficient Memory"
04FC2	04FC2	Erreur "Directory Recursion"
04FCE	04FCE	Erreur "Undefined Local Name"
04FDA	04FDA	Erreur "Invalid Data Card"
04FE6	04FE6	Erreur "Object in Use"
04FF2	04FF2	Erreur "Port not Available"
04FFE	04FFE	Erreur "No room in port"
0500A	0500A	Erreur "Object not in port"
05016	05016	Erreur "Undefined Xlib Name"
05331	05331	compose un objet (N..2:Tout,1:Binaire Système)
05445	05445	→programme interne (N..2:Tout,1:Binaire Système)
05459	05459	→LIST interne (N..2:Tout,1:Binaire Système)
0546D	0546D	→algébrique interne (N:...,1:Binaire Système)

05481	05481	→UNIT interne (N..2:Tout,1:Binaire Système)
054AF	054AF	explose un objet → (N..2:Tout,1:Binaire Système)
05944	05944	BYTES interne → (2:Binaire Système,1:Entier)
059CC	059CC	Binaire Système→Binaire (1:Binaire Système)
05A03	05A03	Binaire→Binaire Système (1:Entier Binaire)
05A51	05A51	Caractère → Binaire Système (1:Caractère)
05A75	05A75	Binaire Système → Caractère (1:Binaire Système)
05AB3	05AB3	Change l'en-tête (2:Tout,1:Binaire Système)
05ACC	05ACC	Change l'en-tête (2:Tout,1:Binaire Système)
05B15	05B15	Chaîne → Nom Global (1:Chaîne)
05BE9	05BE9	Nom Local ou Global → Chaîne (1:Nom Global/Nom Local)
05E81	05E81	→TAG interne (2:Tout,1:Chaîne)
05E9F	05E9F	→TAG interne (1:Liste {Tout Nom_Global})
05EC7	05EC9	OBJ→ interne (1:Signé)
05EEA	05EEC	change l'en-tête du premier élement de la liste en Nom Global
05F2E	05F2E	→TAG interne (2:Tout,1:Nom Global/Nom Local)
05F61	05F61	MEM interne → (1:Binaire Système)
06657	06657	NEWOB interne
06E8E	06E8E	Pas d'opération
06F8E	06F8E	EVAL interne (1:Tout sauf Algebrique/Liste/Signé)
06F9F	06F9F	EVAL interne (1:Programme)
06FB7	06FB7	itère la boucle
0714D	0714D	saute l'objet suivant
0715C	0715C	saute les deux objets suivants
071A2	071A2	boucle
071AB	071AB	sort de la boucle
071E5	071E5	Sort de la boucle
07334	07334	Next (boucle interne)
073A5	073A5	Step interne (1:Binaire Système)
073C3	073C3	For 0 to (TOS)-1 (1:Binaire Système)
073CE	073CE	For 1 to (TOS)-1 (1:Binaire Système)
073DB	073DB	For 1 to TOS (1:Binaire Système)
073F7	073F7	For TOS to (TOS-1)-1 (2,1:Binaire Système)
07497	07497	Détruit les variables locales
074D0	074D0	Stocke les variables locales (N..2:Tout,1:Liste)
074E4	074E4	Stocke les variables locales

		(N..:Tout,..:Noms,1:Binaire Système)
0764E	0764E	Stocke les messages d'erreurs
		(1:Matrice,2:Binaire Système)
076AE	076AE	DETACH du répertoire HOME interne
		(1:Binaire Système)
07709	07709	ATTACH au répertoire HOME interne
		(1:Binaire Système)
07D27	07D27	STO interne (2:Tout,1:Nom Local)
07E50	07E50	→XLIB
07E76	07E76	→XLIB (1:System Entier Binaire)
08309	08309	UPDIR interne
08696	08696	STO interne (2:Tout,1:Nom Global)
08CCC	08CCC	XLIB→
08D08	08D08	STO interne (2:Répertoire,1:Nom Global)
08D5A	08D5A	Rappel du répertoire en cours
08D82	08D82	Rappel du répertoire HOME
08D92	08D92	HOME interne
08DD4	08DD4	Si TOS = répertoire HOME → VRAI/FAUX
0C612	715B1	SYS interne
		(1:Binaire Système sur sx,1:Adresse Virtuelle sur gx)
0CBAE	0CBAE	Erreur "Nonexistent Alarm"
0CBB7	0CBB7	Erreur "Invalide Date"
0CBC4	0CBC4	Erreur "Invalide Time"
0CBFA	0CBFA	TIME interne
0CC0E	0CC0E	DATE interne
0CC39	0CC39	DDAYS interne
0CC5B	0CC5B	DATE+ interne
0CD2B	0CD2B	→DATE interne
0CD3F	0CD3F	CLKADJ interne
0CD53	0CD53	→TIME interne
0D2A3	0D2A3	WSLOG Interne
0D304	0D304	TSTR interne
0DDA8	0DDA8	ACKALL interne
0DDC1	0DDC1	ACK interne
0DF01	0DF01	'Alarms' (Nom Global)
0DF28	0DF28	'Alarms' (Nom Global)
0E235	0E235	Rapelle la liste 'Alarms'
0E3DF	0E3DF	RCLALARM interne

0E402	0E402	Rapelle la nième Alarme (1:Binaire Système)
0E724	0E724	DELALARM interne
0EAD7	0EAD7	FINDALARM interne (1:Nombre Réel)
0EB31	0EB31	FINDALARM interne (1:List)
0EB81	0EB81	TICKS interne
0F34E	0F34E	OBJ→ interne (1:Unité)
10F54	10F54	Erreur "Can't edit Null Char"
10F64	10F64	Erreur "Invalide user Function"
10F74	10F74	Erreur "No current Equation"
10F86	10F86	Erreur "Invalid Syntax"
10F96	10F96	Erreur "Invalid PPAR"
10FA6	10FA6	Erreur "Non Real Result"
10FB6	10FB6	Erreur "Unable to Isolate"
10FC6	10FC6	Erreur "Halt not allowed"
10FD6	10FD6	KILL interne
10FE6	10FE6	Erreur "Last Stack Disabled"
10FF6	10FF6	Erreur "Last Command Disabled"
11006	11006	Erreur "Wrong Argument Count"
11016	11016	Erreur "Circular Reference"
11026	11026	Erreur "Directory not Allowed"
11036	11036	Erreur "Non Empty Directory"
11046	11046	Erreur "Invalid Definition"
11056	11056	Erreur "Missing Library"
11066	11066	Erreur "Name Conflict"
11076	11076	CONT interne
112EC	112EC	Efface les derniers arguments
1132D	1132D	mode Alpha ON
1133A	1133A	mode Alpha OFF
11347	11347	mode Shift-Bleu ON (Shift-Vert sur g/gx)
11354	11354	mode Shift-Bleu OFF (Shift-Vert sur g/gx)
11361	11361	mode Shift-Orange ON (Shift-Violet sur g/gx)
1136E	1136E	mode Shift-Orange OFF (Shift-Violet sur g/gx)
11543	11543	mode Alpha pour plusieurs caractères
1314D	1314D	TEXT interne
137B6	137B6	Rappelle la position courante → Ligne, Colonne:Binaire Système
1400E	1400E	ERR0 interne
1404C	1404C	ERRN interne

14065	14065	ERRM interne
14088	14088	→STR interne (1:Tout)
1415A	1415A	BEEP interne (2:Nombre Réel,1:Nombre Réel)
141B2	141B2	BEEP interne (2:Binaire Système,1:Binaire Système)
14378	14378	HALT interne
15007	15007	DOERR interne (1:Nombre Réel)
1501B	1501B	DOERR interne (1:Entier Binaire)
1502F	1502F	DOERR interne (1:Binaire Système)
15048	15048	DOERR interne (1:Chaîne)
15717	15717	STEQ interne (1:Tout)
1572B	1572B	RCEQ interne
15744	15744	RCEQ interne → Contenu + VRAI/FAUX
15758	15758	'EQ' non évalué (Nom Global)
1592D	1592D	met le dernier jeton RPL à <0h> et verifie DEPTH ≥ 1
166E3	166E3	FIX interne (1:Binaire Système)
166EF	166EF	SCI interne (1:Binaire Système)
166FB	166FB	ENG interne (1:Binaire Système)
16707	16707	STD interne (1:Binaire Système)
167D8	167D8	<ABCd> → "ABC: " (1:Binaire Système)
167E4	167E4	<ABCd> → "ABC: " (1:Binaire Système)
16CA7	16CA7	Erreur "Bad Argument Value"
1848C	1848C	PATH interne
184E1	184E1	CRDIR interne
18513	18513	STO interne (2:Tout,1:Nom Global)
1854F	1854F	PURGE interne (1:Nom Global)
18595	18595	PGDIR interne (1:Nom Global)
186E8	186E8	TVARS interne (1:Nombre Réel)
18706	18706	TVARS interne (1:Liste)
18779	18779	VARS interne
189FC	189FC	Code de configuration pour la bibiothèque 2
18A51	18A51	continue RPL
18A5B	18A5B	Sauve le dernier RPL et verifie DEPTH ≥ 3
18A68	18A68	Verifie DEPTH ≥ 3
18A80	18A80	Sauve le dernier RPL et verifie DEPTH ≥ 2
18A8D	18A8D	Verifie DEPTH ≥ 2
18AA5	18AA5	Sauve le dernier RPL et verifie DEPTH ≥ 1
18AB2	18AB2	Verifie DEPTH ≥ 1
18B6D	18B6D	Sauve le dernier RPL et verifie DEPTH ≥ 5

18B7A	18B7A	Verifie DEPTH ≥ 5
18B92	18B92	Sauve le dernier RPL et verifie DEPTH ≥ 4
18B9F	18B9F	Verifie DEPTH ≥ 4
18C92	18C92	Erreur "Undefined Name"
18CA2	18CA2	Erreur "Bad Argument Value"
18CB2	18CB2	Erreur "Bad Argument Type"
18CC2	18CC2	Erreur "Too Few Arguments"
18EBA	18EBA	EVAL interne (1:Algebraic/List/Program)
18ECE	18ECE	Sauve le dernier RPL, vérifie DEPTH ≥ 1 et contrôle les arguments
18EDF	18EDF	Sauve le dernier RPL, vérifie DEPTH ≥ 2 et contrôle les arguments
18EF0	18EF0	Sauve le dernier RPL, vérifie DEPTH ≥ 3 et contrôle les arguments
18F01	18F01	Sauve le dernier RPL, vérifie DEPTH ≥ 4 et contrôle les arguments
18F12	18F12	Sauve le dernier RPL, vérifie DEPTH ≥ 5 et contrôle les arguments
18FB2		Vérifie le type des arguments
194F7	194F7	Deux Nombres Réels → Binaire Système
1950B	1950B	Deux Binaire Système → Nombres Réels
1957B	1957B	ASR (XLIB 2 0)
1959B	1959B	RL (XLIB 2 1)
195BB	195BB	RLB (XLIB 2 2)
195DB	195DB	RR (XLIB 2 3)
195FB	195FB	RRB (XLIB 2 4)
1961B	1961B	SL (XLIB 2 5)
1963B	1963B	SLB (XLIB 2 6)
1965B	1965B	SR (XLIB 2 7)
1967B	1967B	SRB (XLIB 2 8)
1969B	1969B	R→B (XLIB 2 9)
196BB	196BB	B→R (XLIB 2 10)
196DB	196DB	CONVERT (XLIB 2 11)
1971B	1971B	UVAL (XLIB 2 12)
1974F	1974F	UNIT (XLIB 2 13)
19771	19771	UBASE (XLIB 2 14)
197A5	197A5	UFACT (XLIB 2 15)
197C8	197C8	UFACT interne

197F7	197F7	TIME (XLIB 2 16)
19812	19812	DATE (XLIB 2 17)
1982D	1982D	TICKS (XLIB 2 18)
19848	19848	WSLOG (XLIB 2 19)
19863	19863	ACKALL (XLIB 2 20)
1987E	1987E	ACK (XLIB 2 21)
1989E	1989E	→DATE (XLIB 2 22)
198BE	198BE	→TIME (XLIB 2 23)
198DE	198DE	CLKADJ (XLIB 2 24)
198FE	198FE	STOALARM (XLIB 2 25)
19928	19928	RCLALARM (XLIB 2 26)
19948	19948	FINDALARM (XLIB 2 27)
19972	19972	DELALARM (XLIB 2 28)
19992	19992	TSTR (XLIB 2 29)
199B2	199B2	DDAYS (XLIB 2 30)
199D2	199D2	DATE+ (XLIB 2 31)
19A72		'ALRMDAT' (Nom Global)
19DE2		Affiche alarme suivante
1A105	1A105	CRDIR (XLIB 2 32)
1A125	1A125	PATH (XLIB 2 33)
1A140	1A140	HOME (XLIB 2 34)
1A15B	1A15B	UPDIR (XLIB 2 35)
1A16F	1A16F	UPDIR interne
1A194	1A194	VARS (XLIB 2 36)
1A1AF	1A1AF	TVARS (XLIB 2 37)
1A1D9	1A1D9	BYTES (XLIB 2 38)
1A1FC	1A1FC	BYTES interne (1:Tout sauf Nom Global)
1A265	1A265	BYTES interne (1:Nom Global)
1A2BC	1A2BC	NEWOB (XLIB 2 39)
1A303	1A303	KILL (XLIB 2 40)
1A31E	1A31E	OFF (XLIB 2 41)
1A339	1A339	DOERR (XLIB 2 42)
1A36D	1A36D	ERR0 (XLIB 2 43)
1A388	1A388	ERRN (XLIB 2 44)
1A3A3	1A3A3	ERRM (XLIB 2 45)
1A3BE	1A3BE	EVAL (XLIB 2 46)
1A3FE	1A3FE	IFTE (XLIB 2 47)
1A4CD	1A4CD	IFT (XLIB 2 48)

1A4F0	1A4F0	IFT interne (2:Nombre Réel,1:Tout)
1A513	1A513	IFT interne (2:Symbolique,1:Tout)
1A52E	1A52E	SYSEVAL (XLIB 2 49)
1A547	1A547	SYSEVAL interne (1:Entier Binaire)
1A584	1A584	DISP (XLIB 2 50)
1A5A4	1A5A4	FREEZE (XLIB 2 51)
1A5C4	1A5C4	BEEP (XLIB 2 52)
1A5E4	1A5E4	→NUM (XLIB 2 53)
1A604	1A604	LASTARG (XLIB 2 54)
1A71F	1A71F	WAIT (XLIB 2 55)
1A738	1A738	WAIT Interne (1:Nombre Réel)
1A7B5	1A7B5	WAIT Interne (1:Nombre Réel > 0)
1A858	1A858	CLLCD (XLIB 2 56)
1A873	1A873	KEY (XLIB 2 57)
1A8BB	1A8BB	CONT (XLIB 2 58)
1A8D8	1A8D8	= (XLIB 2 59)
1A995	1A995	NEG (XLIB 2 60)
1A9F9	1A9F9	# 8010h
1AA1F	1AA1F	ABS (XLIB 2 61)
1AA6E	1AA6E	CONJ (XLIB 2 62)
1AABD	1AABD	π (XLIB 2 63)
1AADF	1AADF	MAXR (XLIB 2 64)
1AB01	1AB01	MINR (XLIB 2 65)
1AB23	1AB23	e (XLIB 2 66)
1AB45	1AB45	i (XLIB 2 67)
1AB67	1AB67	+ (XLIB 2 68)
1ACD7	1ACD7	+ (XLIB 2 69)
1AD09	1AD09	- (XLIB 2 70)
1ADEE	1ADEE	* (XLIB 2 71)
1AF05	1AF05	/ (XLIB 2 72)
1B02D	1B02D	^ (XLIB 2 73)
1B185	1B185	XROOT (XLIB 2 74)
1B1CA	1B1CA	XROOT (XLIB 2 75)
1B278	1B278	INV (XLIB 2 76)
1B2DB	1B2DB	ARG (XLIB 2 77)
1B32A	1B32A	SIGN (XLIB 2 78)
1B374	1B374	SQRT (XLIB 2 79)
1B426	1B426	SQ (XLIB 2 80)

1B4AC	1B4AC	SIN (XLIB 2 81)
1B505	1B505	COS (XLIB 2 82)
1B55E	1B55E	TAN (XLIB 2 83)
1B5B7	1B5B7	SINH (XLIB 2 84)
1B606	1B606	COSH (XLIB 2 85)
1B655	1B655	TANH (XLIB 2 86)
1B6A4	1B6A4	ASIN (XLIB 2 87)
1B72F	1B72F	ACOS (XLIB 2 88)
1B79C	1B79C	ATAN (XLIB 2 89)
1B7EB	1B7EB	ASINH (XLIB 2 90)
1B830	1B830	ACOSH (XLIB 2 91)
1B8A2	1B8A2	ATANH (XLIB 2 92)
1B905	1B905	EXP (XLIB 2 93)
1B94F	1B94F	LN (XLIB 2 94)
1B9C6	1B9C6	LOG (XLIB 2 95)
1BA3D	1BA3D	ALOG (XLIB 2 96)
1BA8C	1BA8C	LNP1 (XLIB 2 97)
1BAC2	1BAC2	EXPM (XLIB 2 98)
1BB02	1BB02	(XLIB 2 99)
1BB41	1BB41	FACT (XLIB 2 100)
1BB6D	1BB6D	IP (XLIB 2 101)
1BBA3	1BBA3	FP (XLIB 2 102)
1BBD9	1BBD9	FLOOR (XLIB 2 103)
1BC0F	1BC0F	CEIL (XLIB 2 104)
1BC45	1BC45	XPON (XLIB 2 105)
1BC71	1BC71	MAX (XLIB 2 106)
1BCE3	1BCE3	MIN (XLIB 2 107)
1BD55	1BD55	RND (XLIB 2 108)
1BDD1	1BDD1	TRNC (XLIB 2 109)
1BE4D	1BE4D	MOD (XLIB 2 110)
1BE9C	1BE9C	MANT (XLIB 2 111)
1BEC8	1BEC8	D→R (XLIB 2 112)
1BEF4	1BEF4	R→D (XLIB 2 113)
1BF1E	1BF1E	→HMS (XLIB 2 114)
1BF3E	1BF3E	HMS→ (XLIB 2 115)
1BF5E	1BF5E	HMS+ (XLIB 2 116)
1BF7E	1BF7E	HMS- (XLIB 2 117)
1BF9E	1BF9E	RNRM (XLIB 2 118)

