JPC

MARS 1994

NUMERO 94

Le numéro 40 F

A PROPOS DU CLUB

HP28

HP48

HP95 / HP100

EDITORIAL

La période la plus critique pour les nerfs d'un éditeur de journal est probablement la courte période où la maquette est chez l'imprimeur. En effet, c'est souvent à ce moment (et malgré de nombreux contrôles durant la mise en pages) qu'il s'aperçoit des multiples erreurs (fôtes d'ortografe, mots manquants ou en trop, phrases mal rédigées, erreurs de sélection des polices de caractères, mauvaise présentation de certains listings ou tableaux, NDLR non totalement rédigées, comme le mois dernier...). Il faut donc se faire violence pour ne pas se ruer chez l'imprimeur pour arrêter les machines ! Et il faut reconnaitre que dans le dernier JPC, de nouveaux records ont été atteints. Excusez nous donc et espérons que vous trouverez moins d'erreurs dans ce numéro...

Comme le mois dernier, vous trouverez dans ce journal un programme Basic pour les HP95 et HP100. Le problème avec ce genre de programmes est que l'on frôle de très près la limite que nous nous sommes fixés : ne parler que des machines de poche. Alors, devons nous seulement passer des programmes écrits pour le Swift!Basic, qui est le language destiné à nos palmtops ? Ou passer aussi des programmes écrits dans d'autres langages (Gwbasic...) ? Si nous utilisons d'autres languages, les applications décrites doivent-elles êtres limitées aux considérations "techniques" des palmtops ou aux programmes déstinés à être utilisés "sur le terrain" ? Votre avis est bien sûr le bienvenu.

Le problème avec la loi de Murphy (vous savez, la fameuse tartine beurée...) est qu'il en existe une version spécifique pour les éditeurs de journaux périodiques : C'est systématiquement quelques jours seulement avant la date de mise en page du journal que nous recevons les nouvautés. Cette loi s'est encore vérifiée ce mois ci avec la réception des Goodies Disks #9. Laissez nous donc un peu de temps pour y jeter un coup d'oeil approfondi, et vous y trouverez une présenttion complète dans le prochain numéro.

Continuons par une mauvaise nouvelle : HPUC (le club Espagnol) nous a informé ce mois ci de son intention de stopper la publication de son journal *HP Forum*. Ceci pour deux raisons principales : Tout d'abord, Carles Crespo, son président/fondateur, a décidé d'abandonner définitivement le club, suite à de très graves problèmes de santé. Ensuite, malgré qu'il y ait plus de deux cent adhérents, seulement quatre personnes semblent être réellement actives et participent réellement au journal. En fait, ils sont déjà confrontés aux même problèmes que rencontrent, depuis de longues années, les clubs plus anciens. Espérons qu'il reviendront vite sur leur décision, car il serait bête d'abandonner un travail si bien commencé...

Pour finir avec une note plus humoristique, vous n'êtes pas sans savoir que notre cher Ministre de la Culture et de la Francophonie veut faire voter une loi interdisant l'utilisation de termes étrangers. Si le terme *palmtop* (utilisé deux fois dans cet éditorial) peut être facilement remplacé par *Ordinateur de Poche* devrons nous remplacer Ram par MV (Mémoire Vive) ? Devrons nous aussi traduire nos sigles préférés (RPL, PCMCIA) ? Bien entendu, vous pouvez ici aussi nous donner votre avis. Et qui sait, si certains trouvent des équivalences originales ou amusantes, nous sommes bien sûr prêts à les publier dans un prochain numéro !

CONSEILS AUX AUTEURS POTENTIELS

Je profite du fait que les pages générales sont peu occupées ce mois-ci pour répéter quelques petits conseils aux personnes désirant écrire des articles :

- Si vous hésitez quand au choix des articles, n'hésitez pas à nous téléphoner. Suivant vos domaines d'intérêts, nous pourrons toujours vous aider à en trouver un qui peut intéresser d'autres adhérents.

- Pour les nouveaux adhérents, demandez nous d'abord si le sujet que vous voulez aborder n'a pas déjà été traité en profondeur quelques mois avant. Cela pourrait éviter de vous faire perdre du temps.

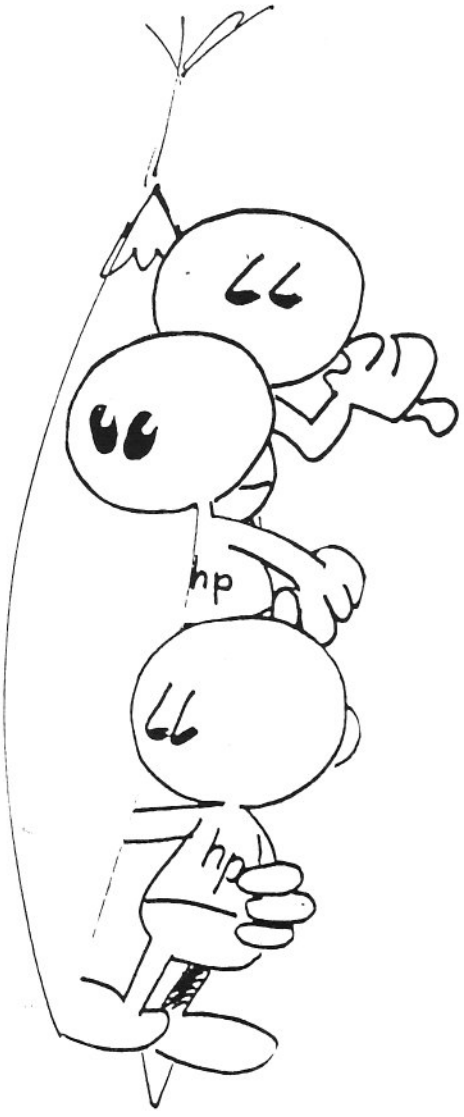
- Si vous êtes en train de travailler sur un article, n'hésitez pas à nous en envoyer une première version. Nous pourrons alors vous dire si certaines choses doivent être changées ou gagneraient à être supprimées. Ou au contraire, nous pourrons peut-être vous donner de quelques idées rendant l'article plus complet.

- Si vous venez d'acheter un nouveau produit, prévenez nous vite pour nous dire que vous voulez en faire une présentation. Cela pourrait éviter que plusieurs personnes travaillent sur la même chose en même temps !

- Au delà de ces questions de fond, pensez à suivre scrupuleusement les indications notées sur la rubrique *Ah, vous écrivez*, située dans les pages centrales. En effet, très peu de personnes en tiennent compte, et nous envoient leurs articles avec les paragraphes frappés "au kilomètre", et pire avec des tableaux dépassant 50 caractères de largeurs (taille d'une colonne de JPC), nous donnant souvent un surcroît de travail très apréciable. Suivez aussi les directives concerant les formats de fichiers (articles et programmes) à envoyer.

En résumé, n'hésitez pas à faire appel à moi ou aux autres membres du bureau. Nous sommes là pour vous aider. Et n'oubliez pas que cela peut toujours nous donner des idées pour nos propres articles !

Jacques Belin (123)



HP28

INTEGRATION NUMERIQUE DANS LE CHAMP COMPLEXE

Introduction commune HP28 et HP48

Nos HP28 et HP48 nous permettent de faire des intégrations numériques, mais seulement pour des fonctions de variables réelles. Il est tout de même possible de faire ce type d'intégration dans le champ complexe à l'aide de la stratégie :

Soit *F(z)* à intégrer de *u* à *v* avec:
- *z* variable complexe
- *u* limite inférieure complexe
- *v* limite supérieure complexe

F(z) sera intégrée successivement pour ses parties réelles et complexes.

Les bornes d'intégration sont rendues réelles par un changement de variable.

Particularités de programmation :

Pour faire le changement de variable nous opérons en mode symbolique.

Pour ne pas interférer avec d'autres variables la nouvelle variable aura un nom peu probable dans ce domaine (ici: '°').

Spécificités HP28

Le jeu des instructions HP48 dans le domaine des expressions algébriques est plus complet que celui des HP28. En particulier la fonction ↑MATCH est inconnue de ces dernières. Il fallait donc la réinventer ce qui rend le programme HP28 nettement plus long que celui pour HP48.

Rappel sur la fonction ↑MATCH pour ce qui est juste nécessaire ici :

niv.2	niv.1	→	niv.2	niv.1
'$symb_1$'	{ '$symb_{pat}$' '$symb_{repl}$' }	→	'$symb_2$'	0/1

Il s'agit donc de remplacer dans l'expression '$symb_1$' l'expression '$symb_{pat}$' par '$symb_{repl}$'.

Si cela a été possible, il est renvoyé 1 sinon 0.

Pour simuler ceci j'ai utilisé la fonction POS en opérant avec les expressions (de la fonction et des variables) transformées en chaîne. Il y a sûrement un autre moyen plus élégant d'arriver au résultat.

Voici donc le programmme pour HP28

INTGCP
cksum: #A12Eh 345.5 octets

```
« SPEED                  @ pour le confort
  4 ROLL →STR            @ variable d'intégration:
  2 2 SUB                @ algebraic -> chaine
  OVER NEG ALOG          @ précision p -> 1E-p
  RCLF                   @ état des flags
  → z r f                @ chaîne variable -> z
                         @ précision -> r
                         @ état des flags -> f
  « SCI 36 SF            @ modes arrondi et symbol.
    OVER - DUP           @ v - u
    ° *                  @ (v-u)*(nouvelle var:°)
    ROT +                @ '(v-u)*°+u'
    →STR                 @ transform. en chaîne
    2 OVER SIZE 1 - SUB  @ et suppression des '
    ")" + "(" SWAP +     @ encadre par "(" et ")"
    ROT →STR             @ transforme fonction:
    2 OVER SIZE 1 - SUB  @ algebraic -> chaîne
    WHILE DUP z POS DUP  @ boucle opérant le
    REPEAT DUP2 1        @ changement de variable
      SWAP 1 - SUB       @ dans la fonction
      4 PICK +           @ (équivalent de la
      ROT  ROT 1 +       @ fonction HP48 ↑MATCH)
      OVER SIZE SUB      @ mais traité ici par
      +                  @ l'intermédiaire de
    END DROP             @ chaînes
    "'" SWAP + "'" +     @ encadre par "'"
    STR→                 @ chaîne -> algebraic
    SWAP DROP * DUP      @ drop sur position,
                         @ fonction * u
    RE                   @ partie réelle de la
                         @ fonction transformée
                         @ F(°)
    { °                  @ la nouvelle variable,
    0 1 } r              @ les nouvelles limites
                         @ d'intégration et la
                         @ précision
    ∫                    @ intégration de la
                         @ partie réelle (résultat
                         @ numérique et précision)
    ROT IM               @ partie imaginaire  de
                         @ F(°)
    { ° 0 1 } r          @ variable et limites
    ∫                    @ intégration partie imag
    ROT SWAP R→C         @ erreur en complexe
    RND                  @ arrondi suivant
                         @ précision souhaitée
    'IERR' STO           @ du majorant de l'erreur
    R→C                  @ résultat intégration
                         @ en complexe
    RND                  @ arrondi
    f STOF               @ restauration état flags
  »                      @ ferme environnement des
»                        @ variables locales z r f
```

Mode d'emploi des programmes HP48 et HP28

Le 5ième paramètre en italique n'est nécessaire qu'aux HP28. Celui-ci sera transformé de *N* en *1E-N*.

En ce qui concerne la précision désirée le principe est le même suivant chaque type de machine :

- Pour HP28, fourniture d'une valeur numérique.
- Pour HP48, lié au mode d'affichage des valeurs numériques.

Les arguments:

HP28		HP48
niveau 5	l'expression de la fonction	niveau 4
niveau 4	la variable	niveau 3
niveau 3	la limite inférieure	niveau 2
niveau 2	la limite supérieure	niveau 1
niveau 1	*la précision désirée*	

Les limites peuvent être des réels ou complexes.

Le nom de la variable indépendante n'aura qu'une lettre.

Résultats:

Niveau 1:
Sous forme de complexe, la valeur numérique de l'intégrale relative à la précision spécifiée.

Dans la variable 'IERR':
Sous forme de complexe, le majorant de l'erreur relative à la précision spécifiée suivant le mode de chaque machine.

Exemples pour HP-28 et HP-48

Remarque: les résultats sont donnés en supposant le mode STD actif avant le lancement du programme et restauré après exécution de celui-ci sur la HP-28. Pour HP-48 il suffira de les transposer dans le mode d'affichage adopté.

Soit à déterminer la valeur de:

$I = \int F(z)dz$ entre u et v

1) $F(z) = \sqrt{z}/(z+1)$
pour u = (1,0) ou 1
et v = (0,1) ou i
avec la précision : HP28 → 7
HP48 7 SCI

```
5: '√Z/(Z+1/Z)'
4: 'Z'
3: (1,0) ou 1
2: (0,1) ou i
1: 7 (HP28)
```

```
INTGCP → (-.58578644,.53283998)
avec IERR = (5.85782110E-8,5.3282270E-8)
```

Valeur exacte : √2 - 2 + i*(√2 - LN(1+√2))
= (-.58578643763,.532839975352)

Machine légèrement plus rapide : HP28 !!!!

2) soit $F(z) = z^2/(2z - 1)$

à intégrer le long du cercle: $x^2 + y^2 - x = 0$
parcouru dans le sens positif

Ce cercle coupe l'axe des x en 0 et 1.

Conseil pratique :

Comme il est assez fastidieux d'entrer plusieurs fois les arguments, opérez de la manière suivante :

```
« 'Z^2/(2*Z-1)' Z »     'F' STO
```

Il suffira d'activer F pour avoir sur la pile la fonction et la variable pour chaque calcul. Si l'un ou les deux intervalles étaient fixes on pourrait les ajouter dans le programme F.

