JPC

MAI 1994

NUMERO 96

Le numéro 40 F

A PROPOS DU CLUB

HP28

HP48

HP95 / HP100

EDITORIAL

Puisque l'actualité est plutôt calme ce mois ci (en attendant la conférence d'Amsterdam), revenons sur le précédent JPC. Nous avons le regret de vous annoncer que nous avons publié deux informations totalement erronées. El le pire c'est que cela était intentionnel ! Les anciens adhérents auront reconnus ici une mauvaise habitude revenant tous les ans, aux environs du mois d'avril... Que les nouveaux adhérents nous en excusent ! Si certains autres ne s'y sont pas laissé prendre, qu'ils ne s'inquièttent pas, nous essaieront de faire mieux l'année prochaine ! Cependant, ces deux plaisanteries sont basées sur deux informations réelles entendues quelques jours auparavant. Tout d'abord, si il n'est pas possible (à notre connaissance !) de tripler la vitesse de la HP48, il serait possible de le faire pour le HP100 ! Cependant, cette modification du quartz causerait quelques problèmes de compatibilité avec certains autres circuits (Rom, I/O...). Quelques personnes travaillent actuellement sur le sujet. Quant au disque dur externe au format PCMCIA, il n'en existe toujours pas à notre connaissance, et surtout pas à 1500F. Par contre il existerait un lecteur de disquette pouvant se brancher sur un port PCMCIA. Nous n'avons pas encore pu avoir des renseignements fiables sur ces deux informations, mais nous vous tiendrons bien sûr au courant.

A part ça, vous trouverez dans ce numéro la suite de notre travail concernant nos récupérations de programmes pour les HP48 et les palmtops. Il nous reste encore quelques programmes à distribuer, mais il vous faudra probablement patienter quelques mois pour ayons suffisement de matériel pour remplir des nouvelles disquettes.

En attendant, Bonne lecture !


JPC est il
un Journal Peu
Credible ?
...


Donnez-moi mon
JPC sinon je
frappe !

HP28

ALLEGEMENTS EN DOUCEUR

A l'occasion de la publication par Jean-François Garnier d'une liste de points d'entrée pour HP28S (JPC 87 et 88) notre ami nous donnait quelques exemples simples d'utilisation de ces points d'entrée par SYSEVAL interposé donc en User Rpl. Ce procédé qui permet d'étendre les possibilités de la machine tout en restant simple de réalisation a l'inconvénient de consommer des octets car un entier binaire courant se code sur 26 quartets soit 13 octets contre 5 quartets donc 2,5 octets en System Rpl pour appeler l'objet correspondant au point d'entrée. Pour des HP28S cette perte de 10,5 octets n'est pas toujours à négliger. D'autre part si un programme en User Rpl débute par un SYSEVAL cela peut-être dangereux à l'usage car il n'y aura pas de vérification du nombre et du type des arguments comme Jean-François nous le rappelait.

Je vous propose donc d'abord de traduire en System Rpl les petits programmes exemples de Jean-François constituant le début d'une boite à outils.

Boite à outils légers

Pour la syntaxe veuillez revoir l'article cité du JPC 87.

SYSRCL 43.5 octets

```
« # 71BDh SYSEVAL # 4152h SYSEVAL DROP »
```

en System Rpl : 15 octets cksum # AFC2h

```
::  CK1NOLASTWD Dispatch1 ELEVEN SECONDELCOMP ;
```

FRE 46 octets

```
« # 48C1h SYSEVAL # C610h SYSEVAL 2 / »
```

en System Rpl : 15 octets cksum # 7B76h

```
:: MEM UNCOERCE %2 %/ ;
```

CRNAME 25,5 octets

```
« # 44EDh SYSEVAL »
```

en System Rpl : 15 octets cksum # 6631h

```
:: CK1NOLASTWD Dispatch1 THREE $>ID ;
```

B→SB 25.5 octets

```
« # 44EDh SYSEVAL »
```

en System Rpl : 15 octets cksum # 31A2h

```
:: CK1NOLASTWD Dispatch1 ELEVEN HXS>#  ;
```

SB→B 28 octets

```
« # C610h SYSEVAL R→B »
```

en System Rpl : 12.5 octets cksum # 858h

```
:: CK1NOLASTWD UNCOERCE %>#  ;
```

Attention : il n'y que la vérification d'un niveau et non du type d'argument. Il ne semble pas y avoir de solution en System Rpl HP28 pour la vérification d'un argument de type entier système.

COM→ 41 octets

```
« # 3F41h SYSEVAL # C610h SYSEVAL »
```

en System Rpl : 12,5 octets cksum # BDE1h

```
:: CK1NOLASTWD INNERCOMP UNCOERCE ;
```

Attention à donner en argument un objet composé car il n'y a que la vérification de la présence d'un objet. Il faudrait tester avec les différents types d'objets composés pour faire une vérification complète.

→PRG 41 octets

```
« # C53Bh SYSEVAL # 3EEBh SYSEVAL »
```

en System Rpl : 40 octets cksum # 2158h

```
:: CK1NOLASTWD Dispatch1 ONE
 ::
  xDEPTH OVER %>
  ITE :: COERCE ::N ;
      ERRTooFewArg
 ;
;
```

Si le gain d'octets est négligeable l'avantage de ce programme en System Rpl est qu'il vérifie à la fois, la nature de l'argument du niveau 1 : un réel, et le nombre d'objets qui doivent être au-dessus.

PARSE 41 octets

```
« # E4F7h SYSEVAL # C9F4h SYSEVAL »
```

en System Rpl : 22.5 octets cksum # F13Fh

```
::
  K1NOLASTWD Dispatch1 THREE
  ::
  palparse COERCEFLAG
 ;
;
```

On voit qu'en général les programmes sont moins octivores tout en étant plus sécurisants.

Voici deux outils supplémentaires :

R→SB

Convertit un réel en entier système.

15 octets cksum # 31A2h

```
:: CK1NOLASTWD Dispatch1 ONE COERCE ;
```

SB→R

Fonction inverse qui convertit un entier système en réel.

10 octets cksum # F8CBh

```
:: CK1NOLASTWD UNCOERCE ;
```

Attention: même remarque que ci-dessus pour SB→B.

Dans un prochain article nous verrons quelques exemples d'utilisation de ces outils.

Allègements en douceur

On a vu par les mini-programmes ci-dessus qu'il est aventageux de remplacer dans un programme les couples 'entier binaire SYSEVAL' par l'objet en Rom qu'ils représentent. Or pour cela il faut compiler des fichiers sources ce qui n'intéressera peut-être pas tout le monde. Par contre ce qui est à la portée d'un plus grand nombre c'est de faire un programme en User Rpl avec ces ces couples 'entier binaire SYSEVAL' et d'utiliser un programme compilant ces couples. Ceci permettrait de faire un mixage User Rpl et System Rpl ayant l'avantage de la simplicité du User Rpl tout en augmentant les possibilités de programmation sans gachis d'octets avec les entiers binaires. D'autre part cela peut apporter un gain de vitesse, ce qui est encore un plus. Voici donc le programme de compilation. Il supprime aussi les '«' de début et les '»' de fin de programme, faisant encore gagner 5 octets.

COMPIL28

148.5 octets cksum # 9FC9h

```
::
 CK1NOLASTWD Dispatch1 EIGHT
 ::
  TOTEMPOB
CODE
        gosbvl =SAVPTR
        c=dat1 a
        r0=c            * sauve adr. obj
        d0=c
        gosbvl =SKIPOB  * objet suivant
        a=r0
        d1=a            * restaure d1 obj
        cd0ex           * adr obj suivant
        c=c-a  a        * len obj
        c=c-con a,10    * len-prolog et marqueur fin
        d=c    a        * compteur
        d0=a            * pour recopie
        d1=d1+ 5        * saute prologue
        d0=d0+ 5        * dito
        a=dat1 a        * 5 quartets suivant prolog.
        st=0 0          * pour x<<
        lc(5)  =x<<     * code de '<<'
        ?a#c a          * pas de '<<' ?
        goyes  no<<
        st=1   0        * présence de x<<
        d=d-con a,5     * pour - x<< et x>>
        d1=d1+ 5        * saute x<<
no<<    d=d-1  a        * actualise compteur
        gonc   ok
out     govlng =GETPTRLOOP * sortie sans rien faire
ok      c=0    w
        p=     11
        lchex  a54c     * c(w) := 0A45C00000000000
        p= 0
loop    a=dat1 1        * quartet courant
        ?a#0   p        * si le quartet <> 0 alors
        goyes  nxtnib   * suivant
        a=dat1 w        * sinon prend 16 quartets
        ?a#c   w        * <> 0A45C00000000000 ?
        goyes  nxtnib   * oui alors quartet suiv.
        d1=d1- 5        * non alors on remplace
        a=dat1 a        * l'entier binaire et le
        d0=d0- 15       * syseval par l'adresse
        dat0=a a        * correspondante et
        d0=d0+ 5        * on saute cette adresse
        d1=d1+ 16       * on avance après
        d1=d1+ 5        * le syseval
        d=d-con a,16    * actualisation compteur
        gonc   loop
        goc    test>>   * saut pour test si '>>'
nxtnib  dat0=a 1        * charge le quartet
        d0=d0+ 1        * avance d'un quartet
        d1=d1+ 1        * dito
decr    d=d-1  a        * actualise compteur quartets
        gonc   loop
test>> ?st=0   0        * pas de '>>' ?
        goyes  no>>     * alors avance normale
        d0=d0- 5        * si non on recule pour saut
no>>    a=dat1 a        *
        dat0=a a        * chargement de SEMI
        d0=d0+ 5        * après SEMI
        a=r0            * adresse début programme
        d1=a            * @       -       -
        d1=d1+ 5        * @ qui va être la longueur
*                       * de la chaine fictive
```

```
        c=dat1 a          * 5 quartets à sauver
        r2=c              * sauve 5 nibs après prolog.
        cd1ex
        r1=c              * adresse longueur
        govlng =Shrink$ * élimine les nibs inutiles
ENDCODE
CODE
        a=r1              * on récupère l'adresse des
        ad1ex             * 5 quartets de données
        c=r2              * on récupère les 5 quartets
        dat1=c a          * remise en place
        d1=a              * restaure D1
        govlng =Loop      * retour au Rpl
ENDCODE
 ;
;
```

Exemple d'utilisation :

Soit le programme SYSRCL ci-dessus en User Rpl mis au niveau 1. CMP28 renverra :

```
System Objet
System Object DUP
```

Soit un SYSRCL en System Rpl de 12.5 octets contre 43.5 octets pour l'original en User Rpl et également sans vérification d'arguments.

Trois adresses nouvelles par rapport aux listes parues dans JPC et utilisées dans les programmes ci-dessus :

SECONDELCOMP # 20238h (pas d'quivalent HP48)

C'est l'équivalent de TWO NTHELCOMP DROP

ERRTooFewArg # 0C526h (pas d'équivalent HP48 en System Rpl)
Shrink$ # 1E97A appelable par un Govlng

en entrée: R0 contient l'adresse de l'objet
R1 contient l'adresse de la longueur de l'objet
D0 @ après la fin des données
en sortie: R0 R1 inchangés

Les codes des programmes se trouveront dans le *coin des codes*.

Guy Toublanc (276)

POUR VOS MENUS

Il est parfois utile d'avoir à sa disposition des caractères existants mais non accessibles directement au clavier pour créer des noms de variables évitant les conflits entre programmes, ou même pour réaliser des messages très personnalisés. Pour cela on peut recourir à la fonction CHR, mais cela oblige à se référer au manuel. Une autre solution est d'avoir un programme qui crée un menu avec pour labels quelques uns de ces caractères spéciaux. C'est ce que fait le programme MNU qui utilise les caractères :

```
134 33 36 38 124 58 59 140 95 126 64 92 27
```

Ce qui vous donnera par exemple des ! $ & etc..

Ce programme est en System Rpl pour économiser des octets. Son listing a l'aspect de ceux pour HP48 car utilisant le même compilateur HP que celui pour HP48. D'autre part tous ces mnémoniques avec les adresses correspondantes ont été publiés par Jean François Garnier dans les JPC 87 et 88. Dans le n° 87 on trouve le programme CRNAME permettant de créer n'importe quel nom à partir d'une chaîne.

Voici ce programme. On remarquera la chaîne "$ "\86\21..." pour les caractères avec leur valeur ASCII en hexadécimal.

MNU
Cksum: # B6BBh 55.5 octets

```
:: $ "\86\21\24\26\7c\3a\3b\8c\5f\7e\40\5c\1b"
  THIRTEEN
  #1+_ONE_DO (DO)
           DUP INDEX@ DUP SUB$ $>ID SWAP
        LOOP
  DROP THIRTEEN ()N xMENU
;
```

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

Mode d'emploi

Pour obtenir le menu il suffit d'activer MNU. Après passage dans un autre menu, il suffira d'activer la touche CUSTOM pour revenir à ce menu particulier. Vous pourrez introduire dans une chaîne les caractères spéciaux des labels du menu ou obtenir des noms très particuliers pour vos variables.

Si vous vouliez étendre la liste des caractères, il est toujours possible que je vous refasse le programme au cas où cela vous serait impossible de le faire vous-même.

Guy Toublanc (276)

HP48

LES DISQUETTES 4 ET 5 POUR HP48

La disquette 4 : Goodies Disk 9, HPS et Divers

Le mois dernier, je vous présentais la nouvelle disquette pour HP48. Maintenant que j'ai eu le temps d'explorer plus complètement les fichiers récupérés sur CompuServe, voici quelques détails supplémentaires sur certaines applications relevées :

- 48IR : Infos sur codes IR de telécommandes TV et magnétoscopes.
- ASTRO : Calcul de positions astronomiques.
- C8SRC : Source de l'interpréteur Chip-8.
- CYNDRW : Outil de dessin sur 48.
- DISP48 : Interessant à titre d'exemple de source system Rpl.
- DISPLS : Affichage de listes (simulation carte Eq.Lib) en user Rpl.
- GREYSC : images en niveaux de gris.
- ICHING : Divinations chinoises, en system Rpl avec sources.
- KERFIX : Amélioration de la commande XRECV de la HP48GX.
- MATHLB : Belle librairie mathématique.
- MEMSCN : Editeur de mémoire pour GX.
- SRTSCH : Collection "scratch" de petits utilitaires en user Rpl.
- STOFIX : Parade au bug de la commande STO sur la GX.
- SWHALT : Gestion de points d'arrêt (HALT), pour GX semble-t-il.
- TET48 : Tetris 1.502 de Detlef Mueller, avec sources system Rpl.
- IR : Echantillonneur de commandes Infra-Rouge.
- LAPLACE : Calculs sur transformées de Laplace.
- MDISP : Source asm d'un remplaçant de DISP.
- PROOT : La fonction PROOT du 71 pour la 48 (écrit par Bill Wickes).
- TILT : Source system Rpl d'un économiseur d'écran (D. Mueller).