1BFBE	1BFBE	CNRM (XLIB 2 119)
1BFDE	1BFDE	DET (XLIB 2 120)
1BFFE	1BFFE	DOT (XLIB 2 121)
1C01E	1C01E	CROSS (XLIB 2 122)
1C03E	1C03E	RSD (XLIB 2 123)
1C060	1C060	% (XLIB 2 124)
1C0D7	1C0D7	%T (XLIB 2 125)
1C149	1C149	%CH (XLIB 2 126)
1C1B9	1C1B9	RAND (XLIB 2 127)
1C1D4	1C1D4	RDZ (XLIB 2 128)
1C1F6	1C1F6	COMB (XLIB 2 129)
1C236	1C236	PERM (XLIB 2 130)
1C274	1C274	SF (XLIB 2 131)
1C28D	1C28D	SF Interne (1:Nombre Réel)
1C2D5	1C2D5	CF (XLIB 2 132)
1C2EE	1C2EE	CF interne (1:Nombre Réel)
1C313	1C313	FS? (XLIB 2 133)
1C32C	1C32C	FS? interne (1:Nombre Réel)
1C331	1C331	FS? interne (1:Nombre Réel) → VRAI/FAUX
1C360	1C360	FC? (XLIB 2 134)
1C379	1C379	FC? interne (1:Nombre Réel)
1C399	1C399	DEG (XLIB 2 135)
1C3B4	1C3B4	RAD (XLIB 2 136)
1C3CF	1C3CF	GRAD (XLIB 2 137)
1C3EA	1C3EA	FIX (XLIB 2 138)
1C403	1C403	FIX interne (1:Nombre Réel)
1C41E	1C41E	SCI (XLIB 2 139)
1C437	1C437	SCI interne (1:Nombre Réel)
1C452	1C452	ENG (XLIB 2 140)
1C46B	1C46B	ENG interne (1:Nombre Réel)
1C486	1C486	STD (XLIB 2 141)
1C4A1	1C4A1	FS?C (XLIB 2 142)
1C4BA	1C4BA	FS?C interne (1:Nombre Réel)
1C4BF	1C4BF	FS?C interne (1:Nombre Réel) → VRAI/FAUX
1C4CE	1C4CE	Teste et CF le drapeau utilisateur
1C4EC	1C4EC	Teste et CF le drapeau système
1C520	1C520	FC?C (XLIB 2 143)
1C539	1C539	FC?C interne (1:Nombre Réel)

1C559	1C559	BIN (XLIB 2 144)
1C574	1C574	DEC (XLIB 2 145)
1C58F	1C58F	HEX (XLIB 2 146)
1C5AA	1C5AA	OCT (XLIB 2 147)
1C5C5	1C5C5	STWS (XLIB 2 148)
1C5FE	1C5FE	RCWS (XLIB 2 149)
1C619	1C619	RCLF (XLIB 2 150)
1C637	1C637	Rappelle les drapeaux système
1C64E	1C64E	Rappelle les drapeaux utilisateur
1C67F	1C67F	STOF (XLIB 2 151)
1C6A2	1C6A2	STOF interne (1:List)
1C6CF	1C6CF	STOF interne (2:Entier Binaire,1:Entier Binaire)
1C6E3	1C6E3	STOF (système) (1:Entier Binaire)
1C6F7	1C6F7	STOF (utilisateur) (1:Entier Binaire)
1C731	1C731	STOF (système)
1C783	1C783	→LIST (XLIB 2 152)
1C79E	1C79E	R→C (XLIB 2 153)
1C7CA	1C7CA	RE (XLIB 2 154)
1C819	1C819	IM (XLIB 2 155)
1C85C	1C85C	SUB (XLIB 2 156)
1C8EA	1C8EA	REPL (XLIB 2 157)
1C95A	1C95A	LIST→ (XLIB 2 158)
1C973	1C973	explose l'objet (liste,programme) → (N..2:Tout,1:Nombre Réel)
1C98E	1C98E	C→R (XLIB 2 159)
1C9B8	1C9B8	SIZE (XLIB 2 160)
1CAB4	1CAB4	POS (XLIB 2 161)
1CB0B	1CB0B	→STR (XLIB 2 162)
1CB26	1CB26	STR→ (XLIB 2 163)
1CB46	1CB46	NUM (XLIB 2 164)
1CB66	1CB66	CHR (XLIB 2 165)
1CB86	1CB86	TYPE (XLIB 2 166)
1CB90	1CB90	TYPE interne (1:Tout)
1CDD4	1CDD4	TYPE interne (1:Program)
1CE28	1CE28	VTYPE (XLIB 2 167)
1CE55	1CE55	VTYPE interne (1:Nom Global/Nom Local)
1CE82	1CE82	VTYPE interne (1:Signé)
1CEE3	1CEE3	EQ→ (XLIB 2 168)

1CF2E	1CF2E	EQ→ interne (1:Expression Algebrique)
1CF7B	1CF7B	OBJ→ (XLIB 2 169)
1CFD0	1CFD0	OBJ→ interne (1:Expression Algébrique)
1D009	1D009	→ARRY (XLIB 2 170)
1D092	1D092	ARRY→ (XLIB 2 171)
1D0DF	1D0DF	RDM (XLIB 2 172)
1D186	1D186	CON (XLIB 2 173)
1D23F	1D23F	CON interne (2:Nom Global,1:Nombre Réel)
1D262	1D262	CON interne (2:Nom Global,1:Nombre Complexe)
1D28A	1D28A	CON interne (2:Nom Local,1:Nombre Réel)
1D2AD	1D2AD	CON interne (2:Nom Local,1:Nombre Complexe)
1D2DC	1D2DC	IDN (XLIB 2 174)
1D392	1D392	TRN (XLIB 2 175)
1D407	1D407	PUT (XLIB 2 176)
1D484	1D484	PUT interne (3:Nom Global,2:Nombre Réel/Liste,1:Tout)
1D565	1D565	PUT interne (3:Nom Local,2:Nombre Réel/Liste,1:Tout)
1D5DF	1D5DF	PUTI (XLIB 2 177)
1D65C	1D65C	PUTI interne (3:Nom Global,2:Nombre Réel/Liste,1:Tout)
1D747	1D747	PUTI interne (3:Nom Local,2:Nombre Réel/Liste,1:Tout)
1D7C6	1D7C6	GET (XLIB 2 178)
1D825	1D825	GET interne (2:Nom Global/Nom Local,1:Nombre Réel/Liste)
1D8C7	1D8C7	GETI (XLIB 2 179)
1D926	1D926	GETI interne (2:Nom Global/Nom Local,1:Nombre Réel/Liste)
1DD06	1DD06	V→ (XLIB 2 180)
1DE66	1DE66	→V2 (XLIB 2 181)
1DEC2	1DEC2	→V3 (XLIB 2 182)
1E04A	1E04A	INDEP (XLIB 2 183)
1E07E	1E07E	PMIN (XLIB 2 184)
1E09E	1E09E	PMAX (XLIB 2 185)
1E0BE	1E0BE	AXES (XLIB 2 186)
1E0E8	1E0E8	CENTR (XLIB 2 187)
1E101	1E101	CENTR interne (1:Nombre Réel)

1E126	1E126	RES (XLIB 2 188)
1E150	1E150	*H (XLIB 2 189)
1E170	1E170	*W (XLIB 2 190)
1E190	1E190	DRAW (XLIB 2 191)
1E1AB	1E1AB	AUTO (XLIB 2 192)
1E1C6	1E1C6	DRAX (XLIB 2 193)
1E1E1	1E1E1	SCALE (XLIB 2 194)
1E201	1E201	PDIM (XLIB 2 195)
1E22B	1E22B	DEPND (XLIB 2 196)
1E25F	1E25F	ERASE (XLIB 2 197)
1E27A	1E27A	PX→C (XLIB 2 198)
1E29A	1E29A	C→PX (XLIB 2 199)
1E2BA	1E2BA	GRAPH (XLIB 2 200)
1E2D5	1E2D5	LABEL (XLIB 2 201)
1E2F0	1E2F0	PVIEW (XLIB 2 202)
1E31A	1E31A	PIXON (XLIB 2 203)
1E344	1E344	PIXOFF (XLIB 2 204
1E36E	1E36E	PIX? (XLIB 2 205)
1E398	1E398	LINE (XLIB 2 206)
1E3C2	1E3C2	TLINE (XLIB 2 207)
1E3EC	1E3EC	BOX (XLIB 2 208)
1E416	1E416	BLANK (XLIB 2 209)
1E436	1E436	PICT (XLIB 2 210)
1E456	1E456	GOR (XLIB 2 211)
1E4E4	1E4E4	GXOR (XLIB 2 212)
1E572	1E572	LCD→ (XLIB 2 213)
1E58D	1E58D	→LCD (XLIB 2 214)
1E5AD	1E5AD	→GROB (XLIB 2 215)
1E5D2	1E5D2	ARC (XLIB 2 216)
1E606	1E606	TEXT (XLIB 2 217)
1E621	1E621	XRNG (XLIB 2 218)
1E641	1E641	YRNG (XLIB 2 219)
1E661	1E661	FUNCTION (XLIB 2 220)
1E681	1E681	CONIC (XLIB 2 221)
1E6A1	1E6A1	POLAR (XLIB 2 222)
1E6C1	1E6C1	PARAMETRIC (XLIB 2 223)
1E6E1	1E6E1	TRUTH (XLIB 2 224)
1E701	1E701	SCATTER (XLIB 2 225)

1E721	1E721	HISTOGRAM (XLIB 2 226)
1E741	1E741	BAR (XLIB 2 227)
1E761	1E761	SAME (XLIB 2 228)
1E783	1E783	AND (XLIB 2 229)
1E809	1E809	OR (XLIB 2 230)
1E88F	1E88F	NOT (XLIB 2 231)
1E8F6	1E8F6	XOR (XLIB 2 232)
1E972	1E972	== (XLIB 2 233)
1EA9D	1EA9D	≠ (XLIB 2 234)
1EBBE	1EBBE	< (XLIB 2 235)
1EC5D	1EC5D	> (XLIB 2 236)
1ECFC	1ECFC	≤ (XLIB 2 237)
1ED7E	1ED7E	≤ interne (2:Nombre Réel,1:Nombre Réel)
1ED9B	1ED9B	≥ (XLIB 2 238)
1EE1D	1EE1D	≥ interne (2:Nombre Réel,1:Nombre Réel)
1EE38	1EE38	OLDPRT (XLIB 2 239)
1EE53	1EE53	PR1 (XLIB 2 240)
1EE6E	1EE6E	PRSTC (XLIB 2 241)
1EE89	1EE89	PRST (XLIB 2 242)
1EEA4	1EEA4	CR (XLIB 2 243)
1EEBF	1EEBF	PRVAR (XLIB 2 244)
1EEEC	1EEEC	PRVAR interne (1:Signé)
1EF1E	1EF1E	PRVAR interne (1:Liste)
1EF43	1EF43	DELAY (XLIB 2 245)
1EF63	1EF63	PRLCD (XLIB 2 246)
1EF7E	1EF7E	Dérivation Complète (XLIB 2 247)
1EFD2	1EFD2	Dérivation pas à pas (XLIB 2 248)
1F0F5	1F0F5	Dérivation pas à pas interne (2:Expression Algébrique,1:Symbolique)
1F133	1F133	RCEQ (XLIB 2 249)
1F14E	1F14E	STEQ (XLIB 2 250)
1F16E	1F16E	ROOT (XLIB 2 251)
1F1D4	1F1D4	∫ (sur la pile) (XLIB 2 252)
1F201	1F201	∫ interne (sur la pile)
1F223	1F223	∫ (algebrique) (XLIB 2 253)
1F27A	1F27A	∫ interne (algebrique)
1F2C9	1F2C9	Σ (XLIB 2 254)
1F354	1F354	| (usage dans la pile) (XLIB 2 255)

1F38B	1F38B	| interne (usage dans la pile) (2:Symbolique,1:Liste)
1F3F3	1F3F3	| (usage algébrique) (XLIB 2 256)
1F500	1F500	QUOTE (XLIB 2 257)
1F542	1F542	QUOTE interne (1:Expression Algébrique)
1F55D	1F55D	APPLY (sur la pile) (XLIB 2 258)
1F585	1F585	APPLY interne (sur la pile) (2:Liste,1:Nom Global/Nom Local)
1F5C5	1F5C5	APPLY (usage algébrique) (XLIB 2 259)
1F640	1F640	XLIB 2 260
1F8CF	1F8CF	STO interne (2:Tout,1:Expression Algébrique)
1F996	1F996	XLIB 2 261
1F9AE	1F9AE	XLIB 2 262
1F9C4	1F9C4	→Q (XLIB 2 263)
1F9E9	1F9E9	→Qπ (XLIB 2 264)
1FA59	1FA59	↑MATCH (XLIB 2 265)
1FA8D	1FA8D	↓MATCH (XLIB 2 266)
1FABA	1FABA	↑MATCH interne (2:Nombre Réel/Complexe/Symbolique,1:Liste)
1FACE	1FACE	↓MATCH interne (2:Nombre Réel/Complexe/Symbolique,1:Liste)
1FAEB	1FAEB	_ (XLIB 2 267)
1FB5D	1FB5D	RATIO (XLIB 2 268)
1FB87	1FB87	DUP (XLIB 2 269)
1FBA2	1FBA2	DUP2 (XLIB 2 270)
1FBBD	1FBBD	SWAP (XLIB 2 271)
1FBD8	1FBD8	DROP (XLIB 2 272)
1FBF3	1FBF3	DROP2 (XLIB 2 273)
1FC0E	1FC0E	ROT (XLIB 2 274)
1FC29	1FC29	OVER (XLIB 2 275)
1FC44	1FC44	DEPTH (XLIB 2 276)
1FC64	1FC64	DROPN (XLIB 2 277)
1FC7F	1FC7F	DUPN (XLIB 2 278)
1FC9A	1FC9A	PICK (XLIB 2 279)
1FCB5	1FCB5	ROLL (XLIB 2 280)
1FCD0	1FCD0	ROLLD (XLIB 2 281)
1FCEB	1FCEB	CLEAR (XLIB 2 282)
1FD0B	1FD0B	STOΣ (XLIB 2 283)
1FD2B	1FD2B	CLΣ (XLIB 2 284)

1FD46	1FD46	RCLΣ (XLIB 2 285)
1FD61	1FD61	Σ+ (XLIB 2 286)
1FD8B	1FD8B	Σ- (XLIB 2 287)
1FDA6	1FDA6	NΣ (XLIB 2 288)
1FDC1	1FDC1	CORR (XLIB 2 289)
1FDDC	1FDDC	COV (XLIB 2 290)
1FDF7	1FDF7	ΣX (XLIB 2 291)
1FE12	1FE12	ΣY (XLIB 2 292)
1FE2D	1FE2D	ΣX^2 (XLIB 2 293)
1FE48	1FE48	ΣY^2 (XLIB 2 294)
1FE63	1FE63	ΣX*Y (XLIB 2 295)
1FE7E	1FE7E	MAXΣ (XLIB 2 296)
1FE99	1FE99	MEAN (XLIB 2 297)
1FEB4	1FEB4	MINΣ (XLIB 2 298)
1FECF	1FECF	SDEV (XLIB 2 299)
1FEEA	1FEEA	TOT (XLIB 2 300)
1FF05	1FF05	VAR (XLIB 2 301)
1FF20	1FF20	LR (XLIB 2 302)
1FF7A	1FF7A	PREDV (XLIB 2 303)
1FF9A	1FF9A	PREDY (XLIB 2 304)
1FFBA	1FFBA	PREDX (XLIB 2 305)
1FFDA	1FFDA	XCOL (XLIB 2 306)
2001A	2001A	UTPC (XLIB 2 308)
2003A	2003A	UTPN (XLIB 2 309)
2005A	2005A	UTPF (XLIB 2 310)
2007A	2007A	UTPT (XLIB 2 311)
2009A	2009A	COLΣ (XLIB 2 312)
200C4	200C4	SCLΣ (XLIB 2 313)
200F3	200F3	ΣLINE (XLIB 2 314)
2010E	2010E	BINS (XLIB 3 315)
20133	20133	BARPLOT (XLIB 2 316)
20167	20167	HISTPLOT (XLIB 2 317)
2018C	2018C	SCATRPLOT (XLIB 2 318)
201B1	201B1	LINFIT (XLIB 2 319)
201D6	201D6	LOGFIT (XLIB 2 320)
201FB	201FB	EXPFIT (XLIB 2 321)
20220	20220	PWRFIT (XLIB 2 322)
2025E	2025E	BESTFIT (XLIB 2 323)

202CE	202CE	SINV (XLIB 2 324)
202F1	202F1	SINV interne (1:Nom Global)
20314	20314	SINV interne (1:Nom Local)
2034D	2034D	SNEG (XLIB 2 325)
20370	20370	SNEG interne (1:Nom Global)
20393	20393	SNEG interne (1:Nom Local)
203CC	203CC	SCONJ (XLIB 2 326)
203EF	203EF	SCONJ interne (1:Nom Global)
20412	20412	SCONJ interne (1:Nom Local)
2044B	2044B	STO+ (XLIB 2 327)
20482	20482	STO+ interne (2:Tout,1:Nom Global/Nom Local)
204C3	204C3	STO+ interne (2:Nom Global/Nom Local,1:Tout)
20538	20538	STO- (XLIB 2 328)
20583	20583	STO- interne (2:Tout,1:Nom Global/Nom Local)
205A1	205A1	STO- interne (2:Nom Global/Nom Local,1:Tout)
205BF	205BF	STO- interne (2:Matrice,1:Nom Global)
205E2	205E2	STO- interne (2:Nom Global,1:Matrice)
2060C	2060C	STO/ (XLIB 2 329)
2066B	2066B	STO/ interne (2:Tout,1:Nom Global/Nom Local)
20689	20689	STO/ interne (2:Nom Global/Nom Local,1:Tout)
206A7	206A7	STO/ interne (2:Nom Global,1:Nombre Réel/Nombre Complexe)
206E8	206E8	STO/ interne (2:Matrice,1:Nom Global)
20729	20729	STO/ interne (2:Nom Global,1:Matrice)
20753	20753	STO* (XLIB 2 330)
207C6	207C6	STO* interne (2:Tout,1:Nom Global/Nom Local)
207E4	207E4	STO* interne (2:Nom Global/Nom Local,1:Tout)
20802	20802	STO* interne (2:Nombre Réel/Nombre Complexe,1:Nom Global)
2082A	2082A	STO* interne (2:Nom Global,1:Nombre Réel/Nombre Complexe)
2086B	2086B	STO* interne (2:Matrice,1:Nom Global)
208AC	208AC	STO* interne (2:Nom Global,1:Matrice)
208F4	208F4	INCR (XLIB 2 331)
20917	20917	INCR interne (1:Nom Global)
20980	20980	INCR interne (1:Nom Local)
209AA	209AA	DECR (XLIB 2 332)
209CD	209CD	DECR interne (1:Nom Global)

209EB	209EB	DECR interne (1:Nom Local)
20A15	20A15	COLCT (XLIB 2 333)
20A49	20A49	EXPAN (XLIB 2 334)
20A7D	20A7D	RULES (XLIB 2 335)
20A93	20A93	ISOL (XLIB 2 336)
20AB3	20AB3	QUAD (XLIB 2 337)
20AD3	20AD3	SHOW (XLIB 2 338)
20B00	20B00	SHOW interne (2:Symbolique,1:Liste)
20B20	20B20	TAYLR (XLIB 2 339)
20B40	20B40	RCL (XLIB 2 340)
20B81	20B81	RCL interne (1:Nom Global/Nom Local)
20B9A	20B9A	RCL interne (1:Liste)
20CAD	20CAD	RCL interne (1:PICT)
20CCD	20CCD	STO (XLIB 2 341)
20D65	20D65	DEFINE (XLIB 2 342)
20D7E	20D7E	DEFINE interne (1:Expression Algébrique)
20DBF	20DBF	DEFINE interne (1:Nom Global/Nom Local)
20EFE	20EFE	PURGE (XLIB 2 343)
20F35	20F35	PURGE interne (1:Liste)
20F8A	20F8A	PURGE interne (1:PICT)
20FAA	20FAA	MEM (XLIB 2 344)
20FD9	20FD9	ORDER (XLIB 2 345)
20FF2	20FF2	ORDER (1:Liste)
210FC	210FC	CLVAR (XLIB 2 346)
2115D	2115D	TMENU (XLIB 2 347)
21196	21196	MENU (XLIB 2 348)
211E1	211E1	RCLMENU (XLIB 2 349)
211FC	211FC	PVARS (XLIB 2 350)
2123A	2123A	PGDIR (XLIB 2 351)
2125A	2125A	ARCHIVE (XLIB 2 352)
21273	21273	ARCHIVE interne (1:Signé)
2133C	2133C	RESTORE (XLIB 2 353)
2137F	2137F	MERGE (XLIB 3 354)
21398	21398	MERGE interne (1:Nombre Réel)
213D1	213D1	FREE (XLIB 2 355)
21408	21408	FREE interne (2:Nombre Réel/Nom Global/Nom Local,1: Réel)
2142D	2142D	LIBS (XLIB 2 356)

