VOYAGE AU CENTRE DE LA HP48

Paul COURBIS
&
Sébastien LALANDE

VOYAGE AU CENTRE DE LA HP48

VOYAGE AU CENTRE DE LA HP48

Troisième édition

Revue et complétée

Paul COURBIS
&
Sébastien LALANDE

Première édition: Août 1991
Deuxième édition: Mars 1992
Troisième édition : Décembre 2000
Version électronique distribuée sur http://www.courbis.com
avec l'aimable autorisation des éditions Angkor

ISBN 2-87892-003-1

Paul COURBIS
24 ans, ingénieur Civil des Mines de Saint-Etienne, ancien élève de mathématiques spéciales au lycée Pasteur (Neuilly-sur-Seine).

Sébastien LALANDE
22 ans, élève-ingénieur à l'Ecole Nationale Supérieure des Télécommunications de Paris, ancien élève de mathématiques spéciales au lycée Pasteur (Neuilly-sur-Seine).

Des mêmes auteurs:

Voyage au centre de la HP28 c/s

Nous tenons à remercier:

Nos familles respectives pour l'aide et le soutien qu'elles nous ont apportés;

Marc BERNARD DE COURVILLE pour ses nombreuses critiques;

Christophe DUPONT DE DINECHIN pour son programme µSOLVER;

Dominique MOISESCU pour son programme SSAG;

Christophe NGUYEN pour ses programmes CIRCLE et BANNER;

Yann ROUSSE de Maubert Electronic;

Jean TOURRILHES pour ses précieuses corrections;

Tous les membres du groupe comp.sys.hp48;

Ainsi que toutes les personnes qui ont contribué par leurs remarques et leurs conseils à la réalisation de cet ouvrage...

Note au lecteur

Cet ouvrage s'adresse à la fois aux néophytes et aux programmeurs expérimentés: il comprend des chapitres d'initiation à l'utilisation "classique" de la HP48 ainsi que des explications permettant d'accéder à des ressources non révélées par le constructeur... Il a un double but: expliquer comment accéder à toutes ces ressources et servir de boîte à outils.

Il est divisé en quatre parties:

- Une première partie qui familiarise le lecteur avec les principes de base de la HP48: notation polonaise inversée, utilisation de la pile, langage de programmation... Des exercices corrigés sont proposés;
- Une deuxième partie qui présente de manière progressive les ressources cachées de la HP48 sous une forme claire, accessible à tout utilisateur et émaillée de nombreux exercices (corrigés en annexe); ce cours d'initiation au langage machine pourra ensuite servir de manuel de référence pour le programmeur;
- Une bibliothèque de programmes variés prêts à l'emploi. Jeux, programmes mathématiques, utilitaires divers, programmes musicaux... sont au rendez-vous !
- Une série d'annexes contenant des documents de référence pour le programmeur (liste exhaustive des messages d'erreur, liste complète des instructions...).

Il est important de noter que l'existence de différentes versions de HP48 (S ou SX) est prise en compte dans cet ouvrage: tous les programmes, schémas et autres informations, à l'exception de ce qui concerne les modules-mémoire, sont indépendants du type de machine possédée.

Maintenant c'est à vous !

Nous vous souhaitons une agréable lecture !

Table des matières

Deuxième partie
Le langage machine

Les ressources cachées de la HP48:
Comment faire plus que ce que Hewlett-Packard a prévu...

Troisième partie
Bibliothèque de programmes

Toute une collection de programmes utiles et prêts à l'emploi...

Annexes

Réponses aux exercices, documents de référence pour le programmeur glossaire et index...

Première partie

La HP48

Introduction

Vous avez entre les mains une des meilleures machines à calculer du marché, si ce n'est la meilleure...

Elle diffère des autres machines du commerce car elle est beaucoup plus complexe du point de vue matériel, bien plus simple du point de vue de l'utilisateur, et peut vous permettre de résoudre des problèmes d'une haute complexité.

Etant donné le nombre incroyable de fonctions internes et leur puissance, il a fallu élaborer un système d'utilisation très puissant, que tout le monde puisse utiliser, du mathématicien émérite, à l'informaticien le plus compétent, en passant par les physiciens, les statisticiens (...) mais aussi par ceux qui n'entendent rien à tous ces domaines.

L'utilisation de cette machine étant bien différente de celle des calculatrices habituelles, elle apparaît souvent au premier abord comme compliquée alors qu'en réalité c'est certainement la plus simple qui soit. Ce n'est qu'une question d'habitude et en quelques jours, avec un peu de pratique, vous deviendrez un virtuose de la HP48...

Les chapitres de cette première partie sont consacrés à une vision générale de l'utilisation standard de la machine: quelques trucs à savoir, comment faire des programmes simples, comment s'organiser...

Mais attention: ces quelques informations ne peuvent en aucun cas se substituer aux manuels fournis par Hewlett-Packard ! Le but de cette partie n'est que de vous présenter les capacités de votre machine, de manière à vous faciliter la lecture de ces manuels...

En fait, la HP48 permet de faire beaucoup plus de choses que ce que Hewlett-Packard présente dans ses manuels: grâce au langage-machine il est possible d'accéder à de nouvelles ressources, de réaliser des programmes infiniment plus rapides...

C'est pourquoi une deuxième partie vous apprendra, de manière très didactique, abordable par les programmeurs de tous

niveaux, ce qu'est la programmation en langage machine et vous décrira la structure interne de la HP48...

Si vous ne connaissez rien au langage machine ou à l'assembleur voici une bonne occasion de vous en faire une idée...

Mais avant de passer à cela, il convient de bien connaître l'utilisation normale de la machine !

Pour vous aider dans cet apprentissage, des exemples de programmes, depuis des programmes élémentaires jusqu'à des programmes complexes, sont donnés en troisième partie (bibliothèque de programmes).

En les utilisant et en les modifiant au gré de votre imagination, vous serez très rapidement capable de faire vous-même des programmes sophistiqués...

Première approche de la HP48

Votre machine est sous vos yeux, elle est tapissée de boutons et d'inscriptions bleues, oranges et blanches qui ne signifient pas grand chose à première vue...

NON ! Ne partez pas en courant ! C'est comme un sapin de Noël: à première vue ça fait fouillis, mais si on s'y arrête quelques instants, on s'aperçoit que les décorations ont été placées judicieusement, que chacune est à sa place, et que son créateur n'a pas travaillé à la légère, bien au contraire.

Dites vous bien que bientôt vous arriverez à maîtriser l'ensemble et trouverez cela génial...

Avant toute chose, comme chaque appareil électrique, la HP48 a besoin de courant. Vérifiez donc que les trois piles électriques se trouvent bien dans le compartiment (au dos de la machine en bas) et dans le bon sens (la pile du haut et celle du bas ayant le "plus" vers la gauche, celle du milieu l'ayant vers la droite).