La disquette 5 : Outils pour PC

Comme le titre l'indique, cette disquette est destinée aux utilisateurs de PC. Vous y trouverez :

- 48PDL : Version de démo du *Programming Development Link* de HP. C'est un environnement de développement (Rpl utilisateur) déjà ancien (1990) qui n'a eu qu'une diffusion réduite. Cette version de démo est entièrement fonctionnelle, seules quelques fonctions (comme la sauvegarde) sont absentes. L'interêt du PDL est l'aide en ligne, équivalent au Programmer's Reference Manual (de la SX) non fourni avec nos machines, malheureusement, cette version en est presque entierement privée.

- DRAW48 : Programme de création de graphiques pour HP48.
- EMU48 : Sources C d'un émulateur de 48 pour Station Sun (X-Window)... Le portage sur PC est probablement difficile (mais pas impossible), sous Windows par exemple. Ce source est cependant intéressant à étudier (pour connaisseurs).
- GR2DR : Utilitaires en liaison avec DRAW48.
- GROBHP : Créateur de Grob..
- GROBMA : Créateur de Grob sous Windows. Nécessite VBRUN200.DLL.. Ne possédant pas ce dernier fichier, je n'ai pu tester ce soft.
- HPCONV : Conversion d'objets HP48 <-> fichiers MS-Dos.
- HPCPI : Pages de code (CPI) pour afficher le jeu de caractères HP48 sur IBM-PC.
- SRIDE : Environnement de développement system Rpl. Dans ma description de HPS du mois dernier, j'exprimais mon souhait d'un produit analogue pour le system Rpl. Et bien c'est fait avec ce *System Rpl Integrated Development Enviroment*. Sa conception est très semblable : éditeur multifenetre à la Borland, intégration de l'interface à Kermit, mais aussi aux différents outils HP : RPLCOMP, SASM, SLOAD. L'aide en ligne reprend le document RPLMAN.DOC de HP. Il s'agit cependant d'une version dite 'bêta', et il m'est arrivé de constater quelques réaction bizarres, ou même des plantages, de sorte que je vous recommande la prudence, dans l'attente d'une version définitive de ce produit néanmoins très interessant.
- STR104 : Source de l'assembleur Star 1.04.4. Beaucoup de développements en assembleur outre-atlantique se font avec un jeu de mnémoniques différent de celui officiel de HP (NDLR : Celui d'Alonzo Garpieri). C'est le cas d'un certain nombre de sources ASM venant de Compuserve et d'Internet. Cet assembleur écrit par Jan Brittenson vous permettra d'utiliser ces développements. Ce source C peut être compilé sur différentes machines : Dos (Borland et MicroSoft C), Amiga, VMS, Unix. L'executable MS-DOS est fourni.
- VGER7 : Voyager version V1.0-7. Nouvelle version de Voyager (explorateur de la Rom 48) présent sur la Goodies Disk 2.
- VSRC7 : Sources C de Voyager.

Jean-François Garnier (242)

TRUCS ASTUCES ET PETITS UTILITAIRES

Je rappelle que cette rubrique est ouverte à tous et encore plus à ceux qui n'ont que la possibilité de nous transmettre un texte sur papier et non sur disquette, à ceux qui ne peuvent pas passer beaucoup de temps sur un article. Si votre HP48 ne reste pas constamment au fond d'un tiroir ou que vous ne vous contentez pas de l'utiliser seulement pour les quatre opérations arithmétiques alors, vous avez sûrement découvert des astuces lors de son utilisation :

- Raccourcis clavier
- Mini programmes
- Equivalences entre HP48s/sx et HP48g/gx au niveau de l'accès aux commandes sans passer par les menus déroulants des HP48g/gx.
- etc...

Ou vous utilisez des astuces et informations qui vous ont été communiquées et qui pourraient être aussi utiles à d'autres utilisateurs.

2,80 ou même 2,40 francs sont-ils trop pour en faire bénéficier les autres lecteurs de JPC ?

Alors les anciens (ceux qui ne participent jamais) montrez que vous existez encore. Quant aux jeunes montrez que vous êtes curieux et à la recherche d'une utilisation intelligente de votre machine.

Aujourd'hui ce sera pour moi deux petits trucs.

Premier truc

Vous êtes dans un menu HP et vous voulez connaître dans leur intégralité les noms des labels par exemple dans le menu MATH les labels :

- Pour HP48s/sx

PART PROB HYP MATR VECTR BASE (une seule page)

sont en réalité:

PARTS PROB HYP MATRX VECTR BASE

donc certains ont un affichage tronqué

- Pour HP48g/gx

PROB FFT CMPL CONS (pour la 2ième page)

au lieu de:

PROB FFT CMPLX CONST

Vous êtes dans un menu utilisateur avec le même problème et en plus les minuscules apparaissent en majuscules.

Pour une visualisation, dans ces deux cas, il faut faire :

Pour HP48s/sx et comme indiqué dans le manuel :

REVIEW : shift droit [→] et flèche basse [▽]

mais pour HP48g/gx il faut savoir que c'est :

shift gauche [←] et flèche basse [▽]

La touche REVIEW de la HP48s/sx étant remplacée par la touche VIEW :
shift droit [→] et flèche basse [▽]

Cette possibilité pour HP48g/gx n'est indiquée dans aucun manuel.

Deuxième truc

Au clavier et dans un programme on peut entrer dans l'environnement graphique pour digitaliser les coordonnées de points à fournir au programme (par exemple dans l'article de Pierre de Sacy sur les fractales et dans JPC 95 page 17).

Cet accès se fera par:

GRAPH pour HP48s/sx
PICTURE pour HP48g/gx

Le problème est que si on tape dans un programme GRAPH sur HP48g/gx la machine saura traduire cela par PICTURE donc une compatibilité en amont que l'on a pas dans l'autre sens : PICTURE tapé sur une HP48 série S sera inopérant.

Une solution compatible hp48 séries S et G est celle utilisée par Pierre :

Remplacer GRAPH ou PICTURE par « » et PVIEW puis lors de l'affichage de l'objet graphique faire :

shift gauche [←] et flèche gauche [=]

pour entrer dans l'environnement graphique et pouvoir utiliser les possibilités de celui-ci, par exemple la digitalisation.

ou plus simple que tout le monde tape GRAPH. Les programmes auront le même checksum sur HP48S ou G.

C'est tout. Qui fait la suite ?

Guy Toublanc (276)

ALLEGEMENTS EN DOUCEUR

Dans certains cas précis l'usage de SYSEVALs dans un programme en User Rpl permet de faire des choses impossibles avec ce seul langage. Ce procédé qui permet d'étendre les possibilités de la machine tout en restant simple de réalisation a l'inconvénient de consommer des octets car un entier binaire courant se code sur 13 octets et le SYSEVAL occupe encore 2.5 octets soit au total 15.5 octets contre 2,5 octets en System Rpl pour appeler l'objet correspondant au point d'entrée. Pour des HP48S ou HP48G donc sans extensions mémoire cette perte de 13 octets n'est pas toujours à négliger.

La solution royale pour remédier à cela serait de programmer en System Rpl, mais cela pourra rebuter certains. Aussi je vous propose un compilateur qui remplacera chaque couple 'entier binaire SYSEVAL' par l'adresse de l'objet en Rom. Ceci permettra de faire un mixage User Rpl et System Rpl avec l'avantage de la simplicité du User Rpl tout en augmentant les possibilités de programmation sans gachis d'octets avec les entiers binaires. D'autre part cela peut apporter un gain de vitesse, ce qui est encore un plus. Voici donc le programme de compilation. Il supprime aussi le '«' de début et le '»' de fin de programme, faisant encore gagner 5 octets.

S'il y a des amateurs d'assembleur pour les HP48 et HP28 ils pourront voir quelques différences dans les fins de listings causées par les routines Shrink$ qui ne débutent pas et ne se terminent pas de la même manière (voir la rubrique HP28).

COMPIL48
139 octets cksum # 5C33h

```
::
 CK1NoBlame CK&DISPATCH0 EIGHT
 ::
  TOTEMPOB
CODE
        gosbvl =SAVPTR
        c=dat1 a
        r0=c            * sauve adr. obj
        d0=c
        gosbvl =SKIPOB  * objet suivant
        a=r0
        d1=a            * restaure d1 obj
        cd0ex           * adr obj suivant
        c=c-a  a        * len obj
        c=c-con a,10    * len-prolog et marqueur fin
        d=c    a        * compteur
        d0=a            * pour recopie
        d1=d1+ 5        * saute prologue
        d0=d0+ 5        * dito
        a=dat1 a        * 5 quartets suivant prolog.
        st=0 0          * pour x<<
        lc(5)  =x<<     * code de '<<'
        ?a#c a          * pas de '<<' ?
        goyes  no<<
        st=1   0        * présence de x<<
        d=d-con a,5     * pour - x<< et x>>
        d1=d1+ 5        * saute x<<
no<<    d=d-1  a        * actualise compteur
        gonc   ok
        goto   out      * sortie sans rien faire
ok      c=0    w
        p=     11
        lchex  1a52e    * c(w) := 1A52E00000000000
        p= 0
loop    a=dat1 1        * quartet courant
        ?a#0   p        * si le quartet <> 0 alors
        goyes  nxtnib   * suivant
        a=dat1 w        * sinon prend 16 quartets
        ?a#c   w        * <> 0A45C00000000000 ?
        goyes  nxtnib   * oui alors quartet suiv.
        d1=d1- 5        * non alors on remplace
        a=dat1 a        * l'entier binaire et le
        d0=d0- 15       * syseval par l'adresse
        dat0=a a        * correspondante e
        d0=d0+ 5        * on saute cette adresse
        d1=d1+ 16       * on avance après
        d1=d1+ 5        * le syseval
        d=d-con a,16    * actualisation compteur
        gonc   loop
        goc    test>>   * saut pour test si '>>'
nxtnib  dat0=a 1        * charge le quartet
        d0=d0+ 1        * avance d'un quartet
        d1=d1+ 1        * dito
decr    d=d-1  a        * actualise compteur quartets
        gonc   loop
test>> ?st=0   0        * pas de '>>' ?
```

```
        goyes  no>>     * alors avance normale
        d0=d0- 5        * si non on recule pour saut
no>>    a=dat1 a        *
        dat0=a a        * chargement de SEMI
        d0=d0+ 5        * après SEMI
        a=r0            * adresse début programme
        d1=a            * @        -       -
        d1=d1+ 5        * @ qui va être la longueur
*                       * de la chaine fictive
        c=dat1 a        * 5 quartets à sauver
        r2=c            * sauve 5 nibs après prolog.
        gosbvl =Shrink$ * élimine les nibs inutiles
        a=r1            * on récupère l'adresse des
        d1=a            * 5 quartets de données
        a=r2            * on récupère les 5 quartets
        dat1=a a        * remise en place
out     govlng =GETPTRLOOP
ENDCODE
 ;
;
```

Exemple d'utilisation:

Soit le mini programme COM→ qui éclate un objet composé en ses éléments et renvoie aussi le nombre de ces éléments.

```
COM→   41 octets
« # 54AFh SYSEVAL # 18DBFh SYSEVAL »

  External External
```

soit un programme de 10 octets.

De même le programme qui recompose le programme éclaté avec au niveau 1 le nombre des éléments:

```
→PRG  41 octets
« # 18CEAh SYSEVAL # 5445h SYSEVAL »
```

COMPIL48 réduira égalemnt ce programme en User Rpl en un programme de 10 octets.

Le procédé peut être appliqué à des programmes contenant de nombreux SYSEVALs. Ainsi vous pouvez tout en programmant en User Rpl avoir l'impression de faire du System Rpl. Dans un prochain article on verra comment inclure des vérifications d'arguments au début d'un programme avec un SYSEVAL en première position car dans ce cas comme avec les deux mini-programmes ci-dessus il faut être vigilant avant de lancer un programme avec des SYSEVALs.

Les codes se trouveront dans *le coin des codes*.

Guy Toublanc (276)

NOUVEAUX POINTS D'ENTREES

SYSTEM RPL

En étudiant la dernière version du système de développement RPL de Detlef Mueller et Raymond Hellstern (voir le volume 9 des Goodies Disks), j'ai trouvé de nouveaux points d'entrées supportés absents du fichier ENTRIES.ADD distribué sur la disquette kit de développement du club.

Voici ces points d'entrées (pour SX et GX) :

```
=SETMESG      EQU #0764E * (primitive)
=OFFSRRP      EQU #076AE * (primitive)
=NULLID       EQU #15777 * (globname)
=CRLF$        EQU #2E4F0 * (string)
=%80          EQU #320B1 * (real)
=SetDA2bBad   EQU #394CF * (primitive)
=ClrDA2bBad   EQU #394DD * (primitive)
=SetDA3Bad    EQU #394F9 * (primitive)
=MenuMaker    EQU #407FB * (prog)
=MenuKey      EQU #40828 * (prog)
=Modifier     EQU #4085A * (prog)
=DoDelim      EQU #40DD4 * (prog)
=DoDelims     EQU #40DF7 * (prog)
=MenuDef@     EQU #418A4 * (prog)
=ID_X         EQU #4AB1C * (globname)
=PvarsC%0     EQU #4AB2A * (complex)
=ID_Y         EQU #4AB59 * (globname)
```

Notez en particulier les noms globaux ID_X ('X') et ID_Y ('Y'), voir à ce propos l'article de David Fabiani dans JPC94.

Je transmet bien sûr au club les versions à jour de ENTRIES.ADD et ENTRIES.O (NDLR : Ils sont dès à présents inclus dans notre "Kit de développement").

Jean-François Garnier (242)

FRACTALES LEGERES, RAPIDES ET EN COULEURS !

Un petit frère de Mandel

Après les triangles de Sierpinski, les ensembles de Mandelbrot ont fait leur apparition dans JPC mais cette fois sous la signature de Pierre de Sacy que je remercie pour avoir tracé la voie. A priori ce problème de graphisme nécessite beaucoup de variables qu'il faut sauvegarder entre les calculs ce qui rend la programmation en assembleur assez délicate pour éviter des temps d'exécution assez longs. Pierre a appliqué une méthode de programmation qui permet de simplifier le travail avec les pointeurs D1 et D0. Une méthode qui sera sûrement réutilisée car elle est intéressante. Bien que ce problème de fractales se traite en général en virgule flottante, notre ami a pris la bonne solution de le faire en virgule fixe ce qui est très écomique en temps d'exécution et en en mémoire par rapport aux calculs en virgule flottante.

Malgré cela le programme MANDEL occupe 733 octets et ceux qui se sont lancés à taper les codes ont dû certainement faire une pause café avant d'arriver à la 92ième ligne. Je me suis dit qu'on devait bien pouvoir éliminer des octets ici et là. Après quelques essais voici le résultat: 363 octets et une vitesse deux fois et demi plus rapide. Pour arriver à cela j'ai utilisé une programmation classique mais en tenant compte de l'évolution des calculs. D'autre part je me sers du même algorithme que HP pour les multiplications de nombres entiers. Je ne fais pas appel aux routines PIXON de HP ce qui est plus simple et plus rapide dans ce cas particulier.