Avec la précision de type 5 et un partionnement en 4 intervalles du chemin, on obtient:

```
(0,0) à (0.5,-0.5)   →  I1 = (0.125,-0.0536505)
(0.5,-0.5) à (1,0)   →  I2 = (0.375,0.44635)
(1,0) à (0.5,0.5)    →  I3 = (-0.375,0.44635)
(0.5,0.5) à (0,0)    →  I4 = (-0.125,-0.053605)
```

d'où $I = I_1 + I_2 + I_3 + I_4$
I = (0,0.785399)

valeur exacte: $I = i\pi/4$ = (0,0.785398163398)

la conversion du résultat de l'intégration donne avec 5 SCI l'expression:

```
(0,.785399)   →Qπ   '(0,1/4*π)'
```

3) solution à certains problèmes de convergence.

Si des intégrales à variables réelles présentent des problèmes de lenteur de convergence, une transposition dans le champ complexe peut, dans certains cas, résoudre ces problèmes.

Soit à intégrer entre les limites: a = 1 et b = ∝ l'une ou les fonctions :

$I_1 = \int \cos(x)/(x+1/x)\ dx$
$I_2 = \int \sin(x)/(x+1/x)\ dx$

En posant $I = I_1 + I_2$

on obtient l'intégrale de variable complexe:

$I = \int e^{iz} / (z + 1/z)\ dz$

entre les limites: u = 1 et v = 1 + i∝

Sous cette forme, la convergence est rapide, ce qui n'est pas le cas de I_1 et de I_2 pour qui la méthode de Romberg est interminable et celle de Gauss donne des résultats aberrants. Avec I l'intervalle d'intégration peut être réduit considérablement par suite de la convergence rapide.

l'expression de F sera donc:

```
'EXP(i*Z)/(Z+1/Z)'
```

Pour la précision exprimée avec 3 SCI on obtient suivant les intervalles d'intégration :

intervalles	I
1 à (1,10)	(-0.3242 , 0.3824)
1 à (1,5)	(-0.3246 , 0.3813)
1 à (1,7)	(-0.3242 , 0.3823)
1 à (1,8)	(-0.3242 , 0.3824)

donc : I = (-0.3242 , 0.3824)
I_1 = -0.3242
I_2 = 0.3824

Remarques: le but de cet article n'aura été que d'attirer l'attention sur des possibilités que nous offrent nos HP28 et HP48. D'où un exposé mathématique manquant de rigueur et de vocabulaire au goût du jour. Je laisse aux spécialistes le soin d'aller plus loin et avec la rigueur qui s'impose pour tout problème de convergence.

Guy Toublanc (276)

MELI-MELO

Dans la rubrique *Trucs et Astuces* du JPC 93 et à propos du programme pour basculer entre les modes symboliques et numériques, j'avais trouvé une première méthode pour stocker le programme, puis les choses se sont simplifiées mais j'ai mélangé les deux méthodes dans l'article. La phrase du haut de la colonne de droite est :

Et stockez le programme assemblé dans la variable provisoire 'SYMA'.

Pourquoi faire compliqué quand tout est simple ?

Guy Toublanc (276)

HP48

LITTERATURE HP48

Il ne s'agit pas de nouveautés dans l'absolu mais de rééditions remaniées plus ou moins largement. Je les signale pour ceux qui pourraient en ignorer plus ou moins l'existence.

The HP 48 Handbook

Seconde édition 1993 par James Donnelly
Armstrong Publishing Company (USA)
ISBN 1-879828-04-9 (341 pages)

Sous un petit format (12,7 x 18 cm) c'est l'essentiel de ce que l'on trouve dans les différents manuels des HP48. Ce petit livre est très pratique pour retrouver rapidement une information. Cette seconde édition a subi diverses améliorations qui la rendent plus pédagogique et même parfois plus informative que les manuels HP. Evidemment certaines parties ne peuvent que résumer les manuels. Des exemples de programmes bien commentés seront appréciés par ceux qui ont des difficultés à programmer et en particulier avec les nouvelles instructions des HP48 G. Un chapitre de 62 pages nous livre un échantillonnage des Externals avec les mnémoniques HP et des exemples d'applications, mais par l'intermédiaire de SYSEVAL.

Les 146 pages supplémentaires par rapport à la 1ère édition (1990) se justifient pleinement.

De lecture facile cet ouvrage est devenu nécessaire pour tout HPiste (HP48 S ou G) digne de ce nom.

Il est au catalogue d'EduCALC sous la référence #2900 et pour $ 19.95

The HP 48 Pocket Book

1ère édition 1993 par James Donnelly
Armstrong Publishing Company (USA)
ISBN 1-879828-05-7 (56 pages)

Ce fascicule pour HP48G/X est un peu l'équivalent pour HP48S/X de l'aide mémoire que livrait HP pour ces dernières. Sous un faible format (9 x 16,3 cm), mais trop large pour entrer dans l'étui de la machine, nous y trouvons des listes de renseignements et la liste des commandes pour se les remémorer en cas de besoin. C'est le pense-bête des HP48Gistes. Dans ces strictes limites ce fascicule est utile.

Il est au catalogue d'EduCALC sous la référence #2901 et pour $ 6.95

"MATHEZ" LA HP48G/GX
EN 380 programmes

par J.-M. Ferrard
Editions D3I 1993
ISBN 2-908791-12-9 (474 pages)
prix 180F TTC

Cet ouvrage de Jean-Michel Ferrard est divisé en 2 parties d'importance et d'objectif différents.

C'est en quelque sorte un peu la réunion en un seul volume de la maîtrise de la HP48 en deux volumes mais où le premier volume *Programmation et Exercices* est repris ici moins dans l'optique généraliste puisque c'est à partir de problèmes posés que sont données les explications concernant la programmation en User Rpl et la manipulation des objets HP48. Cette première partie d'exercices et d'initiation représente 83 pages et certains de ses programmes peuvent être considérés comme des utilitaires ou des programmes mathématiques complémentaires de ceux de la deuxième partie. Ces programmes sont livrés dans un but pédagogique et ne sont pas forcément optimisés contrairement à ceux de la deuxième partie. D'ailleurs l'auteur incite le lecteur à améliorer les solutions.

Les thèmes abordés nous offrent un bon éventail de programmation en utilisant les objets HP48 :

Manipulation d'objets sur la pile, nombres réels ou complexes, listes, chaînes et tableaux, variables et répertoires, expressions, problèmes graphiques.

Les corrigés sont évidemment commentés.

Le gros de l'ouvrage est occupé ensuite par la très importante bibliothèque de programmes mathématiques, celle-ci ayant évoluée au fil des différentes éditions de la *maitrise* (HP28 puis HP48) *tome 2 : Programmation et Applications*. Lorsque cela s'est présenté les programmes ont bénéficié des nouvelles possibilités des HP48G/X et ont souvent subi une optimisation. Cela conduit à plus de rapidité et moins d'encombrement mémoire.

Je ne m'étendrai pas sur le contenu de cette 2ième partie car les programmes de Jean-Michel Ferrard dans le domaine mathématique sont bien connus et appréciés.

Pour les utilisateurs de HP48G/X et de mathématiques voici donc un ouvrage qui ne décevra pas. D'autre part cette nouvelle formule est plus avantageuse que celle de la maitrise: 180F au lieu de 2 fois 180F.

LES SECRETS DE LA HP48G/GX
par J.-M. Ferrard
Editions D3I 1993

TOME 1 6000 Bonnes Adresses
ISBN 2-908791-10-2 (421 pages)
prix 200F TTC

TOME 2 Externals et Assembleur
ISBN 2-908791-11-0
394 pages
prix 200F TTC

Les deux volumes de la première édition pour HP48S/X avaient été présentés dans JPC 83 pages 13-14 et JPC 87 page 17 aussi je me contenterai de signaler des différences nécessitées par la venue des HP48G/X ou justifiées par un désir d'amélioration.

Pour le tome 1 :

- Un chapitre de 15 pages concernant la mémoire cachée s'est inséré. Des choses pas faciles à expliquer sont exposées concernant les différentes situations et priorités relatives à la Ram et la Rom. En effet le problème s'est compliqué avec les GX dont la port 2 peut contenir jusqu'à 4 Mo de Ram qui doivent être gérés bien que la mémoire adressable du Saturn ne soit que de 512 Ko. On y apprendra ce que sont ces êtres mystérieux : les *access pointers* et comment cela peut fonctionner. On y trouvera aussi les adresses des routines d'accès aux différents ports logiques des numéros 2 à 33. Le mécanisme de CONFIG et UNCONFIG est bien expliqué.

- Le chapitre sur les ports a dû être remanié par la même occasion.

- Le chapitre *Solve et Plot* a disparu alors qu'un certain nombre d'adresses de cet ancien chapitre ont été conservées dans la Big-Liste et même certaines sont repérées comme des points d'entrées supportés. Est-ce un oubli?

- Des listes ont subi des remaniements. Les points d'entrée qui sont supportés sont maintenant signalés et ceux qui ont changé sont repérés. Il y a une table d'équivalence entre adresses HP48G/X et HP48S/X qui ont changé. L'auteur n'a pas oublié ses fidèles lecteurs dont le recyclage est ainsi facilité.

- Une liste récapitulative des points d'entrée, avec les mnémoniques spécifiques de l'auteur et pour l'ensemble des deux tomes, termine le premier tome (Big-Liste).

Pour le tome 2 :

- Un chapitre de 34 pages est venu grossir ce volume: c'est celui concernant la Ram-Système. La liste des adresses est donnée avec les mnémoniques HP et des explications concernant ces adresses. Un plan de cette Ram précède tout cela.

Par ailleurs des chapitres ont été aussi remaniés pour améliorer la forme ou parce que les programmes et outils présentés par l'auteur ont été modifiés.

Si les deux tomes ont grossi de 47 et 31 pages, leur prix n'a pas changé. Pour moi ils constituent une source d'information très précieuse et en particulier remédient aux lacunes de la documentation des outils de développement HP.

Mais comme pour l'ouvrage *Voyage 48G* de P. Courbis je trouve que l'on aurait peut-être pu aller plus loin concernant les spécificités des HP48G/GX. De ce point de vue je reste un peu sur ma faim.

Pour cette nouvelle édition on peut obtenir aussi sur disquette les différents programmes du tome 2.

Si certains trouveront peut-être que signaler des ouvrages qu'ils possèdent depuis un certain nombre de mois leur paraît inutile je leur répondrais qu'il aurait été très utile que ces personnes aient fait cela avant moi, ce qui aurait rendu service à d'autres lecteurs qui sont nouveaux dans le clan HP ou que leur éloignement des grandes villes ne leur permet pas d'être rapidement tenus au courant des nouveautés.

Guy Toublanc (276)

INTEGRATION NUMERIQUE DANS LE CHAMP COMPLEXE

L'intégration numérique dans le champ complexe est possible avec nos HP48 et HP28. C'est ce que j'ai montré dans la rubrique HP28 à laquelle je vous renvoie pour les explications et les exemples valables pour les deux types de machines.

Le programme est plus simple sur HP48 grâce à la fonction ↑MATCH et à la variable 'IERR' qui reçoit un majorant de l'erreur fonction de la précision désirée et précisée avant intégration par le mode d'affichage des valeurs numériques.

Dans le programme on trouve la variable '°' qui est le CHR 176 et s'obtient avec [α] puis [shift bleu] et [6]. Ce choix est justifié pour éviter les éventuels conflits avec d'autres variables utilisées par des programmes.

Si ce que vous aurez glané dans l'article de la rubrique HP28 vous intéresse, vous pourrez vous lancer avec le programme suivant:

INTGCP
cksum: # 5D2Eh 151 octets

```
« RCLF 5 ROLLD -3 CF
  OVER - DUP ' *
  ROT + ROT SWAP 2 →LIST ROT SWAP
  ↑MATCH DROP * 0 1
  3 PICK RE '
  ∫ →NUM
  IERR 0 1 5 ROLL IM '
  ∫ →NUM
  SWAP IERR R→C 'IERR' STO
  R→C
  SWAP STOF
»
```

Il n'est pas inutile de revoir le manuel de la HP48 pour le chapitre sur l'intégration avec ce qui est relatif à la précision des résultats.

Guy Toublanc (276)

PROBLEME DE COMPATIBILITE ENTRE HP48 S/SX et G/GX

Exécutez le programme suivant sur une quelconque HP48 S/SX (versions A à J):

```
«
 [[ 1 1 ]   @ un simple et fictif tableau statistique
  [ 2 2 ]]
 STOΣ
 { X EQ ΣPAR PPAR } PURGE
 SCATERPLOT GRAPH
»
```

Ceci vous laisse dans l'environnement interactif GRAPH, avec un PICT presque blanc (en fait seulement deux points ont été tracés), pressez la touche [FCN] du menu (la plus à droite), ceci tracera la ligne joignant les deux points statistiques.

Après cela abandonnez l'environnement GRAPH, pressez la touche [ON], allez dans le menu VAR et examinez :

```
'PPAR' := { (1,.85) (2,2) X 0 (0,0) FUNCTION Y }
'EQ'   := '0+1*X'
```

Ici tout est normal.

Maintenant transférez 'PPAR' et 'EQ' dans une HP48 G/GX, (par *IR* ou *wire*, c'est sans importance). Si vous utilisez le mode ASCII (indicateur -35 mis à zéro), tout est normal, mais si vous utilisez le mode Binaire, qui est le mode de transfert le plus rapide, vous recevrez côté G/GX des choses étranges :

```
'PPAR' := { (1,.85) (2,2) External 0 (0,0)
          FUNCTION Y }
'EQ'   := '0+1*UNKNOWN'
```

Que s'est-il produit ?