21448	21448	ATTACH (XLIB 2 357)
21461	21461	ATTACH (1:Nombre Réel)
2147C	2147C	DETACH (XLIB 2 358)
21495	21495	DETACH interne (1:Nombre Réel)
214A9	214A9	Réel→Binaire Système et vérifie ≥ <100h> et π <700h>
214F4	2150F	STO interne (2:Tout,1:Signé)
215BF	215BF	STO interne (2:Bibliothèque/Sauvegarde,1:Réel)
21761	21761	RCL interne (1:Signé)
217C7	217C7	EVAL interne (1:Signé)
217F1	217F1	PURGE interne (1:Signé)
21922	21922	Copie les variables du port dans la pile (1:Binaire Système)
21B2F	21B2F	RESTORE interne (1:Sauvegarde)
21B74	21B74	FREE interne (2:Liste,1:Nombre Réel)
21C6F	21C6F	ATTACH interne (1:Binaire Système)
21CBA	21CBA	ATTACH (pas HOME) interne (2:Directory,1:Binaire Système)
21CE5	21CE5	DETACH interne (1:Binaire Système)
21D2B	21D2B	DETACH interne (pas HOME) (2:Directory,1:Binaire Système)
21D54	21D54	LIBS interne
21E75	21E75	XMIT (XLIB 2 359)
21E95	21E95	SRECV (XLIB 2 360)
21EB5	21EB5	OPENIO (XLIB 2 361)
21ED5	21ED5	CLOSEIO (XLIB 2 362)
21EF0	21EF0	SEND (XLIB 2 363)
21F24	21F24	KGET (XLIB 2 364)
21F62	21F62	RECN (XLIB 2 365)
21F96	21F96	RECV (XLIB 2 366)
21FB6	21FB6	FINISH (XLIB 2 367)
21FD1	21FD1	SERVER (XLIB 2 368)
21FEC	21FEC	CKSM (XLIB 2 369)
2200C	2200C	BAUD (XLIB 2 370)
2202C	2202C	PARITY (XLIB 2 371)
2204C	2204C	TRANSIO (XLIB 2 372)
2206C	2206C	KERRM (XLIB 2 373)
22087	22087	BUFLEN (XLIB 2 374)
220A2	220A2	STIME (XLIB 2 375)

220C2	220C2	SBRK (XLIB 2 376)
220DD	220DD	PKT (XLIB 2 377)
220F6	220F6	Affiche les choix Entrée/Sortie (I/O)
224CA	224CA	INPUT (XLIB 2 378)
224F4	224F4	ASN (XLIB 2 379)
22514	22514	STOKEYS (XLIB 2 380)
22548	22548	DELKEYS (XLIB 2 381)
22586	22586	RCLKEYS (XLIB 2 382)
225BE	225BE	→TAG (XLIB 2 383)
225F5	225F5	→TAG interne (2:Tout,1:Chaîne)
22618	22618	→TAG interne (2:Tout,1:Nombre Réel)
22633	22633	DTAG (XLIB 2 384)
22EC3	22EC3	IF (XLIB 1792 0)
22EFA	22EFA	THEN (XLIB 1792 1)
22FB5	22FB5	ELSE (XLIB 1792 2)
22FD5	22FD5	END (XLIB 1792 3)
22FEB	22FEB	→ (XLIB 1792 4)
23033	23033	WHILE (XLIB 1792 5)
2305D	2305D	REPEAT (XLIB 1792 6)
230C3	230C3	DO (XLIB 1792 7)
230ED	230ED	UNTIL (XLIB 1792 8)
23103	23103	START (XLIB 1792 9)
231A0	231A0	FOR (XLIB 1792 10)
2324C	2324C	NEXT (XLIB 1792 11)
23380	23380	STEP (XLIB 1792 12)
233DF	233DF	IFERR (XLIB 1792 13)
23472	23472	HALT (XLIB 1792 14)
2349C	2349C	Commande sans effet (XLIB 1792 15)
234C1	234C1	→ (XLIB 1792 16)
235FE	235FE	» (XLIB 1792 17)
2361E	2361E	« (XLIB 1792 18)
23639	23639	» (XLIB 1792 19)
23694	23694	END (XLIB 1792 22)
236B9	236B9	END (XLIB 1792 23)
2371F	2371F	THEN (XLIB 1792 24)
2378D	2378D	CASE (XLIB 1792 25)
237A8	237A8	THEN (XLIB 1792 26)
23813	23813	DIR (XLIB 1792 27)

23824	23824	PROMPT (XLIB 1792 28)
28A38	28A38	_ interne (1:Symbolique)
2A5D2	2A5D2	DEG interne
2A5F0	2A5F0	RAD interne
2A604	2A604	GRAD interne
2A622	2A622	D→R interne (1:Nombre Réel)
2A655	2A655	R→D interne (1:Nombre Réel)
2A673	2A673	→HMS interne (1:Nombre Réel)
2A68C	2A68C	HMS→ interne (1:Nombre Réel)
2A6A0	2A6A0	HMS+ interne (2:Nombre Réel,1:Nombre Réel)
2A6C8	2A6C8	HMS- interne (2:Nombre Réel,1:Nombre Réel)
2A6F5	2A6F5	MAX interne (2:Nombre Réel,1:Nombre Réel)
2A70E	2A70E	MIN interne (2:Nombre Réel,1:Nombre Réel)
2AFC2	2AFC2	RAND interne
2B044	2B044	RDZ interne (1:Nombre Réel)
2B529	2B529	RND interne (2:Nombre Réel,1:Nombre Réel)
2C09F	2C09F	UTPN interne (3:Nombre Réel,2:Nombre Réel,1:Nombre Réel)
2C149	2C12E	UTPC interne (2:Nombre Réel,1:Nombre Réel)
2C174	2C15E	UTPF interne (3:Nombre Réel,2:Nombre Réel,1:Nombre Réel)
2C19A	2C189	UTPT interne (2:Nombre Réel,1:Nombre Réel)
2C1F3	2C1F3	STOΣ interne (1:Tout)
2C22F	2C22F	CLΣ interne
2C293	2C293	RCLΣ interne → Contenu et VRAI/FAUX
2C2AC	2C2AC	RCLΣ interne
2C2D9	2C2D9	Σ+ interne (1:Nombre Réel)
2C32E	2C32E	Σ+ interne (1:Matrice)
2C423	2C423	Σ- interne
2C535	2C535	NΣ interne
2C558	2C558	MAXΣ interne
2C571	2C571	MEAN interne
2C58A	2C58A	MINΣ interne
2C5A3	2C5A3	SDEV interne
2C5BC	2C5BC	TOT interne
2C5D5	2C5D5	VAR interne
2C684	2C675	COLΣ interne (2:Nombre Réel/Matrice,1:Réel)
2C6A2		Stocke les niveaux 1 à 5 dans 'ΣPAR'

2C6C5	2C6B6	XCOL interne (1:Nombre Réel)
2C6DE	2C6CF	YCOL interne (1:Nombre Réel)
2C84B	2C83C	CORR interne
2C8F5	2C8E6	COV interne
2C94F	2C940	ΣX interne
2C963	2C959	ΣY interne
2C977	2C972	ΣX^2 interne
2C99A	2C99A	ΣY^2 interne
2C9BD	2C9C2	ΣX*Y interne
2CA30	647BB	LR interne
2CB02	2CADF	PREDY interne (1:Nombre Réel)
2CB75	2CB52	PREDX interne (1:Nombre Réel)
2D2E6		Erreur "Non existent ΣDAT"
2D816	2D816	RECN interne (1:Chaîne/Nom Global/Nom Local)
2D9F5	2D9F5	SERVER interne
2E5AB	2E5AB	SEND interne (1:Nom Global/Nom Local)
2E6EB	2E6EB	SEND interne (1:Liste)
2E7EF	2E7EF	KGET interne (1:Chaîne/Nom Global/Nom Local)
2E835	2E835	KGET interne (1:List)
2E876	2E876	FINISH interne
2E8D1	2E8D1	PKT interne (2:Chaîne,1:Chaîne)
2EC84	2EC84	BAUD interne (1:Nombre Réel)
2ECCA	2ECCA	PARITY interne (1:Nombre Réel)
2ED10	2ED10	TRANSIO interne (1:Nombre Réel)
2ED4C	2ED4C	CKSM interne (1:Nombre Réel)
2EDA6	2EDA6	KERRM interne
2EDE1	2EDE1	BUFLEN interne
2EDF5	2EDF5	STIME interne (1:Nombre Réel)
2EE18	2EE18	SBRK interne
2EE6F	2EE6F	XMIT interne (1:Chaîne)
2EE97	2EE97	SRECV interne (1:Nombre Réel)
30794	30794	"HPHP48-M" (Selon le modèle)
315C6	315C6	CLOSEIO interne
31868	31868	CR interne
318A4	318A4	PRSTC interne
318FE	318FE	PR1 interne
31A25	31A25	PRST interne
31D56	31D56	PRVAR interne (1:Nom Global/Nom Local)

31DAB	31DAB	OLDPRT interne
31EE2	31EE2	PRLCD interne
31FFD	31FFD	DELAY interne (1:Nombre Réel)
32F77	32F77	ROOT interne (3 arguments)
3A9CE	3A9CE	OFF interne
3AA0A	3AA0A	Mode α
3AA37	3AA37	Mode α Shift-Orange (Shift-Violet sur g/gx)
40788	40788	Programme vide
40D25	40D25	Touche a a
415C9	415C9	RCLMENU interne
41679	41679	TMENU interne (1:Nombre Réel)
41AA1	41AA1	STOKEYS interne (1:Liste)
41B28	41B28	ASN interne (2:Tout,1:Nombre Réel)
41B3C	41B3C	DELKEYS interne (1:Liste)
41B69	41B69	DELKEYS interne (1:Nombre Réel)
41BA5	41BA5	STOKEYS interne (1:Nom Global/Nom Local)
41BB9	41BB9	DELKEYS interne (1:Nom Global/Nom Local)
4C8F4	4C944	BINS interne (3:Nombre Réel,2:Réel,1:Nombre Réel)
4F37C	4F37C	STO interne (2:Graphique,1:PICT)
53725	53725	SF interne (utilisateur) (1:Binaire Système)
53731	53731	SF interne (système) (1:Binaire Système)
53755	53755	CF interne (utilisateur) (1:Binaire Système)
53761	53761	CF interne (système) (1:Binaire Système)
53778	53778	FS? interne (utilisateur) (1:Binaire Système) → VRAI/FAUX
53784	53784	FS? interne (système) (1:Binaire Système) → VRAI/FAUX
53C37	53C37	HEX interne
53C43	53C43	BIN interne
53C4F	53C4F	OCT interne
53C5B	53C5B	DEC interne
53C96	53C96	STWS interne (1:Nombre Réel)
53CAA	53CAA	STWS interne (1:Binaire Système)
53CF0	53CF0	RCWS interne
53D04	53D04	AND interne (2:Entier Binaire,1:Entier Binaire)
53D15	53D15	OR interne (2:Entier Binaire,1:Entier Binaire)
53D26	53D26	XOR interne (2:Entier Binaire,1:Entier Binaire)
53D4E	53D4E	NOT interne (1:Entier Binaire)

53D5E	53D5E	SL interne
53D6E	53D6E	SLB interne
53D81	53D81	SR interne
53D91	53D91	SRB interne
53DA4	53DA4	RR interne
53DE1	53DE1	RRB interne
53E0C	53E0C	RL interne
53E3B	53E3B	RLB interne
53E65	53E65	ASR interne
54039	54039	RCWS → Binaire Système interne
54954	54954	Dérivation complète interne (2:Symbolique,1:Symbolique)
55927	55927	= interne (2:Tout,1:Tout)
55C3D	55C3D	% interne (2:Symbolique,1:Nombre Réel/Unité)
55C56	55C56	% interne (2:Nombre Réel/Unité,1:Symbolique)
55C6F	55C6F	% interne (2:Symbolique,1:Symbolique)
55C88	55C88	%CH interne (2:Symbolique,1:Nombre Réel/Unité)
55CA1	55CA1	%CH interne (2:Nombre Réel/Unité,1:Symbolique)
55CBA	55CBA	%CH interne (2:Symbolique,1:Symbolique)
55CD3	55CD3	%T interne (2:Symbolique,1:Nombre Réel/Unité)
55CEC	55CEC	%T interne (2:Nombre Réel/Unité,1:Symbolique)
55D05	55D05	%T interne (2:Symbolique,1:Symbolique)
56949	56949	Σ interne (4:Symbolique,3:Symbolique,2:Symbolique,1:Tout)
56A06	56A06	Σ interne (4:Symbolique,3:Symbolique,2:Nombre Réel,1:Tout)
56A4C	56A4C	Σ interne (4:Symbolique,3:Nombre Réel,2:Symbolique,1:Tout)
56AC9	56AC9	Σ interne (4:Symbolique,3:Nombre Réel, 2:Nombre Réel,1:Tout)
572A2	57293	ISOL interne (2:Symbolique,1:Nom Global)
57A0C	57A0C	EXPAN interne (1:Réel/Complexe/Symbolique)
57D90	57D90	COLCT interne (1:Réel/Complexe/Symbolique)
58D75	58D75	SHOW interne (2:Symbolique,1:Nom Global/Nom Local)
591AD	591AD	QUAD interne (2:Symbolique,1:Nom Global)
595DD	595DD	TAYLR interne (3:Symbolique,2:Nom Global,1:Nombre Réel)

59F91	59F91	SIZE interne (1:Symbolique)
61C1C	61C1C	alloue taille en quartets (2:Objet,1:Binaire Système)
62080	62080	Remplace TOS par VRAI
620A0	620A0	Remplace TOS par FAUX
62A34	62A34	RCL interne (1:Nom Global/Nom Local) → Contenu,VRAI/FAUX
62C69	62C69	NEWOB et SWAP interne
632D1	632D1	→Programme et EVAL interne (N..2:Tout,1:Binaire Système)
634F7	634F7	VRAI et FAUX interne
6350B	6350B	FAUX et VRAI interne
63A29	63A29	Stocke TOS dans "dvar' (Nom Local)
63A3D	63A3D	Stocke TOS dans "LNAME' (Nom Local)
63B6E	63B6E	Dérivation pas à pas (non évaluée)
63FE7	63FE7	→programme interne
64775	64775	DTAG interne (1:Tout)
647A2	647A2	DTAG niveau 2 interne

Tests

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03A81	03A81	VRAI
03AC0	03AC0	FAUX
03ADA	03ADA	XOR interne (1:VRAI/FAUX,2:VRAI/FAUX) → VRAI/FAUX
03AF2	03AF2	NOT interne (1:VRAI/FAUX)
03B2E	03B2E	si TOS-1 = TOS (les objets ont même adresse) → VRAI/FAUX
03B46	03B46	si TOS-1 = FAUX, alors DROP TOS, sinon DROP TOS-1
03B75	03B75	si TOS-1 = VRAI, alors DROP TOS, sinon DROP TOS-1
03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX

03CA6	03CA6	si TOS = 0 (Binaire Système) → VRAI/FAUX
03CC7	03CC7	si TOS ≠ 0 (Binaire Système) → VRAI/FAUX
03CE4	03CE4	si TOS-1 < TOS (Binaire Système) → VRAI/FAUX
03D19	03D19	si TOS-1 = TOS (Binaire Système) → VRAI/FAUX
03D4E	03D4E	si TOS-1 ≠ TOS (Binaire Système) → VRAI/FAUX
03D83	03D83	si TOS-1 > TOS (Binaire Système) → VRAI/FAUX
03EB1	03EB1	AND interne (2:Binaire Système,1:Binaire Système)
0712A	0712A	si TOS = VRAI, alors saute l'objet suivant
071C8	071C8	si TOS = FAUX, alors sort de la boucle, sinon itère la boucle
071EE	071EE	si TOS = FAUX, saute les deux objets suivants et itère la boucle
07E99	07E99	Si TYPE(TOS) = xlib → VRAI/FAUX
08DD4	08DD4	Si TOS = répertoire HOME → VRAI/FAUX
0F584	0F584	== interne (2:Réel/Unité,1:Nombre Réel/Unité)
0F598	0F598	≠ interne (2:Réel/Unité,1:Nombre Réel/Unité)
0F5AC	0F5AC	< interne (2:Réel/Unité,1:Nombre Réel/Unité)
0F5C0	0F5C0	> interne (2:Réel/Unité,1:Nombre Réel/Unité)
0F5D4	0F5D4	≤ interne (2:Réel/Unité,1:Nombre Réel/Unité)
0F5E8	0F5E8	≥ interne (2:Réel/Unité,1:Nombre Réel/Unité)
1420A	1420A	> interne (2:Chaîne,1:Chaîne)
142A6	142A6	< interne (2:Chaîne,1:Chaîne)
142BA	142BA	≥ interne (2:Chaîne,1:Chaîne)
142E2	142E2	≤ interne (2:Chaîne,1:Chaîne)
188AF	188AF	si SIZE(TOS) = SIZE(TOS-1) (Chaîne), NEWOB sinon SWAP
1A2DA	1A2DA	Si TOS = Objet ROM → VRAI/FAUX
1A4A3	1A4A3	IFTE interne (3:Nombre Réel,2:Tout,1:Tout)
1A4CD	1A4CD	IFT (XLIB 2 48)
1A4F0	1A4F0	IFT interne (2:Nombre Réel,1:Tout)
1A513	1A513	IFT interne (2:Symbolique,1:Tout)
1A8D8	1A8D8	= (XLIB 2 59)
1E761	1E761	SAME (XLIB 2 228)
1E972	1E972	== (XLIB 2 233)
1EA30	1EA30	== interne (2:Tout,1:Tout)
1EA44	1EA44	== interne (2:Signé/Tout,1:Signé/Tout)
1EA6C	1EA6C	== interne (2:Nombre Réel,1:Nombre Complexe)
1EA76	1EA76	== interne (2:Nombre Complexe,1:Nombre Réel)