Voici le bébé :

```
*****************************************************
*                    MANDELGT                       *
*           363 octets  cksum # EAA7h               *
*              compatible G et S                    *
*                                                   *
* adaptation et optimisation du programme MANDEL    *
* (733 octets  JPC 95) de Pierre Silvestre de Sacy  *
*        par Guy Toublanc         le 20/4/94        *
*                                                   *
* fichier source compilable sur PC par RPLCOMP de HP*
* ou sur HP48 avec ->RPL de D. Mueller (en mettant  *
* en majuscules les mnémoniques de l'assembleur     *
*****************************************************
* arguments et notations :                          *
*  arguments :                                      *
*              niveau 3 : ( Xc,Yc)   1 complexe     *
*              niveau 2 : ( Px,Py)   1 complexe     *
*              niveau 1 :  iter      1 réel         *
*  Xc,Yc: coordonnées du centre de l'image          *
*  Px,Py: pas d'avancement en abscisses et ordonnées*
*  iter : nombres d'itérations de                   *
*         Z(i+1) = Z(i)^2 + Z                       *
*         avec Z(i) = ( Xz(i),Yz(i) )  (boucle Z)   *
*              Z = (Xz,Yz)   pour le point courant  *
* principe de départ:                               *
*         Xc,Yc,Px,Py sont multipliés par           *
*         108576 = # 100000h puis convertis en      *
*         entiers binaires signés pour les calculs  *
*         en virgule fixe : hhh....h , hhhhh        *
*         pour ces arguments tous les calculs se    *
*         feront suivant ce mode                    *
*                                                   *
* les autres variables :                            *
*                                                   *
*  Xzo,Yzo cordonnées du 1er pixel de la 1ère ligne *
*  Xi,Yi coordonnées du 1er point de chaque ligne   *
*                                      (boucle I)   *
*  Xj abscisse du point courant d'une ligne         *
*                                      (boucle J)   *
*  Xz,Yz coordonnées du point courant pour le départ*
*  de l'itération de la boucle Z                    *
*  Xz(i),Yz(i) voir ci-dessus                       *
*                                                   *
* principe général :                                *
*     chaque ligne est parcourue de gauche à droite *
*     par la boucle J, pour chaque point on fait n  *
*     itérations (iter) pour calculer Z(i) avec la  *
*     boucle Z, si abs(Z(i) >= 2 : sortie de boucle *
*     sinon après les n itérations on fait PIXON    *
*     le balayage (de haut en bas) des 64 lignes    *
*     effectué par la boucle I                      *
*                                                   *
* pour commencer : conversion des données en entiers*
* binaires stockés dans un entier binaire long de   *
* 5*16 digits                                       *
*****************************************************
::
 CK3&Dispatch # 221              * vérif. des args
 ::
  '                              * objet non évalué
   ::                            * sous-programme de
    MINUSONE UNCOERCE %1+ %* DUP * conversion des
    %0 %< ITE :: %CHS %># HXS 1  * réels x # 100000h
                 0 SWAP bit- ;   * en entiers
              %>#  SWAP          * signés
   ;
  4UNROLL %ABS %># SWAP C%>%     * iter -> hexa
  5PICK EVAL                     * pas Yp -> hexa
  5PICK EVAL &HXS &HXS SWAP C%>% * pas Xp -> hexa
  4PICK EVAL                     * Yc
  4ROLL EVAL &HXS &HXS           * Xc
  SIXTYFOUR BINT_131d MAKEGROB   * affiche grob
  GROB>GDISP ZEROZERO WINDOW#    * blanc 131x64
*************************************************
* niveau 1 : entier binaire contenant:          *
*            iter , Px , Py , Xc , Yc           *
```

```
*            soit 5*16 quartets de données      *
*************************************************
CODE
        gosbvl  =PopASavptr
        d1=a            * @ l'entier binaire
        d0=(5)  #01f79
        a=dat0  a
        d0=a
        a=dat0  a
        d0=a            * @ PICT
        d0=d0+  16      * saute prolog. long. dims
        d0=d0+  4       * et @ début données PICT
        c=0     w
        p=      4
        cpex    5
        d=c     w       * d(w) := # 400000h : limite
        d1=d1+  10      * @ iter
        c=dat1  a
        c=c-1   a       * iter - 1
        rstk=c          * sauve iter sur rstk
        d1=d1+  12      * @ index lignes I
        lc(2)   63      * 64 -1 lignes pour loopx
        dat1=c  b       * index lignes I
        d1=d1+  4       * saute index absc., @ Px
        a=dat1  w       * Px
        b=a     w
loopx   a=a+b   w       * pour 65*Px et calcul Xzo
        c=c-1   b
        gonc    loopx
        d1=d1+  16      * @ Yp
        lc(2)   30
        d=c     b       * index boucle y
        c=dat1  w       * Yp
        b=c     w
loopy   b=b+c   w       * pour 32*Py et calcul Yzo
        d=d-1   b
        gonc    loopy
        d1=d1+  16      * @ Xc
        c=dat1  w       * Xc
        c=c-a   w       * Xc - 65*Px
        dat1=c  w       * Xzo début de ligne
        d1=d1+  16      * @ Yc
        a=dat1  w       * Yc
        d1=d1-  16      * @ sauvegarde Xzo
        a=a+b   w       * Yzo = Yc + 32*Py
*************************************************
* d1 @ Xzo    à d1- 16  Py   à d1- 32 Px         *
*             à d1- 36  index lignes (I=63)     *
*             à d1- 34  index abscisses (J)     *
* iter sur rstk   p= 0   r0 := Xzo    r1 := Yzo *
* d(w) := 4 000..h     d(b) index pour boucle Z *
* d0 @ début données du pict                    *
* a(w) := Yi = Yzo     c := Xj = Xzo            *
*************************************************
loopI   r1=a            * Yi
        r0=c            * Xj = Xzo
        r2=c            * Xz(i)
        d1=d1-  16      * @ Py
        d1=d1-  16      * @ Px
        d1=d1-  2       * @ index J abscisses
        lc(2)   130     * (131 - 1) pixels
        dat1=c  b       * index J pixels
        d=0     xs      * position pixel dans nib.
        d=d+1   xs      * initialisation
loopJ
*************************************************
* d1 @ J   d(xs) = 1 index position dans le quartet *
*                  et pour incrémenter d0       *
* p= 0          J = 130   r0  r1  r2  r3 actualisés *
* d0 @ début ligne I                            *
*************************************************
        c=rstk
        d=c     b       * index itérations
        rstk=c
loopZ
*************************************************
* calcul Xz(i)^2   Yz(i)^2  2Xz(i)*Yz(i)        *
* entrée :                                      *
*          d1 @ J                               *
* principe :                                    *
*          si ( Z(i)^2 ) >= 4 donc abs(Z(i)) >= 2 : *
*          sortie pour le pixel suivant de la ligne *
*          sinon PIXON                          *
*************************************************
        r3=a            * Yz(i) -> r3(w)
        gosub   sqr     * a(w) := Yz(i)^2
        r4=a            * Yz(i)^2 -> r4(w)
        a=r2            * Xz(i)
        gosub   sqr     * a(w) := Xz(i)^2
        c=r4            * Yz(i)^2
        c=c+a   w       * Xz(i)^2 + Yz(i)^2
        ?d<c    w       * limite dépassée ?
        goyes   nopixon
        c=r4            * Yz(i)^2
        a=a-c   w       * Xz(i)^2 - Yz(i)^2
        c=r0            * Xz
        a=a+c   w       * Xz(i+1)
        ar2ex           * Xz(i+1) -> r2  a(w) := Xz(i)
        a=a+a   w       * 2*Xz(i)
        c=r3            * Yz(i)
        gosub   mpy     * 2*Xz(i)*Yz(i)
        c=r1            * Yz
        a=a+c   w       * Yz(i+1)         (-> r3)
nextZ   d=d-1   b       * index iter
        gonc    loopZ

pixon   c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
nopixon d=d+d   xs      * pour position pixel suivant
        gonc    idnib   * si pas encore position 3
        d0=d0+  1       * après bit 0 -> 1 -> 2 ->
        d=d+1   xs      * bit 3 on revient à bit 0
*                       * et on avance d'un quartet
idnib   c=dat1  b       * index J
        c=c-1   b
```

```
        goc     outJ    * on sort de la boucle J
        dat1=c  b       * sinon sauve index J
        d1=d1+  2       * @ Px
        a=r0            * X(j)
        c=dat1  w       * Px
        c=a+c   w       * X(j+1) = X(j) + Px
        r0=c            * X(j+1) -> r0
        r2=c            * Xz(i) = X(j+1)
        a=r1            * Yz -> Yz(i)      (-> r3)
        d1=d1-  2       * @ index J
nextJ   goto    loopJ

outJ    d0=d0+  2       * nib 34  puis ligne I + 1
        d1=d1-  2       * @ index I
        c=dat1  b       * index I
        c=c-1   b
        goc     out     * sortie générale
        dat1=c  b       * sinon sauve index I
        d1=d1+  4       * saute les 2 index
        d1=d1+  16      * @ Py
        a=r1            * Y(i)
        c=dat1  w       * Py
        a=a-c   w       * Y(i+1) = Y(i) - Py (->r1/r3)
        d1=d1+  16      * @ Xzo
        c=dat1  w       * Xzo
nextI   goto    loopI   * a:= Y(i+1) c:= Xzo d1 @ Xzo
out     c=rstk          * enlève iter de la rstk
        govlng  =GETPTRLOOP * retour au Rpl
*****************************************************
* produit de 2 nombres < à 2 en valeur absolue      *
* en entrée : a(w) & c(w) les 2 nombres à multiplier*
* en sortie :  a(w) = a*c    c(w) = 0               *
*****************************************************
sqr     c=a     w       * pour le carré
mpy     st=0    2       * pour le signe
        ?a=0    s       * positif ?
        goyes   apos    * oui alors test autre fact.
        a=-a    w       * sinon valeur absolue
        st=1    2       * pour négatif
apos    ?c=0    s       * positif ?
        goyes   mpy0    * oui alors produit
        c=-c    w       * sinon valeur absolue
        ?st=1   2       * 1er facteur négatif ?
        goyes   neg     * alors produit positif
        st=1    2       * sinon produit négatif
        gonc    mpy0    * saut pour le produit
neg     st=0    2       * pour produit positif
mpy0    b=0     w       * initialisation produit
        abex    w       * a: produit b: muliplicande
mpy1    sb=0            * pour bit 0 nul
        csrb            * divise par 2
        ?sb=0           * bit 0 nul ?
        goyes   mpy2    * alors produit inchangé
        a=a+b   w       * sinon produit partiel
mpy2    b=b+b   w       * multiplicande * 2
        ?c#0    w       * reste bits multiplicateur ?
        goyes   mpy1    * oui on itération
        gosbvl  =ASRW5  * sinon divise par # 100000h
        ?st=0   2       * produit positif
        rtnyes          * oui retour
        a=-a    w       * sinon entier signé
        rtn
ENDCODE
  GBUFF                 * rappelle PICT sur la pile
 ;                      * pour sauvegarde éventuelle
;
```

On remarquera que j'adopte une méthode différente de celle de Pierre pour l'affichage. Mon programme permet comme MANDEL d'avoir le GROB sur la pile à la fin de l'exécution pour sauvegarde éventuelle mais aussi de pouvoir réafficher simplement avec PICTURE sur HPG/GX ou GRAPH sur S/SX tant que l'on a pas modifié le PICT.

Pour réaliser la vue d'ensemble avec les paramètres donnés par Pierre:

```
( -0.5 ,    0 )
( 0.03 , 0.03 )
          20
```

il faut sur ma HP48GX 149 secondes contre 375 avec le programme de Pierre.

Bébé grandit

Pierre avait donné une possibilité de modifier son programme MANDEL pour produire non pas deux ensembles de pixels mais trois ensembles. Mais il faut faire la modification et recompiler le fichier source. Il est possible de faire autrement pour avoir, sans modifier le programme, les deux solutions: il suffit de fournir le nombre d'itérations avec :

- le signe positif pour la solution à deux ensembles
- le signe négatif pour la solution à trois ensembles

D'autre part il est un peu rageant, lorsque la figure est symétrique verticalement de devoir attendre autant de temps pour réaliser la partie basse après la partie haute. Le programme ci-dessous calcule tous les points de la partie haute puis effectue une symétrie qui se fait pratiquement instantanément d'où un temps divisé par deux.

Enfin on peut avoir besoin d'interrompre le programme autrement que par un arrêt système ce qui fait perdre tout ce qui a été réalisé. Mon progamme sait détecter l'appui sur la touche [ATTN] (HP48S/SX) ou CANCEL (HP48G/GX) pour interrompre le programme en douceur.

Tous ces plus se soldent évidemment par un surplus d'octets, 78.5 exactement.

Remarque pratique: pour l'interruption insister sur l'appui de la touche.