Bien sûr ce n'est pas un problème de transmission; en clair, la routine SX qui calcule et trace la fonction sélectionnée (LINear, LOGarithmic, EXPonential ou POWer) pour ajuster les données statistiques, crée, s'il n'est déjà présent dans le répertoire courant, un 'PPAR' dans lequel la valeur par défaut pour la DEPeNDent variable est 'X', et, par conséquent, une 'EQ' qui est fonction de 'X'. Les noms globaux 'X' inclus dans 'PPAR' et 'EQ' ne sont pas stockés comme de "vrais" objets, mais plutôt par le biais de deux adresses/ pointeurs de la ROM S/SX, où 'X' est réellement stocké, et cela pour économiser quelques octets de RAM. Malheureusement ces adresses spécifiques S/SX, # 4A154h dans 'PPAR' et # 4A19Eh dans 'EQ' ne sont pas restées inchangées sur les G/GX, mais disparurent, se déplacèrent, ou je ne sais quoi d'autre mais le résultat ne change pas :

En mode Binaire le transfert d'objets d'une HP48 à une autre conserve la stucture interne binaire, s'ils sont de vrais objets, ou simplement envoie leurs adresses s'ils sont référencés par un pointeur, comme dans le cas de 'X'. Mais l'adresse # 4A154h dans 'PPAR' ne signifie rien sur une G/GX, qui ainsi est affichée External, qui n'est pas dans ce cas un pointeur à une partie bien définie du code mais simplement une mauvaise adresse.

De même, # 4A19Eh dans 'EQ' est renvoyé dans le nom global 'UNKOWN' qui est la manière dont les algébriques représentent la plupart des objets non alloués, autres que les nombres réelles/complexes, unités, noms globaux/locaux et fonctions.

Les G/GX stockent par défaut, et reconnaissent le nom global 'X' comme deux autres adresses de ROM qui sont également présentes dans les S/SX :

```
# 47A59h et 4AB1Ch
```

En fait, aucune de ces adresses n'est garantie rester stable dans les futures versions HP48 G/GX, n'étant pas donnée dans la liste *HP NOMAS* des points d'entrée supportés.

Le transfert en mode ASCII, au contraire, travaille sans problèmes parce que le transfert se fait au moyen des caractères ASCII (de la manière dont les objets apparaissent dans la ligne de commande), qui sont envoyés et reçus (et parsés) en série, sans tenir compte de la structure interne des objets eux-mêmes.

David Fabiani (HPCC 650)
(traduit de l'anglais par Guy Toublanc)

David Fabiani est milanais et, bien qu'apportant sa participation au club anglais HPCC et à 48Sxtant, journal de notre ami Robert Pulluard, a eu la gentillesse de nous communiquer cet article.

Cette découverte doit nous rendre vigilants, en particulier pour les fichiers HP48 sauvegardés sur support magnétique puis diffusés par le club. De là à demander aux auteurs de programmes de transmettre les fichiers HP48 en User Rpl sous les deux formes, Binaire et ASCII, il n'y a qu'un pas.

En plus du problème des adresses différentes il y a aussi le fait que pour la procédure ci-dessus les G/GX ne génèrent plus la variable globale 'X' dans le répertoire courant. Est-ce pour éviter les conflits avec d'autres programmes qui utiliseraient cette même variable ?

G.T

CALCULS ASTRONOMIQUES

ACTE II

Dans mon précédent article (JPC 90), je vous ai décrit les routines de calculs des positions des principales planètes dans le système solaire. Cette fois, nous allons déterminer les coordonnées de ces objets dans notre ciel local. Nous ajouterons la lune à notre liste d'objets célestes.

Un objet dans le ciel est repéré par sa direction (0-360° par rapport au nord) et sa hauteur (0-90°). Pour passer des coordonnées écliptiques à ces coordonnées locales, il faut effectuer un double changement de repère: passage du repère héliocentrique dans le plan de l'écliptique au repère terrestre dans le plan de l'équateur, puis passage au repère lié au lieu géographique (longitude, latitude) et à l'heure de l'observation (la terre tourne).

1ère opération : passage au repère terrestre

Les planètes :

La première opération consiste à changer d'origine en retranchant le vecteur position de la terre, puis à effectuer une rotation correspondant à l'angle d'inclinaison de l'axe de la terre par rapport au repère sidéral. Cet angle, de l'ordre de 23°, est calculé par la routine `FA` dans la variable `Eps`.

La lune :

La routine `LUNE` fournit les coordonnées de la lune dans le repère écliptique centré sur la terre.

Le soleil :

Nous connaissons les coordonnées de la terre par rapport au soleil, donc nous connaissons l'inverse.

2ème opération : passage au repère local

Nous devons effectuer ici 2 rotations : une premiere rotation autour de l'axe Z d'un angle égal à la longitude plus l'angle de rotation de la terre sur elle-même, puis une deuxième rotation autour de l'axe Y d'un angle égal à la lattitude.

Réalisation des rotations :

Pour faire effectuer une rotation à un vecteur, une méthode simple consiste à utiliser une matrice rotation.

Les routines `MRX`, `MRY` et `MRZ` fournissent les matrices rotations autour des axes *x*, *y* et *z*. Elles prennent en entrée l'angle de rotation. Les mathématiques nous disent que la matrice rotation est constituée des nouvelles coordonnées des anciens vecteurs de base. La HP48 connaissant les coordonnées sphériques, ne nous ennuyons pas avec les sinus et cosinus et construisons directement la matrice.

Par exemple, `MRX` est défini comme suit :

```
«
  → a        @ angle de rotation
  «
    1 0 90       →V3 OBJ→ DROP
    1 90 90 a + →V3 OBJ→ DROP
    1 90 a       →V3 OBJ→ DROP
    { 3 3 }      →ARRY TRN
  »
»
```

En effet, les vecteurs de base ont pour nouvelles coordonnées :

```
[ 1 ∠0  ∠   90   ]   inchangé
[ 1 ∠90 ∠ (90+a) ]
[ 1 ∠90 ∠    a   ]
```

La fonction →ARRY ne permettant pas de construire une matrice à partir de vecteurs, les séquences "OBJ→ DROP redécomposent les éléments des vecteurs formés par →V3. TRN sert à transposer la matrice car →ARRY construit celle-ci par ligne et non pas par colonne.

Nouvelles fonctions :

A coté de CALC, DT et POSPL, apparaissent dans ASTRO :

LIEU : variable liste contenant la longitude et la latitude du lieu d'observation (en degrés décimaux).

Vous devez définir cette variable par vous-même.

A titre d'exemple, Paris correspond approximativement à { 2.3 48.7 } et Grenoble à { 5.43 45.1 }.

La longitude est positive si le point est à l'est du méridient de Greenwich, et la latitude est positive dans l'hémisphère nord.

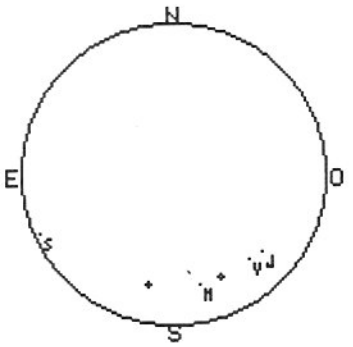
CIEL : cette fonction, utilisant POSPL (à exécuter donc après CALC) fournit une liste contenant les positions des astres dans le repère équatorial. Le résultat est dans POSEQ, dans l'ordre Vénus, Mars, Jupiter, Saturne, Soleil, Lune.

AFFCIEL : routine d'affichage des astres dans le ciel utilisant POSEQ, DT et LIEU.

PLANET : programme principal attendant la date et l'heure, et enchaînant CALC, CIEL et AFFCIEL.

Une précision : le temps utilisé dans tous ces programmes est le temps universel (encore appelé heure du méridient de Greenwitch), retranchez donc une ou deux heures de l'heure locale suivant la saison.

L'illustration jointe à cet article a été créée par : 16.111993 13 PLANET. J'ai choisi cette date car tous les astres sont présents dans le ciel. Les planètes sont représentées par un point et identifiées par une lettre, tandis que le soleil et la lune sont représentés par un rond et une croix.



AFFEQ : j'ai créé cette routine pour valider les résultats de mes programmes, je vous la laisse car elle peut intéresser certains d'entre vous. Elle convertit les coordonnées équatoriales de POSEQ du format sphérique utilisé par la HP48 au format astronomique : déclinaison droite et ascension en heures (ou degrés), minutes, secondes.

Par exemple, comparons les résultats de la HP48 aux données des éphémérides du 15/08/93 (source : revue *Ciel & Espace* juillet-août 93).

Faire 15.081993 0 CALC CIEL AFFEQ.

Résultats:

	HP48	Ciel & Espace
Mars	12h07mn42s	12h07mn42s
	-0°14'58''	-0°14'55''
Jupiter	12h45mn59s	12h46mn02s
	-3°40'43''	-3°41'09''
Saturne	22h00mn11s	22h00mn14s
	-13°48'38''	-13°49'41''

Ce qui n'est pas trop mal...

Nouvelles fonctions à ajouter dans le sous répertoire OBJ NDLR : Voir JPC 90 page 18 pour la première partie des Sources :

FA a été optimisée et complétée.

JD a été réécrite et grandement simplifiée grâce à DDAYS.

EQUAT, LOCAL, MRX, MRY, MRZ réalisent les changements de repères.

PPAR, DESSIN, FFOND, GASTR réalisent l'affichage graphique.

LUNE calcule la position de notre satellite.

Ce modeste programme de planétarium pourrait bien sûr être amélioré :
- animation de la rotation de la voûte céleste,
- gestion des autres planètes ou encore affichage les principales étoiles.

Je ne compte pas vous en proposer à court terme d'évolution (je me suis lancé dans un autre projet), aussi je laisse ces développements "au lecteur, à titre d'exercice".

J'espère cependant vous avoir convaincu de la puissance de la HP48 quant au calcul vectoriel dans l'espace et aux changements de repère.

A vos claviers, et à bientôt.

Programmes :

Répertoire ASTRO :

```
@-------------------------------------------
@PLANET
@  calcul et affichage du planetarium
@ entrees:
@  niveau2: date (ou 0 si date courante)
@  niveau1: heure (ou 0 si heure courante)
@-------------------------------------------
PLANET
«
  IFERR
    DUP2 DROP2
  THEN
    "ERREUR: ENTREZ
LA DATE ET L'HEURE"
    DOERR
  END
  CALC CIEL AFFCIEL
»

@-------------------------------------------
@CIEL
@  positions des objets (Vénus, Mars, Jupiter,
@  Saturne, Soleil, Lune)
@  dans le repère équatorial
@ entrees:
@  POSPL
@  arguments calculés par FA
@ sorties:
@  POSEQ
@-------------------------------------------
CIEL
«
  POSPL          @ rappelle positions planètes
  OBJ→ DROP
  4 ROLL         @ récupère terre
  5 ROLLD        @ construit liste
  4 →LIST        @  des 4 autres planètes
  → t p          @ terre positions
  «
    1 4 FOR i
      p i GET t -
    NEXT
    4 →LIST
    t NEG +      @ ajoute soleil à la liste
  »
  OBJ            @ va dans DIR OBJ
  SETM
  LUNE           @ calcul pos lune
  +              @ ajoute à la liste
  Eps            @ angle inclinaison equateur
  EQUAT          @ convertit pos objets
                 @ dans repère équatorial
  UPDIR          @ retour dans ASTRO
  'POSEQ' STO
»

@-------------------------------------------
@AFFCIEL
@  affichage du ciel local
@ entrées:
@  POSEQ, LIEU, DT,
@  Tsg0 (calculé par FA)
@ sorties:
@  PICT
@-------------------------------------------
AFFCIEL
«
  POSEQ          @ positions équatoriales
  DT 2 GET       @ rappelle heure
  LIEU           @          et lieu
  OBJ SETM
  Tsg0           @ t. sidéral de greenwitch OTU
  ROT ROT
  LOCAL          @ conv. repère local
  DESSIN
  UPDIR
»

@-------------------------------------------
@AFFEQ
@  conversion des positions équatoriales
@  exprimées en coord. polaires (HP48)
@  en coord. astronomiques :
@  ascension droite (0 à 24 h)
@  et déclinaison (0 à 90 °)
@ entrées:
@  POSEQ
@  mode DEG et coord. sphériques
@ sorties:
@  niveau 1 : liste des coordonnées
```

```
@-------------------------------------------
AFFEQ
«
  POSEQ DUP SIZE
  → p n
  «
    1 n FOR i
      p i GET V→ ROT DROP
      SWAP 360 / 24 * 24 MOD →HMS
      SWAP 90 SWAP - →HMS 2 →LIST
    NEXT
    n →LIST
  »
»



@ lieu d'observation
@ { longitude latitude }
@ longitude <0 si ouest
@ latitude <0 si sud
LIEU
{ 5.43 45.1 }
```