1EA9D	1EA9D	≠ (XLIB 2 234)
1EB51	1EB51	≠ interne (2:Tout,1:Tout)
1EB65	1EB65	≠ interne (2:Signé/Tout,1:Signé/Tout)
1EB8D	1EB8D	≠ interne (2:Nombre Réel,1:Nombre Complexe)
1EB97	1EB97	≠ interne (2:Nombre Complexe,1:Nombre Réel)
1EBBE	1EBBE	< (XLIB 2 235)
1EC40	1EC40	< interne (2:Nombre Réel,1:Nombre Réel)
1EC5D	1EC5D	> (XLIB 2 236)
1ECDF	1ECDF	> interne (2:Nombre Réel,1:Nombre Réel)
214A9	214A9	Réel→Binaire Système et vérifie ≥ <100h> et π <700h>
21638	21638	Si TYPE(TOS) = Nombre Réel, alors execute prochain sinon saute
22EC3	22EC3	IF (XLIB 1792 0)
22EFA	22EFA	THEN (XLIB 1792 1)
22F22	22F22	THEN interne (1:Nombre Réel)
22F4F	22F4F	THEN interne (1:Symbolique)
22FB5	22FB5	ELSE (XLIB 1792 2)
22FD5	22FD5	END (XLIB 1792 3)
23033	23033	WHILE (XLIB 1792 5)
2305D	2305D	REPEAT (XLIB 1792 6)
23085	23085	REPEAT interne (1:Nombre Réel)
230A3	230A3	REPEAT interne (1:Symbolique)
230C3	230C3	DO (XLIB 1792 7)
230ED	230ED	UNTIL (XLIB 1792 8)
23103	23103	START (XLIB 1792 9)
23144	23144	START interne (2:Nombre Réel,1:Nombre Réel)
23167	23167	START interne (2:Réel/Symbolique,1:Symbolique)
23180	23180	START interne (2:Symbolique,1:Nombre Réel)
231A0	231A0	FOR (XLIB 1792 10)
231E1	231E1	FOR interne (2:Nombre Réel,1:Nombre Réel)
23213	23213	FOR interne (2:Réel/Symbolique,1:Symbolique)
2322C	2322C	FOR interne (2:Symbolique,1:Nombre Réel)
2324C	2324C	NEXT (XLIB 1792 11)
2326A	2326A	NEXT interne
23380	23380	STEP (XLIB 1792 12)
233A8	233A8	STEP interne (1:Symbolique)
233C1	233C1	STEP interne (1:Nombre Réel)
233DF	233DF	IFERR (XLIB 1792 13)

23472	23472	HALT (XLIB 1792 14)
23694	23694	END (XLIB 1792 22)
236B9	236B9	END (XLIB 1792 23)
2371F	2371F	THEN (XLIB 1792 24)
2378D	2378D	CASE (XLIB 1792 25)
237A8	237A8	THEN (XLIB 1792 26)
2A738	2A738	Si TOS < 0 (Nombre Réel) → VRAI/FAUX
2A799	2A799	Si TOS > 0 (Nombre Réel) → VRAI/FAUX
2A81F	2A81F	< interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A871	2A871	< interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A87F	2A87F	> interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A88A	2A88A	> interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A895	2A895	≥ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8A0	2A8A0	≥ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8AB	2A8AB	≤ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8B6	2A8B6	≤ interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8C1	2A8C1	== interne (2:Nombre Réel,1:Nombre Réel) → VRAI/FAUX
2A8CC	2A8CC	≠ interne (2:Nombre Réel,2:Nombre Réel) → VRAI/FAUX
5380E	5380E	Si TOS = VRAI alors 1 sinon 0
544D9	544D9	== interne (2:Entier Binaire,1:Entier Binaire)
544EC	544EC	≠ interne (2:Entier Binaire,1:Entier Binaire)
54500	54500	> interne (2:Entier Binaire,1:Entier Binaire)
5452C	5452C	≥ interne (2:Entier Binaire,1:Entier Binaire)
5453F	5453F	≤ interne (2:Entier Binaire,1:Entier Binaire)
54552	54552	< interne (2:Entier Binaire,1:Entier Binaire)
54565	54565	IFTE interne (3:Symbolique,2/1:Réel/Complexe/Symbolique)
55927	55927	= interne (2:Tout,1:Tout)

5599A	5599A	AND interne (2:Symbolique,1:Nombre Réel)
559B3	559B3	AND interne (2:Nombre Réel,1:Symbolique)
559CC	559CC	AND interne (2:Symbolique,1:Symbolique)
559E5	559E5	OR interne (2:Symbolique,1:Nombre Réel)
559FE	559FE	OR interne (2:Nombre Réel,1:Symbolique)
55A17	55A17	OR interne (2:Symbolique,1:Symbolique)
55A30	55A30	XOR interne (2:Symbolique,1:Nombre Réel)
55A49	55A49	XOR interne (2:Nombre Réel,1:Symbolique)
55A62	55A62	XOR interne (2:Symbolique,1:Symbolique)
55A7B	55A7B	== interne (2:Symbolique,1:Nombre Réel/Complexe/Unité)
55A94	55A94	== interne (2:Complexe,1:Nombre Réel/Complexe/Unité)
55AAD	55AAD	== interne (2:Symbolique,1:Symbolique)
55AC6	55AC6	≠ interne (2:Symbolique,1:Nombre Réel/Complexe/Unité)
55ADF	55ADF	≠ interne (2:Nombre Réel/Complexe/Unité,1:Symbolique)
55AF8	55AF8	≠ interne (2:Symbolique,1:Symbolique)
55B11	55B11	< interne (2:Symbolique,1:Nombre Réel/Unité)
55B2A	55B2A	< interne (2:Nombre Réel/Unité,1:Symbolique)
55B43	55B43	< interne (2:Symbolique,1:Symbolique)
55B5C	55B5C	> interne (2:Symbolique,1:Nombre Réel/Unité)
55B75	55B75	> interne (2:Nombre Réel/Unité,1:Symbolique)
55B8E	55B8E	> interne (2:Symbolique,1:Symbolique)
55BA7	55BA7	≤ interne (2:Symbolique,1:Nombre Réel/Unité)
55BC0	55BC0	≤ interne (2:Nombre Réel/Unité,1:Symbolique)
55BD9	55BD9	≤ interne (2:Symbolique,1:Symbolique)
55BF2	55BF2	≥ interne (2:Symbolique,1:Nombre Réel/Unité)
55C0B	55C0B	≥ interne (2:Nombre Réel/Unité,1:Symbolique)
55C24	55C24	≥ interne (2:Symbolique,1:Symbolique)
618F7	618F7	Si TOS = VRAI, alors DROP, suivant et retour sinon saute suivant
6194B	6194B	Si TOS = VRAI, alors prends et retourne
61993	61993	Si TOS = VRAI, alors execute suivant/retout sinon saute suivant
619AD	619AD	Si TOS = VRAI, saute suivant, sinon execute suivant et retour

619BC	619BC	Si TOS ≠ VRAI alors saute suivant
61A02	61A02	Si TOS = VRAI alors SF le drapeau CARRY
61A2C	61A2C	Si TOS ≠ VRAI, alors retour
61A3B	61A3B	Si TOS = VRAI, alors retour
61AD8	61AD8	Si TOS = VRAI, alors execute/saute, sinon saute/execute
61B72	61B72	Si TOS ≠ VRAI, alors DROP
61F1B		Si TOS = VRAI, alors SWAP
62025	62025	teste TYPE(TOS) = Caractère, remplace TOS par VRAI/FAUX
62035	62035	teste TYPE(TOS) = Nom Global, → VRAI/FAUX
6203A	6203A	teste TYPE(TOS) = Nom Global, remplace TOS par VRAI/FAUX
6204A	6204A	teste TYPE(TOS) = Unité, → VRAI/FAUX
6204F	6204F	teste TYPE(TOS) = Unité, remplace TOS par VRAI/FAUX
62063	62063	Verifie En-tête → VRAI/FAUX
62115	62115	teste TYPE(TOS) = Nom Local → VRAI/FAUX
6211A	6211A	teste TYPE(TOS) = Nom Local remplace TOS par VRAI/FAUX
6212A	6212A	teste TYPE(TOS) = Binaire Système → VRAI/FAUX
6212F	6212F	teste TYPE(TOS) = Binaire Système, remplace TOS par VRAI/FAUX
6213F	6213F	teste TYPE(TOS) = Entier Binaire → VRAI/FAUX
62144	62144	teste TYPE(TOS) = Entier Binaire, remplace TOS par VRAI/FAUX
62154	62154	teste TYPE(TOS) = Chaîne → VRAI/FAUX
62159	62159	teste TYPE(TOS) = Chaîne, remplace TOS par VRAI/FAUX
62169	62169	teste TYPE(TOS) = Nombre Réel → VRAI/FAUX
6216E	6216E	teste TYPE(TOS) = Nombre Réel, remplace TOS par VRAI/FAUX
6217E	6217E	teste TYPE(TOS) = Complexe → VRAI/FAUX
62183	62183	teste TYPE(TOS) = Nombre Complexe, remplace TOS par VRAI/FAUX
62193	62193	teste TYPE(TOS) = Matrice → VRAI/FAUX
62198	62198	teste TYPE(TOS) = Matrice, remplace TOS par VRAI/FAUX

621A8	621A8	teste TYPE(TOS) = Fonction → VRAI/FAUX
621AD	621AD	teste TYPE(TOS) = Fonction, remplace TOS par VRAI/FAUX
621BD	621BD	teste TYPE(TOS) = Répertoire → VRAI/FAUX
621C2	621C2	teste TYPE(TOS) = Répertoire remplace TOS par VRAI/FAUX
621D2	621D2	teste TYPE(TOS) = Expression Algébrique → VRAI/FAUX
621D7	621D7	teste TYPE(TOS) = Expression Algébrique, remplace TOS par VRAI/FAUX
621E7	621E7	teste TYPE(TOS) = Programme → VRAI/FAUX
621EC	621EC	teste TYPE(TOS) = Programme, remplace TOS par VRAI/FAUX
621FC	621FC	teste TYPE(TOS) = Graphique → VRAI/FAUX
62201	62201	teste TYPE(TOS) = Graphique, remplace TOS par VRAI/FAUX
62211	62211	teste TYPE(TOS) = Liste → VRAI/FAUX
62216	62216	teste TYPE(TOS) = Liste, remplace TOS par VRAI/FAUX
62226	62226	teste TYPE(TOS) = Signé → VRAI/FAUX
6222B	6222B	teste TYPE(TOS) = Signé, remplace TOS par VRAI/FAUX
62256	62256	teste TYPE(TOS-1) = complexe, remplace TOS par VRAI/FAUX
6289B	6289B	Si TOS > TOS-1 (Binaire Système) → VRAI/FAUX
62D81	62D81	Si TOS-1 ≤ TOS (Nombre Réel) alors SWAP
62D9F	62D9F	Si pop TOS ≠ VRAI alors SWAP
62F1B	62F1B	Si TOS = VRAI, alors SWAP
62F43	62F43	Si TOS = VRAI, alors DROP, sinon SWAP/DROP
62F5C	62F5C	Si TOS = VRAI, alors SWAP/DROP, sinon DROP
6317D	6317D	Si TOS = VRAI, alors UVAL interne
63399	63399	Si TOS-1 > TOS (Binaire Système), alors saute suivant
633B2	633B2	ELSE interne
633C6	633C6	UNTIL interne
6347F	6347F	DUP et FOR 0 to (TOS)-1 (1:Binaire Système) interne
635C4	635C4	≠ interne (2:Tout,1:Tout) → VRAI/FAUX
6365F	6365F	OVER et Si TOS-1 < TOS (Binaire Système) → VRAI/FAUX interne

63673	63673	Si TOS < 3 (Binaire Système) → VRAI/FAUX interne
63687	63687	DUP et Si TOS < 7 (Binaire Système) → VRAI/FAUX interne
636C8	636C8	Si TOS ≠ 2 (Binaire Système) →VRAI/FAUX interne
636DC	636DC	OVER et Si TOS-1 > TOS (Binaire Système) → VRAI/FAUX interne
636F0	636F0	Si TOS > 1 (Binaire Système) → VRAI/FAUX interne
63A6F	63A6F	DUP et Si TOS est { } → VRAI/FAUX interne
63B19	63B19	si TOS ≠ VRAI, alors Erreur "Bad Argument Value"
63B2D	63B2D	Si TYPE(TOS) ≠ Nombre Réel, alors Erreur "Bad Argument Type"
63B46	63B46	Si TOS ≠ VRAI, alors Erreur "Bad Argument Type"
63CFE	63CFE	Si SAME, alors execute suivant/retour, sinon saute suivant
63D3A	63D3A	Si TOS-1 = TOS (Binaire Système), saute suivant, sinon execute
63D67	63D67	Si TOS-1 > TOS (Binaire Système), execute suivant, sinon saute

Unités

Adresse HP48s/sx	Adresse HP48g/gx	Commentaire de l'effet de l'adresse
03B97	03B97	SAME interne (1,2:Tout) → VRAI/FAUX
05089	05089	UVAL interne (1:Unité)
052FA	052FA	+ interne (2:Liste,1:Tout)
05481	05481	→UNIT interne (N..2:Tout,1:Binaire Système)
0F33A	0F33A	UNIT interne
0F34E	0F34E	OBJ→ interne (1:Unité)
0F371	0F371	CONVERT interne
0F561	0F561	Exprime le résultat numériquement en unités MKSA
0F584	0F584	== interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F598	0F598	≠ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5AC	0F5AC	< interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5C0	0F5C0	> interne (2:Nombre Réel/Unité,1:Réel/Unité)

0F5D4	0F5D4	≤ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5E8	0F5E8	≥ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F5FC	0F5FC	ABS interne (1:Unité)
0F615	0F615	NEG interne (1:Unité)
0F62E	0F62E	SIN interne (1:Unité)
0F660	0F660	COS interne (1:Unité)
0F674	0F674	TAN interne (1:Unité)
0F6A2	0F6A2	+ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F774	0F774	- interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F792	0F792	* interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F823	0F823	/ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F841	0F841	INV interne (1:Unité)
0F878	0F873	^ interne (2:Nombre Réel/Unité,1:Réel/Unité)
0F8FA	0F8FA	XROOT interne (1:Unité)
0F913	0F913	SQ interne (1:Unité)
0F92C	0F92C	√ interne (1:Unité)
0F945	0F945	UBASE interne (1:Unité)
0FB6F	0FB6F	MAX interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FB8D	0FB8D	MIN interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FBAB	0FBAB	% interne (2:Unité,1:Nombre Réel)
0FC3C	0FC3C	%CH interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FCCD	0FCCD	%T interne (2:Nombre Réel/Unité,1:Réel/Unité)
0FCE6	0FCE6	SIGN interne (1:Unité)
0FCFA	0FCFA	IP interne (1:Unité)
0FD0E	0FD0E	FP interne (1:Unité)
0FD22	0FD22	FLOOR interne (1:Unité)
0FD36	0FD36	CEIL interne (1:Unité)
0FD68	0FD68	RND interne (2:Unité,1:Nombre Réel)
0FD8B	0FD8B	TRNC interne (2:Unité,1:Nombre Réel)
140AB	140AB	DISP interne (2:Tout,1:Nombre Réel)
15978	15978	→STR interne (1:Tout)
15B13	15B13	→STR interne (1:Tout)
15B31	15B31	→STR interne (1:Tout)
18513	18513	STO interne (2:Tout,1:Nom Global)
196DB	196DB	CONVERT (XLIB 2 11)
1971B	1971B	UVAL (XLIB 2 12)
1974F	1974F	UNIT (XLIB 2 13)
19771	19771	UBASE (XLIB 2 14)

197A5	197A5	UFACT (XLIB 2 15)
197C8	197C8	UFACT interne
1AB67	1AB67	+ (XLIB 2 68)
1AC93	1AC93	+ interne (2:Tout,1:Liste)
1ACA7	1ACA7	+ interne (2:Chaîne,1:Tout)
1ACBB	1ACBB	+ interne (2:Tout,1:Chaîne)
1ACD7	1ACD7	+ (XLIB 2 69)
1AD09	1AD09	- (XLIB 2 70)
1ADEE	1ADEE	* (XLIB 2 71)
1AF05	1AF05	/ (XLIB 2 72)
1B02D	1B02D	^ (XLIB 2 73)
1B278	1B278	INV (XLIB 2 76)
1CA3A	1CA3A	SIZE interne (1:Unité)
1FAEB	1FAEB	_ (XLIB 2 267)
1FB31	1FB31	_ interne (1:Nombre Réel/Unité)
28A38	28A38	_ interne (1:Symbolique)

Remarques

La plupart des nouvelles fonctions des HP48g et gx sont en ROM cachée (zone de 80000h à FFFFFh). On ne peut donc pas les localiser simplement. Lorsque la ROM est cachée (cas normal), la mémoire de la HP48gx est organisé de la manière suivante :

```
00000-7FFFF ROM        256k
                       (au lieu de 512k)
80000-BFFFF RAM        128k
C0000-FFFFF Port 1     128k
```

1 • Accès à la ROM cachée *PIK*

Le programme est à enregistrer sous le nom : PIK

Son checksum est : #F838h (54 octets). Le programme suivant est un PEEK qui travaille sur la ROM cachée, de #C0000h à #FFFFFh.

Voici le listing de ce programme :

```
02D9D   Entête de programme
196BB   B→R             cette astuce sert
1969B   R→B             à vérifier le type
02DCC   Entête de code
00044   68 quartets
147     C=DAT1    A
137     CD1EX
06      RSTK=C
179     D1=D1+    10    Positionnement sur
147     C=DAT1    A     l'adresse
137     CD1EX
06      RSTK=C
340000C LCHEXC0000      Nous désactivons
804     UNCNFG          l'adressage en C0000
15BF    A=DAT1    16    Lecture de la donnée
                        Réadressage :
805     CONFIG          Taille = C0000h
805     CONFIG          et Adresse = C0000h
07      C=RSTK
137     CD1EX
159F    DAT1=A    16
07      C=RSTK          Ecriture de la donnée
137     CD1EX
159F    DAT1=A    16
07      C=RSTK
137     CD1EX
142     A=DAT0    A     Fin du code
164     DO=D0+    5
808C    PC=(A)

0312B   Fin du programme
```

Pour utiliser ce programme, saisissez le listing suivant sans espaces ni sauts de lignes. Son checksum doit être : #488Fh (98 octets) Executez alors ASS.

```
"D9D 20B B69 1B9 691 CCD 204 400 014 713
706 179 147 137 063 400 00C 804 15B F80
580 507 137 159 F07 137 142 164 808 CB2
130 0"
```

Utilisation :
Entrez une adresse : par exemple #C5000, puis executez le programme précédent.

2 • Adresse d'un objet en RAM *ADR*

Ce programme est à enregistrer sous le nom : ADR

Son checksum est : #EC0Ah (48,5 octets)

Listing

La chaîne suivante doit être saisie sans espaces ni sauts de lignes. Son checksum est : #AE58h (89 octets). Executez alors ASS pour fabriquer le programme.

```
"D9D 20E 4A2 0A0 000 000 007 566 0DB BF1
CCD 207 200 014 317 414 713 616 914 013
61C 414 216 480 8C4 423 0B2 130"
```

Exemple d'utilisation:

```
11 [ENTER]
'C' STO
C [ENTER]
ADR
```

renvoie l'adresse du chiffre que l'on vient de stocker.

18

Extensions Bibliothèques, matrices, affichage, compression

1 Fabrication de bibliothèques

Quelques rappels sur l'utilisation des bibliothèques...

Pour être utilisée, une bibliothèque doit se trouver dans l'un des ports (port 0, 1 ou 2 sur la HP48sx et port 0 sur la HP48s).

Après son stockage, il faut attacher la bibliothèque (Cette opération peut être automatisée).
Pour l'attacher, il faut choisir un répertoire (en général HOME). Le choix de ce répertoire est important, car seuls ce répertoire et ses répertoires fils pourront utiliser la bibliothèque.

Pour cela, on exécute *numéro* ATTACH, où *numéro* est le numéro de la bibliothèque.

Si l'on veut enlever une bibliothèque...

Si elle est stockée dans un port extérieur (HP48sx ou gx), il faut d'abord que le port soit ouvert (interrupteur sur la carte).