Dans le listing suivant je ne reproduit que les changements par rapport à MANDELGT, vous laissant le soin de raccorder les morceaux.

```
*****************************************************
*                    MAND3CSI                       *
*          441.5 octets  cksum # 422Ch              *
*                                                   *
* Extension du programme MANDELGT permettant de     *
* traiter 2 fois plus rapidement les ensembles de   *
* Mandelbrot lorsqu'il y a symétrie (Yc = 0)        *
* et d'afficher des pixels fonction de la parité    *
* du nombre d'itérations avec la possibilité de     *
* stopper le programme sans arrêt système           *
*                                                   *
*    Auteur : Guy Toublanc       le 21/4/94         *
*****************************************************
* arguments et notations :                          *
*  arguments :                                      *
*              niveau 3 : ( Xc,Yc)   1 complexe     *
*              niveau 2 : ( Px,Py)   1 complexe     *
*              niveau 1 :  iter      1 réel         *
*  Xc,Yc: coordonnées du centre de l'image          *
*  Px,Py: pas d'avancement en abscisses et ordonnées*
*  iter : nombres d'itérations de                   *
*         Z(i+1) = Z(i)^2 + Z                       *
*         avec Z(i) = ( Xz(i),Yz(i) )  (boucle Z)   *
*              Z = (Xz,Yz)   pour le point courant  *
*        si iter est négatif un troisième ensemble  *
*        de pixels sera affiché                     *
* principe de départ:                               *
*         iter,Xc,Yc,Px,Py sont multipliés par      *
*         108576 = # 100000h puis convertis en      *
*         entiers binaires signés pour les calculs  *
*         en virgule fixe : hhh....h , hhhhh        *
*         pour ces arguments tous les calculs se    *
*         feront suivant ce mode sauf pour iter     *
*         qui après détermination de son signe      *
*         reprendra sa valeur sans signe            *
*                                                   *
* les autres variables :                            *
*    ..........................................     *
* principe général :                                *
*     chaque ligne est parcourue de gauche à droite *
*     par la boucle J, pour chaque point on fait n  *
*     itérations (iter) pour calculer Z(i) avec la  *
*     boucle Z :                                    *
*       si abs(Z(i) >= 2 alors sortie de boucle     *
*         avec 2 cas pour le nombre d'itérations k: *
*               si k pair alors PIXON               *
*               sinon point suivant                 *
*       sinon après les n itérations on fait PIXON  *
*     le balayage (de haut en bas) des 64 lignes    *
*     effectué par la boucle I ou des 33 lignes     *
*     hautes jusqu'à l'axe de symétrie si Yc = 0    *
*                                                   *
* intégration dans la boucle principale I d'un test *
* d'appui de la touche [ATTN]/[CANCEL] pour sortie  *
* de programme sans destruction de PICT             *
*                                                   *
* pour commencer : conversion .....                 *
*****************************************************
::
 CK3&Dispatch # 221
 ::
  '
   ::
    MINUSONE UNCOERCE %1+ %* DUP
    %0 %< ITE :: %CHS %># HXS 1
                 0 SWAP bit- ;
               %>#  SWAP
   ;
  DUP4UNROLL EVAL C%>%
  4PICK EVAL
  4PICK EVAL &HXS &HXS ROT C%>%
  4PICK EVAL                    <********
  ..............................          *
  ..............................          *   dito
CODE                                      * MANDELGT
=out<c>in  EQU  #01eec                    *
        gosbvl  =PopASavptr     <********
        gosbvl  =DisableIntr
        d1=a                    <********
        d0=(5)  #01f79                    *
  ..............................          *   dito
  ..............................          * MANDELGT
        d1=d1+  10      * @ iter          *
        c=dat1  w               <********
        clrst           * pour non symétrie et
        ?c=0    s       * iter positif
        goyes   twocolors
        st=1    0       * pour 3 couleurs
        c=-c    w       * valeur absolue de iter
twocolors
        gosbvl  =CSRW5  * vraie valeur de iter
        c=c-1   a               <********
        rstk=c                            *   dito
        d1=d1+  12                        *
        lc(2)   63                        * MANDELGT
        dat1=c  b               <********
        cd1ex
        r0=c            * sauve @ index I
        cd1ex           * restaure d1 et c(a)
        d1=d1+  4               <********
        a=dat1  w                         *
  ..............................          *  dito
  ..............................          * MANDELGT
        a=dat1  w       * Yc              *
        d1=d1-  16              <********
        ?a#0    w       * Yc <> 0 ?
        goyes   calcYzo * oui alors non symétrie
        st=1    1       * pour symétrie verticale
        cr0ex           * c(a) := adr. I   r0 := Xzo
```

```
        cd1ex           * c(a) := adr. Xzo d1 @ I
        rstk=c          * sauve adr. Xzo
        lc(2)   32      * (n lignes à balayer) - 1
        dat1=c  b       * index I
        c=rstk          * récupère adr. Xzo
        d1=c            * d1 @ Xzo
        c=r0            * Xzo
calcYzo a=a+b   w               <********
loopI   r1=a            * Yi            *
  ........................              *
loopJ                                   *   dito
  ........................              *
loopZ                                   * MANDELGT
  ........................              *
        c=r4            * Yz(i)^2       *
        c=c+a   w               <********
        ?c<d    w       * limite non dépassée ?
        goyes   cont    * oui on continue
        ?st=0   0       * non alors 2 couleurs ?
        goyes   nopixon
        c=d     b       * nombre d'itérations à faire
        ?cbit=0 0       * parité paire ?
        goyes   nopixon
        gonc    pixon   * non alors PIXON
cont    c=r4                    <********
        a=a-c   w               *
        c=r0            * Xz            *   dito
  ........................              *
  ........................              * MANDELGT
        c=dat1  w       * Xzo           *
nextI   goto    loopI           <********
?sym    ?st=0   1       * non symétrie ?
        goyes   out     * oui alors sortie générale
        ad0ex
        d0=a            * @ début ligne sous axe
        c=0     a
        lc(2)   29      * n-1 lignes à symétriser
        d=c     a       * index lignes
        lc(2)   (2*34)  * offset pour reculer au
        b=c     a       * début ligne au-dessus
looprow a=a-b   a       * ligne au-dessus
        d1=a            * @  -     -
        lc(2)   16      * (m-1) octets par ligne
loop2nib
        a=dat1  b       * octet du grob au-dessus axe
        dat0=a  b       * symétrise 8 pixels à la fois
        d0=d0+  2       * 8 pixels ou octet suivant
        d1=d1+  2       *         dito
        c=c-1   b       * index d'octets - 1
        gonc    loop2nib
        ad1ex           * a := adr. fin ligne copiée
        d=d-1   a       * index de lignes - 1
        gonc    looprow
out     c=rstk          * enlève iter de la rstk
        gosbvl  =AllowIntr * autorise les interrupt.
        govlng  =GETPTRLOOP     <********
  ........................              *
  ........................              *   dito
ENDCODE                                 * MANDELGT
  GBUFF                                 *
 ; ;                            <********
```

Bébé prend des couleurs

Le progamme MAND3CSI peut générer deux ensembles de points et il est possible d'aller au delà mais avec l'affichage standard HP48 à deux couleurs il faut y renoncer. La solution est d'utiliser la technique des nuances de gris, technique employée par certains bien avant la sortie des HP48g/gx (et donc de *voyage g/gx*).

Le principe est simple. Prenons l'exemple de ce que fait le programme MAND4CSI ci-dessous. Celui-ci génère 3 ensembles de points que nous dénommerons A B C. On fera:
1 -un affichage avec l'ensemble A
2 -un affichage avec les ensembles A + B
3 -un affichage avec les ensembles A + B + C

Ainsi l'ensemble A sera affiché pendant le temps 3*T
l'ensemble B sera affiché pendant le temps 2*T
l'ensemble C sera affiché pendant le temps T

et l'on aura la graduation d'image:
l'ensemble A en gris foncé
l'ensemble B en gris moyen
l'ensemble C en gris clair

avec le blanc nous aurons ainsi quatre couleurs.

Le programme MAND4CSI génèrera un GROB de hauteur égale à 3 écrans. Chaque tiers correspondant à un écran contiendra:

le premier tiers les ensemble A,B,C
le deuxième tiers les ensemble A,B
le troisième tiers l'ensemble A