Sous-répertoire OBJ :

```
@-------------------------------------------
@JD
@  conversion de la date et de l'heure
@  en temps exprimé en jours Julian
@  par rapport au 1er janvier 2000 12h
@ entrees:
@  niveau2: date
@  niveau1: heure
@ sorties:
@  niveau1: temps en jours julian
@-------------------------------------------
JD
«
  HMS→ 24 / .5 - 1.012000 ROT DDAYS +
»



@-------------------------------------------
@FA
@ calculs des arguments fondamentaux
@ entrees:
@  niveau1: temps en jours Julian
@           par rapport au 1/01/2000 12h
@ sorties:
@  Lm,Gm,Fm,Ls,Gs,L2,G2,F2,L4,G4,F4,L5,G5,L6,G6
@  Ts, TsgO, Dms,Om,Eps,Dl
@  mode DEG et coord. sphériques
@-------------------------------------------
FA
«
  SETM
  DUP 13.17639643 * 218.31624 + 'Lm' STO
  DUP 13.06499295 * 134.96292 + 'Gm' STO
  DUP 13.22935027 * 93.27276  + 'Fm' STO
  DUP .985647348  * 280.46592 + 'Ls' STO
  DUP .9856       * -2.47464  + 'Gs' STO
  DUP 1.602169    * 181.97928 + 'L2' STO
  DUP 1.60213022  * 50.40828  + 'G2' STO
  DUP 1.60214407  * 105.29928 + 'F2' STO
  DUP .524071181  * -4.55292  + 'L4' STO
  DUP .52402078   * 19.38816  + 'G4' STO
  DUP .524050085  * 305.88984 + 'F4' STO
  DUP .083091215  * 32.25888  + 'L5' STO
  DUP .08309121   * 20.35116  + 'G5' STO
  DUP .033459736  * 47.9862   + 'L6' STO
  DUP .033459736  * 317.87532 + 'G6' STO

  DUP 36525 / 1 + 'Ts' STO   @ temps séculaire

  .5 - FLOOR .5 + 36525 / 1 +
  DUP .000387083 * 36000.76893 + *
  99.69098325 + 360 MOD
  'TsgO' STO      @ temps sidéral
                  @ de Greenwitch à 0h TU

  Lm Ls - 'Dms' STO
  Lm Fm - DUP 'Om' STO

  DUP COS 400 / Ts 1.3056E-2 * -
  23.4522222 +
  'Eps' STO       @ angle de l'équateur/écliptique

  SIN 4.7222E-3 * 'Dl' STO
»

@-------------------------------------------
@SETM
@ set mode DEG, et coord. sphériques
@-------------------------------------------
SETM
«
  DEG -15 SF -16 SF
»

@-------------------------------------------
@EQUAT
@  conversion en coordonnées équatoriales
@ entrées :
@  niveau 2: liste de coord. écliptiques
@            centrées sur la terre
@  niveau 1: angle du plan de l'équateur
@            par rapport à l'écliptique
@  mode DEG et coord. sphériques
@ sortie :
@  niveau 1: liste de coord. équatoriales
@-------------------------------------------
EQUAT
«
  NEG MRX         @ matrice rotation X
```

```
  OVER SIZE
  → p r n          @  positions rotation nombre
  «
    1 n FOR i
      r p i GET *
    NEXT
    n →LIST
  »
»

@-------------------------------------------
@LOCAL
@ entrées :
@  niveau 4: liste de positions équatoriales
@  niveau 3: temps sidéral de Greenwitch à 0h TU
@  niveau 2: heure TU de l'observation
@  niveau 1: liste longitude, latitude
@             du lieu d'observation
@  mode DEG et coord. sphériques
@ sortie:
@  niveau 1: liste d'observations locales
@-------------------------------------------
LOCAL
«
  OBJ→ DROP        @ {...} TsgO heure long lat
  90 SWAP - MRY    @ matrice rotation Y
  4 ROLLD          @ {...} rot.Y TsgO heure long
  ROT ROT          @ {...} rot.Y long TsgO heure
  15.04106863 * +  @ {...} rot.Y long GST
  + MRZ            @ matrice rotation Z
                   @ ATTENTION:
  *                @ multiplication non commutative !
                   @ l'ordre a de l'importance
  OVER SIZE
  → p r n          @ positions rotation nombre
  «
    1 n FOR i
      r p i GET *
    NEXT
    n →LIST
  »
»

@-------------------------------------------
@MRX, MRY, MRZ
@ Matrices rotations X, Y, Z
@ entrées:
@  niveau 1: angle rotation
@  mode DEG et coord. sphériques
@ sortie:
@  niveau 1: matrice rotation
@-------------------------------------------
MRX
«
  → a
  «
    1 0 90       →V3 OBJ→ DROP
    1 90 90 a +  →V3 OBJ→ DROP
    1 90 a       →V3 OBJ→ DROP
    { 3 3 }      →ARRY TRN
  »
»

MRY
«
  → a
  «
    1 0 90 a -  →V3 OBJ→ DROP
    1 90 90     →V3 OBJ→ DROP
    1 180 a     →V3 OBJ→ DROP
    { 3 3 }     →ARRY TRN
  »
»

MRZ
«
  → a
  «
    1 a NEG 90  →V3 OBJ→ DROP
    1 90 a - 90 →V3 OBJ→ DROP
    1 0 0       →V3 OBJ→ DROP
    { 3 3 }     →ARRY TRN
  »
»

@ param. graphiques
PPAR
{ (-100,-100) (100,100) X 0 (0,0) FUNCTION Y }

@-------------------------------------------
@DESSIN
@ Dessin du ciel
@ entrées:
@  niveau 1: liste des positions des objets
@  FOND, GASTR
@  mode DEG et coord. sphériques
@ sortie:
@  PICT
@-------------------------------------------
DESSIN
«
  FOND
  IFERR
    PICT STO
  THEN             @ si FOND ne contient pas encore
    DROP2          @ le GROB, on le construit
    FFOND
  END
  -19 SF           @ ->V2 construit un complexe
  DUP SIZE → p n
  «
    1 n FOR i
      p i GET V→ DUP
      IF 90 < THEN
        SWAP NEG 90 - →V2 C→PX OBJ→ DROP
        1 - SWAP 1 - SWAP 2 →LIST
```

```
      PICT SWAP GASTR i GET GOR
    ELSE
      DROP2
    END
    DROP
  NEXT
 »
 { } PVIEW
»

@-----------------------------------------
@FFOND
@ construction du fond
@ sortie:
@  FOND, PICT
@-----------------------------------------
FFOND
«
 # 131d DUP PDIM
 (0,0) 90 0 360 ARC   @ très lent!
 PICT { # 63d # 0d }   "N" 2 →GROB GOR
 PICT { # 63d # 124d } "S" 2 →GROB GOR
 PICT { # 0d # 63d }   "E" 2 →GROB GOR
 PICT { # 126d # 63d } "O" 2 →GROB GOR
 PICT RCL 'FOND' STO
»

@-----------------------------------------
@GASTR
@ GROBs des astres :
@ vénus, mars, jupiter, saturne, soleil, lune
@-----------------------------------------
GASTR
{
 GROB 6 8 0020008282820101
 GROB 6 8 0020008283828282
 GROB 6 8 0020000202028283
 GROB 6 8 0020000380010281
 GROB 6 8 2050200000000000
 GROB 6 8 2070200000000000
}

@-----------------------------------------
@ LUNE
@ Calcul de la position de la lune dans
@ le repère écliptique centré sur la terre
@ entrees:
@  arguments calculés par FA
@  mode DEG et coord. sphériques
@ sortie:
@  niveau 1: position
@-----------------------------------------
LUNE
«
 0 0 0 → r b l
 «
  Gm SIN 6.289 *
  Gm Dms 2 * - SIN 1.2739 * -
  Dms 2 * SIN .65833 * +
  Gm 2 * SIN .21361 * +
  Gs SIN .1856 * -
  Fm 2 * SIN .1144 * -
  Gm Dms - 2 * SIN .05889 * -
  Gm Dms 2 * - Gs + SIN .05722 * -
  Gm Dms 2 * + SIN .05333 * +
  Dms 2 * Gs - SIN .04583 * +
  Gm Gs - SIN .04111 * +
  Dms SIN .03472 * -
  Gm Gs + SIN .03056 * -
  Fm Dms - 2 * SIN .015278 * -
  Gm Fm 2 * + SIN .0125 * -
  Gm Fm 2 * - SIN 90 / +
  Gm Dms 4 * - SIN .01056 * -
  Gm 3 * SIN 100 / +
  Gm 2 * Dms 4 * - SIN .00861 * -
  Gm Gs - Dms 2 * - SIN .007778 * +
  Dms 2 * Gs + SIN 150 / -
  Gm Dms - SIN .0052778 * +
  Dms Gs + SIN 200 / +
  Lm 360 MOD +
  Dl -
  'l' STO

  Fm SIN 5.1281 *
  Gm Fm + SIN .28056 * +
  Gm Fm - SIN 3.6 / +
  Fm Dms 2 * - SIN .17333 * -
  Gm Fm - Dms 2 * - SIN .055278 * -
  Gm Fm + Dms 2 * - SIN .046389 * -
  Fm Dms 2 * + SIN .0325 * +
  Gm 2 * Fm + SIN .0172 * +
  Gm Fm - Dms 2 * + SIN .009167 * +
  Gm 2 * Fm - SIN 112.5 / +
  Fm Dms 2 * - Gs + SIN 120 / -
  Gm Dms - 2 * Fm + SIN 225 / -
  Gm Fm + Dms 2 * + SIN 240 / +
  'b' STO

  60.3629
  Gm COS 3.27746 * -
  Gm Dms 2 * - COS .57994 * -
  Dms 2 * COS .46357 * -
  Gm 2 * COS .08904 * -
  23454.8 /  'r' STO

  r l 90 b - →V3 "Lune" →TAG
 »
»
```

Jean-François Garnier (242)

L'INSERTION DE DATAS EN ASSEMBLEUR

Dans un programme en assembleur, l'utilisateur peut être amené à placer des data, c'est à dire non pas des instructions saturn mais des données destinées à être lues par un pointeur. La question se pose alors : comment obtenir l'adresse de ces data puisqu'elle change à chaque fois que l'on déplace le code dans la mémoire. Deux méthodes sont à retenir :

1ère méthode:

```
Gosub Data      Appel à un sous-prog juste avant les
                data. L'adresse de la ligne suivante
                (i.e. des data) est empilée sur rstk
@d  $....       Les  data elles-mêmes
*Data           Label où doit commencer le sous-prog
                (qui n'en est pas un)
c=rstk          On désempile la rstk. L'adresse des
                data va dans C champ A.
```

2ème méthode:

```
A=PC            A contient l'adresse de la ligne
                suivante
GoinC A         C contient l'offset aux data (cette
                instruction est propre à
                ASM-Flash Nouvelle version)
A=A+C A         On additionne les deux. A(A)
                contient l'adresse des data.
...
$...            Les data eux-mêmes
```

La première méthode est légèrement plus courte (3.5 octets) et présente l'avantage de n'utiliser qu'un seul registre. La seconde est intéressante car elle permet de placer les data n'importe où dans le code (c'est particulièrement utile pour les librairies auto-modifiantes).

Les Data sont en fait placées dans le code telles quelles à la compilation. Il faut donc qu'elles soient déjà retournées. Pour les grobs (les data la plupart du temps en sont), cela n'a aucune importance car ils sont traités par la machine quartet par quartet (le saturn retourne par blocs). Les data peuvent être entrés sous deux formes :

```
    $+data sous forme héxa
ou  ø+---- ---- ----- ascii
```

Enfin, les data rendent souvent impossible le désassemblage d'un code par un désassembleur car celui-ci essaye essaie de les interpréter comme des instructions de LM, ce qui donne n'importe quoi. L'instruction RTNSXM, codée par 00 apparait par exemple fort fréquemment au désassemblage. Certains désassembleurs toutefois comme UNASS-1.2 reconnaissent la séquence Gosub ... C=RSTK et en conséquence laissent intacts ce qui se trouve entre.

Arthur Ripoll (590)

VERIFICATION DES ARGUMENTS DANS UNE LIBRAIRIE

Avant d'exécuter un programme en système-rpl ou en assembleur, il vaut mieux généralement vérifier que les arguments présents sur la pile sont bien du bon type, si l'on ne veut pas risquer de perdre le contenu de sa mémoire.

L'utilisateur peut créer un programme pour cette vérification (comme CHK dans Voyage au centre de la HP). Mais il dispose également de routines en ROM prévues à cet effet (je ne parlerai dans cet article que des adresses prévues pour les programme au sein d'une librairie, mais il existe des adresses équivalentes pour les programmes RPL).

Les avantages sont :

- Elles sont dans la Rom et ne prennent donc pas de mémoire.
- L'erreur créée affichera le nom du XLIB faisant erreur.
- L'exécution du programme peut être différente selon les objets sur la pile.
- Une sauvegarde de la pile est effectuée dans LASTARG et LASTSTACK.

L'inconvénient :

Le nombre maximal d'argument vérifié est 5 (taille d'un entier-système) voire moins pour certains objets "rares".

Voici les adresses :

Vérification uniquement du nombre

```
LCHK0    # 18A1E h
LCHK1    # 18AA5 h
LCHK2    # 18A80 h
LCHK3    # 18A5B h
LCHK4    # 18B92 h
LCHK5    # 18B6D h
```

Vérification du nombre et des types

```
LCHKT1   # 18ECE h
LCHKT2   # 18EDF h
LCHKT3   # 18EF0 h
LCHKT4   # 18F01 h
LCHKT5   # 18F12 h
```

Les types des arguments contrôlés par les adresses de la seconde colonne sont passés sous forme d'entiers-système selon le tableau suivant :

Objets de type courant

```
0  Objet qcq      | 8  Programme
1  Réel           | 9  Algébrique
2  Complexe       | A  Alg  ou nombre
3  Chaîne         | B  Entier binaire
4  Tableau        | C  Graphique
5  Liste          | D  Objet taggé
6  Nom Global     | E  Unité
7  Nom Local      |
```

Objets "rares" codés sur 2 quartets

```
0F   Nom XLIB       | 8F  Librairie
1F   Entier-système | 9F  Backup
2F   Répertoire     | AF  Lib Data
3F   Réel long      | BF  Réservé 1
4F   Complexe long  | CF     "    2
5F   Tableau chaîné | DF     "    3
6F   Caractère      | EF     "    4
7F   Code           |
```

Prenons un exemple pour clarifier les choses :

```
PRG
 $ 18ECE
   00033          Si 2 chaînes ...
    +                ... alors +
   00010          Si 1 réel et un objet qcq
   PRG                ... alors
    DROP                    DROP
    LN                      LN
   END
 END
```

C'est un sorte de stucture `SWITCH ... CASE ....`

Les fonctions de RPL classiques utilisent ces adresses pour vérifier les arguments sur la pile et utiliser les points d'entrée adéquats. Voici pour exemple comment se décompose la fonction `SQ` :

```
PRG
 $ 18ECE
   00001
    $ 1B47B
   00002
    $ 1B48F
   00004
    $ 36435
   0000A
    $ 54F9A
   0000E
    $ 0F913
 END
```

Pour "démonter" ainsi une fonction RPL on peut utiliser l'adresse `# 54AFh` avec la fonction sur la pile.On voit déjà les horizons ouverts par cette étude des fonctions User-RPL : la découverte de nouvelles adresses plus précises, plus rapides (mais aussi plus dangeureuses). Il suffit de vérifier les arguments une fois pour toutes au début du programme, puis on peut employer les adresses trouvées par la méthode ci-dessus.