I) Le clavier

La deuxième chose à faire est de la mettre en marche. Jusque là tout est simple, il suffit d'appuyer sur le bouton [ON] qui est la première touche en bas à gauche (c'est écrit en blanc).

Au-dessus (c'est-à-dire dans la direction de l'écran à cristaux liquides) se trouvent deux touches [↱] (bleue) et [↰] (orange).

Les inscriptions blanches inscrites sur la touche correspondent en général à l'action d'un simple appui sur la touche.

Les inscriptions bleues au dessus d'une touche correspondent à l'appui de [↱] (bleue) suivi de l'appui de cette touche. De même les inscriptions oranges correspondent à un appui de [↰] suivi d'un appui sur cette touche.

Ainsi [↱] [STO] exécute la commande RCL (qui comme nous le verrons plus tard effectue un ReCaLl, c'est à dire rappelle le contenu d'une variable).

Au dessus de [↰] se trouve la touche [α].

Si vous appuyez une seule fois sur [α] la prochaine touche appuyée correspondra à une lettre (celle inscrite en blanc à droite de certaines touches).

Par exemple, [α] puis [SIN] donnera la lettre "S", alors qu'un simple appui sur [SIN] exécutera la fonction sinus.

Pour rester en mode alphabétique, il convient d'appuyer deux fois de suite sur [α].

Pour sortir de ce mode, il suffit d'appuyer une autre fois sur [α].

Ainsi, pour taper 'AB' il faut appuyer consécutivement sur les touches: ['] [α] [α] [A] [B] [ENTER].

II) L'écran

Il est divisé en 3 parties:

- Au dessus de la barre horizontale se trouve l'état de la machine. Vous y trouverez entre accolades ({ }) le répertoire courant (voir chapitre 3 pour vous familiariser avec l'arborescence).

  Peuvent s'y trouver aussi de petits chiffres (1, 2, 3, 4, et 5) indiquant l'état de certains indicateurs de la machine, l'indication du mode de mesure des angles (RAD, pour le mode "radians", ou GRAD, pour le mode "gradians", rien n'apparaissant en mode "degrés") ainsi que la date et l'heure.

- En dessous, séparées de la première zone par une barre horizontale, sont affichées 4 lignes:

  Il s'agit de l'affichage de la pile (voir le chapitre 2).

- Enfin, la troisième zone représente le "menu" courant qui est constitué de 6 cases noires dans lesquelles se trouvent des noms évoquant la fonction des 6 touches blanches situées juste en dessous (première rangée du clavier).

  Ainsi la touche blanche [ ]A engendre-t-elle la fonction dont le nom figure sur la première case du menu (en bas à gauche de l'écran) et ainsi de suite pour les suivantes.

  Une case surmontée d'une petite barre horizontale correspond à l'accès à un sous-menu.

  Ces notions de menus et de sous-menus seront revues dans le chapitre 3...

  Ainsi, si vous appuyez sur 'MEMORY' ([↰] puis [VAR]), conduira à afficher le menu de gestion de la mémoire: "MEM", "BYTES", "VARS", ORDER", "PATH", "CRDIR". Appuyer alors sur la touche [ ]C revient à exécuter la fonction VARS...

Cette première vision de la HP48, orientée vers l'aspect physique de la machine, est maintenant terminée.

Nous allons à présent entrer dans le monde merveilleux de l'utilisation de cette fantastique machine...

Exercices:

A-1-1: Quelle séquence de touches doit-on utiliser pour accéder à "=" ?

A-2-2: Même question pour 'RCL".

La notation polonaise inversée

La HP48 utilise un mode de calcul appelé "la notation polonaise inversée" (en anglais Reverse Polish Notation, RPN) qui repose sur le principe de pile.

Définissons tout d'abord le principe de pile...

I) La pile

Imaginez une pile d'assiettes... La seule assiette accessible à un instant donné est celle du dessus (la première).

La HP48 stocke les données temporaires de la même manière: elle les empile... et vous les montre à l'écran (du moins pour les 4 dernières entrées) précédées de leur numéro d'ordre (1:, 2:, 3: et 4:). Evidemment cela ne ressemble plus trop à notre pile d'assiettes puisque la première est celle du bas... Mais le principe reste le même !

Suivant le principe de pile, seule la donnée au niveau 1 (la plus en bas de l'écran) est disponible... Heureusement il existe des commandes permettant d'influer sur l'ordre des éléments ! Mais avant de les étudier, apprenons à placer des données dans cette fameuse pile...

La HP48 gère plusieurs types de données (réels, entiers, chaînes de caractères, noms, programmes, équations, objets graphiques, etc...). Chacun de ces types d'éléments est susceptible d'être placé dans la pile.

Pour cela, il suffit de taper l'intitulé de l'objet et de presser [ENTER] qui valide cette entrée.

Par exemple, pour placer le réel 123 dans la pile, il suffit de taper la séquence de touches: [1] [2] [3] [ENTER].

L'écran change alors d'aspect, et sa partie centrale présente un affichage du type suivant:

Cela signifie que la pile contient un élément, 123, placé au niveau 1...

Remarque: la HP48 n'affiche que les quatre premiers niveaux de la pile, mais celle-ci peut être beaucoup plus importante (limitée seulement par la mémoire disponible).

II) Calculer en RPN

Les différentes fonctions de la HP48 (addition, soustraction...) vont donc devoir prendre leurs données dans la pile... Et leur résultat ? Très logiquement elles le remettent dans cette pile !

Ce style de notation est souvent déroutant pour l'utilisateur débutant habitué à la notation standard. Mais avec l'usage il se rendra vite compte qu'elle est plus performante. En particulier, elle évite l'usage de parenthèses, car la pile sert au stockage des données intermédiaires: par exemple pour calculer (2+3)*(4+5), on effectuera les commandes suivantes:

- On part de la pile vide (si elle ne l'est pas, utilisez la commande CLR ([↱] [←]) qui la nettoie, et que nous reverrons bientôt). L'affichage central est alors:

- [2] [ENTER] La pile est alors:

- [3] [ENTER] La pile est alors:

```
4:
3:
2:                                2
1:                                3
```

  Remarquez que le "3" a poussé le "2" au deuxième niveau, ce qui est normal puisque la nouvelle "assiette du dessus" est "3"...