Ensuite, il faut se placer dans le répertoire où la bibliothèque a été attachée. Puis il faut la détacher en tapant: *numéro* DETACH.

Ensuite il faut taper : *port : numéro PURGE* où *port* est le numéro du port où la bibliothèque est stockée.

1
DFL

Sous-programme de →LIB

Ce programme est à enregistrer sous le nom : DFL

Son checksum est : #62C8h (124,5 octets)

Listing :

```
« SWAP 1 3 PICK SIZE FOR S OVER OVER
SWAP S IF GET POS DUP THEN OVER 1 3 PICK
1 - SUB ROT ROT 1 + 1000 SUB + ELSE DROP
END NEXT SWAP DROP »
```

2
F&R

Sous-Programme de →LIB

Ce programme est à enregistrer sous le nom : F&R

Son checksum est : #CC71h (90 octets)

Listing :

Compiler avec ASS la chaîne ci-après.Avant d'exécuter ASS, pour vérifica-

tion, tapez DUP BYTES. Le résultat doit être # 8E43h 170.
Il faut rentrer la chaîne sans espaces ni sauts de lignes.

```
"D9D 208 6A8 12B F81 119 203 330 0D9 D20
592 309 FF3 02A 170 2C2 30A 321 659 230
1B5 468 813 07C C30 EE1 70D 9D2 0CA 130
952 36C 121 639 150 A32 166 365 004 736
BCE 26D 623 6A6 F36 B21 305 E17 044 230
A21 16B 213 0B2 130"
```

Remarque : Le symbole & s'obtient en tapant :
[α] [ORANGE] [ENTER]
sur une HP48s ou sx
et
[α] [VIOLET] [ENTER]
sur une HP48 g ou gx

3
CRC

Sous-programme de →LIB

Ce programme est à enregistrer sous le nom : CRC

Son checksum est : #8723h (144 octets)

Listing :

Compiler avec ASS la chaîne ci-après. Avant de taper ASS, pour vérification, tapez DUP BYTES. Le résultat doit être # F5ACh 278.
Il faut rentrer la chaîne sans espaces ni sauts de lignes.

```
"D9D 20D 295 12B F81 D00 40D 9D2 088 130
636 508 813 09F F30 4EC 307 F81 67A C61
E4A 20A 000 000 000 756 60C AF0 6CC D20
8A0 008 F14 660 CC8 FB9 760 D81 431 301
691 741 431 311 79D 014 A31 03B 6A3 190
```

```
9EA 803 07B 6A7 F30 34F 000 00E F2D 734
180 10D 0CF 480 CA6 8FF 147 F61 457 210
141 161 CD5 BA8 D34 150 100 FC1 470 EF6
120 FE0 EF2 110 0EF E01 442 30B 213 0B2
130"
```

4
RVHX

Sous-programme de →LIB.

Ce programme est à enregistrer sous le nom : RVHX

Son checksum est : #CA34h (36 octets)

Listing

Compiler avec ASS la chaîne ci-après. Avant de exécuter ASS, pour vérification, tapez DUP BYTES. Le résultat doit être # FC37h 60.
Il faut rentrer la chaîne sans espaces ni sauts de lignes.

```
"D9D 207 F49 132 230 84E 204 044 143 535
B20 405 923 0A8 526 337 50B 213 0"
```

5
AFFV

Sous-programme de →LIB.

Ce programme est à enregistrer sous le nom : AFFV

Son checksum est : #ED16h (210 octets)

Listing :

```
« { $ROMID $TITLE $CONFIG $VISIBLE $HID-
DEN $VARS $MESSAGE } 1 7 FOR K DUP K GET
DUP IFERR RCL THEN DROP " -" END SWAP
#5BE9h SYSEVAL 2 20 SUB ":" + SWAP →STR
+ K DISP NEXT 3 FREEZE DROP »
```

Remarque :
Le caractère $ s'obtient en tapant :

```
[α] [ORANGE] [4]
```

sur une HP48 s ou sx
et

```
[α] [VIOLET] [4]
```

sur une HP48 g ou gx

6
MKLIB

Sous-programme de →LIB.

Ce programme est à enregistrer sous le nom : MKLIB

Son checksum est : #6BDEh (465,5 octets)

Listing :

```
«  DUP EVAL {
{ "ROMID" « '$ROMID' STO AFFV » }
{ "TITLE" « '$TITLE' STO AFFV » }
{ "CONFIG" « '$CONFIG' STO AFFV » }
{ "HIDDEN" « '$HIDDEN' STO AFFV » }
{ "VISIBLE" « '$VISIBLE' STO AFFV » }
{ "VARS" « '$VARS' STO AFFV » }
{ "MESSAGE" « '$MESSAGE' STO AFFV » }
{ "→LIB" « UPDIR RCL 'LIB' SWAP OVER STO
→LIB LIB » }
```

```
AFFV } TMENU AFFV »
```

Remarque :
MKLIB laisse le répertoire intact et crée un répertoire intermédaire 'LIB'. →LIB remplace le répertoire par une bibliothèque.

Ce programme automatise la création de librairies à l'aide d'un menu interactif.

7
→LIB

Fabrique une bibliothèque

Ce programme est à enregistrer sous le nom : →LIB

Son checksum est : #96CAh (1800 octets)

Listing :

```
« DUP IF VTYPE 15 ≠ THEN #202h DOERR END
DUP EVAL IF $ROMID DUP TYPE THEN B→R END
3 RVHX IF $HIDDEN DUP TYPE 5 == THEN
VARS SWAP DFL ELSE DROP IF $VISIBLE DUP
TYPE 5 ≠ THEN DROP VARS END END DUP ORDER
IF $TITLE DUP TYPE 2 ≠ THEN DROP "00"
ELSE
#5B15h SYSEVAL DASS 6 999 SUB DUP 1 2
SUB + END $VARS PURGE $MESSAGE ROT {
$ROMID $TITLE $MESSAGE $VARS $VISIBLE
$HIDDEN } DUP PURGE DFL SIZE VARS DUP
SIZE "29E20" 7 PICK + OVER { } 0 ROT 1 -
FOR I OVER I 3 RVHX + + NEXT SWAP DROP
ROT { } 1 5 PICK FOR I OVER I GET DASS +
NEXT → R T M V N X U A « { } "" 1 N FOR
I U I
GET DUP RCL SWAP PURGE DASS 1 N FOR J A
```

```
J GET X J GET F&R NEXT IF I V ≤ THEN
"E1632BEF22BEF22D6E20E16321C4321C432D6E2
0"  OVER 6 15 SUB POS "000" "8" IFTE ROT
SWAP + X I GET 6 11 SUB + ELSE SWAP END
DUP SIZE 4 ROLL SWAP + ROT ROT SWAP +
NEXT SWAP DUP SIZE DUP 1 + 5 * DUP 5
RVHX ROT 1 SWAP FOR I OVER I 5 * - 4
PICK I GET + 5 RVHX + NEXT ROT IF U
'$CONFIG' POS DUP THEN GET ROT + 10 + 5
RVHX ELSE DROP2 SWAP DROP "00000" END
"A0000" SWAP + "E4A20" + SWAP + SWAP +
IF M TYPE 4 == THEN DUP SIZE 5 + 5 RVHX
SWAP + M
DASS + ELSE "00000" SWAP + END DUP SIZE
5 + 5 RVHX SWAP + "E4A20" + 1 16 START {
} NEXT 1 V FOR I I A OVER GET 6 99 SUB
OVER 1 - 3 RVHX + DUP SIZE 5 - 2 / DUP 3
+
ROLL ROT + ROT + SWAP ROLLD NEXT "" V 1
→LIST 0 CON 85 "" 1 16 START IF 5 ROLL
DUP { } SAME THEN DROP "00000" + ELSE
ROT ROT OVER 6 PICK SIZE + 5 RVHX + ROT
5 ROLL 5 ROLL 1 4 PICK SIZE FOR I OVER
SIZE 4 PICK I 1 + GET SWAP OVER 1 - 5 *
- PUT SWAP 3 PICK I GET + SWAP 2 STEP 5
ROLLD 5 ROLLD DROP END SWAP 5 - SWAP
NEXT SWAP
4 PICK SIZE + 5 RVHX + ROT 3 PICK OVER
SIZE CON 4 ROLL - SWAP 1 V FOR I OVER I
GET 5 RVHX + NEXT SWAP DROP + DUP SIZE 5
+ 5 RVHX SWAP + + T R + SWAP + DUP SIZE
9 + 5 RVHX SWAP + DUP CRC B→R 4 RVHX +
"04B20" SWAP + ASS UPDIR SWAP DUP PURGE
STO » »
```

Lancement du programme :

Saisir le nom du répertoire où se situent
les programmes de la bibliothèque puis saisir

```
→LIB [ENTER]
```

Ces programmes ont été créés suite au travail de Monsieur Frank Ochoa.
Exemple d'utilisation :

Créez le répertoire ESSAI : `'ESSAI' CRDIR`

Entrez dedans : `ESSAI [ENTER]`

Créez les paramètres nécessaires

```
800 [ENTER]
'$ROMID' [STO]

"PROPRIETAIRE DE LA HP" [ENTER]
'$TITLE' [STO]

« TEXT CLLCD "Jean Dupont
1 rue Léon Blum
PARIS"  1 DISP 3 WAIT » [ENTER]
'DEBUT' [STO]

« DEBUT »  '$CONFIG' [STO]

{ $CONFIG }
'$HIDDEN' [STO]
UPDIR [ENTER]

'ESSAI' [ENTER]

→LIB [ENTER]
```

Et au bout de quelques secondes, le répertoire ESSAI est remplacé par la bibliothèque

```
Library 800: PROPRIETAIRE DE LA HP
```

Tapez :

```
ESSAI [ENTER]
0 [STO]
```

Puis *ON-C.* Cette bibliothèque n'est volontairement pas attachée car elle n'a aucune utilité. Si on avait voulu l'attacher, il aurait fallu rajouter dans `$CONFIG` :

```
800 ATTACH
```

800 étant le numéro de cette bibliothèque ($ROMID)

La signification des variables

$ROMID	Numéro de la bibliothèque
$CONFIG	Programme auto-exécuté
$TITLE	Titre de la bibliothèque
$MESSAGE	Matrice de messages d'erreurs *(facultatif)*
$VARS	Liste des variables *(facultatif)*
$VISIBLE	Liste des noms qui doivent apparaître *(facultatif)*
$HIDDEN	Liste des noms qui doivent disparaître *(facultatif)*

2
Décomposition d'une bibliothèque

Ce programme fait appel à la procédure prédédemment vue F&R, en plus de ASS et DASS.

@
RCLX

Sous programme de LIB-→

Ce programme est à enegistrer sous le nom : RCLX

Son checksum est : #FB80h (33,5 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : # EEBDh (55 quartets). Exécutez ensuite **ASS**.

```
"D9D 207 F49 105 E70 881 309 9E7 039 916
4B2 A24 423 09C 2A2 B21 30"
```

ß
LIB→

Ce programme décompose une bibliothèque en ses éléments initiaux.

Ce programme est à enregistrer sous le nom : LIB→

Son checksum est : #28DDh (862,5 octets) pour la version HP48 s/sx
Son checksum est : #C557h (862,5 octets) pour la version HP48 g/gx

Version pour HP48 s et sx

Listing :

```
« "LIB" OVER + OBJ→ DUP CRDIR EVAL DUP
ATTACH LIBS IF OBJ→ THEN DROP2 IF DUP ""
≠ THEN '$TITLE' STO 0 END DROP ELSE "Lib
inconnue" DOERR END DUP # 18CEAh SYSEVAL
DUP2 DASS 4 8 SUB "29E" SWAP + 0 « IF
ROT →STR DUP 1 5 SUB "XLIB" == THEN DROP
"X" ROT + "." + SWAP + 2 CF ELSE ROT ROT
DROP2 2 SF END # 5B15h SYSEVAL » → N X I
P  « DUP # 8143h SYSEVAL # 3A81h SYSEVAL
IF == THEN '$CONFIG' STO END #811Ch
SYSEVAL #3A81h SYSEVAL  IF ==  THEN
'$MESSAGE' STO END  '$ROMID' STO {}
```

```
'$VISIBLE' STO  { $CONFIG } '$HIDDEN'
STO 1 CF DO IF N I RCLX THEN 1 SF END
'I' INCR DROP UNTIL 1 FS?C END 'I' DECR
DROP WHILE 'I' DECR 0 ≥ REPEAT DASS WHILE
DUP X POS DUP REPEAT OVER SWAP DUP 10 +
SUB DUP ASS DUP # 8CCCh SYSEVAL # 1950Bh
SYSEVAL P EVAL DASS  F&R END DROP ASS
SWAP N I P EVAL SWAP OVER STO IF 2 FS?C
THEN '$VISIBLE' ELSE '$HIDDEN' END STO+
END  » »
```

Version pour HP48 g et gx

Listing :

```
« "LIB" OVER + OBJ→ DUP CRDIR EVAL DUP
ATTACH LIBS IF OBJ→ THEN DROP2 IF DUP ""
≠ THEN '$TITLE' STO 0 END DROP ELSE "Lib
inconnue" DOERR END DUP # 18CEAh SYSEVAL
DUP2 DASS 4 8 SUB "29E" SWAP + 0 « IF
ROT →STR DUP 1 5 SUB "XLIB" == THEN DROP
"X" ROT + "." + SWAP + 2 CF ELSE ROT ROT
DROP2 2 SF END # 5B15h SYSEVAL » → N X I
P  « DUP # 8130h SYSEVAL # 3A81h SYSEVAL
IF == THEN '$CONFIG' STO END #8157h
SYSEVAL #3A81h SYSEVAL  IF ==  THEN
'$MESSAGE' STO END  '$ROMID' STO {}
'$VISIBLE' STO  { $CONFIG } '$HIDDEN'
STO 1 CF DO IF N I RCLX THEN 1 SF END
'I' INCR DROP UNTIL 1 FS?C END 'I' DECR
DROP WHILE 'I' DECR 0 ≥ REPEAT DASS WHILE
DUP X POS DUP REPEAT OVER SWAP DUP 10 +
SUB DUP ASS DUP # 8CCCh SYSEVAL # 1950Bh
SYSEVAL P EVAL DASS  F&R END DROP ASS
SWAP N I P EVAL SWAP OVER STO IF 2 FS?C
THEN '$VISIBLE' ELSE '$HIDDEN' END STO+
END  » »
```

Utilisation :
Saisissez le numéro de la librairie à décomposer puis exécutez`LIB→`.

3
FABRICATION DE MATRICES

Les deux programmes qui suivent ont pour but l'utilisation des matrices de chaînes. On peut utiliser des matrices de n'importe quel type.

Les matrices de réels et de complexes sont déjà prévues pour l'utilisateur. Or, les matrices de chaînes sont utilisées pour les messages d'erreurs. C'est pourquoi nous avons choisi de présenter ces deux programmes qui faciliterons l'utilisation des matrices de chaînes.

α
A→L

Transforme une matrice en liste de chaînes

Ce programme est à enregistrer sous le nom : A→L

Son checksum est : #715Dh (408 octets)

Listing :

```
« IF DUP TYPE 4 ≠ OVER DASS 11 15 SUB
"C2A20" ≠ OR THEN DROP "A→L Error" DOERR
END DASS 26 → L p « IF L 16 20 SUB
"10000" ≠ THEN "A→L:une dimension"
DOERR END { } L 21 25 SUB "#" SWAP +
STR→ DASS 11 15 SUB "#" SWAP + STR→ B→R
1 SWAP START L p 9999 SUB "C2A20" SWAP +
ASS DUP BYTES SWAP DROP 2 * 5 - p + 'p'
STO + NEXT » »
```

Lancement du programme :
Entrer une matrice de chaînes puis saisir...

```
A→L [ENTER]
```

Exemple d'utilisation :

sur une HP48 s ou sx

```
#72000 [ENTER]
SYS [ENTER]
```

renvoie

```
Array of String
```

Saisir `A→L [ENTER]`
On obtient une liste de messages d'erreurs.

sur une HP48 g ou gx

```
#73E60 [ENTER]
SYS [ENTER]
```

renvoie

```
Array of String
```

Saisir `A→L [ENTER]`
On obtient une liste de messages d'erreurs.

ß
L→A

Transforme une liste de chaînes en matrice

Ce programme est à enregistrer sous le nom : L→A

Son checksum est : #1584h (228,5 octets)

Listing :

```
« → L « "" 1 L SIZE FOR I IF L I GET
TYPE 2 ≠ THEN DROP #202h DOERR END L I
GET DASS 6 9999 SUB + NEXT L SIZE 5 RVHX
SWAP + "C2A2010000" SWAP + DUP SIZE 5 +
5 RVHX SWAP + "8E920" SWAP + ASS » »
```

Lancement du programme :
Saisir une liste de chaînes puis exécuter :

```
L→A [ENTER]
```

Exemple d'utilisation

```
{ "MOT 1" "MOT 2" "MOT 3" } [ENTER]
L→A [ENTER]
```

renvoie

```
Array of String
```

Remarque :
Ce programme utilise RVHX décrit plus haut.

Exemple :
Création de messages d'erreurs.

```
{ "Pas assez d'arguments"
"de Type d'Arguments"
"de Valeur d'Arguments"
"Nom indéfini"
"LASTARG désactivé"
"Expression Incomplète"
"() Implicite inactive"
"() Implicite Active" } [ENTER]

L→A [ENTER]
DUP [ENTER]
```

```
'ERREURS' [STO]

2 [ENTER]
#18CEAh [ENTER]
SYSEVAL [ENTER]
SWAP [ENTER]
#764Eh [ENTER]
SYSEVAL [ENTER]
```

Par exemple exécuter SWAP sans rien sur la pile.

```
SWAP Error:
Pas assez d'Arguments
```

apparaît en haut de l'écran. On peut ainsi tout remplacer.
Remarque :
On peut placer cette séquence dans une bibliothèque.