Il faudra évidemment un programme d'affichage de ce type particulier de GROB

Voici MAND4CSI qui s'emploie comme les précédents programmes et qui a les mêmes fonctionnalités que MAND3CSI. Le listing est donné intégralement même s'il y a des parties communes avec les autres listings afin d'éviter les acrobaties de lecture.

```
*****************************************************
*                    MAND4CSI                       *
*            545 octets  cksum # 5B1Fh              *
*                compatible S et G                  *
*                                                   *
* Extension du programme MANDELGT permettant de     *
* traiter 2 fois plus rapidement les ensembles de   *
* Mandelbrot lorsqu'il y a symétrie (Yc = 0)        *
* et d'afficher des pixels fonction de la parité    *
```

```
* du nombre d'itérations avec la possibilité de     *
* stopper le programme sans arrêt système           *
* Extension du programme MAND3CSI avec la construc- *
* tion de 3 écrans:                                 *
* 1er écr. avec les pixels correspondant au maximmum*
*     d'itérations sans dépassement de limite       *
* 2ième écran avec les pixels du 1er écran plus ceux*
*     correspondant aux dépassements de limite lors *
*     d'une itération dont le bit 0 du rang est de  *
*     parité impaire                                *
* 3ième écran avec les pixels du 2e écran plus ceux *
*     correspondant aux dépassements de limite lors *
*     d'une itération dont le bit 1 du rang est de  *
*     parité impaire                                *
*                                                   *
*    Auteur : Guy Toublanc       le 22/4/94         *
*                                                   *
* fichier source compilable sur PC par RPLCOMP de HP*
* ou sur HP48 avec ->RPL de D. Mueller (en mettant  *
* en majuscules les mnémoniques de l'assembleur     *
*****************************************************
* arguments et notations :                          *
*  arguments :                                      *
*              niveau 3 : ( Xc,Yc)   1 complexe     *
*              niveau 2 : ( Px,Py)   1 complexe     *
*              niveau 1 :  iter      1 réel         *
*  Xc,Yc: coordonnées du centre de l'image          *
*  Px,Py: pas d'avancement en abscisses et ordonnées*
*  iter : nombres d'itérations de                   *
*         Z(i+1) = Z(i)^2 + Z                       *
*         avec Z(i) = ( Xz(i),Yz(i) )  (boucle Z)   *
*              Z = (Xz,Yz)   pour le point courant  *
* principe de départ:                               *
*         Xc,Yc,Px,Py sont multipliés par           *
*         108576 = # 100000h puis convertis en      *
*         entiers binaires signés pour les calculs  *
*         en virgule fixe : hhh....h , hhhhh        *
*         pour ces arguments tous les calculs se    *
*         feront suivant ce mode                    *
*                                                   *
* les autres variables :                            *
*                                                   *
*  Xzo,Yzo cordonnées du 1er pixel de la 1ère ligne *
*  Xi,Yi coordonnées du 1er point de chaque ligne   *
*                                      (boucle I)   *
*  Xj abscisse du point courant d'une ligne         *
*                                      (boucle J)   *
*  Xz,Yz coordonnées du point courant pour le départ*
*  de l'itération de la boucle Z                    *
*  Xz(i),Yz(i) voir ci-dessus                       *
*                                                   *
* principe général :                                *
*     chaque ligne est parcourue de gauche à droite *
*     par la boucle J, pour chaque point on fait n  *
*     itérations (iter) pour calculer Z(i) avec la  *
*     boucle Z :                                    *
*       si abs(Z(i) >= 2 alors sortie de boucle     *
*         avec 3 cas pour le nombre d'itérations k: *
*              si k pair alors PIXON                *
*              si bit 1 de k est pair alors PIXON   *
*              sinon point suivant                  *
*       sinon après les n itérations on fait PIXON  *
*     le balayage (de haut en bas) des 64 lignes    *
*     effectué par la boucle I ou des 33 lignes     *
*     hautes jusqu'à l'axe de symétrie si Yc = 0    *
*                                                   *
* intégration dans la boucle principale I d'un test *
* d'appui de la touche [ATTN]/[CANCEL] pour sortie  *
* de programme sans destruction de PICT             *
*                                                   *
* pour commencer : conversion des données en entiers*
* binaires stockés dans un entier binaire long de   *
* 5*16 digits                                       *
*****************************************************
::
 CK3&Dispatch # 221
 ::
  '
   ::
    MINUSONE UNCOERCE %1+ %* DUP
    %0 %< ITE :: %CHS %># HXS 1
                  0 SWAP bit- ;
               %>#  SWAP
  ;
  4UNROLL %ABS %># SWAP C%>%
  5PICK EVAL
  5PICK EVAL &HXS &HXS SWAP C%>%
  4PICK EVAL
  4ROLL EVAL &HXS &HXS
  BINTC0h BINT_131d MAKEGROB
  GROB>GDISP ZEROZERO WINDOW#
************************************************
* niveau 1 : entier binaire contenant:          *
*            iter , Px , Py , Xc , Yc           *
*            soit 5*16 quartets de données      *
************************************************
CODE
=out<c>in  EQU  #01eec
        gosbvl  =PopASavptr
        gosbvl  =DisableIntr
        d1=a            * @ l'entier binaire
        d0=(5)  #01f79
        a=dat0  a
        d0=a
        a=dat0  a
        d0=a            * @ PICT
        d0=d0+  16      * saute prolog. long. dims
        d0=d0+  4       * et @ début données PICT
        c=0     w
        p=      4
        cpex    5
        d=c     w       * d(w) := # 400000h : limite
        d1=d1+  10      * @ iter
        c=dat1  w
        c=c-1   a       * iter - 1
        rstk=c          * sauve iter sur rstk
```

```
        d1=d1+  12      * @ index lignes I
        lc(2)   63      * 64 -1 lignes pour loopx
        dat1=c  b       * index lignes I
        cd1ex
        r0=c            * sauve @ index I
        cd1ex           * restaure d1 et c(a)
        d1=d1+  4       * saute index absc., @ Px
        a=dat1  w       * Px
        b=a     w
loopx   a=a+b   w       * pour 65*Px et calcul Xzo
        c=c-1   b
        gonc    loopx
        d1=d1+  16      * @ Yp
        lc(2)   30
        d=c     b       * index boucle y
        c=dat1  w       * Yp
        b=c     w
loopy   b=b+c   w       * pour 32*Py et calcul Yzo
        d=d-1   b
        gonc    loopy
        d1=d1+  16      * @ Xc
        c=dat1  w       * Xc
        c=c-a   w       * Xc - 65*Px
        dat1=c  w       * Xzo début de ligne
        d1=d1+  16      * @ Yc
        a=dat1  w       * Yc
        d1=d1-  16      * @ sauvegarde Xzo
        st=0    1       * pour non symétrie
        ?a#0    w       * Yc <> 0 ?
        goyes   calcYzo * oui alors non symétrie
        st=1    1       * pour symétrie verticale
        cr0ex           * c(a) := adr. I   r0 := Xzo
        cd1ex           * c(a) := adr. Xzo d1 @ I
        rstk=c          * sauve adr. Xzo
        lc(2)   32      * (n lignes à balayer) - 1
        dat1=c  b       * index I
        c=rstk          * récupère adr. Xzo
        d1=c            * d1 @ Xzo
        c=r0            * Xzo
calcYzo a=a+b   w       * Yzo = Yc + 32*Py
*****************************************************
* d1 @ Xzo    à d1- 16  Py   à d1- 32 Px            *
*             à d1- 36  index lignes (I=63 ou 32)   *
*             à d1- 34  index abscisses (J)         *
* iter sur rstk   p= 0   r0 := Xzo    r1 := Yzo     *
* d(w) := 4 000..h     d(b) index pour boucle Z     *
* d0 @ début données du pict                        *
* a(w) := Yi = Yzo     c := Xj = Xzo                *
*****************************************************
loopI   r1=a            * Yi
        r0=c            * Xj = Xzo
        r2=c            * Xz(i)
        d1=d1-  16      * @ Py
        d1=d1-  16      * @ Px
        d1=d1-  2       * @ index J abscisses
        lc(2)   130     * (131 - 1) pixels
        dat1=c  b       * index J pixels
        d=0     xs      * position pixel dans nib.
        d=d+1   xs      * initialisation
loopJ
*****************************************************
* d1 @ J   d(xs) = 1 index position dans le quartet *
*                  et pour incrémenter d0           *
* p= 0         J = 130   r0  r1  r2  r3 actualisés  *
* d0 @ début ligne I                                *
*****************************************************
        c=rstk
        d=c     b       * index itérations
        rstk=c
loopZ
*****************************************************
* calcul Xz(i)^2   Yz(i)^2  2Xz(i)*Yz(i)            *
* entrée :                                          *
*          d1 @ J                                   *
* principe :                                        *
*          si ( Z(i)^2 ) >= 4 donc abs(Z(i)) >= 2 : *
*          sortie pour le pixel suivant de la ligne *
*          sinon PIXON                              *
*****************************************************
        r3=a            * Yz(i) -> r3(w)
        gosub   sqr     * a(w) := Yz(i)^2
        r4=a            * Yz(i)^2 -> r4(w)
        a=r2            * Xz(i)
        gosub   sqr     * a(w) := Xz(i)^2
        c=r4            * Yz(i)^2
        c=c+a   w       * Xz(i)^2 + Yz(i)^2
        ?d<c    w       * limite dépassée ?
        goyes   nopixon * sortie de boucle
cont    c=r4            * Yz(i)^2
        a=a-c   w       * Xz(i)^2 - Yz(i)^2
        c=r0            * Xz
        a=a+c   w       * Xz(i+1)
        ar2ex           * Xz(i+1) -> r2  a(w) := Xz(i)
        a=a+a   w       * 2*Xz(i)
        c=r3            * Yz(i)
        gosub   mpy     * 2*Xz(i)*Yz(i)
        c=r1            * Yz
        a=a+c   w       * Yz(i+1)          (-> r3)
nextZ   d=d-1   b       * index iter
        gonc    loopZ

pixon   lc(5)   (34*64) * offset pour autres écrans
        b=c     a       * sauvegarde
        c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
        ad0ex           * adr. quartet 1er écran
        d0=a            * restaure d0
        a=a+b   a       * adr. quartet 2e écran
        ad0ex           * @       -     -    -
        c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
        ad0ex           * adr. quartet 2e écran
        a=a+b   a       * adr. quartet 3e écran
        ad0ex           * @       -     -    -
```

```
        c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
        d0=a            * @ quartet 1er écran
        gonc    nextpix * saut pour le pixel suivant
nopixon c=d     b       * index itérations à faire
        ?cbit=0 0
        goyes   testbit1
        c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
        ad0ex           * adr. quartet 1er écran
        d0=a            * restaure d0
        lc(5)   (34*64) * offset pour 2ième écran
        a=a+c   a       * adr. quartet 2e écran
        ad0ex           * @       -     -   -
        c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
        d0=a            * @ quartet 1er écran
testbit1
        c=d     b
        ?cbit=0 1       * 2e bit pair ?
        goyes   nextpix * oui alors pixel suivant
        c=dat0  xs      * quartet contenant le pixel
        c=c!d   xs      * OR pour noircir le pixel
        dat0=c  xs      * remet en place le quartet
nextpix d=d+d   xs      * pour position pixel suivant
        gonc    idnib   * si pas encore position 3
        d0=d0+  1       * après bit 0 -> 1 -> 2 ->
        d=d+1   xs      * bit 3 on revient à bit 0
*                       * et on avance d'un quartet
idnib   c=dat1  b       * index J
        c=c-1   b
        goc     outJ    * on sort de la boucle J
        dat1=c  b       * sinon sauve index J
        d1=d1+  2       * @ Px
        a=r0            * X(j)
        c=dat1  w       * Px
        c=a+c   w       * X(j+1) = X(j) + Px
        r0=c            * X(j+1) -> r0
        r2=c            * Xz(i) = X(j+1)
        a=r1            * Yz -> Yz(i)      (-> r3)
        d1=d1-  2       * @ index J
nextJ   goto    loopJ

outJ    d0=d0+  2       * nib 34  puis ligne I + 1
        d1=d1-  2       * @ index I
        c=dat1  b       * index I
        c=c-1   b       * I - 1
        goc     ?sym    * test avant sortie générale ?
        dat1=c  b       * sinon sauve index I
        d1=d1+  4       * saute les 2 index
        d1=d1+  16      * @ Py
        gosbvl  =out<c>in
        ?cbit=1 15      * appui de [ATTN]/[CANCEL] ?
        goyes   gout    * oui alors sortie générale
        a=r1            * Y(i)
        c=dat1  w       * Py
        a=a-c   w       * Y(i+1) = Y(i) - Py (->r1/r3)
        d1=d1+  16      * @ Xzo
        c=dat1  w       * Xzo
nextI   goto    loopI   * a:= Y(i+1) c:= Xzo d1 @ Xzo
?sym    ?st=1   1       * symétrie ?
        goyes   sym     * oui alors on continue
gout    goto out        * sinon sortie générale
sym     c=0     a
        lc(2)   16      * index octets par ligne
        d=c     a       * sauvegarde
        lc(5)   (34*64)
        b=c     a       * offset entre écrans
        lc(2)   29      * n-1 lignes à symétriser
        ad0ex
        d1=a            * @ début ligne sous axe
looprow rstk=c          * sauve index lignes
        lc(5)   (2*34)  * offset pour reculer au
*                       * début ligne au-dessus
        a=a-c   a       * ligne au-dessus
        d0=a            * @ -     -
        c=d     a       * index octets par ligne
loopoct rstk=c          * sauve index octets
        c=dat0  b       * octet du grob au-dessus axe
        dat1=c  b       * symétrise 8 pixels à la fois
        ad0ex           * adr. octet 1er écran
        d0=a            * restaure d0
        a=a+b   a       * adr. octet 2ième écran
        ad0ex           * @     -     -     -
        c=dat0  b       *       -     -     -
        d0=a            * restaure d0
        ad1ex           * adr. octet sous axe
        d1=a            * restaure d1
        a=a+b   a       * adr. oct. sous axe 2e écran
        ad1ex           * @     -     -   -   -   -
        dat1=c  b       * symétrise l'octet
        d1=a            * restaure d1 1er écran
        ad0ex           * adr. oct. 1er écran
        d0=a            * restaure d0
        a=a+b   a       * saut 2ième écran
        a=a+b   a       * saut 3ième écran
        ad0ex           * @ oct. au-dessus axe 3e écr.
        c=dat0  b       *   -       -       -   -   -
        d0=a            * restaure d0
        ad1ex           * adr. octet sous axe
        d1=a            * restaure d1
        a=a+b   a       * adr. oct. sous axe 2e écran
        a=a+b   a       * adr. oct. sous axe 3e écran
        ad1ex           * @     -     -   -   -   -
        dat1=c  b       * symétrise l'octet
        d1=a            * restaure d1 1er écran
        d0=d0+  2       * 8 pixels ou octet suivant
        d1=d1+  2       *         dito
        c=rstk
        c=c-1   b       * index d'octets - 1
        gonc    loopoct
        ad0ex           * a := adr. fin ligne copiée
        c=rstk
```

```
        c=c-1   b       * index de lignes - 1
        gonc    looprow
out     c=rstk          * enlève iter de la rstk
        gosbvl  =AllowIntr * autorise les interrupt.
        govlng  =GETPTRLOOP * retour au Rpl
************************************************
* produit de 2 nombres < à 2 en valeur absolue  *
* en entrée : a(w) & c(w) les 2 nombres à multiplier*
*                                               *
* en sortie :  a(w) = a*c    c(w) = 0           *
************************************************
sqr     c=a     w       * pour le carré
mpy     st=0    2       * pour le signe
        ?a=0    s       * positif ?
        goyes   apos    * oui alors test autre fact.
        a=-a    w       * sinon valeur absolue
        st=1    2       * pour négatif
apos    ?c=0    s       * positif ?
        goyes   mpy0    * oui alors produit
        c=-c    w       * sinon valeur absolue
        ?st=1   2       * 1er facteur négatif ?
        goyes   neg     * alors produit positif
        st=1    2       * sinon produit négatif
        gonc    mpy0    * saut pour le produit
neg     st=0    2       * pour produit positif
mpy0    b=0     w       * initialisation produit
        abex    w       * a(w) prod. b(w) muliplican.
mpy1    sb=0            * pour bit 0 nul
        csrb            * divise par 2
        ?sb=0           * bit 0 nul ?
        goyes   mpy2    * alors produit inchangé
        a=a+b   w       * sinon produit partiel
mpy2    b=b+b   w       * multiplicande * 2
        ?c#0    w       * reste bits multiplicateur ?
        goyes   mpy1    * oui on itération
        gosbvl  =ASRW5  * sinon divise par # 100000h
        ?st=0   2       * produit positif
        rtnyes          * oui retour
        a=-a    w       * sinon entier signé
        rtn
ENDCODE
  GBUFF
 ;
;
```

Bon voyage ... ou de zoom en zoom

Pierre nous a donné une méthode pour faire un zoom d'un sous-ensemble mais le procédé a deux contraintes :

il faut modifier le PPAR, celui-ci correspondant à la vue d'ensemble ce qui exclut d'opérer directement de zoom en zoom.

Aussi je vous propose le programme MZOOM qui permet d'aller de zoom en zoom automatiquement.

Mode d'emploi :

- En partant du GROB généré par MAND4CSI ou MANELDGT ou MAND3CSI avec le centre de coordonnées *Xc* et *Yc*, le pas horizontal *Px* et le pas vertical *Py* :

niveau 3 : (Xc , Yc)
niveau 2 : (Px , Py)
niveau 1 : le GROB

MZOOM fait entrer dans l'environnement graphique :

- Déplacez le curseur sur un angle du rectangle que voulez zoomer. Appuyez sur [ENTER] pour digitaliser ce point. Déplacez le curseur à l'angle opposé de la diagonale. Appuyez sur [ENTER]. Si vous avez une HP48s/sx vous pouvez changer la couleur du curseur avec la touche [+-]. Sortez de l'environnement avec CANCEL sur HP48g/gx ou ATTN sur HO48s/sx.

- Répondez au questions:

sym? vous avez choisi une zone que vous désirez symétrique par rapport à l'axe horizontal mais votre digitalisation est imprécise alors le programme fera le nécessaire.

carré? le programme peut corriger l'imprécision de digitalisation pour rendre la zone carrée

- Entrez les réponses dans la même ligne. et validez.

iter = ?

nn nombre d'itérations déjà proposé. Modifiez sinon validez directement si vous acceptez ce choix

Après l'exécution du programme vous trouvez sur la pile les paramètres correspondant au GROB qui est sur la pile et vous pouvez:

- Sauvegarder le GROB

- Si vous avez utilisé MAND4CSI vous pouvez l'afficher en nuances de gris avec le programme D4COL ci-dessous en n'oubliant pas de sauvegarder ou de faire DUP pour continuer de zoomer.

Le processus peut ainsi se répéter automatiquement en activant MZOOM avec les paramètres qu'il avait laissé sur la pile.

MZOOM
484.5 octets cksum # 827Dh

```
« STD
 { # 0d # 0d } { # 130d # 63d } SUB
  PICT PURGE PICT STO
  → C P I
  « C C→R P C→R DUP
```

```
    63 * SWAP 32 * 4
    ROLL + DUP ROT -
    ROT DUP 130 * SWAP
    65 * NEG 5 ROLL +
    DUP ROT + XRNG SWAP
    YRNG PICTURE                @ GRAPH pour HP48S/SX
    "sym.?  oui/non → 1/0
    carre?  oui/non → 1/0"
    "" INPUT OBJ→
    "iter = ? "
    I →STR
    INPUT OBJ→ ROT
    NOT → C1 C2 E I S
    « C1 C2 - C→R
      ABS 63 / SWAP ABS
      130 / SWAP E
      IF
      THEN + 2 / DUP
      END R→C C1 C2
      + 2 / C→R S * R→C
      SWAP I 3 DUPN
    »
  »  MAND4CSI          @ ou MANDELGT ou MAND3CSI
»
```

Exemples :

générons la vue d'ensemble avec:

```
        (-0.5 ,    0)
        (0.03 , 0.03)
                  20

3 DUPN      pour  utilisation ensuite avec MZOOM
MAND4CSI     <-----------------------------------|
                                                 |
sauvegarde et rappel du GROB                     |
sur pile (facultatif)                            |
                                                 |
DUP D4COLOR si GROB généré par MAND4CSI          |
pour visualisation en nuances de gris (facultatif) |
                                                 |
MZOOM >------------------------------------------|
```

Enfin voir en couleurs

Comme il a été expliqué ci-dessus (*Bébé prend des couleurs*) suivant le temps d'affichage des objets graphiques il est possible de les voir en nuances de gris. Le programme D4COLOR permet l'affichage en 3 ou 4 couleurs suivant que l'objet graphique est composé de 2 ou 3 parties correspondant à 2 ou 3 écrans. Ces objets peuvent être réalisés avec le programme MAND4CSI ou par tout autre programme qui construit des objets graphiques suivant ce même principe. D4COLOR détecte si le GROB est de hauteur supérieure à 128 lignes et dans ce cas affichera en 4 couleurs sinon en trois. Dans le cas d'un affichahe en 3 couleurs il y a une temporisation entre chaque affichage pour éviter d'avoir des couleurs claires baveuses (défaut du programme GREY de *voyage G*, encore un programme qui laisse à désirer). De plus il est compatible G et S.

Son emploi est simple : il prend en argument 1 GROB généré par MAND4CSI ou tout autre GROB similaire. La sortie se faisant par l'appui sur [ATTN] (HP48s_) ou [CANCEL] (HP48g_).

Le contraste est à adapter pour obtenir le meilleur aspect.