- [+] qui réalise l'addition:

```
4:
3:
2:
1:                                5
```

- [4] [ENTER] La pile est alors:

```
4:
3:
2:                                5
1:                                4
```

- [5] [ENTER] La pile devient:

```
4:
3:                                5
2:                                4
1:                                5
```

- [+] qui donne:

```
4:
3:
2:                                5
1:                                9
```

- Et enfin [*], d'où le résultat:

```
4:
3:
2:
1:                               45
```

Nous n'avons eu aucune parenthèse à taper, et en plus nous avons pû contrôler les résultats intermédiaires (5 et 9)...

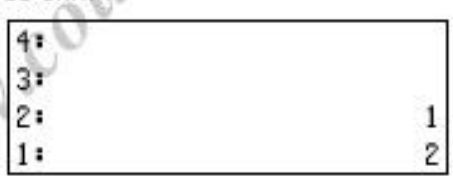
La seule chose à se rappeler est donc: une commande prend ses arguments (les données dont elle a besoin) dans la pile, et y replace ses résultats...

III) Gérer la pile

Nous avons vu que les commandes n'influaient que sur les premiers éléments de la pile, alors est-il impossible d'accéder aux autres ? Non, car tout est prévu: nous avons à notre disposition des commandes de gestion de la pile... En particulier nous disposons au clavier des commandes suivantes:

- SWAP ([↰] [▶]) qui échange les deux premiers éléments de la pile (niveaux 1 et 2). Par exemple:

donnera après SWAP:



- DROP ([↰] [←]) qui enlève l'élément au niveau 1. Par exemple:

donnera:

- CLR ([↱] [←]) qui vide la pile. Appliquée à une pile quelconque, il conduira à:

Mais il en existe d'autres: elles sont accessibles par le menu STK, sous menu de PRG (appuyer sur [PRG] puis [STK], première touche de menu, et n'oubliez pas, les menus s'affichent par pages de six fonctions et deux commandes permettent de passer de page en page: NXT et PREV). Ces commandes de gestion sont:

- OVER place dans la pile une copie de l'élément situé au niveau 2:

conduira à:

- ROT effectue une rotation des 3 premiers éléments de la pile:

donnera:

- ROLL est une fonction similaire, mais qui prend un argument (au niveau 1 de la pile) correspondant au

nombre d'éléments à traiter. Ainsi 2 ROLL correspond à SWAP, 3 ROLL à ROT...

- ROLLD fonctionne comme ROLL mais effectue une rotation dans l'autre sens. Par exemple si la pile est:

```
4:                4
3:                5
2:                6
1:                3
```

Alors ROLLD donnera:

```
4:
3:                6
2:                4
1:                5
```

(ne pas oublier que ROLLD prend un argument, ici 3).

- PICK prend aussi un argument dans la pile. Elle considère alors qu'il s'agit d'un numéro de niveau et copie l'élément qui s'y trouve. 2 PICK correspond donc à OVER. Exemple:

```
4:          123456789
3:                  1
2:                  1
1:                  3
```

alors PICK donnera:

```
4:          123456789
3:                  1
2:                  1
1:          123456789
```

(ne pas oublier que PICK prend un élément dans la pile, ici 3).

- DEPTH renvoie le nombre d'éléments dans la pile, c'est à dire le nombre d'étages occupés. Si la pile est vide on obtiendra donc 0. Exemple:

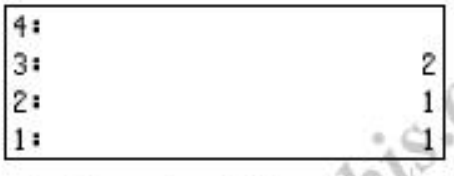
```
4:
3:
2:                33333
1:                44444
```

donnera:

(il y avait 2 éléments dans la pile).

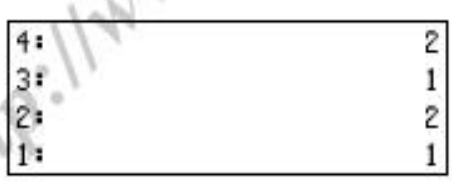
- DUP duplique l'élément au niveau 1:

donne:



- DUP2 duplique les 2 premiers éléments de la pile. Ainsi:

donnera:



- DUPN est une généralisation de DUP et DUP2: elle prend un argument (n) et duplique les n premiers éléments de la pile. Ainsi 1 DUPN correspond à DUP et 2 DUPN à DUP2.
- DROP2 ôte les deux premiers éléments de la pile:

donnera:

```
4:
3:
2:
1:                                3
```

- DROPN est une généralisation de DROP et DROP2. Elle prend un arguments dans la pile (n) et ôte les n premiers éléments de la pile. 1 DROPN correspond à DROP, 2 DROPN à DROP2...

Cette revue des commandes de gestion de la pile est terminée. Comme vous pouvez le constater, le jeu de commandes mis à votre disposition est très complet...

Exercices:

A-2-1: Calculer 5/((3+1)*(9-5))

A-2-2: Si la pile contient:

```
4:
3:                                3
2:                                2
1:                                1
```

comment arriver à:

```
4:
3:                                1
2:                                2
1:                                3
```

A-2-3: Que calcule la séquence de touches suivante ?

[5] [ENTER] [3] [*] [1] [1] [-] [4] [/] [1] [-] [COS]

Quel en est le résultat ?

Bien organiser ses données

La HP48 est un véritable petit ordinateur, et à ce titre elle doit être capable de stocker des données. Celles-ci peuvent être de différents types: réels, entiers, programmes, listes...

Elles peuvent se regrouper en deux familles: données internes (fonctions préprogrammées...) et données utilisateur (celles que vous entrerez dans votre machine).

Toutes ces données apparaissent soit sous la forme d'objets dans la pile (partie centrale de l'écran), soit sous formes d'entrées dans les menus...

I) Menu

Il existe deux types de menus: les menus de fonctions internes et les menus utilisateur. Dans ces derniers, vous verrez apparaître vos propres données. Un menu est une série d'objets accessibles par leurs noms, mis en correspondance avec les six touches du haut du clavier.

Si il y a plus de 6 objets, les autres apparaîtront si l'on fait défiler la liste grâce à [NXT] (NEXT, page suivante, c'est-à-dire les six cases suivantes) et [↰] [NXT] qui est PREV (PREVIOUS, page précédente).

Ainsi: [↰] [VAR] (MEMORY) vous amène dans le menu MEMORY qui rassemble l'ensemble des fonctions internes assurant la gestion de la mémoire. Si vous appuyez sur [ ]A (au dessous de MEM, en bas à gauche de l'écran), la machine renvoie une valeur dans la pile (partie centrale de l'écran). L'affichage ressemblera à:

Lorsque vous avez appuyé sur la touche [ ]A, la HP48 a reconnu que vous désiriez exécuter l'objet MEM et a répondu à votre demande. Cette fonction renvoie la mémoire libre (c'est à dire la place qui reste à votre disposition). Cette valeur est exprimée en octets (voir l'annexe "Binaire, hexadécimal et autres barbaries...").