Autre application : la non-sauvegarde d'arguments. La pile sauvegardée par les adresses est employée lors des interruptions causées par l'appui sur `[ON]`. Ainsi à ceux que cela dérange de trouver 0 sur la pile en ayant appuyé sur `[ON]` dans une structure `0 WAIT DROP`, je conseille de remplacer `WAIT` par `$ 1A738`.

De même, on pourra remplacer `INPUT` par `$ 43395` (arguments = 2 chaînes) ou par `$ 433CC` (arguments = 1 chaîne et une liste).

Arthur Ripoll (590)

DE CHAINE EN CHAINE

Acte II

On ne le dira jamais assez que notre journal JPC ne peut être attractif que par la diversité des sujets abordés et donc par la participation du plus grand nombre de lecteurs et quelque soient les niveaux et les langages de programmation. Ce qui compte avant tout c'est de savoir qu'il existe des besoins et des centres d'intérêts qui sont matière à programmation. Aussi je suis très reconnaissant envers Robert Amram qui nous a révélé une utilisation assez inattendue de sa HP48. Cela m'a rappelé, qu'à l'âge d'or du HP-71, un très bon programmeur finlandais (Tapani Tarvaïnen) avait réalisé un Lex pour JPC Rom : MAPLEX. Ce Lex fait la même chose que le programme ZMC de Robert pour les chaînes et traite même les fichiers texte du HP-71.

Aussi j'ai pensé que cela pourrait peut-être rendre service d'adapter à la HP48 le Lex pour HP-71 et pour les raisons suivantes :

Le programme ZMC de Robert fait usage de REPLACE de la *Tool Library* de James Donnelly. Or un certains nombre de commandes de celle-ci ont été intégrées dans la Rom des HP48 G/GX, ce qui pour les possesseurs de ces dernières leur fera peut-être renoncer à ce produit commercial d'autant qu'il leur faudrait sacrifier 9392 octets de Ram. La commande SREPL de la DEVLIB peut s'utiliser aussi à la place de REPLACE. Mais ceux qui possèdent une HP48S ou G et économisent leurs 32K de Ram, ou n'utilisent pas les autres commandes, se contenteraient d'une solution plus autonome.

Voici donc MAP pour HP48 et ses avantages :

- Simplicité d'emploi :

niveau 3 la chaîne à traiter
niveau 2 la chaîne contenant la suite des caractères à remplacer
niveau 1 la chaîne contenant la suite des caractères de remplacement

MAP → la chaîne transformée

- Economique en mémoire :
Ce programme n'occupe que 148.5 octets et utilise des chaînes de caractères au lieu de listes de chaînes de un caractère. Par exemple pour la liste des douze caractères accentués il faut 77 octets contre 17 pour la chaîne de ces douze caractères mis bout à bout.

- Rapide :

Le traitement d'une chaîne de plus de 3 Ko avec le remplacement de 12 caractères accentués se fait 50 fois plus vite avec MAP qu'avec le programme de Robert.

Exemples d'utilisations

En plus de l'usage que nous a révélé Robert, on peut par exemple :

- Convertir une chaîne de majuscules en minuscules :

niveau 3 la chaîne à traiter
niveau 2 la suite des majuscules dans une chaîne qui peut s'obtenir aussi par :

```
« "" 65 90 FOR I I CHR + NEXT »
```

niveau 1 la suite des minuscules dans une chaîne qui peut s'obtenir aussi par :

```
« "" 97 122 FOR I I CHR + NEXT »
```

MAP → la chaîne transformée

- Convertir un fichier texte en simple ASCII tapé sur un IBM PC et transféré dans une HP48 où il sera utilisé sous forme de chaîne.

niveau 3 la chaîne à traiter
niveau 2 la suite des caractères accentués du PC qui peut s'obtenir par :

```
« { 137 131 136 130 133 138 139 140 147
    151 150 135 } "" 1 12
    FOR I OVER I GET CHR + NEXT SWAP DROP »
```

ou tout simplement en tapant sur le PC :
"ëâêéàèïîôùûç"
Le fichier texte de cette seule ligne étant importé dans la HP48
niveau 1 la chaîne des caractères correspondants de la HP48:
"ëâêéàèïîôùûç"
qui peut s'obtenir par :

```
« { 235 226 234 233 224 232 239 238 244
    249 251 231 } "" 1 12
    FOR I OVER I GET CHR + NEXT SWAP DROP »
```

ou sur une HP48G/GX avec CHARS

MAP → la chaîne transformée

- Convertir un texte (chaîne) de votre HP48 en un fichier texte en simple ASCII à exporter sur un IBM PC.

Il suffit de procéder comme ci-dessus en permutant les chaînes des niveaux 1 et 2.

Tout ceci peut permettre de résoudre les problèmes des cas suivants :

Vous n'avez pas de PC mais vous avez la possibilité de recevoir (par l'intermédiaire d'un autre HPiste ou d'un serveur) la documentation issue d'un IBM PC et relative à des programmes ou des librairies, alors MAP est tout indiqué pour rendre cette documentation lisible sur votre HP48.

Vous n'avez toujours pas de PC mais vous avez concocté sur votre HP48 un article pour JPC. Un appel à MAP et vous voilà prêt à transmettre votre travail par deux voies possibles: un autre HPiste vous transférera cela sur disquette à envoyer à PPC, ou vous venez à une réunion du club et Jacques récupérera dans son HP95 votre prose qu'il pourra visualiser immédiatement.

Remarques:

- Mettre en correspondance les caractères HP48 et PC, donc respecter le même ordre.

- Les transferts HP48 <-> PC se feront par Kermit en mode ASCII avec translate code 1. Pour le transfert PC -> HP48 le fichier texte du PC commencera et se terminera par les délimiteurs de chaîne ". Lors d'une réception sur PC d'une chaîne HP48, il faudra éliminer la première ligne de l'en-tête Kermit ainsi que les délimiteurs de chaîne au début et en fin du fichier texte du PC.

- On pourra transposer les exemples ci-dessus à tout type de micro-ordinateur.

- On pourra aussi étendre le nombre de caractères à transposer d'une machine à l'autre, il suffit de trouver les correspondances et de respecter le même ordre dans les chaînes des niveaux 1 et 2.

- On peut très bien concevoir un programme faisant les conversions dans les deux sens. Pour cela il suffit de mettre les deux chaînes des niveaux 1 et 2 dans une liste:

```
{ "CCCC....CCCCC" "cccc....ccccc" }
```

Par exemple si "cccc....ccc" est la chaîne pour les caractères type PC et "CCCC...CCCC" la chaîne côté HP48, la liste étant stockée dans la variable 'PC48' on peut avoir le programme ←PC48→ :

```
«
  PC48 OBJ→ DROP
  "→PC: 0  →HP48: 1"
  "" INPUT OBJ→
  IF THEN SWAP END
  MAP
»
```

avec au niveau 1 : la chaîne à traiter

il suffit de répondre avec 0 pour la conversion dans le sens HP48→PC et 1 dans le sens PC→HP48.

Pour ceux qui n'utiliseraient MAP que pour cette seule application j'ai intégré l'ensemble de ce programme, les chaînes de caractères accentués et MAP, dans le programme ←PC48→ en System Rpl et assembleur. Pour cela le début du fichier source de MAP doit être modifié comme suit :

←PC48→
213.5 octets cksum # AC95h

```
::
 CK1NoBlame CK&DISPATCH1 str
 ::
  $ "\eb\e2\ea\e9\e0\e8\ef\ee\f4\f9\fb\e7"
  $ "\89\83\88\82\85\8a\8b\8c\93\97\96\87"
  $ "\8DPC: 0  \8DHP48: 1"
  NULL$ xINPUT xOBJ>
  %0> ?SWAP
  ROT TOTEMPOB
CODE ..........
              suite idem MAP
```

Les avantages : cela ne coûte que 213.5 octets et il n'y a qu'une chose à stocker.

Bien d'autres applications sont possibles. Et j'espère que des lecteurs nous feront part des leurs.

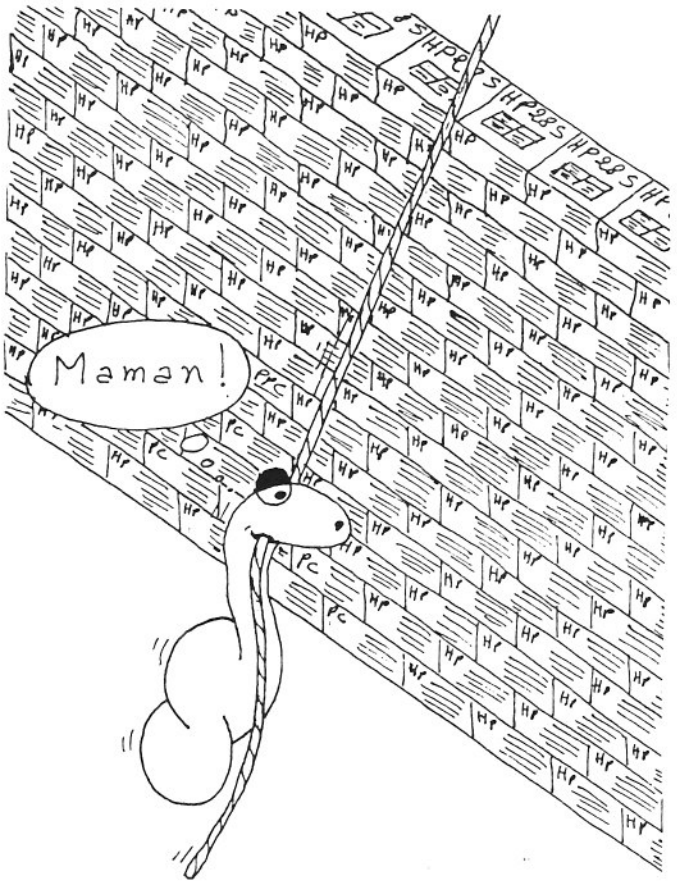
Voici pour les amateurs de programmation en assembleur le listing commenté du fichier source de MAP.

```
*****************************************************
*****************************************************
*                        MAP                        *
*          (cksum: # 33FAh  148.5 octets)           *
*     remplacement des caractères d'une chaîne      *
* Création: 10/02/94  Adaptation par Guy Toublanc   *
*                     du Lex Maplex (JPC 46 7/87)   *
*                     pour HP71 de Tapani Tarvainen *
*                                                   *
* Syntaxe: niveau 3 <- str0 : chaîne à traiter      *
*          niveau 2 <- str1 : chaîne des caractères *
*                             à remplacer           *
*          niveau 1 <- str2 : chaîne des caractères *
*                             de remplacement       *
*                             correspondant à str1  *
*                                                   *
*            MAP  →  la chaîne transformée          *
*****************************************************
*****************************************************
```

```
::
 CK3NOLASTWD CK&DISPATCH1 # 333
 ::
  ROT TOTEMPOB
CODE

*****************************************************
* Après vérification de la présence de 3 niveaux de *
* chaînes nous avons :                              *
*                      niveau 3 <- str2             *
*                      niveau 2 <- str1             *
*                      niveau 1 <- str0 recréée     *
* puis nous créons une chaine vecteur strv de 256   *
* caractères de valeurs 0 à 255 qui sera aussi une  *
* table de positionnement de caractères             *
*****************************************************

       gosbvl =SAVPTR        * sauve les pointeurs
       lc(5)  512            * longueur en quartets
       gosbvl =MAKE$N        * R0 := adr. strv créée
*                              D0 @ données de strv
       cd0ex
       r1=c                  * sauve adr. données strv
       d0=c                  * @ données strv
       c=0    b              * pour initialiser strv

init   dat0=c b              * boucle d'initialisation
       d0=d0+ 2              * de strv avec caractères
       c=c+1  b              * de valeurs 0 à 255
       gonc   init           * pour la future table

       gosbvl =GETPTR        * restitue les pointeurs
       gosbvl =GetStrLenStk  * C(A) := len str0
*                              D1 @ données de str0
       c=c-1  a              *
       goc    bad0           * refus si chaine nulle
       rstk=c                * sauve (len - 1) de str0
       cd1ex                 *
       rstk=c                * sauve adr. données str0
       gosbvl =GETPTR        * restitue les pointeurs
       d1=d1+ 5              * @ niveau 2 : str2
       gosbvl =GetStrLenStk  * même principes str0
       rstk=c                * sauve len de str2
       cd1ex                 *
       rstk=c                * sauve adr. données str2
       gosbvl =GETPTR        * restitue les pointeurs
       d1=d1+ 10             * @ niveau 3 : str1
       gosbvl =GetStrLenStk  * même principes str0
       b=c    a              * sauve len de str1
       c=rstk                *
       d0=c                  * @ données str2
       c=rstk                *
       a=c    a              * len de str2
       c=r1                  *
       d=c    a              * adresse table
       ?a=b   a              * len str2 = len str1 ?
       goyes  ok             *
bad2   c=rstk                * si non dépile et erreur
       c=rstk                *
bad0   lchex  00203          * pour bad argument value
       govlng =GPErrjmpC     * sortie avec ce message
ok     b=b-1  a              *
       goc    bad2           * erreur si str1 et str2
*                              de longueur nulle
tabl   c=0    a              * boucle de vectorisation
       c=dat1 b              * valeur carac. n de str1
       c=c+c  x              * transforme en offset
       c=c+d  a              * adresse position j
       cd1ex                 * @ position j dans table
       a=dat0 b              * caractère n de str2
       dat1=a b              * charge en position j
       cd1ex                 * restaure D1
       d1=d1+ 2              * @ caract. n + 1 de str1
       d0=d0+ 2              * @ caract. n + 1 de str2
decr   b=b-1  a              * compteur car. str1 et str2
       gonc   tabl           *

       c=rstk                *
       d1=c                  * @ données str0
       c=rstk                *
       a=c    a              * (len - 1) de str0

loop   c=0    a              * boucle de traduction
       c=dat1 b              * caractère n de str0
       c=c+c  x              * transforme en offset
       c=c+d  a              * adresse dans table
       d0=c                  * @ position j dans table
       c=dat0 b              * caract. de remplacement
*                              ou non suivant caract.
*                              n de str0
       dat1=c b              * charge dans str0
       d1=d1+ 2              * @ caract. n + 1 de str0
       a=a-1  a              * actualise compteur
       gonc   loop           *

       govlng =GETPTRLOOP    * restaure pointeurs et
*                              retour au RPL
ENDCODE
  UNROT2DROP                 * drop sur str1 et str2
 ;
;
```

Voir la liste des codes dans *Le coin des codes*.