4
FABRICATION D'UN AFFICHAGE

Ces programmes nécessitent les programmes fabriquant des bibliothèques, décrits ci-dessus.Dans ce répertoire, saisir :

```
'STACK' CRDIR STACK [ENTER]
```

et rentrer les programmes suivants :

1
NEW

Crée un nouvel affichage
Ce programme est à enregistrer sous le nom : NEW
Son checksum est : #F8C4h (47 octets)

Listing :

Ce programme doit être assemblé avec ASS. Rentrez la chaîne ci-dessous, tapez DUP BYTES. Le résultat doit être en mode HEX : # 7409h 84

Si c'est le cas tapez ASS, puis 'NEW' STO. Sinon, vérifiez la saisie (La chaîne doit être saisie sans espaces ni sauts de lignes).

```
"D9D 208 E58 32A 170 9CB 04F 668 384 E20
60C 473 930 3E2 53C 302 4E5 E40 454 048
BE4 082 783 BA1 70B 213 0"
```

Lancement du programme :
Saisir

```
NEW [ENTER]
```

une fois que la bibliothèque sera crée et que l'on aura saisi ON-C

2
ST5

Passe en mode d'affichage 5 lignes

Ce programme est à enregistrer sous le nom : ST5

Son checksum est : #EDDBh (22,5 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : # EBE2h (35 quartets). Exécutez ensuite **ASS**.

```
"D9D 209 FF3 055 735 F71 401 373 5B2
130"
```

3
ST7

Passe en mode d'affichage 7 lignes

Ce programme est à enregistrer sous le nom : ST7

Son checksum est : #B725h (22,5 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : # 7936h (35 quartets). Exécutez ensuite **ASS**.

```
"D9D 209 FF3 052 735 F71 401 673 5B2
130"
```

4
L790.4

Sous-programme

Ce programme est à enregistrer sous le nom : L790.4

Son checksum est : #70D4h (198,5 octets)

Listing :

```
# 32C2h SYS
#1CB90h SYS
#60F9Bh SYS
#18CEAh SYS
26 #18CEAh SYSEVAL
```

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : # 32A7h (25 quartets). Exécutez ensuite **ASS**.

```
"D9D 203 8D3 068 926 B21 30"
```

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : # 69D0h (248 quartets). Exécutez ensuite **ASS**.

```
"D9D 202 C23 0CC D20 B10 001 431 74E 78F
B97 608 DC7 530 CC9 50E 4A2 051 000 000
000 000 000 000 00A E35 84E 20A 0C4 96E
647 562 7E6 16C 637 322 300 FAC 188 130
997 A28 DA1 6D9 D20 84E 20A 0C4 472 716
E63 736 279 647 322 308 98D 1C2 A20 510
009 4E6 475 627 E61 6C6 5F5 223 223 0CA
F06 322 304 423 032 230 B21 304 423 0B2
130"

# 32C2h SYS
#1CB90h SYS
#60F9Bh SYS
#18CEAh SYS
19 #18CEAh SYSEVAL
# 3D19h SYS
#61A2Ch SYS
# 3223h SYS
"Commande "
#225F5h SYS
# 3223h SYS

18
→PGM    puis 'L790.4'  STO
```

5
L790.5

Sous-programme

Ce programme est à enregistrer sous le nom : L790.5

Son checksum est : #FFFh (52 octets)

Listing:

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : #5AC5h (88 quartets). Exécutez ensuite **ASS**.

```
"D9D 201 4F8 384 E20 60C 473 930 3E2 938
2F8 384 E20 60C 473 930 3E2 63A 5F8 3D0
0A3 37F 83A C1A 344 193 B21 30"
```

6
L790.6

Sous-programme

Ce programme est à enregistrer sous le nom : L790.6

Son checksum est : #F9A2h (40 octets)

Listing :

```
#3FF9h SYS
#53778h SYS
#61A3Bh SYS
#42402h SYS
#619ADh SYS
'L790.7'
#3947Bh SYS
7 →PGM
'L790.6' STO
```

7
L790.7

Sous-programme

Ce programme est à enregistrer sous le nom : L790.7

Son checksum est : #53Fh (260 octets)

Listing :

```
#1314Dh SYS
# 3FF9h SYS
#53778h SYS
# 712Ah SYS
#39B0Ah SYS

#39AF1h SYS
# 3188h SYS
```

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**. Le résultat doit être : #4DDAh (235 quartets). Exécutez ensuite **ASS**.

```
"D9D 208 DA1 6C6 B46 089 93C 2A2 090 000
020 2A5 D80 2A1 708 813 03E 280 881 307
775 14C 536 881 308 DA1 6D9 D20 322 309
EB5 0B2 130 06A 93B BF0 667 326 C2A 207
000 0C5 5E2 26C AF0 6EF 116 636 500 DB4
64E C30 64B 30E E17 060 A93 5E1 70C 072
6C0 021 2C2 308 DA1 62F F93 C2A 209 000
002 020 8F1 159 230 B21 30"

#60FBBh SYS
#61AD8h SYS

#12635h SYS
#64BD0h SYS
# 402Bh SYS
```

```
#64D24h SYS
# 4035h SYS
#11A6Dh SYS
#64BD0h SYS
# 4071h SYS
#64D24h SYS
# 407Bh SYS
#6389Eh SYS
#39A83h SYS
#12635h SYS
# 3FEFh SYS
# 402Bh SYS
#64BD0h SYS
# 407Bh SYS
#6389Eh SYS
18 →PGM

#39958h SYS
# 3223h SYS
#12635h SYS
# 3FEFh SYS
# 4035h SYS
#11679h SYS
#12635h SYS
# 4017h SYS
# 3295h SYS
#619BCh SYS
# 3E2Dh SYS
# 4035h SYS
#11679h SYS
19 →PGM

'L790.8'
3 →PGM

#38FD2h SYS
#39523h SYS
7 →PGM
'L790.7' STO
```

8
L790.8

Sous-programme

Ce programme est à enregistrer sous le nom : L790.8

Son checksum est : #94Eh (226 octets)

Listing :

```
# 4099h SYS
#63EEDh SYS
#39682h SYS
# 40A3h SYS
#63EEDh SYS
"GRAD"
"DEG"
# 4003h SYS
5 →PGM
# 3FEFh SYS
# 40B7h SYS
6 →PGM

#39632h SYS

# 408Fh SYS
#63EEDh SYS
# 4085h SYS
#63EEDh SYS
"R" 149 CHR + 216 CHR +
151 CHR 149 CHR + "Z" +
4 →PGM
# 4085h SYS
#63EEDh SYS
"P/Q"
"XYZ"
4 →PGM
```

```
# 40CBh SYS
# 40B7h SYS
# 4161h SYS
7 →PGM

#39632h SYS

#41A8Dh SYS
#61AD8h SYS
#3FFA8h SYS
#61AD8h SYS
"USR1"
"USER"
4 →PGM
"NRL "
#64C3Eh SYS
#64C2Ah SYS
#64CA2h SYS
7 →PGM

#39632h SYS
#3981Bh SYS
#39632h SYS
#39853h SYS
#39632h SYS
"       "        (7 espaces)
# 4161h SYS
# 4161h SYS
#1CCB9h SYS
#39632h SYS
15 →PGM
'L790.8' STO
```

9
L790.9

Sous-programme

Ce programme est à enregistrer sous le nom : L790.9

Son checksum est : #D362h (38,5 octets)

Listing :

```
#42402h SYS
# 3AF2h SYS
#38FB9h SYS
#629BCh SYS
'L790.10'
#394A5h SYS
6 →PGM
'L790.9' STO
```

10
L790.10

Sous-programme

Ce programme est à enregistrer sous le nom : L790.10

Son checksum est : #231Ah (62,5 octets)

Listing :

```
#1314Dh SYS
#38FB9h SYS
#53A90h SYS
#61993h SYS
#39BF3h SYS
#3958Bh SYS
#39FD2h SYS
# 4E5Eh SYS
'L790.12'
# 4EB8h SYS
```

```
'L790.11'
#38FEBh SYS
12 →PGM
'L790.10' STO
```

11
L790.11

Sous-programme

Ce programme est à enregistrer sous le nom : L790.11

Son checksum est : #E1Dh (57,5 octets)

Listing :

```
#18308h SYS
# 4CE6h SYS
#61896h SYS
'L790.10'
#64E82h SYS
# 4D87h SYS
#38908h SYS
#39FC1h SYS
'L790.12'
#39FD2h SYS
10 →PGM
'L790.11' STO
```

12
L790.12

Sous-programme

Ce programme est à enregistrer sous le nom : L790.12

Son checksum est : #C6BDh (67 octets)

Listing :

```
#39F6Fh SYS
# 3FF9h SYS
#53778h SYS
#61AD8h SYS
#6256Ah SYS
# 3DEFh SYS
#63CBDh SYS
#62E67h SYS
# 73C3h SYS
#5182Fh SYS
#62CCDh SYS
#6256Ah SYS
#44197h SYS
'L790.13'
#51843h SYS
# 73A5h SYS
# 3244h SYS
17 →PGM

'L790.12' STO
```

13
L790.13

Sous-programme

Ce programme est à enregistrer sous le nom : L790.13

Son checksum est : #6A68h (112 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**.
Le résultat doit être : # 1A92h (206 quartets). Exécutez ensuite **ASS**.

```
"D9D 208 813 0DA 916 84E 207 0C4 739 303
E21 353 EF1 166 B22 639 916 84E 207 0C4
739 303 E21 353 65F 93D A91 684 E20 70C
473 930 3E2 135 32C 230 F0E 93D A91 684
E20 70C 473 930 3E2 135 30B F93 399 168
4E2 070 C47 393 03E 213 536 272 684 E20
70C 473 930 3E2 134 3B2 130"
```

14
L790.14

Sous-programme

Ce programme est à enregistrer sous le nom : L790.14

Son checksum est : #D14Dh (122 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**.
Le résultat doit être : # 59DDh (226 quartets). Exécutez ensuite **ASS**.

```
"D9D 201 3D2 69E 550 45C 363 C37 008 316
84E 207 0C4 739 303 E22 323 122 709 8E3
69B E93 32F 934 C01 6D1 236 CB9 161 EE9
3D4 436 852 309 BC2 6A6 526 E93 303 C37
0F2 815 3F2 162 C23 094 270 0ED 301 227
0CB D30 9FF 308 773 58D A16 FED 30A 652
692 421 433 707 A72 671 040 91D 30C B91
6A7 593 B21 30"
```

15
L790.15

Sous-programme

Ce programme est à enregistrer sous le nom : L790.15

Son checksum est : #21C5h (29,5 octets)

Listing :

```
'L790.16'
# 3FF9h SYS
2 →PGM
'L790.15' STO
```

16
L790.16

Sous-programme

Ce programme est à enregistrer sous le nom : L790.16

Son checksum est : #6546h (47 octets)

Listing :

```
#61AD8h SYS
'L790.17'
#167D8h SYS
# 3223h SYS
# 3FF9h SYS
#53778h SYS
# 712Ah SYS
# 3E2Dh SYS
```

```
#12429h SYS
9 →PGM
'L790.16' STO
```

17
L790.17

Sous-programme

Ce programme est à enregistrer sous le nom : L790.17

Son checksum est : #32A6h (42 octets)

Listing :

```
#39FB0h SYS
#619ADh SYS
'L790.18'
# 3223h SYS
#167D8h SYS
# 3223h SYS
#15CCFh SYS
7 →PGM
'L790.17' STO
```

18
L790.18

Sous-programme

Ce programme est à enregistrer sous le nom : L790.18

Son checksum est : #D6EEh (32 octets)

Listing :

```
#16969h SYS
# 6FD1h SYS
'L790.19'
3 →PGM
'L790.18' STO
```

19
L790.19

Sous-programme

Ce programme est à enregistrer sous le nom : L790.19

Son checksum est : #A659h (61,5 octets)

Listing :

```
'L790.4'
#1795Ah SYS
#53914h SYS
# 4E5Eh SYS
'L790.20'
# 4EB8h SYS
#159AFh SYS
#159B4h SYS
6 →PGM
# 3223h SYS
#1686Ah SYS
5 →PGM
'L790.19' STO
```

20
L790.20

Sous-programme

Ce programme est à enregistrer sous le nom : L790.20

Son checksum est : #E3BEh (184 octets)

Listing :

```
# 3188h SYS
"CCD203100034559208DB5026" ASS
#61993h SYS
'L790.21'
#62020h SYS
#61993h SYS
"Caractère "
# 3223h SYS
# 52EEh SYS
3 →PGM
#1CB90h SYS
27
# 3B97h SYS
#61993h SYS
"CCD20B1000143174E78FB97608DC7530" ASS
# 59CCh SYS
#54061h SYS
# 400Dh SYS
#6326Dh SYS
#65686h SYS
# 525Bh SYS
"FB92042" ASS
# 525Bh SYS
9 →PGM
#6217Eh SYS
# 4085h SYS
```

```
#53784h SYS
# 3B46h SYS
# 408Fh SYS
#53784h SYS
# 3AF2h SYS
# 3B46h SYS
#61993h SYS
# 5D2Ch SYS
#162ACh SYS
# 3223h SYS
#162ACh SYS
#6545Dh SYS
# 52EEh SYS
# 3223h SYS
# 5193h SYS
8 →PGM
#159EBh SYS
23 →PGM
'L790.20' STO
```

21
L790.21

Sous-programme

Ce programme est à enregistrer sous le nom : L790.21
Son checksum est : #813Ah (34,5 octets)

Listing :

```
# 2A5B0h SYS
# 159EBh SYS
"LR "
# 3223h SYS
# 5193h SYS
5 →PGM
'L790.21' STO
```

22
L790.22

Sous-programme

Ce programme est à enregistrer sous le nom : L790.22

Son checksum est : #50C8h (52 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**.
Le résultat doit être : # CCAEh (86 quartets). Exécutez ensuite **ASS**.

```
"D9D 203 223 0E5 E40 419 358 4E2 070 C47
393 03E 223 338 BE4  0D9 D20 229 351 DE4
0B2 130 229 353 223 0B2 130"
```

23
L790.23

Sous-programme

Ce programme est à enregistrer sous le nom : L790.23
Son checksum est : #55A2h (64,5 octets)

Listing :

Ne saisissez si espace, ni saut de ligne. Executez ensuite **DUP BYTES**.
Le résultat doit être : # 14BFh (111 quartets). Exécutez ensuite **ASS**.

```
"D9D 20C 493 596 961 A59 71E 342 42C 230
381 263 991 684 E20 70C 473 930 3E2 234
32C 230 891 26C B91 6A4 A61 E5E 40A 9A6
18B E40 85B 61B 213 0"
```

24
L790.24

Sous-programme

Ce programme est à enregistrer sous le nom : L790.24

Son checksum est : #575Dh (85 octets)

Listing :

```
# 4085h SYS
#53784h SYS
# 408Fh SYS
#53784h SYS
# 3AF2h SYS
# 3B46h SYS
#619ADh SYS
#17518h SYS
# 54AFh SYS
#63E89h SYS
# 3FEFh SYS
""
2 →PGM
#1613Fh SYS
# 3AC0h SYS
# 3AC0h SYS
#610C4h SYS
""
"/"
#619CBh SYS
#1756Dh SYS
6 →PGM
14 →PGM

'L790.24' STO
```

Maintenant, Tapez :

```
{ L790.4 L790.5 L790.6 L790.7 L790.8
L790.9 L790.10 L790.11 L790.12 L790.13
L790.14 L790.15 L790.16 L790.17 L790.18
L790.19 L790.20 L790.21 L790.22 L790.23
L790.24  $CONFIG }
'$HIDDEN' STO

  790
'$ROMID' STO

  "NEW STACK"
'$TITLE' STO

  « 790 ATTACH »
'$CONFIG' STO

  UPDIR
  'STACK' →LIB
```

Au bout de quelques minutes, vous obtiendrez votre bibliothèque.

Ce que NEWSTACK permet :

- Quand la bibliothèque est créée, exécuter STACK 0 STO puis ON-C,
- Exécuter NEW pour lancer le programme,
- ST5 pour passer en mode 5 lignes,
- ST7 pour passer en mode 7 lignes.

L'affichage est modifié. :

- Les Externals apparaissent sous la forme de leur adresse s'ils n'ont pas de contenu.

  `# 3223h SYS` *renvoie* `$ 3223h`
  au lieu de `External`

- Les longs réels apparaissent:

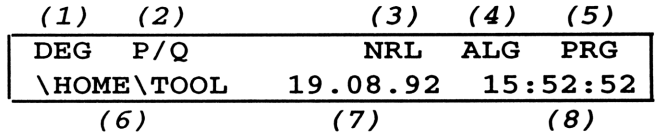
  `#2A458h SYSEVAL` *renvoie* `LR 3.14159265359`
  au lieu de `Long Real`

• Les caractères apparaissent:

`# 654CDh SYSEVAL` *renvoie au lieu de*	`Caractère A` `Character`

• Le mode rationnel (voir les premiers programme du livre) est affichable. Si l'on saisit `-15 SF -16 CF` (cf KEYQ), `(5,3)` sera affiché `5/3.`

L'affichage du haut de l'écran est modifié.
En mode 5 lignes :



Zone 1 (1) : *Affichage du mode angulaire (unité de mesure d'angles).*

DEG : mode degrés
RAD : mode radians
GRAD: mode grades

Zone 2 (2) : *Affichage des couples.*

XYZ : coordonnées cartésiennes
rθZ : coordonnées cylindriques
rθØ : coordonnées sphériques
P/Q : affichage des rationnels

Zone 3 (3) : *Affichage du mode d'utilisation du clavier.*
NRL : clavier normal
USR1 : clavier redéfini pour 1 touche
USER : clavier redéfini

Zone 4 (4) : *Mode de saisie algébrique.*
Zone 5 (5) : *Mode de saisie des programmes.*
Zone 6 (6) : *Répertoire courant*

Zone 7 *(7)* : *Date courante.*
Zone 8 *(8)* : *Heure courante.*

Note : Ce programme laisse la possibilité à l'utilisateur de la HP48 d'affichier les **Internals** sous une forme encore plus pratique.
Il faut pour cela créer deux fichiers :

```
Linternals
Ltranscrit
```

Ils devront être sous le format suivant :
Par exemple :

```
Linternals
[ #3223h #3188h }

Ltranscrit
{ SWAP DUP }
```

car l'adresse 3223h correspond à SWAP et l'adresse 3188h correspond à DUP. Ces adresses sont cataloguées dans le chapitre 17. Ces deux listes de même taille peuvent être aussi grande que la mémoire le permet.

Ainsi, si vous effectuez :

```
#3223h
SYS
```

Vou verrez sur l'écran :

```
Internal : SWAP
```

au lieu de

```
External
```

Remarque :
Cette liste d'adresses peut être aussi grande que vous le désirez. Seule la mémoire de votre calculateur la limite. Si vous êtes amenés à programmer assez souvent en adresses Internals, vous comprendrez assez vite l'utilité de ces deux fichiers !

5
Compression-décompression

Nous vous proposons un ensemble de quatre programmes. Trois servent à compresser des données selon des techniques différentes. Le dernier programme restitue l'orginal de la compression, quelque soit la technique utilisée.

Les programmes seront présentés dans l'ordre suivant :

- **LZ10** Ce compacteur est basé sur la méthode de Lempel-Ziv. Il fonctionne sur des quartets. Il est très efficace sur les programmes RPL normaux.

- **LZ8B** Ce compacteur fonctionne encore à partir de la méthode de Lempel-Ziv. Il travaille sur des octets avec un buffer de 256 octets. Il donne surtout de bons résultats avec des chaînes de caractères. (formules, aides-mémoires ou encore sources assembleur)

- **HUFF** Ce compacteur utilise la méthode de Huffmann dynamique sur des quartets. Il donne en général de moins bons résultats que les autres, mais il permet souvent de gagner un peu de place en recompactant quelquechose de déjà compacté.

- **UNPK** Ce programme décompacte toutes les archives précédentes.

Remarque : lors du compactage, ces programmes, à l'exception de UNPK, utilisent un GROB écran pour économiser la mémoire. Ceci provoquera une perturbation de l'affichage de l'écran pendant l'éxécution des programmes.