Si vous appuyez sur [NXT], vous pourrez accéder aux autres fonctions du menu MEMORY...

Remarque: cette promenade dans le menu est cyclique: si, arrivé à la dernière page de menu, vous appuyez encore sur [NXT], vous vous retrouverez à la première page...

Autre exemple: [↰] [MODES] vous met dans le menu MODES qui se décompose en 4 pages ressemblant à:

page 1: [STD] [FIX] [SCI] [ENG] [SYM] [BEEP]

page 2: [STK] [ARG] [CMD] [CNC] [ML] [CLK]

page 3: [DEG] [RAD] [GRAD] [XYZ] [R<Z] [R<<]

page 4: [HEX] [DEC] [OCT] [BIN] [FM,] [ ]

Les pages 1, 2, 3 et 4 s'enchaînent dans cet ordre avec NXT et dans l'ordre inverse avec PREV.

Si vous appuyez sur [CLK] en fin de page 2, l'heure et la date apparaissent et disparaissent en haut de l'écran à chaque appui et [CLK] devient [CLK•] et réciproquement. Lorsqu'un "•" apparait dans une case du menu cela signifie que l'option en question est active.

C'est grâce à ces menus que l'on peut paramétrer le fonctionnement de la HP48 (mode de mesure des angles...).

Dans certains menus se trouvent de petites barres au dessus de certaines cases. Tel est le cas pour le menu PROGRAMS (accessible en appuyant sur [PRG]).

Cette barre signifie que si vous appuyez sur la touche correspondante, vous aurez accès à un menu, sous-menu du premier.

Cette structure peut se décrire par une arborescence:

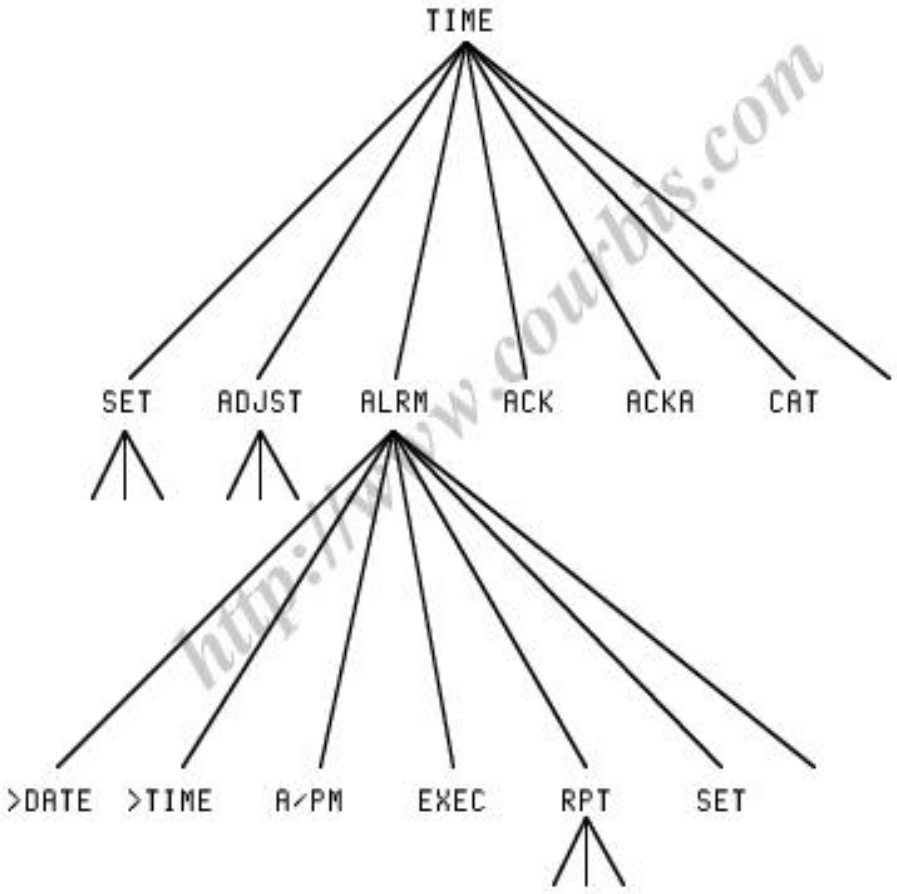
II) L'arborescence des menus

Pour expliquer ceci nous utilisons la notion d'arbre.

Le menu principal (le premier) s'appelle la racine. Dans cette racine on voit des cases "normales" et éventuellement d'autres qui possèdent une petite barre sur le dessus.

Ces cases "spéciales", correspondant à des sous-menus, sont des branches qui partent vers d'autres menus.

Au bout de chaque branche se trouve un nouveau menu qui est sous-menu de la racine. Par exemple, pour TIME ([↰] [4]), on a l'arborescence suivante (représentée partiellement):



Ces sous-répertoires peuvent eux-mêmes avoir des sous-répertoires en plus des données qu'ils contiennent (par exemple

REPT constitue un sous-menu du sous-menu ALRM) et ainsi de suite.

Pour nommer les menus les uns par rapports aux autres, on parle de menu-père et de menu-fils. Ce sont deux menus reliés par une branche, le père est celui le plus proche de la racine, le fils est celui le plus éloigné...

III) Le menu "VAR"

Le menu "VAR" est votre menu. C'est là que vous pouvez stocker vos données, créer vos propres sous-menus...

La racine du menu VAR a un nom particulier: HOME. Pour descendre dans un sous-menu, il suffit d'appuyer sur la touche correspondante à une case de menu (avec la petite barre au dessus) ou de taper son nom.

Pour revenir au menu- père il suffit de faire faire [↰] ['] (UP).

Et pour revenir directement à la racine: [↱] ['] (HOME).

On appelle menu VAR courant, le répertoire dans lequel nous nous trouvons à un instant donné.

Pour stocker une donnée, il suffit de la mettre dans la pile, de rentrer un nom (suite de caractères entre ') et de le STOcker...

Exemple: 512 [ENTER] (place le réel 512 dans la pile), puis:

['] [α] [α] [A] [B] [C] [ENTER]

L'écran ressemblera alors à:

```
4:
3:
2:                    512
1:                  'ABC'
```

Tapez [VAR] (pour vous placer dans votre menu) puis [STO] (pour stocker le réel). [ABC] apparaît à gauche (au début) du menu courant...

Pour rappeler la donnée, il suffit de faire ['] [α] [α] [A] [B] [C] [ENTER] [↱] [STO].