Voici ce programme.

```
*****************************************************
*                      D4COLOR                      *
*            153 octets   cksum # D219h             *
*         compatible HP48G/GX et HP48S/SX           *
*                                                   *
* programme d'affichage en trois ou quatre nuances  *
* de gris d'un objet graphique composé de 2 ou 3    *
* parties correspondant à 2 ou 3 écrans.            *
* La succession des affichages créant ces nuances   *
* de gris.                                          *
*     Auteur : Guy Toublanc          le 24/4/94     *
*                                                   *
* argument : 1 objet graphique spécifique pour ces  *
*            affichages                             *
*****************************************************
:: CK1NoBlame CK&DISPATCH1 TWELVE
  :: TURNMENUOFF
CODE
=savleftmargin   EQU #00100
=adrptrbitmap    EQU #00120
=savrightmargin  EQU #00125
=savvideosynchro EQU #00129
=savVDISP1       EQU #01F79
=DISP1CTLs       EQU #7050E   * pour HP48S/SX
=VDISP1s         EQU #7055B   * pour HP48S/SX
=DISP1CTLg       EQU #8068D   * pour HP48G/GX
        gosbvl   =PopASavptr
        gosbvl   =DisableIntr * interdit les inter.
        d0=a             * @ le grob
        d0=d0+   10      * @ la hauteur du grob
        st=0     0       * pour plus de 2 écrans
        a=dat0   a       * hauteur grob
        lc(5)    128     * hauteur 2 écrans
        ?a>c     a       * plus de 2 écrans ?
        goyes    3scr    * oui alors 3scr
        st=1     0       * sinon pour 2 écrans
3scr    d0=d0+   10      * @ données du grob
        ad0ex            * adresse 1er écran
        b=a      a       * adr. 1er écran -> b(a)
        lc(3)    (34*64) * offset entre écrans
        a=a+c    a       * adresse 2ième écran
        c=a+c    a       * adresse 3ième écran
```

```
        d=c     a       * sauvegarde
        d0=(5)  =savleftmargin
        c=dat0  s
        ?abit=0 0       * adresse paire ?
        goyes   nop     * oui on saute
        lchex   c
        dat0=c  1       * marge à 4
        d0=(2)  =savrightmargin
        c=0     a
        c=c-1   a       * pour # FFFh (-1)
        dat0=c  x       * marge à -1
nop     d1=(5)  =adrptrbitmap
        d0=(2)  =savvideosynchro
loop    c=b     a       * adresse 1er écran
        gosub   dispscr * affichage 1er écran
        c=a     a       * adresse 2ième écran
        gosub   dispscr * affichage 2ième écran
        ?st=0   0       * 3ième écran ?
        goyes   3Scr    * oui pour affichage
        lchex   800     * pour une temporisation
waitloop c=c-1  x       * boucle d'attente
        gonc    waitloop
        goc     ?out    * on saute affich. 3e écr
3Scr    c=d     a       * adresse 3ième écran
        gosub   dispscr * affichage 3ième écran
?out    gosbvl  #01eec  * out=c c=in
        ?cbit=0 15      * ? pas d'appui sur ATTN
        goyes   loop    * oui on reboucle
out     d0=(2)  =savleftmargin
        dat0=c  s       * restaure marge gauche
        lc(5)   =VDISP1s
        d1=(5)  =savVDISP1
        a=dat1  a
        d1=(5)  =DISP1CTLs
        ?a=c    a       * hp48s/sx ?
        goyes   restore * oui on garde d1
        d1=(5)  =DISP1CTLg * pour hp48g/gx
restore c=dat1  8       * récupère val. bitmap
        d0=(2)  =adrptrbitmap
        dat0=c  8       * restaure bitmap initial
        gosbvl  =AllowIntr * autorise les interrupt
        govlng  =GETPTRLOOP * retour rpl

dispscr dat1=c  a       * charge adresse écran n
vsync1  c=dat0  1       * bit 1 de la synchro vid
        ?cbit=1 1       * toujours à 1 ?
        goyes   vsync1  * oui on attemd encore
vsync2  c=dat0  1       * après passage à 1
        ?cbit=0 1       * est-il repassé à 0 ?
        goyes   vsync2  * si oui on reboucle
        rtn
ENDCODE
 ;
;
```

Aspects pratiques

Le programme MAND4CSI est polyvalent et peut très bien remplacer MANDELGT et MAND3CSI. En effet vous pouvez extraire du GROB l'une des trois parties. Voici un petit programme MPART pour vous aider à faire ces extractions. Son emploi est simple:

niveau 2 : le GROB généré par MAND4CSI
niveau 1 : 1 réel pour le numéro de la partie

MPART
243.5 octets cksum # 544Bh

```
«
  CASE 3 OVER ==
    THEN DROP { # 0d # 0d } { # 130d # 63d }
    END 2 ==
    THEN { # 0d # 64d } { # 130d # 127d }
    END { # 0d # 128d } { # 130d # 191d }
  END SUB
»
```

1 donnera le grob correspondant à MANDELGT ou bien à MAND3CSI avec iter positif.
2 donnera le grob correspondant à MAND3CSI acec iter négatif.
3 donnera le grob correspondant à l'union des trois ensembles de points fournis par MAND4CSI.

Ainsi en prenant les parties 1 et 2 vous aurez un GROB visualisable en 3 couleurs avec D4COLOR. Dans ce cas il sera peut-être nécessaire d'adapter le contraste.

Lorsque le GROB généré par MAND4CSI est stocké en PICT et que vous faites PICTURE sur HP48g/gx ou GRAPH sur HP48s/sx n'oubliez pas que que le GROB apparaît à mi-hauteur, pour voir la partie classique allez jusqu'en bas.

Vous pourrez constater des différences de détails avec les images des ensembles de Mandebrot dans les livres, cela est normal car la définition de l'écran d'un HP48 n'est pas comparable à celle d'une image obtenue par un ordinateur.

Vous trouverez les codes de tous ces pogrammes dans *le coin des codes*.

CONCLUSION

Comme quoi le travail de l'un peut en engendré un autre. Merci encore à Pierre du sien qui, s'il était perfectible, m'a permis d'attaquer ce problème de fractales et de graphisme avec plaisir.

Guy Toublanc (276)

HP95 / HP100

POUR QUELQUES DISQUETTES DE PLUS...

Comme annoncé le mois dernier, voici deux nouvelles disquettes de programmes destinés aux HP95 et HP100. Mais si celles que nous vous avions présenté précédement contenaient des programmes pouvant fonctionner sans problèmes sur les deux machimes, ce n'est plus le cas avec la distribution de ce mois ci. Faites donc attention pour vos commandes.

Rapellons que, comme pour les précédents articles, celui ci ne présente que les programmes qui me semblent les plus intéressants, et non le contenu total des disquettes.

Disquette HP95

Notez tout d'abord que cette nouvelle disquette contient quelques nouvelles versions de programmes déjà présents sur les deux premières disquettes. Il m'a semblé plus simple d'agir ainsi, car cela evite aux anciens utilisateurs d'avoir à faire des mises à jour alors que le nombre de fichiers modifiés est très faible.

Directory : \UTILS

Il peut être utile, lors du lancement de certains programmes MS-DOS, de pouvoir positionner la fenêtre 40×16 de l'affichage du HP95 à un emplacement autre que le coin Supérieur-Gauche. Un petit utilitaire, nommé WINPOS, vous évite les toujours désagréables pressions simultanées de [Alt][Flèches].

Si vous trouvez que la lecture des fichiers textes sur un écran aussi large que celui du HP95, peut être fatiguant à la longue (à cause de l'obligation de faire des longs retours à la ligne), vous pouvez utiliser VR95, qui permet de basculer l'affichage du texte de 90°, et de prendre le HP95 en main de la même façon qu'un livre. De plus, l'affichage s'effectue en utilisant une des nombreuses polices, en différents styles (gras, proportionnel) et plusieurs tailles.

Si vous avez pris la (bonne) habitude d'effectuer très régulierement des sauvegarde du contenu du HP95 sur votre IBM PC, mais trouvez les différentes manipulations à effectuer trop lourdes, vous pouvez utiliser CARDBAK qui sauvegarde le contenu total d'une carte mémoire dans un fichier unique du PC. Cela se fait en utilisant d'un coté (le PC) le programme CARDBAK.EXE et de l'autre la commande $SERVER (non documentée) du HP95. Mais le fait que vous stockez sur le PC un fichier unique vous interdit d'effectuer des modifications de vos sauvegardes directement sur le PC. De plus il ne permet pas de sauvegarder le contenu du disque C: et ne fonctionne pas avec les cartes de type SunDisk. Mais d'un autre côté, les gros avantages de ce programme sont la facilité d'utilisation, et surtout que vous n'avez besoin d'aucun programme à charger sur le HP95. Donc économie d'espace disque et aucun problème en cas de perte totale du contenu de la machine.

Un des petits défauts du HP95 est que si une alarme survient lorsque l'on est sous le DOS, elle est ignorée. DOSALARM corrige ce problème et ajoute une commande permettant d'activer une alarme à une heure donnée.

Directory : \CALC

Parmi les reproches faits à la calculatrice installée dans le HP95, les manques de fonctions statistiques et de conveersions de bases (binaire, hexadécimal...) revienent souvent. Les programmes STCALC et HEXCALC résolvent ces problèmes. La grande diférence entre ce dernier et CMCALC (inclus dans la disquette 1) est que c'est un programme DOS, et non System Manager.

Directory : \TEXTES

Il contient une petite sélection de fichiers texte, dont certains auraient pu faire l'objet d'articles, si j'avais eu le temps de m'en occuper... Notez qu'is sont tous en Anglais.

Tout d'abord, pour ceux que cela pourrait intéresser, j'ai inclus le communiqué officiel annoncant le HP95LX.

Pour les <encore> débutants, un fichier FAQ.95 contenant les réponses aux questions les plus courament posées.

Enfin, pour les curieux, deux fichiers (HP95.HW et INTHE.95) donnant quelques informations sur le coté "hardware" et certains aspects cachés de la machine.

Directory : \DIVERS

Si vous voulez montrer les possibilités du HP95 à vos proches, quoi de mieux que le programme inclus dans la carte de démonstration faite par HP lors de la sortie de la machine ?

Directory : \IMAGES

De nombreuses images au format PCX 240×128, pour ceux qui veulent changer leur "Topcard". Les sujets

sont très divers, mais on peut noter un certain attrait autour de la série "Star Trek"...

En prime, un programe (PIXER) permettant de manipuler ces fichiers.

Disquette HP100

Directory : \UTILS

Dans un précédent JPC, je vous avais présenté une commande permettant de modifier la valeur du Zoom. Cependant, faute de renseignements suffisants, je n'avais pas inclus d'options permettant de basculer en affichage monochrome ou 4 niveaux de gris. Heureusement, la commande ZOOM termine le travail là où je l'avais commencé.

Si vous avez des problèmes en utilisant des programmes de communication sous le System Manager, c'est probablement dû au fait que cette interface modifie d'elle-même l'état de la liaison série. Pour corriger cela, vous pouvez utiliser ASERCTL, un petit résident vous permettant de l'activer ou de la désactiver.

Que ce soit pour transférer le contenu de votre répertoire téléphonique HP95 vers le HP100, ou convertir au format texte un fichier de Base de données du HP100, quelques petits utilitaires sont inclus dans la disquette.

Si vous trouvez les caractères du HP100 peu lisibles (surtout en mode 80×25), un petit programme (HELV100) permet de les remplacer par des polices "Helvetica". Le gain de visibilité est tout simplement fabuleux...

Et si vous trouvez que les caractères affichés par HELV100 ne sont toujours pas assez lisibles, vous n'avez plus qu'à utiliser FCL pour créer les votres !

Tout comme 95BUDDY (inclus dans notre disquette n°1) est devenu le plus célèbre programme écrit pour le HP95, 100BUDDY semble être bien parti pour faire de même. Pour ceux qui ne le connaissent pas, il s'agit d'un programme étendant les possibilités et l'ergonomie des applications internes. Je ne peux que les conseiller de l'essayer. Cependant, puisque le HP100 ne possède plus qu'une seule langue de travail, il ne fonctionne pas totalement sur le version Française de la machine. Voir l'article suivant pour plus d'informations à ce sujet.

Si vous avez lu ma présentation de la disquette précédente, vous ne devriez pas avoir de problèmes pour savoir à quoi sert un programme s'appelant VR100...

Par le même auteur que le programme précédent, CLIPVUE permet de transférer le contenu du presse-papier dans un fichier, ou au contraire de charger le presse-papier avec le contenu d'un fichier texte.

Enfin, si vous désirez utiliser des programmes MS-DOS relativements gourmands en mémoire, il existe maintenant un driver EMS spécifique au HP100, utilisant le disque C: pour simuler la mémoire étendue.

Directory : \TEXTES

Tout comme pour le HP95, j'ai inclus le communiqué officiel annonçant le HP100LX.

Plus quelques fichiers apportant des précisions à propos de certains aspects non cités dans la documentation du HP100 : Les icônes ou les paramètres à utiliser pour mieux contrôler l'exécution des applications externes à partirt du System Manager.

Directory : \IMAGES

Quelques images au format PCX 640×200. Je n'ai pas encore eu le temps de décider du contenu exact de ce directory.

Directory : \ICONES

Un des grandes nouveautés du System Manager du HP100 est la possibilité d'attacher des icônes aux à certaines applications. Ce directory contient donc quelques icônes, correspondant soit à des programmes connus, soit sont totalement inventées pour votre propre usage.

Directory : \DIVERS

Le fichier 42ALAR.ZIP contient une très importante collection de fichiers d'alarmes musicales. A ce propos, si vous désirez écouter plusieurs fichiers ".snd", les manipulations deviennent vite très fastidieuses, car il faut à chaque fois recopier le fichier d'alarme dans le directory C:_DAT, sous le nom ALARM.SND, puis acceder à l'écran de configuration des alarmes... Pour simplifier ces manipulations, je vons conseille d'assigner la macro suivante à la touche F1 (par exemple) :

```
{F2}c:\_dat\alarm.snd{Entrée}{Entrée}{Menu}obfas{Gest}
```

Ensuite, il ne vous suffit plus qu'à vous placer sur le fichier ".snd" que vous voulez écouter, puis à appuyer sur [Fnt][F1] pour que le HP100 fasse tout le travail à votre place. Y compris le retour sur le Filer après que la musique ait été testée.

Notez cependant que si certains fichiers donnent des résultats très spectaculaires ("Eruption", par exemple), d'autres, comme "Getup" contiennent des périodes, durant parfois plus de dix secondes, au cours desquelles aucun son n'est émis. Ne touchez donc à rien et attendez que la macro se termine en retournant sur le Filer.

Directory : \JEUX

Pour finir, cédant à une forte pression (enfin, pas si forte que ça !) de certains utilisateurs, j'ai ajouté ici quelques petits jeux fonctionnant en mode CGA. Bien sûr, je n'ai pas pu eu le droit d'y mettre "Flight Simulator", mais certains d'entre eux, mmême si il sont déjà ancien, devraient vous faire passer quelques heures...

Parmi ceux ci, GTHOR, qui est un des meilleurs programmes d'Othello. Si il est très difficile de le battre sur un PC équipé d'un 386 rapide (même au niveau 1 pour quelqu'un battant presque systématiquement la version incluse dans Windows) il revient à un niveau plus civilisé dès qu'il tourne sur le HP100.

Parmi les jeux d'echecs, j'utilise CHESS (oui, je sais, c'est très original comme nom !). D'un assez bon niveau, il possède en plus deux particularités pour un programe de son age (1985) : un mode démo et une option d'affichage en 3D, assez agréable sur le HP100. Il est à noter cependant qu'il semble y avoir un problème dans la gestion du temps de réflexion, puisque le temps effectif est trois ou quatre fois suppérieur à ce qui est sélectionné. Autre remarque (valable aussi pour d'autres programmes) : Au lancement, il demande le type d'adaptateur utilisé. Répondez "Couleur" et jamais "Hercule".

Parmi les autres archives, un autre jeux d'échecs, chinois cette fois ci. Le fichier ZIP contient deux programmes indépendants CCHESS et XQ. Je ne saurais vous dire lequel est le meilleur, mais XQ possède l'avantage d'être infiniment plus compact.

Et en prime...