1
LZ10

Programme de compactage LZ10

A enregistrer sous le nom : LZ10
Son checksum est : #B383h (553 octets)

Listing :

Saisissez la chaine suivante sans espaces ni sauts de ligne. Exécutez alors **DUP BYTES** pour vérifier que le checksum est bien : #D176h (1094 octets). Si c'est le cas, éxécutez la commande **ASS**. Sinon, vérifiez votre saisie.

```
"D9D 205 AA8 188 130 209 505 362 186 C36
CCD 20B F30 08F 146 60D 606 340 500 0C2
8FA E26 118 934 88B 201 441 690 715 C31
63C E10 A13 210 411 806 119 061 1A0 611
4CA 102 147 108 174 143 818 F09 818 F09
103 340 040 0C2 109 131 341 400 0AF 015
171 7FC E56 FD1 840 841 118 135 D20 6AC
114 311 913 416 F16 414 011 B13 484 214
314 616 08A 67F 136 134 111 8BA 568 72D
085 2DA 07D 606 AC9 80D F17 416 315 371
567 1C4 183 916 F10 C40 290 28F 132 130
07D 606 80C FAC 520 649 F07 136 06A C1A
4D4 D38 726 060 709 4DF 281 B43 4D6 000
C21 341 43A C21 468 A68 007 60B 016 4B4
65B E76 120 711 3E2 CEC 6E6 C6E 615 431
62A C91 544 160 D28 128 18F 231 321 046
FF0 071 437 050 D9D 20B 213 033 920 779
201 192 0C2 A20 69A 208 4E2 0D6 E20 29E
204 7A2 0E4 A20 CCD 200 4B2 0E1 B20 8BA
200 713 4AC 214 68A 682 727 130 D15 C01
601 544 160 132 104 344 000 060 701 64B
465 BC8 717 0D3 851 E73 482 000 8A7 C41
```

```
10E AE4 131 114 130 319 F14 C16 115 371
507 16F 17F 153 715 071 6F1 7F1 5B7 158
716 713 210 484 1D2 DA0 7E4 E25 508 500
612 211 C8B 650 850 11A 102 113 C01 30C
111 013 1CE 15B 017 015 801 60C E5F E13
710 834 FF3 00E 954 311 113 111 313 0FA
CE1 5B0 158 016 017 0CE 5FE 34F F30 00E
F16 320 11B 135 119 134 153 715 071 6F1
7F1 431 408 706 064 3D1 101 317 320 136
10C 070 710 907 108 114 8F3 866 18D F66
301 141 308 610 034 800 00E B58 1FA CEC
6C6 C6E 614 C16 160 10D BCE C61 5C0 160
137 EB1 35C F15 B01 580 160 170 CF5 FE8
410 1B9 F06 881 302 095 059 230 4EC 30C
5F2 6B2 130"
```

Remarque : LZ10 fabrique des objets qui ont pour type "Library Data". Cet objet ne sera fabriqué que si le méthode employée permet de diminuer la taille du fichier.

Mode d'utilisation :
Mettez l'objet que vous voulez compacter sur la pile, puis lancez le programme LZ10.

2
LZ8B

Programme de compactage LZ8B

A enregistrer sous le nom : LZ8B

Son checksum est : #95FCh (474,5 octets)

L'objet créé par LZ8B a pour fomat : Entier Binaire

Listing :

Saisissez la chaine suivante sans espaces ni sauts de ligne. Executez alors **DUP BYTES** pour vérifier que le checksum est bien : #FDBEh (937 octets). Si c'est le cas, exécutez la commande **ASS**. Sinon, vérifiez votre saisie.

```
"D9D 205 AA8 188 130 209 505 362 186 C36
CCD 20E 530 08F 146 60D 606 C68 FAE 261
189 34E 4A2 014 416 907 144 164 CE8 19F
2CE 10A 132 104 118 061 190 614 713 4E6
E61 081 741 438 18F 098 18F 091 033 400
200 C21 091 313 412 000 AF0 151 717 FCE
56F AF1 E5E 511 B13 514 A14 985 111 813
5D2 060 614 B11 913 416 F16 F14 811 B13
414 B14 E16 196 67F 136 134 111 8BA 540
613 713 506 171 D3E 714 B17 114 E16 196
2FE 071 350 713 4DA 078 B7A 007 D60 6DB
066 4AF 07C E56 063 A0D 7CE 47F 34F 000
08B 760 CED 711 413 086 1F5 AC9 D28 128
0D0 110 EAE A13 189 0D0 D0E 415 801 601
5C0 160 1C1 153 115 011 378 091 351 368
091 341 531 150 180 913 420 841 DBE 615
C01 600 711 3E2 81E CE1 4C1 616 150 071
1C1 348 71C 0AC 185 168 30B 455 133 11F
14C 161 118 135 1CF 1CF 153 715 071 7F1
6F1 537 150 716 FD3 136 10C DBE 611 284
0EA 550 850 102 113 C01 30C 1C1 118 135
14B 171 148 161 CF5 1F1 371 08D 981 9F2
CE9 2A2 480 D01 111 311 1B1 341 531 150
180 913 413 780 913 515 311 501 203 4FF
100 0EF 165 201 1B1 351 191 341 537 150
716 F17 F15 371 507 870 606 ACD 861 D51
141 301 10A C9D 281 2EA EA1 311 C18 0D0
D0E 415 801 601 5C0 160 153 115 011 378
091 351 368 091 341 531 150 180 910 C20
071 090 710 811 48F 386 618 DF6 630 B9F
068 813 020 950 592 304 EC3 0C5 F26 B21
30"
```

3
HUFF

Programme de compactage HUFF

A enregistrer sous le nom : HUFF
Son checksum est : #C63h (450,5 octets)

Listing :

Saisissez la chaine suivante sans espaces ni sauts de ligne. Executez alors **DUP BYTES** pour vérifier que le checksum est bien : #8A3Bh (906 octets). Si c'est le cas, exécutez la commande **ASS**. Sinon, vérifiez votre saisie.

```
"D9D 205 AA8 188 130 209 505 362 186 C36
CCD 20F 330 08F 146 60D 606 818 F2F 818
F2F 8FA E26 118 934 A0A 201 441 690 714
4CE 10A 118 061 190 616 413 210 111 AC2
061 431 041 741 438 18F 098 18F 091 031
303 1F1 D7D 111 3D2 E61 441 64D 980 880
F2C 214 416 4D9 F2C 6C6 C21 441 648 18F
2F8 18F 2F1 441 64D 214 416 BE5 A6F 55B
136 81A F08 34A E10 0CA 130 2F1 461 351
47C 618 914 418 B18 90D 56E 118 134 818
F09 818 F0B AF3 350 020 001 401 361 311
79A FF1 5D9 AFF 179 145 136 164 CA0 D59
D20 113 345 200 0CA 130 D21 44A C3A C11
1C1 34E 610 CD2 15E 0DA C6C 6C2 110 CA1
311 470 613 4D2 ACB 812 D5D 3E5 C71 321
301 641 461 341 691 468 A24 0E7 184 146
184 8AE 4DA 4F4 A0C 7A4 F5A FD2 AC9 812
CB1 111 308 18F 934 111 5C0 E41 60F 66B
EF8 16A C58 18F 13D 981 6AC 710 107 135
143 E41 411 371 351 341 8F1 8F1 468 B66
06F 901 8F1 8F1 468 B64 F16 F16 F14 0CC
```

```
141 169 179 15A E15 FE1 5CE 159 E18 913
616 414 413 416 E14 213 016 414 413 406
143 1C9 133 174 141 131 17E 147 135 174
141 131 179 179 147 135 141 169 169 142
130 071 441 351 741 431 311 741 471 318
AA6 069 2F1 43E 414 107 112 CC4 311 021
118 B68 006 615 E07 129 DA0 710 894 BE0
130 AC9 154 4E4 8F3 866 18D F66 30B 9F0
688 130 209 505 923 04E C30 C5F 26B 213
0"
```

L'objet fabriqué par HUFF a pour format : Linked Array (matrice liée)

4
UNPK

Programme de décompactage UNPK

A enregistrer sous le nom : UNPK

Son checksum est : #454Bh (1040,5 octets)

Listing :

Saisissez la chaine suivante sans espaces ni sauts de ligne. Executez alors **DUP BYTES** pour vérifier que le checksum est bien : #65B8h (2069 octets). Si c'est le cas, exécutez la commande **ASS**. Sinon, vérifiez votre saisie.

```
"D9D 20E CE8 1D0 040 29E 208 E14 00D 504
0D9 D20 FD5 501 192 085 200 C1C 163 223
0CC D20 462 001 471 361 691 421 641 360
6D6 063 402 000 C28 FAE 261 189 132 102
07C ACE 10B 071 0CD 606 118 061 741 438
18F 091 01D 134 002 00C 210 813 134 120
00A F01 517 17F CE5 6F1 1C1 351 5F0 170
```

```
80D 089 160 6F7 015 F01 708 0D0 112 130
153 115 011 370 680 913 5D7 136 809 134
153 115 018 091 0A0 713 511 9C9 134 153
115 018 091 34D B13 515 311 501 809 10C
136 809 111 E2D 56B A08 804 314 B17 113
710 C11 A13 414 8E6 E61 0A8 1AF 19C 913
414 8E5 E56 470 0DD 014 B17 113 710 C11
9C2 C21 351 1A1 341 531 150 180 913 413
780 913 515 711 541 A97 136 809 10A 119
C91 341 501 809 134 A9B 154 113 680 911
1E2 D5D 280 980 911 3EA 497 103 D98 19F
2CE 92A 538 0D0 118 135 119 134 153 115
018 091 341 378 091 351 531 150 165 201
191 351 181 341 537 150 717 F16 F15 371
507 203 4FF 100 0EF 16E 3E2 007 108 07D
A13 08F 196 618 DF6 630 B9F 06B 213 096
DC1 D9D 20F D55 004 F46 C1C 163 223 0CC
D20 292 001 471 361 69D 015 A31 631 360
6D6 063 405 000 C28 FAE 261 189 132 102
07C ACE 10B 071 0CD 606 118 061 741 438
18F 091 01D 134 004 00C 210 813 134 140
00A F01 517 17F CE5 6F1 1C1 351 1A1 34D
215 F01 708 08B 032 819 F28 0D0 153 115
01E 613 3CA 104 206 821 808 A15 5D0 1C0
153 317 215 741 37E 610 C81 9F0 819 F08
1AF 19C 213 5D2 812 818 F23 D7E 615 B01
580 160 170 CF5 FE6 EC0 808 B28 41C 0D2
14F 171 819 F28 19F 281 9F2 818 F27 D7E
613 313 1CA 104 15B 015 801 601 70C F5F
E61 80D 015 B0D 6C4 C4C A13 7E6 10C 705
0D9 D20 B21 303 392 077 920 119 20C 2A2
069 A20 84E 20D 6E2 029 E20 47A 20E 4A2
0CC D20 04B 20E 1B2 08B A20 07C 213 514
314 034 500 00D 711 1C0 131 112 130 C1C
E15 A01 590 160 170 CE5 FE1 321 021 181
341 191 35D 934 FF3 00E 958 2FA CE1 5A0
159 016 017 0CE 5FE 34F F30 00E F16 710
153 715 071 6F1 7F1 431 401 1BE B49 010
B61 1E0 710 807 DA1 308 F19 661 8DF 663
0B9 F06 B21 308 DCC 1D9 D20 FD5 501 192
049 200 C1C 163 223 0CC D20 A62 001 431
```

```
321 691 461 300 681 8F2 F8F AE2 611 891
321 010 7CA CE1 0AD 606 143 818 F0E 104
174 143 818 F09 103 130 2FD 116 4D9 819
F2D 7C6 C6C BC6 C6C 214 416 4D9 C6D 7C6
C6C BC6 C6C 214 416 481 8F2 981 8F2 914
416 4E5 0D5 3BD 2E6 144 164 D98 19F 2D7
C6C 6CB C6C 6C2 144 164 D21 441 64D 914
416 4E5 0D5 6C2 034 631 00C A13 02F 146
135 147 C61 891 441 890 D59 E20 113 130
16F 168 D21 44A C31 131 311 7E1 7EA 4F5
711 1C1 348 18F 231 0C1 5E3 D5D 981 D80
8A0 501 741 431 311 791 438 AC7 C17 415
B01 191 34E 610 915 801 CE1 43E 414 113
713 513 418 918 914 68B 660 6E7 018 918
914 68B 64F 169 169 140 CC1 411 691 791
5A9 15F 915 C91 599 189 136 164 144 134
16E 142 130 164 144 134 D71 431 C91 331
741 411 311 7E1 471 351 741 41D B13 517
414 313 117 414 713 18A A60 6A4 F14 3E4
141 112 CC4 901 026 3DE 07D A13 08F 196
618 DF6 630 B9F 06B 213 0B2 130"
```

Remarque : UNPK décompacte à la fois les fichiers compactés par LZ10, LZ8B et HUFF.

Annexe

Programmes et sous-programmes

Certains programmes en utilisent d'autres comme sous-programmes.
Ils sont ici classés répertoire par répertoire.
Les noms de sous-programmes sont suivis du nom placé entre parenthèses du répertoire dans lequel il convient de les placer.
Les sous-programmes d'un même répertoire sont présentés les uns à la suite des autres.

Répertoire HOME

LOOK
Ce programme ne nécessite aucun autre programme.

FIND
Ce programme ne nécessite aucun autre programme.

PGM→
Ce programme ne nécessite aucun autre programme.

→PGM
Ce programme ne nécessite aucun autre programme.

ASS
Ce programme ne nécessite aucun autre programme.

SYS
Ce programme ne nécessite aucun autre programme.

DASS
Utilise le programme :
ASS (HOME)

ERREUR
Ce programme ne nécessite aucun autre programme

Répertoire ARITHM

Sous répertoire de HOME

EDIVI
Utilise le programme :
ASS (HOME)

DECOMP
Utilise les programmes :
EDIVI (ARITHM)
ASS (HOME)

PGCD
Utilise le programme :
ASS (HOME)

PPCM
Utilise les programmes :
PGCD (ARITHM)
ASS (HOME)

Q→R
Utilise les programmes :
PGCD (ARITHM)
ASS (HOME)

R→Q
Utilise les programmes :
PGCD (ARITHM)
ASS (HOME)

QPLUS
Utilise les programmes :
QSIMPL R→Q Q→R PGCD (ARITHM)
ASS (HOME)

QMOINS
Utilise les programmes :
QSIMPL R→Q Q→R PGCD (ARITHM)
ASS (HOME)

QMULT
Utilise les programmes :
Q→R R→Q PGCD (ARITHM)
ASS (HOME)

QDIV
Utilise les programmes :
Q→R R→Q PGCD (ARITHM)
ASS (HOME)

QSIMPL
Utilise les programmes :
Q→R PGCD (ARITHM)
ASS (HOME)

QPUIS
Utilise les programmes :
Q→R R→Q PGCD (ARITHM)
ASS (HOME)

QNEG
Utilise le programme :
ASS (HOME)

QTEST
Utilise les programmes :
QSIMPL Q→R PGCD (ARITHM)
ASS (HOME)

QSQ
Utilise les programmes :
Q→R PGCD (ARITHM) et ASS (HOME)

QINV
Utilise les programmes :
Q→R PGCD (ARITHM)
ASS (HOME)

KEYQ
Utilise les programmes :
QPLUS QMOINS QMULT QDIV QINV QTEST QSQ
R→Q Q→R PGCD (ARITHM)
ASS (HOME)

Répertoire POLY

Sous répertoire de ARITHM

PSIMPL
Utilise le programme
ERREUR (HOME)

VAL
Utilise le programme :
PSIMPL (POLY)
ERREUR (HOME)

EVALP
Utilise les programmes :
QPUIS QMULT QPLUS QSIMPL Q→R R→Q PGCD

(ARITHM)
ASS ERREUR (HOME)

OPER
Utilise le programme :
ERREUR (HOME)

PPLUS
Utilise les programmes :
OPER (POLY)
QPLUS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

PMOINS
Utilise les programmes :
OPER (POLY)
QMOINS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

PCONST
Utilise les programmes :
OPER (POLY)
QMULT Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PMULT
Utilise les programmes :
PCONST PPLUS OPER (POLY)
QPLUS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

PPUIS
Utilise les programmes :
PMULT PCONST PPLUS OPER (POLY)
QPLUS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

PDIV
Utilise les programmes :
PSIMPL PPUIS PCONST PPLUS PMULT PMOINS OPER (POLY)
QDIV QMULT QPLUS QSIMPL QMOINS Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PDIVC
Utilise les programmes :
PCONST OPER (POLY)
QINV QDIV QNEG QMULT QPLUS QSIMPL Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PPGCD
Utilise les programmes :
PSIMPL PDIV PPUIS PCONST PPLUS PMULT PMOINS OPER (POLY)
QDIV QMULT QPLUS QSIMPL QMOINS Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PPPCM
Utilise les programmes :
PPGCD PDIV PMULT PSIMPL PPUIS PCONST PPLUS PMOINS OPER (POLY)
QDIV QMULT QPLUS QSIMPL Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PCOMP
Utilise les programmes :
PMULT PCONST PPLUS OPER (POLY)
QPLUS QSIMPL QMULT R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

PTRANS
Utilise les programmes :
QMULT QPLUS QSIMPL
Q→R R→Q
PGCD (ARITHM)
ASS ERREUR (HOME)

PDER
Utilise les programmes :
QMULT Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PINT
Utilise les programmes :
PSIMPL (POLY)
QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

P→L
Utilise les programmes :
Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

Répertoire RACINES

Sous répertoire de POLY

DIVIS
Utilise le programme :
ERREUR (HOME)

RACP
Utilise les programmes :
DIVIS QDIV

QMULT QPLUS
QMOINS (RACINES)
PSIMPL PDIV
PPUIS PCONST PPLUS
PMULT PMOINS PDER
OPER EVALP (POLY)
QPUIS QSIMPL
Q→R
R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

QPLUS
Ce programme ne nécessite aucun autre programme.

QMOINS
Ce programme ne nécessite aucun autre programme.

QMULT
Ce programme ne nécessite aucun autre programme.

QDIV
Ce programme ne nécessite aucun autre programme.
QDIV
Ce programme ne nécessite aucun autre programme.

QNEG
Ce programme ne nécessite aucun autre programme.

EPS
Ce programme ne nécessite aucun autre programme.

Répertoire ALGO

Sous répertoire de ARITHM

BEZOUT
Ce programme ne nécessite aucun autre programme.

NEWTON
Ce programme ne nécessite aucun autre programme.

TRI
Ce programme ne nécessite aucun autre programme.