Vous pouvez aussi faire [↱] [ABC] où [ABC] est la touche du menu correspondant à ABC. De même si [ABC] existe déjà dans le menu pour stocker quelque chose sous le même nom (ce qui efface l'ancienne donnée), il suffit de mettre dans la pile le nouveau contenu et de faire [↰] [ABC].

Si le contenu d'une case de menu n'est pas un programme, vous pouvez rappeler son contenu en appuyant simplement sur la touche correspondante. Ainsi pour rappeler le réel 512 précédemment stocké, suffit-t-il d'appuyer sur la touche [ABC]...

Vous pouvez créer un sous-répertoire avec la fonction [CRDIR] dans le menu MEMORY en tapant un nom (par exemple 'DIREC'), puis en appuyant sur la touche [CRDIR].

Grâce à cette possibilité de créer des sous-menus, vous pourrez regrouper vos différentes données par affinités.

Par exemple, si vous entrez dans votre machine des programmes de mathématiques, des programmes en langage machine et des programmes de jeux, vous aurez tout intérêt à créer 3 sous-menus dans le menu HOME: 'MATHS', 'LM', et 'JEUX'.

Vous placerez dans chacun d'eux les données correspondantes, ce qui vous permettra de les retrouver facilement...

Trois autres commandes sont importantes à connaître dans le cas du menu VAR: il s'agit de UPDIR ([↰] ['']) qui permet de remonter d'un étage dans l'arborescence de VAR (pour passer du menu-fils au menu père), HOME ([↱]['']) qui permet de remonter directement à la racine de VAR. Enfin, la commande PATH (dans le menu memory: [↰][VAR][PATH]) permet de savoir où l'on se trouve dans l'arborescence de VAR. Cette commande renvoie une liste contenant la suite des noms des menus (le premier élément est donc toujours HOME).

Exercices:

A-3-1: Créer un sous-répertoire EXO du menu HOME regroupant trois variables A B et C, contenant respectivement les réels 1 2 et 3.

A-3-2: Combien de sous-menus contient le menu MTH ?

La programmation de la HP48

Jusqu'à présent, nous avons utilisé diverses commandes de la HP48.

Il est possible de créer ses propres commandes en utilisant ces dernières.

La HP48 ne possède pas seulement des commandes de base, mais un véritable langage de programmation...

Ce langage s'appelle le RPL (Reverse Polish Lisp ou Lisp Polonais Inversé)... Pourquoi ce nom étrange ? Parce que ce langage dérive d'un autre, le LISP ("LISt Processor" ou "Lot of Insane and Stupid Parenthesis" selon les auteurs).

Ce langage, très puissant et utilisé en intelligence artificielle, est malheureusement assez difficile d'emploi du fait de sa syntaxe: chaque commande de base s'écrit entre parenthèses... D'où une profusion de "(" et de ")" dans les programmes, ce qui les rend peu lisibles...

Mais la notation polonaise inversée, comme nous l'avons vu, permet de se passer de parenthèses...

Le RPL était né !

Ce langage gère des objets... Ce terme peut paraître assez vague, et c'est bien son but ! En effet la HP48 fait le moins de distinctions possibles sur le type des entités qu'elle manipule: la fonction utilisée est générique et s'adapte au cas particulier qui lui est proposé...

Ainsi, c'est le même "+" qui servira à additionner deux réels, deux entiers, deux matrices mais aussi un réel à un entier, une chaîne de caractères à une liste...

Grâce à cette aptitude de la fonction à s'adapter, il est possible de réaliser facilement des programmes complexes, qui, dans la plupart des cas, seront eux aussi génériques...

Par exemple: si la pile contient les réels 2 et 3:

appuyer sur + conduira à:

Ce qui est bien le résultat de 2+3...

Si vous y mettez "ABC" et "DEF":

Alors "+" réalisera l(addition (ou plus exactement la concaténation) de chaînes et la pile sera alors:

La commande s'est donc bien adaptée au types des arguments qui lui sont fournis...

Les concepts de base du Reverse Polish Lisp étant fixés, nous allons maintenant étudier les méthodes de programmation de la HP48...

I) Méthodes

Comme nous l'avons vu, un programme est un groupe de commandes; dans le cas du RPL, ce groupe de commandes est signalé par deux symboles: « et ».

Un programme en RPL est donc une suite de commandes mises entre « et ».

Prenons un exemple:

Pour calculer la puissance cubique d'un réel, nous taperions le réel puis la séquence suivante:

[3] [y^x]

Mais si nous avions beaucoup de cubes à calculer, il serait intéressant d'automatiser cette procédure. Nous allons pour cela créer le programmes CUBE1...

Comme nous l'avons vu, un programme est une suite de commandes délimitée par deux caractères spéciaux: « et ».

Tapons donc le programme...

Remarque: en cas de faute de frappe, la touche "←" permet d'effacer le caractère à gauche du curseur. En cas de faute grave, appuyer sur [ON] effacera tout ce que vous avez tapé (sans détruire le contenu de la pile).

- Pour commencer un programme, il faut un caractère spécial. Celui-ci s'obtient grâce à la séquence [↰] [-]. Comme vous l'avez remarqué, le caractère réciproque (») s'affiche lui aussi. L'écran ressemble alors à:

```
2:
1:
«
»
```

  et un curseur clignote à droite du "«". C'est là que vos prochains caractères vont apparaître...

- La première commande à effectuer est de placer 3 dans la pile, tapons donc [3] puis un espace ([SPC]) qui servira de séparateur...

- La seconde est y^x, appuyons donc sur cette touche... Surprise, ce n'est pas "y^x" qui s'affiche mais le symbole "^". Ce symbole signifie lui aussi "mise à la puissance".

Notre écran est alors:

Avec le curseur à droite de "^"...

- Notre programme est terminé, il ne reste plus qu'à le valider en tapant [ENTER]. L'écran sera alors:

Le programme est maintenant dans la pile, c'est le premier objet puisqu'il se trouve au niveau un...

Nous pourrions exécuter le programme en tapant [EVAL], mais cela nous conduirait à une erreur (puisque la pile ne contient pas assez de données) et nous perdrions le programme (une fois exécuté, il disparaîtrait de la pile).

Nous allons donc le stocker dans une variable:

['] [α] [α] [C] [U] [B] [E] [1] [ENTER] [STO]

Si maintenant vous appuyez sur la touche [VAR], vous devez voir "CUBE1" dans la case de gauche du menu... C'est votre programme !

Maintenant mettez un nombre dans la pile, appuyez sur [VAR] si vous ne l'avez pas déjà fait, appuyez sur la touche correspondant à CUBE1, le nombre dans la pile sera alors mis au cube en appuyant sur une seule touche au lieu de trois !