Guy Toublanc (276)


Maman !

HP95 / HP100

LE NOUVEAU HP100

Depuis peu de temps, une nouvelle version du HP100 (équipée de 2 Mo de mémoire) est disponible. Comme toujours, lors de la sortie d'une nouvelle machine, de nombreux utilisateurs vont être confrontés au cruel dilème consistant à choisir quel modèle acquérir. Cet petit article a donc pour but de les aider dans leur choix.

Tout d'abord, parlons un peu chiffres. La nouvelle version est mise sur le marché au prix de 7000 F. C'est a dire le prix de la version 1 Mo il y a encore peu de temps. Une bonne nouvelle est que l'ancien modèle a vu son prix décroitre de façon importante, puisqu'il ne coute plus que 5000 F.

Pour les personnes ne possédant pas encore de HP100 (ni de HP95), cette différence de 2000F dans le prix des deux HP100 est assez importante a mon avis, pour se demander si il n'est pas plus intéressant de prendre une version à 1Mo, plus une carte Ram de 2 Mo. En effet, même si on doit dépenser un peu plus (puisque ces cartes coutent encore près de 3000F), on aura alors une capacité totale de 3Mo, soit un gain de 50% pour 1000F de plus (on peut noter d'ailleurs que le Méga-octet "HP" coute 2000F, alors qu'une carte ramenne le prix a 1500F/Mo, alors qu'il y a des composants en plus). De plus, il est toujours plus conseillé d'avoir le maximum de ses données sur cartes, car celles-ci sont moins sujettes aux pertes de données en cas de plantage complet de la machine.

Si vous possédez déjà un HP95, avec une carte Ram (ou mieux, une Carte SunDisk), le doute n'est pas permis, la version 1 Mo suffit.

Pour les personnes possédant déjà un HP100, je pense qu'il est plus intéressant, pour les raisons énoncées ci-dessus de reporter leur achat sur une carte SunDisk.

Cependant, la nouvelle version est plus conseillée pour les personnes désirant utiliser des cartes Rom dont le contenu ne peut pas être copié dans une carte Ram (que ce soit a cause d'une taille trop importante du contenu ou une protection anti-copie). Il en est de même pour les cartes d'interfaces (telles que les Modem/Fax) qui demandent souvent beaucoup de mémoire vive (en Ram System et disque) pour fonctionner. Et qui ne peuvent évidement pas être retirées en cas d'utilisation...

Jacques Belin (123)

UNE COMMANDE CACHEE DANS LE HP100

Lors de mes premières investigations concernant le HP100, j'ai très vite décelé la présence d'un fichier caché dans le directory `\BIN` du disque C:. Son nom : `PUSHKEYS.COM`.

Hewlett-Packard n'ayant divulgué aucune information sur ce fichier (ce programme, en fait), je me suis livré à une rapide analyse de son code, ce qui m'apporta quelques informations dont voici les plus importantes :

- Ce programme permet de placer des codes de touches dans le buffer du clavier, afin de simuler la pression de celles-ci et d'éviter ainsi des manipulations répétitives lorsque l'on entre dans certaines applications. (Pour cela, il n'y avait pas besoin de le désasssembler : son nom était suffisement explicite !).
- Il s'installe en Résident.
- Il nécessite l'entrée d'un fichier contenant le code des touches à stocker dans le buffer.

C'est ce dernier point qui m'apporta le plus de problème car l'analyse du code de traitement du fichier me montra rapidement que ce fichier avait une forme assez particulière, composée d'un en-tête contenant des offsets vers des données situées plus loin.

En fait, n'ayant pas le temps de d'approfondir mes recherches sur le format exact de ce fichiers, je m'étais résigné à abandonner le sujet. D'autant plus que même si je trouvais les informations recherchées, l'utilisation de la commande me paraissait trop lourde puisqu'elle me semblait nécessiter l'utilisation d'un éditeur binaire pour créer le fichier. Enfin, la dernière raison me faisant hésiter était le fait que la commande était non seulement non-documentée, mais aussi cachée. Etait-ce parce que c'était une commande abandonnée au dernier moment (suite à la découverte d'un bug ou une autre raison) et qu'il avait été plus simple pour les développeurs de la cacher (en armant un bit dans l'entrée du directory), plutôt que de l'enlever de la Rom, ce qui aurait pu poser des problèmes de positionnement de données dans celle-ci ?

Heureusement, il y a quelques jours, j'ai recu le journal du club étranger Prompt-HPGC. Dans celui-ci se trouvait un long article article de Raan

Young, un développeur de HP Corvallis bien connu des utilisateurs de HP-75, et de ceux qui ont lu le compte-rendu de la démonstration de la liaison pager/mobidem parue dans le JPC d'octobre 1992. Ce nouvel article décrivait, entre autres, le fonctionnement de cette fameuse commande ! En fait, si il ne faisait pas officiellement partie de l'équipe de développement du HP100LX, il en était suffisement proche pour y faire inclure ce programme, qu'il a écrit en collaboration avec Dave Suvak.

Heureusement que j'ai eu des problemes quant à l'utilisation de cette commande et que je n'ai pas pu en faire un article plus tôt commme je le voulais, car cela aurait fait un peu désordre qu'un utilisateur grille la politesse à l'auteur du programme !

Ceci dit, passons aux choses sérieuses : Comment utiliser ce programme ?

Tout d'abord, pour répondre à une des interrogations de Raan Young, je confirme que le programme est bien présent dans la version Française du HP100 !

Ensuite, comme je l'avais bien déduit, il permet bien de remplir le buffer du clavier avec le contenu d'un fichier. Or, et c'est là que j'ai failli, il faut savoir que le HP100 permet de créer des fichiers de Macros (c'est à dire des séquences de pressions de touches que l'on activera plus tard en appuyant sur une touche assignée à cette effet) à partir du System Manager. Et ce sont ces mêmes fichiers qui peuvent être utlisés par PUSHKEYS.

En fait, pour créer ce fichier, il est nécessaire d'utiliser, sur le HP100, l'utilitaire *Macro Systèmes* prévu à cet effet dans le System Manager (normalement, c'est la dernière icône de la première ligne de l'écran proncipal, il est aussi accessible en pressant [Ctrl][SUITE]). Cette fonction permet d'assigner dix macros aux touches [Fnt][F1] à [Fnt][F10], soit en éditant la fiche de macro, soit en mode "apprentissage" (voir la documentation de la machine pour plus de détails).

Comme sur le HP95, les macros sont stockées dans le fichier _DAT\SETUP.ENV. Cependant, il est possible sur le HP100 de les extraire pour les stocker dans un fichier séparé, qui pourra être bien sûr relu plus tard. Ceci permet d'avoir plusieurs séries d'assignations dans la machine, et de les changer suivant ses besoins, ce qui n'était pas possible sur le HP95.

Les macros sont stockées dans un fichier à l'aide du menu Fichier/Sauver sous... Raan Young conseille d'utiliser l'extension .PSH.

Une fois qu'un fichier de Macro est créé, nous pouvons entrer dans le vif du sujet.

Syntaxe et Utilisation de PUSHKEYS

Avant d'exécuter une Macro, il est tout d'abord nécessaire d'installer le programme en résident. Cela se fait en entrant l'ordre suivant :

```
d:\bin\pushkeys /i
```

Si tout va bien, le programme affichera le message Pushkeys Installed.

L'option /D permet de désinstaller le programme.

Ensuite, nous pouvons charger le fichier de Macros, par l'ordre :

D:\bin\pushkeys *nom_de_fichier*

Ceci chargera tout d'abord le contenu du fichier en mémoire. Les données prises en compte seront uniquement les codes de touches, et non les étiquettes (commentaires) mises dans la fiche de defintion de la touche. Il est à noter aussi que si plusieurs macros ont été définies dans le même fichier, elles seront toutes chargées et exécutées l'une après l'autre. Il n'y a pas de moyen de sélectionner une macro spécifique.

Ensuite, PUSHKEYS s'activera 18 fois par seconde et regardera si le buffer du clavier est vide. Si c'est le cas, il y placera une de celles qui est présente dans sa zone de mémoire. De son coté, si l'application en cours du HP100 demande une touche, il la retirera du buffer, permettant ainsi à PUSHKEYS d'en placer une autre.

Il est à noter que le programme commence son traitement dès qu'il est chargé. *Ceci veut dire que nous ne pourrons pas lancer* PUSHKEYS, *puis entrer ensuite à la main le programme destiné à bénéficier de ces macros.* Pour contrer ce problème, il faudra donc créer un petit fichier batch, contenant deux lignes : le PUSHKEYS, immédiatement suivi du nom de l'application. Pour plus de détails, reportez vous aux exemples ci-dessous.

Si vous désirez interrompre l'exécution de PUSHKEYS, il suffit de taper [CTRL][BREAK]. Ceci arrêtera définitivement l'exécution de la macro.

Pendant que le programme est en train d'exécuter une macro, il est tout à fait possible à l'utilisateur de taper sur le clavier. Ces données seront alors traitées immédiatement (c'est à dire entre les codes de touches mis dans le buffer par PUSHKEYS).

Afin de rendre cette possibilité plus pratique, une fonctionnalité a été ajoutée à PUSKEYS : Il est possible de spécifier un intervale de temps au cours duquel PUSHKEYS arrête d'alimenter le buffer du clavier, permettant ainsi de pouvoir entrer à loisir des données complémentaires. Cela se fait en insérant dans le texte de définition des macros des sequences des caractères ayant le format #ddd#, où ddd est un nombre d'unités de temps de 1/18e de secondes. Par exemple #60 arêtera l'exécution de la macro pendant un peu plus de 3 secondes.

Hormis le fait qu'il permet à l'utilisateur d'entrer des données complémentaires, ce système de délai est utile dans d'autres cas. Par exemple, lorsqu'une macro doit provoquer la sortie du System Manager, avant de continuer le traitement il faut mettre un petit délai d'attente, car le System Manager vide alors le contenu du buffer du clavier. Si PUSHKEYS stocke des touches à ce moment là, elles seront irrémédiablement perdues.

Il est aussi à noter que puisqu'il s'agit d'une possibilité propre à PUSHKEYS, il est fortement déconseilé d'utiliser des macros modifiées ainsi dans le gestionnaire de Macro du System Manager, car cela provoquerait des effets imprévisibles.

Exemples de création de Macro

Macro utilisable sous le System Manager

Supposons que nous voulions créer une macro permettant, après une réinitialisation de la machine, d'accéder immédiatement à un fichier, nommé TOTO.TXT, que nous voulons éditer.

Pour cela, nous devons créer la macro suivante (partant du moment où nous somme sur l'écran d'accueil, celui avec le Topcard Affiché) :

{Suite}	Accès à l'écran d'affichage des Icônes.
{Mémo}	Accès à l'éditeur
{Menu}	Affichage de la ligne de Menu.
f	Affichage du menu fichier
o	Fiche d'ouverture de fichier
toto.txt	Entrée du nom de fichier
{Entrée}	Ouverture du fichier.

Une fois que nous aurons entré les codes {Suite}{Mémo}{Menu}fototo.txt{Entrée}, sur la ligne prévue pour la touche [fnt][F1], dans l'éditeur de Macro, et avoir vérifié que d'autre touches n'avaient pas été définies, sauvons cette macro sous le nom TOTO.PSH.

Ensuite, vérifions que le fichier CONFIG.SYS contient bien la ligne SHELL=COMMAND.COM /P.

Enfin, créons un fichier AUTOEXEC.BAT tel que celui ci :

```
d:\bin\pushkeys /i
d:\bin\pushkeys c:\_dat\toto.psh
$sysmgr
```

Une fois le fichier stocké, chaque fois que l'on pressera [Ctrl][Alt][Del], nous accèderons directement au fichier TOTO.TXT, que l'on pourra éditer, avant de continuer notre travail.

Macro Utilisable sous DOS

Dans le même ordre d'idée, supposons que nous voulions garder une trace de la date et de l'heure des dernières réinitialisations (volontaires ou involontaires !) dans un fichier CRASH.TXT.

Pour cela il semblerait suffire de lancer les instructions :

```
date >>crash.txt
time >>crash.txt
```

Le problème est que ces deux commandes affichent des lignes telles que :

```
Current time is  18:42:25
Enter new time :
```

et attendent la pression de la touche [Enter] pour continuer.