Ne sachant pas encore quand nous serons en mesure de vous présenter une autre disquette commune pour les HP95 et HP100 (j'ai passé tout ce qu'il y avait d'intéressant), vous trouverez dupliqué dans les deux nouvelles disquettes certains oublis et quelques nouvautés qui auraient du passer dans la disquette parue le mois dernier.

Par exemple, pour les utilisateurs de cartes de type SunDisk (non Stackées) : 1KCLUST, un programme reformattant la carte avec des clusters ayant une taille de 1ko, au lieu de 4. D'où une économie très apréciable d'espace disque. Bien sûr, n'exécutez ce programme qu'une fois que vous aurez sauvegardé le contenu de votre carte. D'autant plus que je n'ai pas encore eu l'occasion de le tester...

Ici se termine la présentation de ces deux disquettes. Il ne vous reste plus qu'à en découvrir le reste du contenu...

Jacques Belin (123) •

SOYEZ COPAINS AVEC 100BUDDY

Les anciens utilisateurs de HP95 connaissent 95BUDDY, un impressionnant programme écrit par Jeffrey Mattox permettant d'accroitre de façon très significative l'ergonomie du System Manager. Depuis quelques mois, le même auteur a renouvelé l'exploit en sortant une version spécifique au HP100.

Malheureusement, si la version dédiée au HP95 fonctionnait parfaitement (à condition qu'on configure la machine en version Anglaise), la nouvelle version pose plus de problèmes, puisqu'il est annoncé qu'elle ne fonctionne pas sur la version Française du HP100.

Cela est dû principalement au fait que le HP100 ne contient plus qu'une seule langue dans la Rom, et que le programme fait souvent appel à des resources dépendantes des messages ou des menus, sous la forme d'appels de macros clavier.

Par exemple, il est possible, sur la version US, de sélectionner l'ordre d'affichage des fichiers en appuyant sur une des touches [Ctrl][F], [Ctrl][E], [Ctrl][S] ou [Ctrl][D], suivant le fait que l'on veille un tri par nom, extension, date ou taille. En fait, BUDDY (que ce soit pour le HP95 ou le HP100) ne répète rien d'autre que les actions que vous feriez manuellement, en accédant aux différents menus grâce aux lettres "clés". Soit, pour l'exemple cité, la simulation de la pression des touches [Menu], [O] (menu Option), [S] (ligne "Sort" puis sélection de l'ordre de tri. Malheureusement, "Sort" se dit "Tri" en Français... Le programme ne trouve donc pas le bon menu, et est donc perdu pour la suite des opérations.

D'autant plus que dans notre exemple, la lettre "S" est utilisée comme clé pour accéder à la ligne "DOS". D'où un comportement assez particulier...

Cependant, d'autres fonctions ne fonctionnent pas pour d'autres raisons plus complexes à expliquer. Cet article a donc pour but de vous indiquer les fonctions supportées par la version Française, ainsi que de vous indiquer quelques moyens pour contourner certains problèmes.

Afin de simplifier les choses, je suivrais ici l'organisation de la documentation du programme, en conservant les titres et les numéros des chapitres originaux. Je n'entrerais pas dans les détails du fonctionnement de chaque fonction. Il vous est donc conseillé de vous munir de la doc afin de savoir de quoi je parle exactement, et si possible d'avoir installé le programme sur votre HP100 et au moins jeté un coup d'oeil sur les différents écrans du "frontal" `100BUDDY.EXM`.

A ce propos, notez que, pour effectuer ces essais, j'ai systématiquement activé toutes les fonctions sélectionnables par le "frontal". Ne vous étonnez donc pas qu'une fonction ne fonctionne pas sur votre machine, alors que j'ai dit le contraire. Dans ce cas, vérifiez que vous ne devez pas changer une option.

3. Instalation et usage général

Aucun problème pour toute cette partie, si ce n'est que pour la désinstalation (chap. 3.5), j'ai crashé le HP100. Mais cela est peut-être dû tout simplement à la méthode particulière que j'utilise pour lancer le System Manager.

4.1 Fonctions générales

La plupart des fonctions décrites dans ce chapitre fonctionnent parfaitement, à l'exception de `MENU-DEL` (simulation de macro, comme expliqué plus haut) et `FN-Q`, dont je n'ai pas pu obtenir une visualisation de l'état du flag et tester le foctionnement des fonctions associées.

Si il est possible d'afficher l'écran d'aide de Buddy (en double-cliquant sur `F1`), j'ai constaté quelques problèmes lors du retours sur l'application précédente.

Enfin, je n'ai pas pu tester toute la partie concernant le mot de passe, car elle n'est accessible qu'aux utilisateurs enregistrés.

4.2 Conversions Clavier

Toutes les fonctions agissent parfaitement (n'oubliez cependant pas d'activer les différentes options dans le fichier "EXM") à l'exception de la touche `DEL`, qui est censée annuler la dernière opération. En fait, elle efface totalement le caractère.

4.3 SmartCaps

Aucun problème.

4.4 Fenetres d'ouvertures de fichiers

Ici, par contre, aucune des fonctions ne semblent fonctionner. A l'exception du positionnement sur un fichier dont on a tapé la première lettre.

4.5 Liens avec d'autres applications

Je n'ai pas eu le temps d'approfondir cette section, mais son fonctionnement semble peu sartisfaisant.

5.1 Fonctions du Filer

Parmi les fonctions que j'ai pu tester, il semble y avoir autant de fonctions correctes que de fonctions invalides. Par contre, certains problèmes peuvent être contournés.

Fonctions correctes :

- Le curseur descend bien sur la deuxième ligne des fichiers.
- Bon affichage des valeurs des batteries, de l'état du port série, du timeout et de la mémoire vive disponible.
- Modification de la valeur du Timeout et de l'état du port série.
- Affichage du contenu d'un fichier quelquonque, à l'aide de la touche `[Entrée]`.
- Descente d'une ligne du pointeur après sélection d'un fichier.
- Descente dans l'arborescence grace aux touches `[\]` et `[.]`.
- Gestion des touches (flèches, Del...) pendant la visualisation d'un fichier texte.

Fonctions incorectes :

- Pas d'affichage du flag "Q". Je n'ai pas testé la fonction associée.
- Apparament, mauvais fonctionnement de la gestion du lancement des fichier "BAT".

- Lorsque l'on appuie \- Impossibilité de lancer un programme correspondant à l'extension du fichier sélectionné.

- En fait, impossibilité de lancer automatiquement l'édition d'un fichier visualisé dans le Filer : l'éditeur se charge bien, mais aucun fichier n'est chargé. Dommage, c'est une des fonctions les plus importantes de Buddy...

- Action des combinaisons de touches `Fn-I`, `F6`, `F9`, `F10`, `F2-F2`, `F3-F3`, `Ctrl-F..Ctrl-D`, `=`, `Menu-f-o` ou `y`.
- Fonctions relatives aux recherches de chaines.
- Le contrôle du Light Sleep (par `Ctrl-L` semble fonctionner, mais l'affichage du message se fait dans la zone des fichiers !

Contournement de problèmes :

- Certaines commandes de déplacement du pointeur de fichier, ainsi que l'accèe aux répertoires sont inopérants dans certains cas (par exemple lorsque l'écran est divisé en deux). Dans ce cas, je vous conseille de désactiver Buddy (par ςΓυσμ≠ς-≠), d'effectuer vos modifications, puis de réctiver Buddy (par ςΓυσμ≠ς+).

5.2 Fonctions du Setup

Tous les affichages fonctionnent, mais il n'est pas possible de lancer les écrans de configuration à l'aide des touches de fonctions, car elles agissent comme des macros (voir au début de l'article).

5.3 Fonctions de l'éditeur

Aucune commande ne fonctionne, si ce n'est les combinaisons `Ctrl-DEL` et `Ctrl-BACKSPACE`, lest touches `F6`, `F7`, `F8` et les fonctions citées dans le chapitre 5.6.

5.4 Phone, Note taker, Database

Voir chapitre 5.6.

5.5 Fonctions de l'agenda

Ne fonctionnent pas, à l'exception des fonctions citées dans le chapitre 5.6

5.6 Fonctions du "Note Taker"

Toutes les fonctions semblent être activables.

5.7 Fonctions de Lotus

Aucune ne semble fonctionner, si ce n'est l'affichage concernant les touches fonctions.

5.8 Fonctions de la calculatrice

La touche "p" ne semble pas pouvoir être activée.

5.9 Heure mondiale

Non testé (fonctions réservées aux utilisateurs enregistrés.

5.10 Application Manager

Semble ne pas fonctionner correctement.

5.11 Macros Systeme

Les touches `F9` et `F10` ne fonctionnent pas correctement.

5.12 Fonctions Dos

Le contrôle de la taille du curseur sous DOS semble fonctionner, mais il semble (cela m'est arrivé une fois) qu'il y ait un problème pouvant entrainer un crash du HP100 lorsque l'on appuie sur la touche `Fn-C`.

Les fonctions de SmartCaps semblent fonctionner parfaitement.

Je n'ai pas testé les fonctions relatives à la liaison série.

6. Fonctions de 100BUDDY.EXM

Fonctionnement normal (application System Manager classique.

J'espère que cet article vous aura donné suffisement d'informations pour que vous puissiez décider si vous allez installer ce programme.

Mais avant de terminer, je voudrais dire (pour répondre à certaines personnes) que ce sujet montre qu'il est tout à fait possible de faire des articles sans avoir aucune notion de programmation. Ici il suffisait de lire la doc et d'essayer les différentes fonctions. J'espère donc qu'il vous donnera aussi des idées d'articles...

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 différent 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 obtiendriez un fichier au transfert impossible dans la HP48.

Programmes HP48 et HP28S

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 le programme correspondant à votre machine:
- ASSCOD48 pour HP-48 ou ASSCOD28 pour HP-28S
(*attention aux parties distinctes HP-48 ou HP-28S*)
cela avec la plus grande attention car un mauvais fonctionnement peut entraîner un désordre fatal pour les objets contenus dans votre machine. La commande SPEED (JPC-85 page 12) peut être incluse au début de ASSCOD28. NEWLINE (↵) s'obtient sur HP-48 par [flèche bleue] puis [.].

Pour tout assemblage de chaîne de codes procédez de manière suivante :

1- lancez le programme ASSCOD48 ou ASSCOD28
2- donnez le nombre d'octets (1/2 octet compris) puis validez avec ENTER.
3- tapez chaque ligne de codes après son numéro à 3 chiffres et sans les espaces puis validez.
4- tapez la somme de contrôle sm, validez. S'il y a erreur la ligne de codes sera demandée à nouveau. Pour HP-28 l'appui sur EDIT fera apparaître la ligne des codes qui pourra être corrigée, ensuite relancez avec CONT.
5- lorsque l'objet apparaîtra sur la pile stockez le dans la variable donnée en tête.
Nota: des versions ASSCODxx plus performantes (System Rpl et Assembleur) existent dans JPC numéro 95.

ASSCOD48 1114.5 octets **ASSCOD28** 1032.5 octets
cksum # 621Eh cksum # 55CEh

```
@ ******************* HP-48 et HP-28S ******************
« RCLF « CLLCD 3 DISP 1 DISP ""
DO DUP 4 DISP
 DO UNTIL KEY END
@ ************ HP-48 ************ @@ ******* HP-28S ******
 IF DUP 51 == THEN DROP 1        @@ IF DUP "ENTER" ==
 ELSE IF DUP 55 == THEN DROP     @@ THEN DROP 1
   1 OVER SIZE 1 - SUB           @@ ELSE
  ELSE                           @@  IF DUP "BACK" ==
   CASE DUP 17 < THEN 54 + END   @@  THEN DROP 1 OVER
        DUP 66 < THEN 7 - END    @@    SIZE 1 - SUB
        DUP 76 < THEN 20 - END   @@  ELSE
        DUP 86 < THEN 33 - END   @@
        DROP 48 END CHR          @@
@ ******************* HP-48 et HP-28S ******************
  + DUP SIZE  1 + 5 MOD NOT 1  FC? AND IF THEN " " +
    END END 0 END UNTIL END  »
HEX 64 STWS "nombre d'octets ?" ""
@ ************ HP-48 ************ @@ ******* HP-28S ******
INPUT                            @@ 1 SF 3 PICK EVAL
@ ******************* HP-48 et HP-28S ******************
1 CF STR→ 2 * 16 DUP2 / IP 3 ROLLD MOD
DUP2 0 > + SWAP 1 + # 0h ""  1 5 ROLL
FOR i
 DO "ligne " i 1 - R→B # 1000h + →STR 4 6 SUB + DUP
  "                                              @ NEWLINE
  codes ?" + "---- ---- ---- ----" 7 PICK i <
  IF THEN 6 PICK DUP 4 / IP + 1 SWAP OVER - SUB END
  1 FS?C IF THEN DROP2 SWAP
@ ************ HP-48 ************ @@ ******* HP-28S ******
  OVER " a corriger" +           @@
  { -1 } ROT + INPUT             @@ CLMF HALT
@ ******************* HP-48 et HP-28S ******************
  ELSE 8 PICK EVAL END DUP
  WHILE DUP " " POS DUP
  REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 + 25 SUB +
  END DROP 0 OVER SIZE 1 SWAP
  FOR j OVER j DUP SUB NUM j * + NEXT
  6 PICK + DUP # FFFh AND
  "                                              @ NEWLINE
  somme de controle ?" 6 ROLL SWAP + "==="
  11 PICK EVAL "#" SWAP + STR→ ==
  IF THEN 1 ELSE DROP2 1000 1 BEEP 1 SF 0 END
 UNTIL END ROT 5 ROLL DROP2 3 ROLLD +
 NEXT 5 ROLLD 4 DROPN
@ **** HP-48 **** @@ ********** HP-28S **********
"GROB 8 " OVER   @@ "" SWAP 1 OVER SIZE
SIZE 2 / " " +   @@ FOR j DUP j DUP 15 + SUB "#"
+ SWAP + STR→    @@  OVER SIZE 1
# 4017h SYSEVAL  @@  FOR i OVER i DUP SUB + -1 STEP
# 62B9Ch         @@  SWAP DROP STR→ ROT SWAP
SYSEVAL NEWOB    @@  # 3B82h SYSEVAL  SWAP 16
SWAP STOF »      @@ STEP DROP # 20238h SYSEVAL
                 @@ # 4F3Dh SYSEVAL SWAP STOF CLMF »
```