Il existe d'autres manières d'effectuer le calcul. En voici quelques-unes présentées comme tous les programmes de la bibliothèque (partie 3 de ce livre):

```
CUBE2 (# D649h)
«
  DUP DUP * *
»
```

```
CUBE3 (# E4F0h)
«
  → A
  «
    A A * A *
  »
»

CUBE4 (# 4526h)
«
  → A
  'A*A*A'
»
```

Ce listing s'interprète de la manière suivante:

- En caractères gras, se trouve le nom de l'objet;
- A côté du nom, entre parenthèses, se trouve la valeur du checksum de l'objet, qui permet de vérifier que ce dernier a été correctement entré (pour calculer ce checksum, placer le nom de l'objet dans la pile, par exemple 'CUBE2' et exécuter la fonction BYTES. Celle-ci renvoie deux valeurs: la valeur du checksum et la taille de l'objet. Le checksum est ici donné en hexadécimal, il faut donc se placer dans ce mode, par HEX, pour effectuer la comparaison);
- Dessous, jusqu'au nom suivant se trouve le listing de l'objet, c'est à dire l'aspect qu'il aura une fois tapé.

Pour entrer ces objets il faut donc:

- Entrer l'objet lui-même (comme nous l'avons fait pour CUBE1) et le mettre dans la pile (en le validant par [ENTER]);
- Mettre son nom dans la pile;
- Taper [STO].

Quelques remarques sur les quatre programmes:

- CUBE1 utilise la fonction interne déjà programmée: la puissance notée "^" qui prend deux arguments dans la pile: un réel et la puissance à laquelle vous voulez l'élever. CUBE1 s'occupe de mettre la puissance dans la pile (3), c'est à vous de spécifier le réel...
- CUBE2 utilise la pile. La fonction DUP duplique le niveau 1 de la pile (ce qui est très rapide comme toutes les fonctions de manipulation de pile). L'utilisation de

trois "DUP" permet d'obtenir 3 exemplaires de l'objet dans la pile, que l'on multiplie ensuite entre eux. Par exemple, si l'utilisateur lance 'CUBE2' sur la pile:

Après le premier DUP on aura:

Après le second:

Après la première multiplication:

Après la seconde:

Ce qui est bien le cube de 5...

- CUBE3 utilise le concept de "variable locale". Nous avons déjà vu ce qu'était une variable lorsque nous avons stocké des objets. Une variable locale n'est visible que par le programme pour lequel elle est déclarée. Pour créer une telle variable on utilise le symbole "→" suivi de un ou plusieurs noms de variables puis d'un "«" qui signifie que la liste de noms s'arrête là. Cela va créer, pour la partie de programme entre "«" qui

suit et le "»" correspondant, les variables correspondates, en utilisant les valeurs qui étaient dans la pile; dans cette partie du programme toute utilisation du nom d'une de ces variables rappellera la valeur qu'elle a prise par "→".

Quelques remarques:

- "→" conserve l'ordre d'empilage: si la pile contient 5 au niveau 2 et 42 au niveau 1, la séquence "→ A B" placera 5 dans la variable A et 42 dans la variable B...
- Si une variable locale a le même nom qu'une autre variable, c'est le contenu de la variable locale la plus proche qui est utilisée. Par exemple dans le cas du programme:

  « 1 → A « 2 → A « A » » »

  On place 1 dans une première variable locale A, puis 2 dans une variable locale de même nom, alors, lors de l'utilisation de A, c'est la valeur 2 qui sera utilisée...
- Les variables locales auront nécessairement toutes disparues lorsque le programme se terminera (normalement, par erreur ou par interruption);
- Par opposition aux variables locales, qui ne sont visibles que localement, on parlera de variables globales à propos des variables du menu VAR, celles-ci étant visibles de partout...

• CUBE4 est semblable à CUBE3 mais au lieu de faire suivre "→ A" par un objet programme on le fait suivre par une expression algébrique qui joue le même rôle.

CUBE1 est le plus court mais si l'utilisateur oublie de mettre une donnée dans la pile, il obtiendra non seulement un message d'erreur "^ Error: Too Few Arguments", mais aussi l'apparition d'un 3 dans la pile ce qui n'est pas très "propre"... Au contraire les autres programmes commencent par une fonction qui teste la présence d'un objet dans la pile avant toute autre chose... En fait c'est le programme suivant:

```
CUBE (# C875h)
«
  → A
  'A^3'
»
```

qui est le plus court et le plus performant, mais aussi le plus correctement programmé. Il est possible d'écrire un tel programme, grâce à l'existence d'une fonction interne "^". En règle générale, pour faire vos programmes, vous aurez à choisir parmi les méthodes de CUBE2 ou CUBE3, tout en sachant que:

- CUBE2 est le plus rapide.
- CUBE3 est bien programmé car il utilise des variables locales pour stocker les entrées et la pile pour les calculs, mais plus lent que CUBE2 car le rappel d'une variable locale est plus lent que l'exécution d'un DUP.

Il faut surtout éviter au maximum le genre de programme suivant:

```
« 'A' STO A A * A * 'A' PURGE »
```

Très lent car il doit créer et détruire une variable globale, qui peut écraser des données préexistantes (si une variable globale A existe déjà) et peut laisser des traces s'il est interrompu (la variable globale A peut subsister). Cependant avoir recours à un tel style de programmation est parfois nécessaire...

II) Variables et arborescence

Nous avons vu qu'une variable locale est une variable qui n'est visible que d'une partie bien définie d'un programme, qui apparaît au début de l'exécution de cette partie et qui disparaît à la fin.

Nous avons vu qu'une variable globale est une donnée stockée dans le menu VAR ou dans un de ses menus-fils... Il est possible pour des variables de porter des noms identiques. En effet, on peut avoir des variables de même nom dans des menus utilisateur différents, ainsi que des variables locales...

Alors comment savoir quel contenu va être utilisé lorsqu'on appelle une variable ? Il suffit de savoir comment la HP48 recherche ce contenu:

- Première étape: elle recherche si une variable locale porte ce nom, en commençant par les variables les plus récemment crées;
- Ensuite, si elle n'a pas trouvé, elle regarde dans le menu VAR courant si la variable existe. Si tel est le cas, elle prend le contenu. Sinon, et si ce menu n'est pas HOME, elle passe au menu-père. Sinon la variable n'existe pas

et au lieu de prendre le contenu de la variable, la HP48 va utiliser le nom de celle-ci (variable mise entre "' ").

Cette capacité de la HP48 à gérer des variables locales permet une technique de programmation classique: la récursivité...

III) Récursivité

Il existe des problèmes mathématiques dits récursifs, c'est-à-dire qui se référencient eux-mêmes. Par exemple, le calcul de la valeur d'une fonction f en un point n peut être tel que:

- f(n)=g(f(n-1)) où g est une fonction connue et calculable;
- il existe un nombre n_0 tel que $f(n_0)$ est connu et vaut f_0.