Pour eviter cela, il y a deux solutions :

- Créer un fichier "générique" de macro ne contenant que la ligne {Entrée}, et utilisable de comme ceci :

```
d:\bin\pushkeys /i
d:\bin\pushkeys c:\_dat\enter.psh
date >>crash.txt
d:\bin\pushkeys c:\_dat\enter.psh
time >>crash.txt
echo . >>crash.txt        (une petite séparation)
d:\bin\pushkeys /d
```

- Ou mieux, créer une macro qui fait tout le traitement (les deux entrées servant respectivement à terminer l'entrée de la commande, puis à en sortir) :

```
date >>crash.txt{Entrée}{Entrée}
time >>crash.txt{Entrée}{Entrée}
echo . >>crash.txt{Entrée}
```

et appelable comme ceci :

```
d:\bin\pushkeys /i
d:\bin\pushkeys c:\_dat\crash.psh
d:\bin\pushkeys /d
```

Dans les deux cas, nous obtiendrons un fichier contenant ceci :

```
Current time is  Sun 27.01.1994
Enter new date (dd-mm-yy):
Current time is  18:42:25
Enter new time:
.
Current time is  Sun 27.01.1994
Enter new date (dd-mm-yy):
Current time is  18:37:25
Enter new time:
.
```

Evidement, il y a de nombreuses lignes inutiles, mais le problème vient des commandes DATE et TIME, pas de PUSHKEYS.

Utilisation de PUSKEYS sur le HP95

Comme me l'a indiqué l'examen de son code, PUSKEYS ne fait appel qu'à des appels systèmes tout à fait standard. Ce qui fait que les possesseurs des deux machines peuvent tout à fait l'utiliser sur le HP95. Malheureusement, cette machine ne permettant pas d'extraire des fichiers de Macros, il est nécessaire de les créer tout d'abord sur le HP100, puis les transférer sur HP95. Attention cependant aux points suivants :

- Ne tentez pas d'effectuer des opérations impossibles sur le HP95.
- Si vous créez les macros en mode "apprentissage", rappele vous que les menus du System Manager sont totalement différents sur les deux machines et que les macros ne stockent que les codes des touches pressées, et non leur effet. Par exemple, ce sont les manipulations des touches fléchées qui sont stockées, et non le fait que l'on pointe sur la fonction *Sauvegarde de fichier...*
- Si vous accédez aux menus en appuyant sur les lettres clefs, faites attention qu'elle correspondent bien entre les deux machines, surtout si vous utilisez deux machines configurées avec une langue de travail différente.

Jacques Belin (123)

RADOTAGES

Claude Shannon, un grand théoricien de l'Information, a postulé l'exsitence d'une loi de composition de la langue écrite, montrant que les mots ne sont composés que d'un nombre relativement limité de combinaisons de lettres, déterminant ainsi ses principales caractéristiques, dont sa sonorité.

Nous allons, dans cette article, essayer de voir si cette loi peut être vérifiée, à l'aide d'un petit programme tournant sur HP95.

Mais tout d'abord, définissons un petit algorithme permettant de manipuler un texte pour en tirer des mots de façon semi-aléatoire :

0- Choisir le premier caractère d'un mot présent dans un texte.
1- Noter le caractère pointé.
2- Parcourir le texte "en anneau" (en reprenant au début quand on arrive à la fin du texte), jusqu'à ce qu'on arrive sur une nouvelle occurence du caractère que l'on vient de noter.
3- Déplacer le pointeur d'un caractère vers la droite.
4- Si ce nouveau caractère est "légal" (une lettre), reprendre à partir de l'étape numéro 1.
5- Si ce caractère est un espace, un chiffre, une ponctuation ou tout autre caractère spécial, le "mot" que l'on vient de créer est terminé. On parcoure donc le texte jusqu'à ce qu'on pointe sur un nouveau caractère "légal", qui deviendra la première lettre d'un nouveau "mot". Puis nous reprenons l'exécution à partir de l'étape numéro 1.

Une fois cet algorithme défini, choisissons un texte quelconque : par exemple, tout simplement, le premier paragraphe de cet article (de *Claude Shannon* à *sonorité*).

Prenons maintenant un mot au hasard dans ce texte : par exemple, *principales* (avant dernière ligne).

Maintenant, exécutons l'algorithme :
- Notons la première lettre du mot : **P**.
- Parcourons le texte, à partir de cette lettre, afin de retrouver la même. Nous n'avons pas à aller loin, puisque elle se trouve dans le même mot, entre le *i* et le *a*.
- Décalons nous sur le caractère suivant. C'est une lettre, donc nous pouvons continuer le traitement.
- Notons la : **A**.
- Comme auparavant, parcourons le texte pour trouver une autre lettre *a*. Nous la trouvons en deuxième position du mot *caractéristiques*.

- Décalons nous sur le caractère suivant. C'est une lettre, donc nous pouvons continuer le traitement.
- Notons la : **R.**
- Recherchons maintenant une autre lettre *r*, qui se trouve dans le même mot, entre le *é* et le *i*.
- Décalons nous sur le caractère suivant. C'est une lettre, donc nous pouvons continuer le traitement.
- Notons la : **I.**
- Recherchons maintenant une autre lettre *i*, qui se trouve dans le même mot, entre le *t* et le *q*.
- Décalons nous sur le caractère suivant. C'est une lettre, donc nous pouvons continuer le traitement.
- Notons la : **Q.**
- Les derniers mots du paragraphe ne contenant pas de *q*, reprenons la recherche à partir du début du texte. Nous trouvons la lettre désirée dans le mot *que* suivant le mot *montrant*.
- Décalons nous sur le caractère suivant. C'est une lettre, donc nous pouvons continuer le traitement.
- Notons la : **U.**
- La prochaine occurence de la lettre *u* se trouve dans la mot *que* suivant (entre *composées* et *d'un*.
- Décalons nous sur le caractère suivant. C'est une lettre, donc nous pouvons continuer le traitement.
- Notons la : **E.**
- La lettre *e* suivante se trouve à la fin du mot *nombre*.
- Décalons nous sur le caractère suivant. C'est un espace, donc le traitement est terminé.

En mettant bout-à-bout toutes les lettres que nous vennons de trouver, nous obtenons le "mot" PARIQUE. Bien qu'il nous semble étranger au premier abord, puisqu'il ne fait pas partie de notre vocabulaire, il pourrait très bien s'agir d'un mot tout à fait "normal". En fait, il est très proche du mot *Barrique*, si l'on admet une petite faute d'orthographe !

L'explication de cela est que l'algorithme a créé le "mot" en utilisant des couples de lettres existant déjà dans les mots du texte original (*pa*, *ar*, *ri*, *iq*, *qu* et *ue*). En fait, l'algorithme ne crée rien, il ne fait qu'utiliser des couples existant dans le texte. Et puisque le couple *rp* n'y est pas présent (même si on peut le trouver facilement dans la langue Française), nous ne trouverons pas de mots comprenant ce couple dans les "mots" que le programme pourra trouver. De même, aucun mot du texte ne commençant par la lettre *v*, nous ne trouverons pas de "mot" commençant par cette lettre.

Un des effets intéressants de cette loi est que si nous prenons un texte de base écrit en Français et que nous obtenions des "mots" ayant une consonnance Française, le phénomêne devrait (je n'ai pas vérifié) être le même dans le cas d'autres langues, telles que l'Anglais ou en Allemand. Puisque nous parlons de langues, il faut aussi noter que nous sommes un peu désavantagés par rapport à l'Anglais à cause de la présence des accents dans notre typographie. Il serait intéressant de voir si la loi que nous étudions aujourd'hui serait modifiée suivant le fait que nous laissions ou retirions ces accents sur les lettres.

D'autre part, vous pouvez noter que l'exemple donné est un peu "pipé", puisque parmi le mots du texte initial se trouve "Claude Shannon". Les noms propres ne possédant pas obligatoirement la même structure que les mots communs, il est probable que ce type de mots fausse un peu les résultats. De plus l'un de ces éléments, "Shannon", est un nom Anglais, qui possède un couple de lettres ("sh") plus commun dans la langue Anglaise que dans notre vocabulaire.

Afin d'étudier plus en détail cet algorithme, j'ai écrit le petit programme Swift!Basic suivant :

```
SET LEN 120
* Création d'un tableau de 1000 lignes
*  de 120 caractères maxi.
DIM A$(1000)
DEF STR B
SET LEN 64
DEF STR R
DEF STR S
* R$ = Chaine contenant les caractères acceptés
R$="abcdefghijklmnopqrstuvwxyzàaéèêëîïû"
* Demande nom du fichier et position de départ
INPUT "Fichier a traiter : ";F$
INPUT "Octet de depart : ";Po
P=Po              * P = position pointeur sur ligne
OPEN "i",1,F$     * Ouverture fichier
I=1               * I = numéro de ligne
* LECTURE FICHIER DANS BUFFER
WHILE NOT EOF(1)
  INPUT #1;B$
  IF B$<>"" THEN
    B$=LOWER$(B$) * Conversion chaîne en minuscules
    B$=B$+" "     * ajoute un espace en fin de ligne
    A$(I)=B$
    I=I+1
  ENDIF
WEND
N=I-1         * N = Nombre de lignes du tableau
I=1
CLOSE 1       * fermeture fichier source
S$=""         * On commence par une chaîne vide
*  ANALYSE
REPEAT
  C$=MID$(A$(I),P,1)       * C$ = Caractère pointé
  IF INSTR(R$,C$)<>0 THEN  * si caractère "légal"
    S$=S$+C$               * on l'ajoute au "mot"
    REPEAT
      GOSUB Nextchar       * et on passe au suivant
    UNTIL C$=MID$(A$(I),P,1) * Jusquà ce qu'on trouve
                             * le même
    GOSUB Nextchar         * et on passe au suivant
```

```
  ELSE                      * SI CARACTERE NON LEGAL
    PRINT S$                * affichage "mot" trouvé
    S$=""                   * réinitialisation du "mot"
    REPEAT
      GOSUB Nextchar        * on passe les autres
                            * caractères non légaux
    UNTIL INSTR(R$,MID$(A$(I),P,1))<>0
  ENDIF
 * on s'arrete si on est revenu à la position initiale
 * ou si une touche est pressée
UNTIL (I=1 AND P=Po) OR LEN(INKEY$(0))<>0
PRINT "   Fin d'exécution"
END

*  Routine d'incrémentation du pointeur
»Nextchar
  P=P+1
  IF P>LEN(A$(I)) THEN
    P=1
    I=I+1
    IF I>N THEN
      I=1
    ENDIF
  ENDIF
RETURN
END
```

Si vous regardez bien ce listing, vous remarquerez que j'ai stocké le texte du fichier en mémoire, afin d'accélérer le traitement. Cependant, vous remarquerez aussi que j'ai stocké les différentes lignes dans un tableau de chaînes. Il aurait été plus intéressant de concaténer toutes ces chaines dans une seule, ce qui aurait simplifié le programme tout en l'accélérant, mais j'ai dû procéder ainsi car le Swift!Basic n'accepte que des chaines ne dépassant pas 256 caractères.

Pour utiliser ce programme, il suffit d'entrer le nom du fichier contenant le texte à analyser, puis d'entrer un nombre correspondant à la position du caractère de départ (sur la première ligne du fichier uniquement, pour cette version du programme).

Il est préférable de choisir un texte comportant peu de mots de moins de cinq lettres, car cela permettra d'éviter d'avoir trop de "mots" de deux ou trois caractères. En fait, un des autres effets intéressants de cet algorithme est aussi que, statistiquement, la distribution en nombre de lettres des "mots" trouvés est la même que pour le texte original...

Il est aussi préférable de choisir un texte assez long, ce qui évitera de "boucler" trop rapidement et permettra de trouver un plus grand nombre de "mots".

Notez aussi que le fait qu'il n'y ait qu'une seule occurence d'une lettre dans le texte ne bloquera pas le programme, puisque si cette lettre est trouvée, le programme exécutera une boucle complete du texte pour revenir sur cette lettre, puis se déplacera sur le caractère suivant.

Une fois le programme lancé, il nous suffira de voir sur l'écran les différents "mots" trouvés.

Par exemple, si nous entrons dans un fichier le texte du premier paragraphe de cet article, puis choisirons comme départ la première lettre du mot *théoricien*. Soit le 26ème caractère (espaces et signes de ponctuation compris) de la première ligne.

L'exécution du programme nous donnera les "mots" suivants :

tisiontsombis
dérice
lant
qunomincacl
iostitrelittes
ctithan
d
ue
mon
ntét
alanormpome
cis
defoncos
mpra
shé
lomot
conai
s
parique
reripoite
lainté
cin
a
le
sé
detrma
pongrie
lécos
qud

En examinant la liste, on peut tout de suite éliminer certaines choses telles que *qunomincacl*, *mpra* ou *detrma*, ainsi que les "mots" de une ou deux lettres (bien que *le* et *mon* soient tout à fait admissibles).

D'autres "mots" semblent faire l'affaire au premier coup d'oeil, mais doivent être écartés parce qu'il manque une lettre. C'est le cas, par exemple, du "mot"

alanormpome, pour lequel il aurait peut être menqué un "o" entre le "m" et le "p".

Certains "mots" semblent bizarres, mais tout à fait possibles : *reripoite*, *pongrie*, ou *iostitrelittes* me faisant penser à un terme du milieu médical.

Enfin, certains autres (*dérice* et *parique*, déjà cité) semblent tout à fait normaux, puisque nous avons dans notre vocabulaire des mots tels que *dérive*. Et *lainté*, qui ferait un excellent participe passé d'un hypothétique verbe "lainter".