Répertoire DL

Sous répertoire de ARITHM

TAYLOR
Utilise les programmes :
Tous les programmes du répertoire *TAYLOR*

DPLUS
Utilise les programmes :
QPLUS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

DMOINS
Utilise les programmes :
QPLUS QNEG QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

DMULT
Utilise les programmes :
QMULT QPLUS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

DDIV
Utilise les programmes :

PDIVC PCONST OPER (POLY)
QINV QDIV
QNEG QMULT
QPLUS QSIMPL
Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

VAL
Utilise le programme :

ERREUR (HOME)

D1PLUS
Utilise les programmes :

QMOINS QMULT QDIV QSIMPL Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DSIMPL
Utilise le programme

ERREUR (HOME)

DCOMP
Utilise les programmes :

DSIMPL DMULT DPLUS (DL)
QMULT QPLUS QSIMPL R→Q Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

DPUIS
Utilise les programmes :

D1PLUS DMULT
DCOMP DDIV
DSIMPL DPLUS (DL)
PCONST OPER (POLY)

QINV QDIV
QNEG QPUIS
QMOINS QMULT QPLUS QSIMPL Q→R R→Q
PGCD (ARITHM)
ASS ERREUR (HOME)

DSIN
Utilise les programmes :
QMULT QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DCOS
Utilise les programmes :
QMULT QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DSINH
Utilise les programmes :
QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DCOSH
Utilise les programmes :
QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DEXP
Utilise les programmes :
QINV Q→R PGCD (ARITHM)
ASS ERREUR (HOME)

DLN
Utilise les programmes :
QMULT QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DASIN
Utilise les programmes :
QMULT QDIV Q→R R→Q PGCD (ARITHM)

ASS ERREUR (HOME)

DASINH
Utilise les programmes :
QMULT QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DATAN
Utilise les programmes :
QNEG QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DATANH
Utilise les programmes :
QDIV Q→R
R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DTAN
Utilise les programmes :
QMULT QPLUS
QSIMPL
Q→R R→Q
PGCD (ARITHM)
ASS ERREUR (HOME)

DTANH
Utilise les programmes :
QMULT QPLUS QSIMPL Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

DNEG
Utilise les programmes :
QNEG (ARITHM)
ASS ERREUR (HOME)

Répertoire MATRICES

Sous répertoire de ARITHM

TR
Utilise les programmes :
QPLUS QSIMPL Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

CM→M
Utilise les programmes :
MSIMPL (MATRICES)
PGCD (ARITHM)
ASS ERREUR (HOME)

MSIMPL
Utilise les programmes :
PGCD (ARITHM)
ASS ERREUR (HOME)

MMULT
Utilise les programmes :
MSIMPL CM→M (MATRICES)
PGCD (ARITHM)
ASS ERREUR (HOME)

MPLUS
Utilise les programmes :
MSIMPL CM→M (MATRICES)
PGCD (ARITHM)
ASS ERREUR (HOME)

MMOINS
Utilise les programmes :
MSIMPL CM→M (MATRICES)
PGCD (ARITHM)
ASS ERREUR (HOME)

LEVERRIER
Utilise les programmes :
TR MSIMPL CM→M (MATRICES)
QPLUS QSIMPL
QDIV Q→R R→Q
PGCD (ARITHM)
ASS ERREUR (HOME)

MINV
Utilise les programmes :
LEVERRIER
MSIMPL
CM→M (MATRICES)
QINV QNEG QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

MCOMAT
Utilise les programmes :
MSIMPL CM→M (MATRICES)
QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

MPOLC
Utilise les programmes :
MSIMPL CM→M (MATRICES)
QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

PENT
Utilise les programmes :
PPCM Q→R QMULT Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

MDET
Utilise les programmes :
MSIMPL CM→M (MATRICES)
QMULT QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

MRESOL
Utilise les programmes :
LEVERRIER MSIMPL CM→M (MATRICES)
QINV QNEG QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

MCONST
Utilise les programmes :
MSIMPL CM→M (MATRICES)
PGCD (ARITHM)
ASS ERREUR (HOME)

MPUIS
Utilise les programmes :
LEVERRIER MSIMPL CM→M (MATRICES)
QINV QNEG QDIV Q→R R→Q PGCD (ARITHM)
ASS ERREUR (HOME)

Répertoire APPLIM

Sous répertoire de MATRICES

JORDAN
Utilise les programmes suivants :
MSIMPL MINV MMULT CM→M
LEVERRIER (MATRICES)
SIMP SYSMIN BKER FLB? L→ARR →MATR
GETTL (APPLIM)
QMULT PGCD QINV QNEG QDIV Q→R
R→Q (ARITHM)
ERREUR ASS (HOME)

GETL
Utilise le programme suivant :
ASS (HOME)

SIMP
Utilise le programme suivant :
PGCD (ARITHM)

FLB?
Ce programme n'utilise aucun sous-programme.

SYSMIN
Utilise les programmes suivants :
GETL SIMP FLB? (APPLIM)
PGCD (ARITHM)
ASS (HOME)

L→ARR
Ce programme n'utilise aucun sous-programme.

BKER
Ce programme utilise les programmes suivants :
SIMP FLB? (APPLIM)
PGCD (ARITHM)

→MATR
Ce programme n'utilise aucun sous-programme.

Répertoire ALG

Sous répertoire de MATRICES

A→M
Utilise le programme :
ERREUR (HOME)
M→A
Utilise le programme suivant :
MSIZE (ALG)

MSIZE
Utilise le programme :
ERREUR (HOME)

MOPER
Utilise le programme suivant :
MSIZE (ALG)
ERREUR (HOME)

MPLUS
Utilise le programme suivant :
MOPER MSIZE (ALG)
ERREUR (HOME)

MMOINS
Utilise le programme suivant :
MOPER MSIZE (ALG)
ERREUR (HOME)

MCONST
Utilise le programme suivant :
MOPER MSIZE (ALG)

MMULT
Utilise le programme suivant :
ERREUR (HOME)

MTR
Utilise le programme suivant :
ERREUR (HOME)

MTRN
Utilise le programme suivant :
MSIZE (ALG)
ERREUR (HOME)

LEVERRIER
Utilise le programme suivant :
A→M MMULT MTR MCONST MPLUS MSIZE

MCOLCT MOPER (ALG)
ERREUR (HOME)

MINV
Utilise le programme suivant :

A→M MMULT MTR MCONST MPLUS MSIZE
MCOLCT MOPER (ALG)
ERREUR (HOME)

MCOMAT
Utilise le programme suivant :

A→M MMULT MTR MCONST MPLUS MSIZE
MCOLCT MOPER MTRN (ALG)
ERREUR (HOME)

MPOLC
Utilise le programme suivant :

A→M MMULT MTR MCONST MPLUS MSIZE
MCOLCT MOPER (ALG)
ERREUR (HOME)

MDET
Utilise les programmes suivants :

MPOLC A→M MMULT
MTR MCONST MPLUS MSIZE MCOLCT MOPER
(ALG)
ERREUR (HOME)

MCOLCT
Utilise le programme suivant :

ERREUR (HOME)

MRESOL
Utilise les programmes suivants :

A→M MMULT MTR MCONST MPLUS MSIZE
MCOLCT MOPER MTRN (ALG)
ERREUR (HOME)

MAT→
Utilise le programme suivant :
ERREUR (HOME)

→MAT
Ce programme ne nécessite aucun autre programme.

Répertoire APPLIA

Sous répertoire de ALG

GAUSS
Utilise les programmes :
QINV QMULT QMOINS MSIZE (ALG)
ERREUR (HOME)

RSYST
Utilise les programmes :
GAUSS (APPLIA)
QDIV QNEG MTRN MAT→ →MAT MSIZE (ALG)
ERREUR (HOME)

ESPPE
Utilise les programmes :
RSYST (APPLIA)
A→M MCONST MMOINS MOPER MSIZE (ALG)
PGCD (ARITHM)
ERREUR (HOME)

Répertoire OUTILS

Sous répertoire de HOME

TRIGO
Ce programme ne nécessite aucun autre programme.

Répertoire GEOMETRIE

Sous répertoire de HOME

ENVELOPPES
Utilise les programmes :
tmin tmax A B C DA DB DC h (GEOMETRIE)

tmin
Ce programme ne nécessite aucun autre programme.

tmax
Ce programme ne nécessite aucun autre programme.

A
Ce programme ne nécessite aucun autre programme.

B
Ce programme ne nécessite aucun autre programme.

C
Ce programme ne nécessite aucun autre programme.

DA
Ce programme ne nécessite aucun autre programme.

DB
Ce programme ne nécessite aucun autre programme.

DC
Ce programme ne nécessite aucun autre programme.

h
Ce programme ne nécessite aucun autre programme.

PPAR
Ce programme ne nécessite aucun autre programme.
DEVELOPPEE

Utilise les programmes :
tmin tmax X Y DX DY DDX DDY h

X
Ce programme ne nécessite aucun autre programme.

Y
Ce programme ne nécessite aucun autre programme.

DX
Ce programme ne nécessite aucun autre programme.

DY
Ce programme ne nécessite aucun autre programme.

DDX
Ce programme ne nécessite aucun autre programme.

DDY
Ce programme ne nécessite aucun autre programme.

CONICS
Utilise les programmes suivants :
CENTRE TEST CST ANGLE SITU PARAM
PARAB DEGENERE DEGX DEGY DEGE2 (GEOMETRIE)

CLEAN
Ce programme n'utilise aucun sous-programme.

TEST
Ce programme n'utilise aucun sous-programme.

CENTRE
Ce programme utilise les programmes suivants :
DEGENERE DEGE2 DEGX DEGY (GEOMETRIE)

ANGLE
Ce programme n'utilise aucun sous-programme.

SITU
Ce programme n'utilise aucun sous-programme.

PARAM
Ce programme n'utilise aucun sous-programme.

PARAB
Ce programme n'utilise aucun sous-programme.

CST
Ce programme n'utilise aucun sous-programme.

DEGENERE
Ce programme n'utilise aucun sous-programme.

DEGX
Ce programme n'utilise aucun sous-programme.

DEGY
Ce programme n'utilise aucun sous-programme.

DEGE2
Ce programme utilise les programmes suivants :
DEGX DEGY (GEOMETRIE)

Répertoire PHYSIQUE

Sous-répertoire de HOME

BODE
Ce programme utilise les programmes suivants :
ETUDE PRECIS SRND ATTENDS G0 SEE (PHYSIQUE)

ETUDE
Ce programme utilise le programme suivant :
SRND (PHYSIQUE)

PRECIS
Ce programme utilise le programme suivant :
ATTENDS (PHYSIQUE)

SRND
Ce programme n'utilise aucun sous-programme.

ATTENDS
Ce programme n'utilise aucun sous-programme.

G0
Ce programme n'utilise aucun sous-programme.

SEE
Ce programme n'utilise aucun sous-programme.

GIBBS
Ce programme n'utilise aucun sous-programme.

Répertoire CHIMIE

Sous-répertoire de HOME

PH
Ce programme utilise les programmes suivants :
AF BF A1 B1 A2 A3 AM CALC (CHIMIE)

AF
Ce programme n'utilise aucun sous-programme.

BF
Ce programme n'utilise aucun sous-programme.
A1
Ce programme n'utilise aucun sous-programme.

B1
Ce programme n'utilise aucun sous-programme.

A2
Ce programme n'utilise aucun sous-programme.

A3
Ce programme n'utilise aucun sous-programme.

AM
Ce programme n'utilise aucun sous-programme.

CALC
Ce programme n'utilise aucun sous-programme.

MASS
Ce programme utilise le programme suivant :
MDATA (CHIMIE)

MDATA
Ce programme n'utilise aucun sous-programme.

Répertoire EQUADIFF

Sous répertoire de HOME

TRACER
Ce programme ne nécessite aucun autre programme.

ITER
Ce programme ne nécessite aucun autre programme.

DIFF1
Utilise les programmes :
TRACER ITER (EQUADIFF)
DIFF2
Utilise les programmes :
TRACER ITER (EQUADIFF)

Table des matières

Disquette

Mode d'emploi

Vous trouverez dans ce livre une disquette contenant les programmes décrits dans les pages précéentes.
NB : Si vous utilisez un ordinateur Macintosh (équipé du FDHD 1,44 Mo), vous devez d'abord transferer le contenu de la disquette à l'aide du programme *Apple File Exchange (AFE)* vers votre disque dur.

1
Mise en place de la *communication*

- Connectez le cable de l'extension HP entre votre calculateur et votre ordinateur.
- Lancez KERMIT sur votre ordinateur. Ce logiciel se situe sur la disquette fournie avec votre kit.

- Une fois le programme lancé, configurez le logiciel sur votre ordinateur :

Sur un compatible PC, saisissez

`SET PORT 1` remplacez 1 par le numéro du port série dans lequel votre cable est branché.

```
SET BAUD 9600
SET PARITY NONE
```

Sur un ordinateur Macintosh, sélectionnez le menu **Settings**, puis l'option **Communications**.
Veuillez à ce que la vitesse soit de 9600 bauds, la parité nulle et que le port soit bien celui dans le quel vous avez branché votre cable.

Placez vous dans le répertoire où se trouvent les fichiers du livre.

Allumez votre calculateur HP48.
Appuyez sur [ORANGE] et [PRG] puis appuyez sur [F] si vous possédez une HP48 s ou sx.
Sélectionnez le menu [IO] si vous possédez une HP48g ou gx.

Modifiez les données du menu, si nécessaire, pour obtenir la configuration suivante :

```
IR/Wire :          wire
ASCII/Binary :     BINARY
Baud :             9600
Parity :           none (0)
```

Vous êtes maintenant prêts pour débuter le transfer...

2
Choix du répertoire à transférer

La majeure partie des programme de l'ouvrage *HP48 en prépa* se situe sur cette disquette. Trouvez à partir du livre, le répertoire dans lequel le programme doit se trouver. Placez-vous y sur votre HP48.

Par exemple, si le répertoire est **RACINES**, saisissez sur votre HP48 :

```
HOME ARITHM POLY RACINES
```

A ce moment, effectuez l'opération similaire sur votre ordinateur. La correspondance des répertoires est la suivante :

Sur la HP48	Sur l'ordinateur
HOME	\
ARITHM	ARITHM
POLY	POLY
RACINES	RACINES
ALGO	ALGO
DL	DL
MATRICES	MATRICES
APPLIM	APPLIM
ALG	ALG
APPLIA	APPLIA
GEOMETRIE	GEOMETRI
EQUADIFF	EQUADIFF
PHYSIQUE	PHYSIQUE
CHIMIE	CHIMIE
Chapitre 18	APPLIC

Si vous n'arrivez pas à changer de répertoire, reportez-vous à votre manuel concernant votre système d'exploitation.

Par exemple, pour aller dans le répertoire **RACINES**, saissez sur un ordinateur PC :

```
CD \ARITHM\POLY\RACINES
```

A ce moment, vous êtes prêts pour transférer les fichiers...

3
Transfert des fichiers

Pour la majorité des fichiers, à l'exception de ceux dont le nom suit, il suffit de faire :

```
'nom_fichier' KGET
```

Pour les fichiers suivants, il faut faire :

```
{ { nom_1 nom_2 } } KGET
```

nom_1 et nom_2 sont les noms des fichiers tels qu'ils apparaissent dans l'ouvrage *HP48 en prépa*.

nom_1	nom_2
RXQ	R→Q
QXR	Q→R
PGMX	PGM→
XPGM	→PGM
CMXM	CM→M
MATX	MAT→
XMAT	→MAT
AXM	A→M
MXA	M→A
LXA	L→A
XLIB	→LIB
LEVERRIE	LEVERRIER
PXL	P→L
T	t
TMAX	tmax
TMIN	tmin
H	h
DEVELOP	DEVELOPPEES
ENVELOP	ENVELOPPES

Pour les programmes qui ont deux versions (xs etg x), remplacez nom_1 par nom_1.sx ou nom_1.gx selon la version de votre calculateur.

NB 1 : La bibliothèque créant un nouvel affichage sur la HP48 est sauvegardée sous le nom NEWSTACK dans le répertoire racine (\) de la disquette.
Tapez :

```
'NEWSTACK' KGET
NEWSTACK 0 STO
[ON-C] puis NEW
```

Une copie des répertoires complets se trouve de plus dans le répertoire racine de la disquette.

NB 2 : Les utilisateurs de Macintosh (équipés de FDHD 1,44 Mo) disposant du tableau de bord Echange Mac-PC peuvent utiliser directement le contenu de la disquette.

Le CLUB !

Vous avez des idées ?
Vous trouvez des astuces ?
Vous programmez et vous souhaitez savoir ce que d'autres ont pu découvrir... Alors sachez qu'il existe un club d'utilisateurs de calculatrices graphiques ! Les principe est très simple : vous envoyez vos découvertes et nous vous faisons parvenir la "lettre du club" qui rassemble des trouvailles de nos membres.

Une adresse à retenir :

Club Calculatrices
BP n° 203-16
75765 Paris Cedex 16

CATALOGUES ET "LIVRES MICRO"

Je désire recevoir gratuitement :

❒ les catalogues DUNOD/TECH suivants : ▫ Informatique
▫ Photo-Vidéo ▫ Electronique

❒ la revue "Livres Micro" sur les nouveautés DUNOD/TECH

❒ le(s) catalogue(s) DUNOD suivants :

- ▫ Management, Marketing
- ▫ Economie, Gestion
- ▫ Lettres
- ▫ Sciences
- ▫ Sciences humaines
- ▫ Enseignement technique

❒ le catalogue de la Librairie Technique Texas Instruments

QU'EN PENSEZ-VOUS ?

Pour nous permettre de vous proposer des livres répondant toujours davantage à vos besoins, merci de nous faire part de vos remarques et suggestions sur : ***HP-48 en prépa 2e édition*** (042012)

Ce livre vous donne-t-il toute satisfaction ? .....

Avez-vous des commentaires à formuler ? .....

.....

.....

Avez-vous déjà acquis des livres ❒ Dunod/Tech ❒ Dunod ❒ Texas ?

Si oui lesquels ? .....

.....

Qu'en pensez-vous ? .....

.....

.....

Où les avez-vous acquis ? ▫ Librairie ▫ Par correspondance
▫ Boutique micro ▫ Stage de formation

Votre centre d'intérêt ? ▫ PC (ou compatibles) ▫ Macintosh
▫ Autre.....

❒ Mr ❒ Mme ❒ Mlle..... Prénom .....

Société ..... Profession .....

Adresse .....

.....

Code Postal ⌊_⌊_⌊_⌊_⌊_⌋ Ville .....

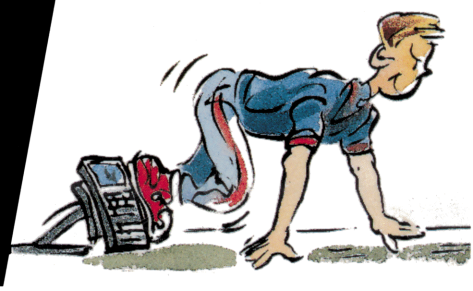
Merci de renvoyer à :

DUNOD/TECH
Courrier des lecteurs
BP 20
92122 MONTROUGE Cedex

Imprimé en France. - JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° 213629R. - Dépôt légal : Octobre 1993

CALCULATRICES

HP 48
G • GX • S • SX
en prépa



PC
Mac

Disquette 3" 1/2 PC/Mac incluse, pour charger directement les programmes du livre sur votre HP 48, à l'aide du kit de connexion disponible chez votre revendeur Hewlett-Packard.

Faites le plein de votre HP 48 !

500 pages de solutions rapides et efficaces, mais surtout **exactes**, à vos problèmes :

- calculs et affichage des fractions rationnelles **exacts**,
- calculs sur les développement limités **exacts**,
- matrices et polynômes **symboliques** et **rationnels**,
- calcul des racines de polynômes **exact** pour les racines rationnelles, approché pour les autres.

Et, bien sûr, cette nouvelle édition vous apporte une brassée de programmes toujours plus puissants ! Entre autres :

- jordanisation d'une matrice carrée de taille quelconque,
- résolution d'un système quelconque,
- calcul de l'inverse, de la comatrice, du déterminant, du polynôme caractéristique d'une matrice, que ses coefficients soient déterminés ou non,
- tracé d'enveloppes de droites ou de développées,
- décomposition de coniques,
- étude et tracé de diagrammes de Bode,
- calcul de pH...

Entraînez-vous sur les sujets de concours corrigés proposés dans ce livre pour découvrir à quel point il va vous être très vite indispensable !

En prime et pour faire bonne mesure, une initiation à l'assembleur et une foule de "trucs et astuces" et d'utilitaires bien sympathiques : programmes de base en deux versions dont une en assembleur (détaillé et expliqué) pour gagner du temps, création et décomposition de bibliothèques directement sur votre calculateur, création de matrices de chaînes de caractères pour personnaliser les messages d'erreurs, affichage permanent sur 5 ou 7 lignes avec de nombreuses améliorations, compactage et décompactage des données, etc, etc... Et s'il vous manque encore quelque chose, dites-le nous !

9 782100 020126
ISBN 2 10 002012 9 Code 042012

Environnement : ***Calculateurs Hewlett-Packard HP 48 G, GX, S et SX***

Thème : ***Programmation***

Utilisateur : ***Etudiant, Ingénieur***

DUNOD
EDITEUR