Nous sommes parfaitement capables de calculer f(n), pour n quelconque supérieur à n_0 puisqu'il suffit d'appliquer plusieurs fois la première formule à $f(n_0)=f_0$ connu, puis à $f(f(n_0))$, puis à $f(f(f(n_0)))$...

Inversement on peut dire: pour calculer f(n), je suppose f(n-1) connu et je fais mon calcul, pour calculer f(n-1), je suppose f(n-2) connu et je fais mon calcul... Calculons par exemple la fonction factorielle. Nous savons que:

- factorielle(n) vaut n*factorielle(n-1);
- factorielle de 0 vaut 1.

Pour calculer factorielle n, nous écririons donc:

- si n vaut 0 je sais faire, c'est 1 !
- si n est plus grand que 0, je dois d'abord calculer factorielle(n-1) et le multiplier par n...

Ce qui se programme directement:

```
FACTORIELLE (# 83DBh)
« → N
  « IF
      N 0 ==
    THEN
      1
    ELSE
      N 1 - FACTORIELLE N *
    END
  »
»
```

Explication du programme:

- On commence par prendre une valeur dans la pile et on la place dans la variable locale N;
- On teste ensuite si N contient 0:
  - si tel est le cas, "on sait faire" et on renvoie la valeur 1 (qui correspond à factorielle(1)) dans la pile...
  - sinon on commence par demander le calcul de factorielle(N-1) que l'on multiplie ensuite par le contenu de N.

Pour bien comprendre le fonctionnement d'un programme récursif, il faut garder à l'esprit que lorsqu'un programme "s'appelle lui-même", c'est une copie de ce programme qui est exécutée, copie qui n'a rien à voir avec le premier...

Regardons par exemple le calcul de factorielle(2). Pour calculer ce nombre nous aurons besoin de la valeur de factorielle(1) donc de factorielle(0) que l'on sait enfin calculer...

Trois copies de 'FACTORIELLE' vont donc s'enchaîner... Observons-les:

Copie 1	Copie 2	Copie 3
C'est celle que l'on appelle avec la valeur 2 dans la pile: pour elle N contient le réel 2...		
N est différent de 0, elle doit donc utiliser factorielle(1), elle place 1 (2-1) dans la pile et appelle factorielle		
Elle attend la réponse... et N vaut toujours 2 pour elle.	Factorielle est lancée avec 1 dans la pile... factorielle(1) ? on ne sait pas faire on doit appeler factorielle pour 1-1=0 !	

Copie 1	Copie 2	Copie 3
Elle attend toujours... N continue à valoir 2.	Elle aussi attend	Factorielle est lancée, elle trouve 0 dans la pile... Elle sait faire et place 1 !
Devinez quoi, elle attend encore. Pourquoi voudriez-vous que N change ?	On trouve le résultat dans la pile (1) on le multiplie par N (1) d'où le résultat: 1	
Ca y est: le résultat de factorielle(1) est arrivé, on le multiplie par N (2), d'où le résultat (2)...		

Le principe est le même quelle que soit la valeur du premier N. Le voici par exemple résumé pour 5:

N=5, f(4)=?					
N=5,...	N=4, f(3)=?				
N=5,...	N=4,...	N=3, f(2)=?			
N=5,...	N=4,...	N=3,...	N=2, f(1)=?		
N=5,...	N=4,...	N=3,...	N=2,...	N=1, f(0)=?	
N=5,...	N=4,...	N=3,...	N=2,...	N=1,...	N=0, f(0)=1
N=5,...	N=4,...	N=3,...	N=2,...	N=1, f(0)=1 ⇒ f(1)=1	
N=5,...	N=4,...	N=3,...	N=2, f(1)=1 ⇒ f(2)=2		
N=5,...	N=4,...	N=3, f(2)=2 ⇒ f(3)=6			
N=5,...	N=4, f(3)=6 ⇒ f(4)=24				
N=5, f(4)=24 ⇒ f(5)=120					

D'où factorielle(5)=120...

Tout au long de ce chapitre, vous avez acquis quelques notions de base de programmation... A présent, il faut vous perfectionner: en essayant de bien comprendre le fonctionnement de petits programmes (ceux des manuels de la HP48 ou ceux de la bibliothèque de programmes de ce livre), ainsi qu'en en écrivant vous même... Voici donc quelques exercices:

Exercices:

A-4-1: Ecrire un programme additionnant deux réels pris dans la pile. Peut-il additionner deux chaînes de caractères ?

A-4-2: Que fait le programme suivant ?

```
« → A B « A B + A B * / » »
```

A-4-3: Ecrire un programme récursif calculant le $n^{ième}$ terme de la suite de Fibonacci U_n définie par:

- Si n est supérieur ou égal à 2, $U_n=U_{n-1}+U_{n-2}$;
- $U_0=U_1=1$.

Bien présenter ses données

Jusqu'à présent, nous avons découvert les capacités de calcul, de stockage et de programmation de la HP48. Mais savoir calculer, stocker des données ou écrire des programmes n'est pas suffisant...

En effet, la mémoire de la HP48 est très importante (32 Ko de base, jusqu'à 280 Ko pour la HP48sx munie de deux cartes 128 Ko, soit l'équivalent de plus de 200 pages de texte...). Il est donc important de bien organiser et de bien présenter ses programmes et données, de manière à pouvoir "s'y retrouver" plus tard.

Pour ce faire, il existe quelques techniques que nous allons étudier ici.

I) Faciliter l'accès aux données

Dans le chapitre 3, nous avons étudié l'arborescence des menus: c'est un des éléments essentiels d'une bonne présentation des données et programmes car cette arborescence nous permet de classer les différentes variables et programmes par affinité (tous les programmes mathématiques ensemble dans un menu 'MATHS', tous les programmes matriciels dans un sous-menu...).

De plus, au sein d'un même menu, nous pouvons classer les variables grâce à la fonction ORDER. Cette commande prend en argument une liste contenant les noms des variables dans l'ordre souhaité par l'utilisateur.

Ainsi, nous pourrons mettre en premier les programmes importants, suivis par leurs sous-programmes, moins utiles.

Il est aussi essentiel de bien choisir les noms, de manière à ce que la simple vision de l'intitulé d'une variable en évoque son contenu.

Cependant, il serait quelquefois utile d'associer un nom de fonction préexistante ou un dessin à un programme que nous venons de créer. Ceci est possible grâce au menu CST (touche à gauche de VAR).

Ce nouveau menu permet la création de liens entre des objets de la HP48 et un intitulé de touche de menu, sans consommation excessive de mémoire.