Cependant, on peut aussi s'apercevoir que certains "mots", comportant des couples pourtant souvent utilisés sont souvent éliminés. Par exemple, le couple "qu" pose souvent un problème, car il est normalement toujours suivi d'une voyelle. Or, les couples composé de la lettre "u" suivie d'une autre voyelle ne sont pas si fréquent que cela, et la plupart de ceux qui existent sont justement ceux précédés de la lettre "q"...

En tous cas, la plupart des "mots" trouvés (hormis *defoncos*, peut-être) ont tout à fait des consonnances tirées de la langue langue Française. Ce qui prouverait que Shannon avait vu juste...

Si cet exemple, a été relativement fécond en "mots" intéressants, il faut reconnaitre aussi que certains autres apportent beuacoup plus de déchets. Voire ne rapportent rien d'intéressant. Cependant, un simple programme tirant des lettres au hasard n'a aucune chance de trouver autant de "mots" que notre programme (*Si tous les chimpanzés du monde tapaient à la machine...*). En fait, c'est probablement un programme de ce type qui a dû être utilisé par Renault pour trouver des noms tels que *Safrane* ou autres *Xantia*...

Mon programme, tel que je l'ai écrit, est plus une démonstration de l'algorithme qu'un programme complet. Il présente donc certains défauts, et peut sans problème être amélioré en ajoutant les traitements et contrôles suivants :

- Possibilité de partir à partir de n'importe mot du fichier source. Et non seulement ceux de la première ligne comme cela se fait actuellement.
- Contrôle du départ sur la première lettre d'un mot. En effet, si nous voulons respecter totalement l'esprit de la loi de composition des mots, le faudrait pas que le premier "mot" trouvé commence par une lettre choisie aléatoirement, qui peut être une de celles rarement utilisées dans notre langue.
- Stockage de la liste des "mots" trouvés dans un fichier.
- Non-stockage dans ce fichier des "mots" de moins de quatre caractères.
- Contrôle dans la liste des "mots" déjà trouvés, de façon à ce que le programme s'arrête dès qu'il boucle.
- Dès qu'il commence à boucler, reprise du traitement avec une autre valeur de départ. Ceci afin de trouver la liste exhaustive des "mots" existants.
- ...

N'ayant pas pour projet actuel de continuer à travailler sur ce sujet, j'espère que certains d'entre vous seront suffisament intéressés pour continuer ce travail là où je l'ai laissé et nous en faire profiter...

D'autre part, qui sera capable de trouver l'algorithme du programme inverse, reconstituant le paragraphe initial en partant de la liste des "mots" trouvés ?

Note finale : L'idée de cet article (et son titre, car le programme a une certaine propension à boucler en répétant la même chose !) m'est venue en lisant l'excellent (et souvent iconoclaste) *La Théorie du Bordel Ambiant*, écrit par Roland Moreno (l'inventeur de la carte a puce). Bien que ce livre ne soit pas un traité d'informatique et d'électronique (c'est plutôt une auto-biographie, où il expose ses méthodes de pensées pouvant aboutir à la Création), il explique ici ses propres analyses concernant le sujet qui nous préocupe aujourd'hui. Par exemple, il a remarqué qu'en ne constituant son fichier source qu'avec des noms de fleurs, on obtient des "mots" ressemblant étrangement à ces choses... Si vous voulez aller plus loin dans ce sujet, une lecture s'impose. Il est disponible en "Livre de Poche". Enfin, pour la petite histoire, on peut aussi apprendre dans ce livre que Roland Moreno a appris à programmer sur une HP55...

Jacques Belin (123)

LE COIN DES CODES

La compilation de certains programmes, tels ceux écrits en assembleur, nécessitent souvent un logiciel que ne possèdent pas tous nos lecteurs. Le *Coin des Codes* permet de résoudre ce problème.

Note importante :

Même si la présentation des listings est identique, le traitement de ceux-ci est différente suivant le programme d'entrée et la machine de destination. Chaque listing est prévu pour être entré sur sa machine de destination. Par exemple, ne tentez pas d'entrer un programme HP48 dans un fichier MS-DOS à l'aide du programme MAKEDOS. Vous obtiendrez un fichier que vous ne pourrez pas transférer dans la HP48.

Programmes HP48

Par rapport aux méthodes habituelles sur HP48, notre méthode effectuant un calcul local du checksum en fin de chaque ligne permet de faciliter la recherche d'erreurs, par rapport à une même recherche dans une chaîne de plusieurs centaines d'octets.

Par mesure de sécurité sauvegardez vos programmes et fichiers, éventuellement verrouillez vos cartes.

Tapez les deux programmes ASSCOD48 et INPUT48 avec la plus grande attention car leur mauvais fonctionnement peut entraîner un désordre fatal pour les objets contenus dans votre machine. Newline (←) s'obtient sur HP-48 par [flèche bleue] puis [.].

ASSCOD48
996.5 octets cksum # 5D1Bh

```
« RCLF HEX 64 STWS 1 SF "" 'tmpcod' STO
"nombre d'octets                          @ + NEWLINE
                                          @ NEWLINE
" 17 INPUT48 1 CF STR→ 2 * 16
DUP2 / IP 3 ROLLD MOD DUP2 0 > +
3 ROLLD 1 + 4 ROLL # 0h DUP → n r f s o « 1 SWAP
FOR i
 DO "ligne " i 1 - R→B # 1000h + →STR 4 6 SUB + DUP
  "                                       @ NEWLINE
  chaine                                  @ + NEWLINE
  " + "---- ---- ---- ----" n i <
  IF THEN r DUP 4 / IP + SWAP OVER 1 SWAP OVER -
      SUB SWAP 18 +
  ELSE 38 END 'o' STO +
  "                                       @ NEWLINE
  " + 1 FS?C
  IF THEN ROT SWAP CLLCD 1 DISP HALT
  ELSE o INPUT48 END DUP
  WHILE DUP " " POS DUP
  REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 + 25 SUB
   IF DUP " " == THEN DROP ELSE + END
  END DROP 0 OVER SIZE 1 SWAP
  FOR j OVER j DUP SUB NUM j * + NEXT
  s + DUP # FFFh AND
  "                                       @ NEWLINE
  somme de controle                       @ + NEWLINE
   ---                                    @ + NEWLINE
  #" 6 ROLL SWAP + 34 INPUT48 STR→ ==
  IF THEN SWAP DROP 1
  ELSE DROP2 1000 1 BEEP 1 SF 0 END
 UNTIL END 's' STO
 WHILE DUP " " POS DUP
 REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 + 19 SUB +
 END DROP 'tmpcod' DUP RCL ROT + SWAP STO
NEXT f STOF » tmpcod
"GROB 8 " OVER SIZE 2 / " " + + SWAP   @@ ou ASC→
+ STR→ # 4017h SYSEVAL # 56B6h         @@ (JPC-79
SYSEVAL DROP NEWOB                     @@ page 11)
"fin" CLLCD 1 DISP
»
```

INPUT48
412.5 octets cksum # 15C7h

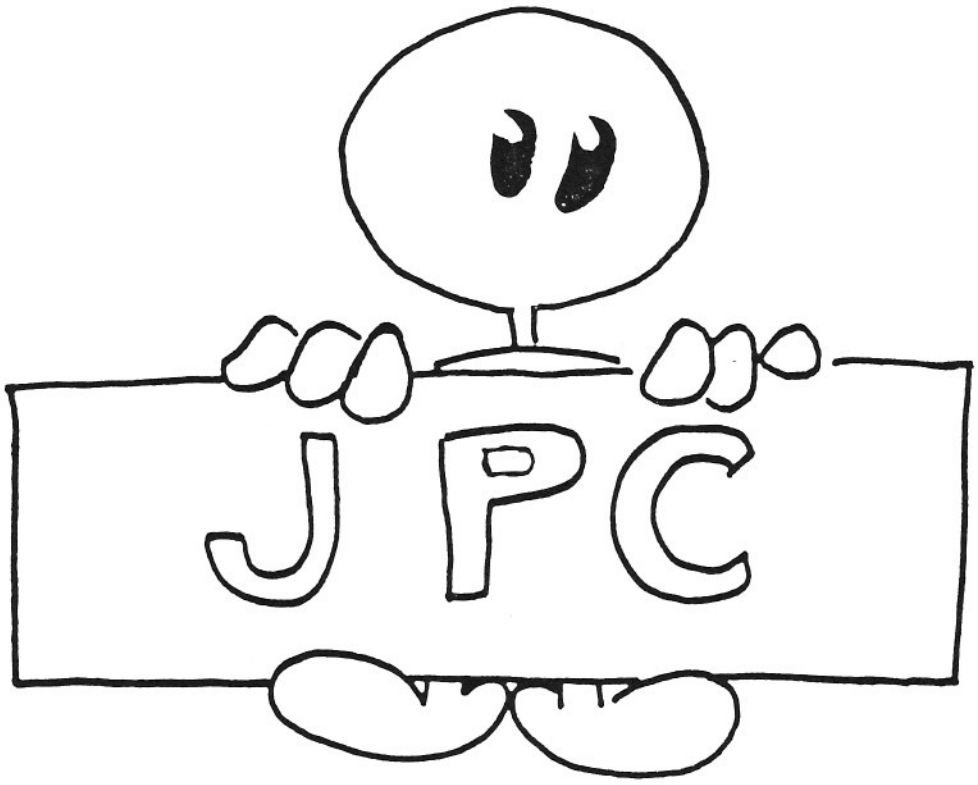
```
« SWAP 1
WHILE OVER CLLCD 1 DISP
REPEAT
 DO UNTIL KEY END
 IF DUP 51 == THEN DROP 0
 ELSE IF DUP 55 == THEN DROP 1 OVER SIZE 1 - SUB
  ELSE
   CASE DUP 17 < THEN 54 + END DUP 66 < THEN 7 - END
     DUP 76 < THEN 20 - END DUP 86 < THEN 33 - END
     92 == THEN 48 END 46 END CHR + DUP SIZE 3 PICK
    - 2 + 5 MOD NOT 1 FC? AND IF THEN " " + END
  END 1 END END SWAP 60 SUB »
```

Donc à partir de maintenant pour tout assemblage de chaîne de codes procédez de manière suivante:

1- lancez le programme ASSCOD48
2- donnez le nombre d'octets (1/2 octet compris) puis validez avec ENTER.
3- tapez chaque ligne de codes, correspondant au numéro de ligne à 3 chiffres, sans les espaces et validez.
4- tapez la somme de contrôle et validez. S'il y a erreur la ligne de codes sera demandée à nouveau après émission d'un BEEP. L'appui sur EDIT fera apparaître la ligne des codes qui pourra être corrigée. Alors relancez avec CONT.
5- stockez le programme assemblé dans la variable donnée en tête.
6- si tout s'est bien déroulé vous pouvez purger tmpcod qui contient la chaîne de codes.

```
MAP                          (HP48)
# 33FAh               148.5 octets

      0123 4567 89AB CDEF    sm

000: D9D2 086A 812B F811    DB9
001: 1920 3330 0D9D 2059    A98
002: 2307 5660 CCD2 0DE0    953
003: 008F B976 0340 0200    49A
004: 8FD7 B501 3610 9134    078
005: AE21 4C16 1B66 56F8    EA0
006: F2D7 608F 4BFF 2CE4    F09
007: B406 1370 68F2 D760    C53
008: 1748 F4BF F206 1370    984
009: 68F2 D760 1798 F4BF    8F2
00A: F2D5 0713 407D A119    610
00B: D78A 0410 7073 4302    15B
00C: 008D 04F0 1CD4 BED2    124
00D: 14FA 36CB 1371 4A14    E6C
00E: 9137 1711 61CD 51E0    BE5
00F: 7135 07DA D214 FA36    A65
010: CB13 414E 14D1 71CC    8D8
011: 57E8 D341 50A2 116B    5F9
012: 2130 B213 0            EFB
```

```
←PC48→                       (HP48)
# AC95h               213.5 octets

      0123 4567 89AB CDEF    sm

000: D9D2 0D29 512B F81D    EBA
001: 0040 D9D2 0C2A 20D1    C72
002: 000B E2EA E9E0 E8EF    DE4
003: EEE4 F9FB F7EC 2A20    DBD
004: D100 0983 8882 858A    B17
005: 8B8C 8397 9697 8C2A    9BC
006: 2052 000D 8053 4A30    5E7
007: 2030 202D 8840 5438    214
008: 3A30 213F D550 AC42    FF7
009: 2B7F C199 7A2B 1F26    E8C
00A: 5923 0756 60CC D20D    CFD
00B: E000 8FB9 7603 4002    8F7
00C: 008F D7B5 0136 1091    532
00D: 34AE 214C 161B 6656    27D
00E: F8F2 D760 8F4B FF2C    2ED
00F: E4B4 0613 7068 F2D7    0CA
010: 6017 48F4 BFF2 0613    E88
011: 7068 F2D7 6017 98F4    C83
012: BFF2 D507 1340 7DA1    A1A
013: 19D7 8A04 1070 7343    5FA
014: 0200 8D04 F01C D4BE    58D
015: D214 FA36 CB13 714A    394
016: 1491 3717 1161 CD51    07F
017: E071 3507 DAD2 14FA    F83
018: 36CB 1341 4E14 D171    C8A
019: CC57 E8D3 4150 A211    928
01A: 6B21 30B2 130          680
```



Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur de JPC, et son siège social est au 56, rue Jean-Jacques Rousseau, 75001 Paris.

PPC Paris est le représentant Français de **HEX** (**H**andhelds **E**uropean Clubs **EX**change), la fédération des principaux clubs Européens d'utilisateurs de calculateurs HP.

La maquette de ce numéro a été préparée et réalisée par Jacques Belin et Asdin Aoufi.

Les dessins sont de Jean-Jacques Dhénin et Paul Courbis.

Directeur de la publication : Jacques Belin
Numéro ISSN : 0762 - 381X

Veuillez adresser toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01.

Imprimé par Paris Copie, 4 rue Linné, 75005 Paris