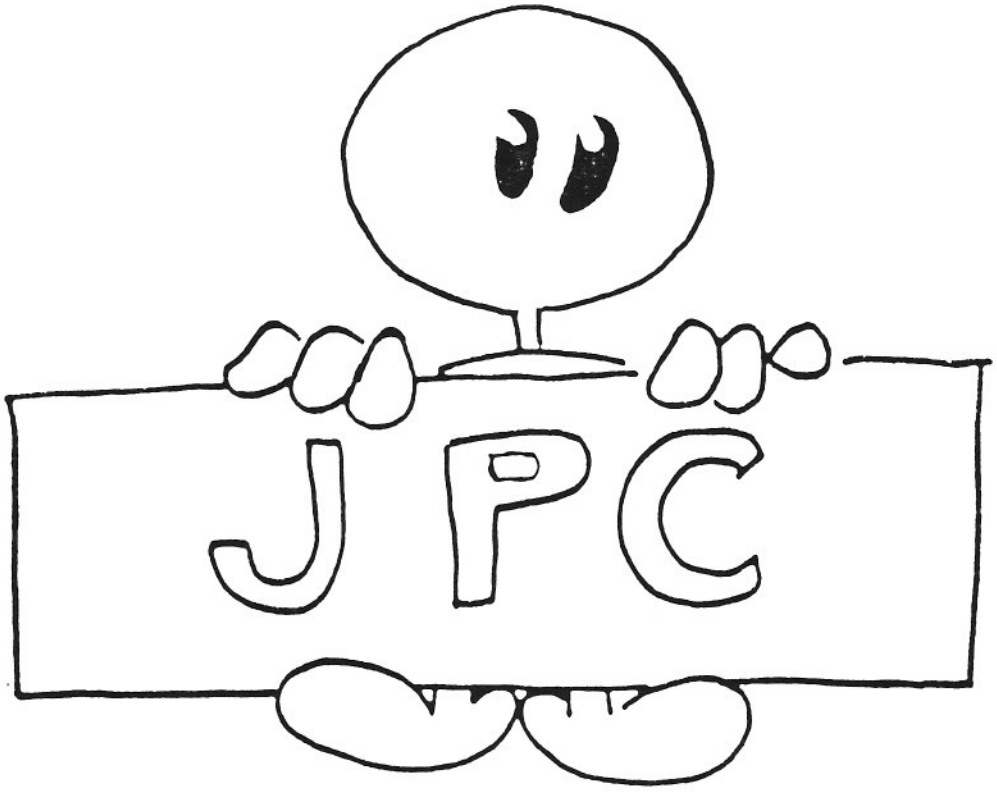
```
SYSRCL                    (HP28)
# AFC2h               15 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C07   E08
001: 1270 8320 209F 20     3B1


FRE                       (HP28)
# 7B76h               15 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01C8 4001 6C0E   D77
001: D211 28D1 109F 20     3A2


CRENAME                   (HP28)
# 6631h               15 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C07   E08
001: C170 DE44 009F 20     485


B→SB                      (HP28)
# 0837h               15 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C07   E08
001: 1270 DA34 009F 20     456


SB→B                      (HP28)
# 0858h            12.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C001 6C03   CC4
001: CE13 09F2 0           660


→PRG                      (HP28)
# 2158h               40 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C03   DC8
001: B170 76C2 0296 7048   A23
002: 1200 BB11 3ACD 376C   916
003: 20B3 5C0B EE30 09F2   78D
004: 0625 C009 F200 9F20   47B
```

```
PARSE                     (HP28)
# F13Fh            22.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C07   E08
001: C170 76C2 07FA E04F   D3F
002: 9C00 9F20 09F2 0      0A1


R→SB                      (HP28)
# 31A2h               15 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C03   DC8
001: B170 B35C 009F 20     44D


SB→R                      (HP28)
# F8CBh               10 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C001 6C00   C94
001: 9F20                  EAF


COMPIL28                  (HP28)
# 9FC9h            149.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C09   E28
001: F170 76C2 0D3F 4069   BC0
002: C20E D000 8F18 0501   7F4
003: 4710 8134 8F49 E201   4E5
004: 1013 1136 E281 8FA9   325
005: D713 0174 1641 4384   EC0
006: 0340 D9E0 8A6E 0850   C6A
007: 818F B417 4CF5 908D   B52
008: 5E52 1AF2 2B33 C45A   9C2
009: 2015 B090 CC21 5379   6E3
00A: 7632 1C41 4318 E140   32A
00B: 1641 7F17 4818 FBF5   240
00C: 2D41 1158 0160 170C   E53
00D: F50C 8605 0184 1431   9C7
00E: 4016 4110 1311 7414   4A7
00F: 710A 1371 098D A79E   37C
010: 169C 20B1 0001 1113   E66
011: 311A 1451 318D CE52   C6F
012: 109F 2009 F20         A6F
```

```
COMPIL48                  (HP48)
# 5C33h              139 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0D29 51D9 F81F   F34
001: 3040 D9D2 0756 60CC   D59
002: D209 E000 8FB9 7601   A93
003: 4710 8134 8F91 0301   658
004: 1013 1136 E281 8FA9   498
005: D713 0174 1641 4384   033
006: 034E 1632 8A6E 0850   D52
007: 818F B417 4CF5 6066   B15
008: 80AF 22B3 4E25 A120   825
009: 15B0 90CC 2153 7976   549
00A: 321C 4143 18E1 4016   15C
00B: 417F 1748 18FB F52D   0BF
00C: 4111 5801 6017 0CF5   DD6
00D: 0C86 0501 8414 3140   8EE
00E: 1641 1013 1174 1471   3FD
00F: 0A8F 1766 1111 1311   F18
010: 1214 18D3 4150 B213   B33
011: 0B21 30               F60


MANDELGT                  (HP48)
# EAA7h              363 octets

     0123 4567 89AB CDEF   sm

000: D9D2 00FE 8111 9201   C36
001: 2200 D9D2 079E 60D9   AA1
002: D20E 9056 FBD8 1262   866
003: 05CB 9A28 8130 4B2A   623
004: 2178 A28D A16D 9D20   495
005: 029A 29F3 45E4 A206   230
006: 0000 0322 300B E35B   F79
007: 2130 9F34 5322 30B2   BBA
008: 130E 9016 009A 29F3   93A
009: 4532 230C 2D50 A321   5D3
00A: 6E8F 60A3 216E 8F60   3F0
00B: A815 0A81 5032 230C   028
00C: 2D50 C121 6E8F 60BB   F40
00D: F06E 8F60 A815 0A81   CB8
00E: 50AD B464 2D46 F851   AB3
00F: 149F 21CF 1462 50F4   84A
010: CCD2 02C1 008F C152   5B0
011: 3131 1B97 F101 4213   195
012: 0142 1301 6F16 3AF2   F2E
013: 2480 F5AF 7179 147C   D63
014: E061 7B31 F314 D173   A84
015: 1537 AF8A 70A6 E59F   A4D
016: 17F3 1E1A E715 77AF   9A2
017: 5A71 A6F5 9F17 F157   810
018: 7B72 1557 17F1 5371   487
019: CFA7 0101 1081 0A1C   13D
01A: F1CF 1C13 1281 4DAA   FD8
01B: 3B27 07AE 7061 0371   C1A
```

```
01C: B010 4112 77A0 11CA   968
01D: 729F 7331 1CB7 A118   713
01E: A7A1 22A7 411B 7680   3E6
01F: 119A 7AA6 F51C 1562   1AC
020: 0E2F 1542 A275 8016   E10
021: 0B27 14FA 6E42 214D   C05
022: 1711 1015 77A7 2108   80B
023: 10A1 111C 16A7 F161   557
024: 1C11 4FA6 E402 14D1   2C2
025: 7317 F111 1577 B7A1   028
026: 7F15 776E 2F07 8D34   E40
027: 150A F684 2948 80BF   CC7
028: 8852 94A3 1BFA 8728   B2D
029: 0852 5508 42AF 1AFC   ABB
02A: 8228 1E83 250A 70A7   81A
02B: 597E CE8F 5E5D 0862   6EF
02C: 00BF 8015 6621 B213   32D
02D: 0B21 30               75A


MAND3CSI                (HP48)
# 422Ch          441.5 octets

     0123 4567 89AB CDEF   sm

000: D9D2 00FE 8111 9201   C36
001: 2200 D9D2 079E 60D9   AA1
002: D20E 9056 FBD8 1262   866
003: 05CB 9A28 8130 4B2A   623
004: 2178 A28D A16D 9D20   495
005: 029A 29F3 45E4 A206   230
006: 0000 0322 300B E35B   F79
007: 2130 9F34 5322 30B2   BBA
008: 1309 9016 E8F6 0C2D   A78
009: 50C1 216E 8F60 C121   7B5
00A: 6E8F 60A8 150A 8150   472
00B: 5923 0C2D 50C1 216E   217
00C: 8F60 BBF0 6E8F 60A8   172
00D: 150A 8150 ADB4 642D   FCD
00E: 46F8 5114 9F21 CF14   DCE
00F: 6250 F4CC D209 6200   AA8
010: 8FC1 5238 F511 1013   66E
011: 11B9 7F10 1421 3014   1C2
012: 2130 16F1 63AF 2248   F23
013: 0F5A F717 9157 7089   C42
014: 4A80 850B FA8F 706D   B9B
015: 0CE0 617B 31F3 14D1   921
016: 3710 8137 1731 537A   590
017: F8A7 0A6E 59F1 7F31   40E
018: E1AE 7157 7AF5 A71A   325
019: 6F59 F17F 1577 B721   0B5
C1A: 5571 7F15 371C F97C   FB8
01B: C185 1128 1370 6310   ACD
01C: 214D 0713 5118 A701   6E7
01D: 0110 810A 1CF1 CF1C   651
01E: 1312 814D AA3B 2707   3F5
01F: AE70 6103 7411 1041   ED8
020: 127A 0111 CA72 9F34   C9B

021: 1860 04AE B808 A063   A2E
022: 5721 1CB7 A118 A7A1   854
023: 22A7 411B 77D0 119A   5ED
024: 7AA6 F5FA 1562 0E2F   4C8
025: 1542 A275 8016 0B27   146
026: 14FA 6E42 214D 1711   DE3
027: 1015 77A7 2108 10A1   A0F
028: 111C 1686 F161 1C11   6C6
029: 4FA6 E4E2 14D1 7317   402
02A: F8FC EE10 808B F351   23F
02B: 1115 77B7 A17F 1577   000
02C: 6E0F 8619 3132 130D   C78
02D: 231D 1D73 144D 5E01   9DD
02E: 3131 0114 B148 1611   554
02F: 71A6 E50F 133C F5FD   579
030: 078F 5E01 08D3 4150   1F3
031: AF68 4294 880B F885   027
032: 294A 31BF A872 8085   DCC
033: 2550 842A F1AF C822   C75
034: 81E8 3250 A70A 7597   9DF
035: ECE8 F5E5 D086 200B   7CE
036: F801 5662 1B21 30B2   423
037: 130                   54A


MAND4CSI                (HP48)
# 5B1Fh            545 octets

     0123 4567 89AB CDEF   sm

000: D9D2 00FE 8111 9201   C36
001: 2200 D9D2 079E 60D9   AA1
002: D20E 9056 FBD8 1262   866
003: 05CB 9A28 8130 4B2A   623
004: 2178 A28D A16D 9D20   495
005: 029A 29F3 45E4 A206   230
006: 0000 0322 300B E35B   F79
007: 2130 9F34 5322 30B2   BBA
008: 130E 9016 009A 29F3   93A
009: 4532 230C 2D50 A321   5D3
00A: 6E8F 60A3 216E 8F60   3F0
00B: A815 0A81 5032 230C   028
00C: 2D50 C121 6E8F 60BB   F40
00D: F06E 8F60 A815 0A81   CB8
00E: 508D D464 2D46 F851   AA2
00F: 149F 21CF 1462 50F4   839
010: CCD2 0E23 008F C152   5AA
011: 38F5 1110 1311 B97F   314
012: 1014 2130 1421 3016   DB6
013: F163 AF22 480F 5AF7   CC8
014: 1791 577C E061 7B31   A21
015: F314 D137 1081 3717   5FC
016: 3153 7AF8 A70A 6E59   4F9
017: F17F 31E1 AE71 577A   38E
018: F5A7 1A6F 59F1 7F15   235
019: 77B7 2155 717F 1537   F05
01A: 1CF8 4197 CC18 5112   BF1
01B: 8137 0631 0214 D071   7AD

01C: 3511 8A70 1011 0810   29F
01D: A1CF 1CF1 C131 2814   F97
01E: DAA3 B270 7AE7 0610   CAA
01F: 37AE 1104 1127 0E11   8AF
020: 1CA7 29F7 D611 CB7A   82A
021: 118A 7A12 2A74 11B7   57B
022: FB11 19A7 AA6F 51C3   44F
023: 4088 00D5 1562 0E2F   1F8
024: 1542 1321 30C0 1321   D15
025: 5620 E2F1 5421 32C0   99A
026: 1321 5620 E2F1 5421   5EC
027: 305F 4AEB 808A 0F21   441
028: 5620 E2F1 5421 3213   FE8
029: 0340 8800 CA13 2156   C29
02A: 20E2 F154 2130 AEB8   A77
02B: 08A1 E015 620E 2F15   7C9
02C: 42A2 7580 160B 2714   3ED
02D: FA6E 4221 4D17 1110   FB6
02E: 1577 A721 0810 A111   B6C
02F: 1C16 3FE1 611C 114F   942
030: A6E4 E214 D173 17F8   732
031: FCEE 1080 8BFC 1111   4CA
032: 1577 B7A1 7F15 7769   292
033: 9E87 1606 790D 2310   EAE
034: 1D73 4088 00D5 31D1   B8D
035: 1321 3106 3444 000E   752
036: A130 DB06 14E1 4D13   49D
037: 2130 C013 214E 1301   033
038: 3313 1C01 3314 D131   C0D
039: 1321 30C0 C013 214E   8C0
03A: 1301 3313 1C0C 0133   4A1
03B: 14D1 3116 1171 07A6   0E3
03C: E54A 1320 7A6E 5980   E55
03D: 78F5 E010 8D34 150A   B76
03E: F684 2948 80BF 8852   935
03F: 94A3 1BFA 8728 0852   677
040: 5508 42AF 1AFC 8228   518
041: 1E83 250A 70A7 597E   35D
042: CE8F 5E5D 0862 00BF   1FC
043: 8015 6621 B213 0B21   E02
044: 30                    E95


D4COLOR                 (HP48)
# D219h             153 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0D29 512B F817   DEA
001: 6040 D9D2 0FC2 E4CC   E0D
002: D205 0100 8FC1 5238   ACA
003: F511 1013 0169 8401   5FD
004: 4234 0800 08B6 5085   235
005: 0169 132D 8320 88CA   033
006: C2D7 1B00 1001 5648   BF9
007: 0860 5130 C15C 0195   884
008: 2D2C E154 31F0 2100   471
009: 1992 D973 70D6 7D60   23A
```

```
00A: 8600 1320 08A3 E5CF   0F6
00B: 480D B745 08FC EE10   00E
00C: 808A FFC1 9001 5443   CB3
00D: 4B55 071F 97F1 0143   96E
00E: 1FE0 5078 A290 1FD8   7E1
00F: 6081 5F71 9021 5C78   500
010: F5E0 108D 3415 0145   0E0
011: 15E0 808B 17F1 5E08   ECE
012: 08A1 7F01 B213 0B21   B53
013: 30                    BE6
```



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