Le mécanisme de fonctionnement de ce menu est simple: lorsqu'on appuie sur la touche [CST], la HP48 recherche une variable 'CST' dont nous allons voir la structure. Si elle ne la trouve pas dans le menu courant, elle va explorer le menu-père et ainsi de suite. Si aucune variable 'CST' n'est trouvée, nous obtiendrions un menu vide sans utilité.

Il est donc possible d'avoir un menu CST par sous-menu de VAR et donc des menus CST adaptés au menu courant (d'où, encore une fois, l'intérêt de bien organiser ses données).

La variable CST doit contenir une liste. Pour chaque élément nous avons plusieurs possibilités:

- Il s'agit d'un nom: dans ce cas chaque touche du menu CST sera associée à la variable de ce nom;
- Il s'agit d'une chaîne de caractères: celle-ci sera mise dans la ligne de commande si l'on appuie sur la touche;
- Il s'agit d'une liste constituée de deux objets: dans ce cas le premier sera l'intitulé de la touche, et le second l'objet associé. Si le premier élément est un objet graphique de 21 colonnes et 8 lignes, alors la fonction sera représentée par le graphique correspondant...
- Tout autre objet sera exécuté. Son intitulé servira de label pour la touche correspondante.

Voici un exemple de menu CST. La règle de présentation des objets est toujours la même: le nom en gras, suivit du listing de l'objet. Pour le rentrer, il conviendra de taper la séquence { ... } suivie de [ENTER] 'CST' STO.

```
CST (# 9D17h)
{ { "A" "Un " } { GROB 21 8
0000000400C10A00E08FFFF0EFFFF1F700C10CFF70000000 "avion " }
{ "in" "dans "} { "the" "le "} {"sky" "ciel "} "!" }
```

Après avoir stocké cet objet, passer en menu CST (en appuyant sur la touche [CST], à gauche de [VAR]). Amusant, non ? A présent, appuyez successivement sur les six touches de menu, de gauche à droite... Bravo ! Votre HP48 vient de réaliser une traduction anglais-français !

Ce menu nous permet donc d'associer des icônes à des fonctions, mais aussi de mélanger fonctions internes de la HP48 et fonctions-utilisateur...

Mais nous pouvons encore faire mieux: ce type d'assignation de fonctions à des touches peut se faire pour la totalité du clavier. Cette redéfinition des touches est alors globale.

Nous allons voir cette capacité de la HP48 à travers un exemple. Voici un petit programme qui joue un air de musique aléatoire:

```
«
  -56 CF 1 10
  START
    4400 RAND * .1 RAND * BEEP
  NEXT
»
```

Tapez-le. L'affichage de la pile est alors:

```
2:
1:    « -56 CF 1 10 START
      4400 RAND * .1 RAND
      * BEEP NEXT »
```

Tapez à présent: [5] [1] [ENTER] [A] [S] [N] [ENTER]

Passez à présent en mode 1USR (par [↰] [α]), puis appuyez sur [ENTER] vous entendez une petite musique !

L'explication est simple: nous avons assigné le programme à la touche [ENTER]. Cette assignation n'est valable que dans un mode particulier: le mode USER. Nous sommes temporairement passé dans ce mode par [↰] [α] (cette séquence fait passer en mode "1USR", mode "USER" actif pour une seule touche). Pour y passer durablement, il convient de taper [↰] [α] [↰] [α]. "USER" s'affiche alors en haut de l'écran. Pour revenir au mode normal: [↰] [α]. Remarque: les touches non redéfinies gardent leurs significations premières en mode USER.

Vous pouvez ainsi redéfinir tout le clavier, y compris la touche ON, pour peu que vous soyez en mode USER ou 1USR. La syntaxe de ASN est la suivante:

```
arg1 arg2 ASN
```

arg1 représente la fonction que l'on désire faire effectuer à la machine lors de l'appui de la touche. Ce peut être un nom de programme, un programme lui-même ou tout autre objet. Un nom particulier est prédéfini: 'SKEY' qui rend à la touche sa fonction standard.

arg2 est un réel qui se décompose ainsi:

- Chiffre des dizaines: numéro de la ligne de la touche (entre 1 et 9, 1 correspondant à la ligne du haut);
- Chiffre des unités: colonne de la touche (entre 1 et 6, 1 correspondant à la première colonne);
- Chiffre des dixièmes: le mode de la touche:
  - 0 ou 1 mode normal
  - 2 mode [↰] (shift-orange);
  - 3 mode [↱] (shift-bleu);
  - 4 mode [α] (alpha);
  - 5 mode [α] [↰] (alpha, shift-orange);
  - 6 mode [α] [↱] (alpha, shift-bleu);

Par exemple, pour remplacer DROP au clavier, il conviendra d'assigner une nouvelle fonction à la touche 56.2.

A noter que la séquence 0 DELKEYS remet toutes les touches dans leur état standard.

Les techniques que nous venons de voir permettent de faciliter l'accès aux données. Nous allons à présent voir comment en faciliter la compréhension...

II) Faciliter la compréhension

Plusieurs méthodes existent pour améliorer cette compréhension des programmes ou de leurs résultats.

Nous en avons retenu trois, importantes et faciles à mettre en œuvre:

- La HP48 permet de placer des commentaires qui commencent par le caractère "@" ([α] [↱] [ENTER]). Malheureusement ces commentaires disparaissent dès l'appui sur [ENTER]... Ils sont donc peu utiles, si ce n'est lorsque l'on stocke les programmes sur un autre ordinateur. Pour laisser des commentaires dans un programme de manière constante, on peut placer une séquence du type "commentaire" DROP, où "commentaire" est le texte désiré. Ce type de remarque restera dans le programme. Vous pouvez en particulier noter le but du programme, sa syntaxe (nombre et type d'arguments en entrée) et quels résultats il renvoie...
- Les messages: il est intéressant de signaler à l'utilisateur ce qui se passe: il est donc souhaitable d'inclure des messages d'erreur et des indications sur le déroulement du programme...
- Expliciter les résultats: quoi de plus difficile à utiliser que le résultat d'un programme lorsqu'on ne sait pas à quoi correspondent les résultats ? Pour simplifier leur lecture, il est utile de les "tagguer", c'est à dire de leur rajouter un prefixe (nom, commentaire...) non pris en compte par les fonctions de la HP48. Cette opération s'effectue à l'aide de la fonction →TAG qui prend en arguments l'objet à tagguer et son tag. Le programme µSOLVER de la bibliothèque de programme utilise cette technique.

La morale de ce chapitre est simple: il vous faut concevoir vos programmes comme si quelqu'un d'autre devait les utiliser. De cette manière, si quelque temps plus tard vous décidez de les reprendre, vous ne rencontrerez pas trop de difficultés...