JPC

OCTOBRE 1993
NUMERO 90

Le numéro 40 F

A PROPOS DU CLUB

Le Bureau — Editorial 1
Courrier du Coeur / SOS 2

HP28

G. Toublanc — Désactivation de l'écran 4
P. Jebeily — Graphismes et animations 4

HP48

G. Toublanc — Les cartes Ram 1Mo Hewlett-Packard 14
G. Toublanc — Programme bizarre et dérives litéraires 14
J.F. Garnier — Calculs astronomiques 18

HP95

J. Belin — Paneau de paramètrage du HP95 24

Le coin des codes 33

EDITORIAL

Il est de règle pour les collaborateurs de certaines sociétés Américaines de ne rester à leur poste que pour une durée de 3 ans environ. Cependant, je voudrais dénoncer un couple d'irréductibles ayant réussi à tenir plus de 53 ans à leur poste ! Sans donner de noms, je vous dirais que ces deux personnes portent les initiales H et P... Plus sérieusement, cette situation vient de changer, puisque David Packard va bientôt quitter le directoire de notre société préférée pour faire valoir son droit à la retraite. Avant qu'il ne nous quitte vraiment, nous ne pouvons que lui dire bravo et merci... Bill Hewlett reste à son poste, et devient de ce fait la plus ancienne personne travaillant dans la société...

Ceux qui ont pu voir la vidéo *Corvallis Plant Tour* au cours de notre réunion anniversaire savent que les développeurs de nos chères machines possèdent un grand sens de l'humour. Il aura cependant fallu plus de vingt ans pour que cela apparaisse dans le code même des machines. En effet, les possesseurs des 48G et GX possèdent maintenant la fabuleuse (!) commande RULES montrant (sous la forme d'un mots croisés) les prénoms des principaux développeurs (Bill Wickes, Jim Donnelly, Dennis York...) ainsi que le nom de code de la machine (Alcuin). Cependant, est ce un hasard ou ce fameux sens de l'humour qui a fait que la Rom de la version Française du HP100LX est datée du 14 Juillet ?

Pourtant, il semblerait qu'à force de s'amuser ces mêmes personnes ont laissé tomber leur vigilance au cours de leur travail. En effet, à mesure que le temps passe, de nombreuses bogues ne cessent d'être déclarées sur la HP48. Si la plupart ne sont en fait que des alertes faites par des utilisateurs ne comprenant pas le fonctionnement exact de leur machine, il n'en est pas moins vrai que les nouvelles HP48 semblent contenir un nombre record de défauts. Bien sûr nous reviendrons en détail sur ce problème dans les prochains JPC, après que nous ayons fait le tri dans tous les reports que nous avons recus de France et de l'étranger. Si en avez trouvé quelques unes, ou plus, que vous avez décidé de faire de leur recherche un nouveau sport, faites nous en profiter ! Et qui sait ? Vous risquez bien de trouver une bogue permettant de soulever un rideau tel que celui qui nous a permis d'accéder à la légendaire *programmation synthétique* de la HP-41 !

Plus sérieusement vous pouvez constater que ce journal est un peu plus épais que d'habitude. Cela est dû principalement à la présence de trois articles particulièrement étoffés. Deux d'entre eux devraient plaire à ceux qui nous accusent de faire trop d'"élitisme", puisqu'ils sont totalement écrits en RPL. Que les autres ne s'inquiètent pas, l'assembleur Saturn devrait sans nul doute être beaucoup plus présent dans le prochain JPC ! Cependant, de la même façon que ce journal n'aurait probablement pas pu paraître aujourd'hui sans le soutien de Paul Jebeily et Jean François Garnier, il ne tient qu'à vous pour nous aider à faire paraitre le prochain numéro dès la semaine prochaine, afin de résorber notre retard. Mais si, c'est possible ! Enfin, si vous nous envoyez tout de suite vos articles bien sûr...

COURRIER DU COEUR

P. Morganti
4, rue Marguerite Bervoets
78280 GUYANCOURT
Tel : (1) 30 40 79 23

Vend :

HP48S : 600 F.

SOS

Guy Toublanc
21, rue Henri Becquerel
59120 LOOS-LES-LILLE

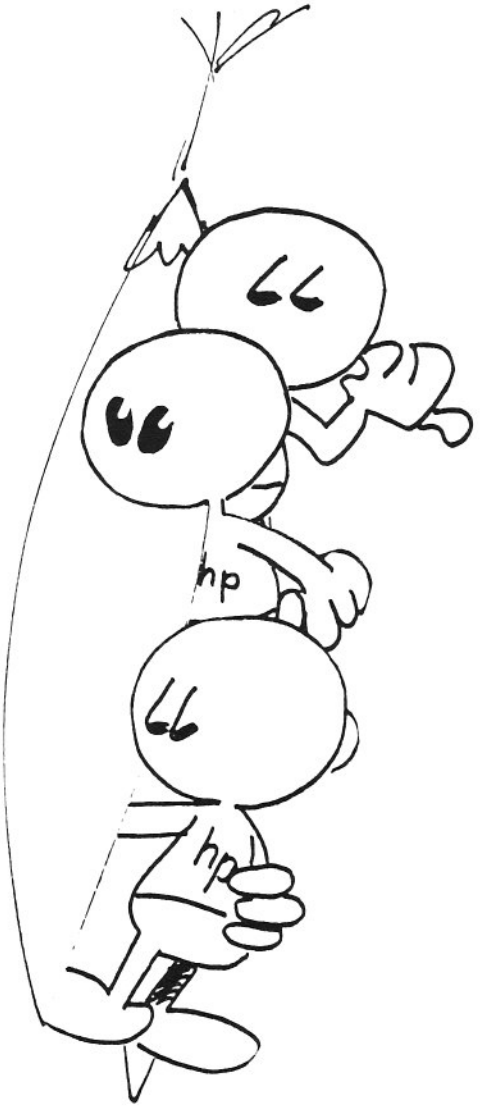
Mon problème:

j'ai une carte d'extension mémoire TDS pour HP48SX de 512 Ko donc partitionnable en quatre bancs de 128 K. Actuellement trois bancs seulement sur quatre sont utilisables. J'ai tout essayé : vidage complet de tous les objets de la cartes puis rechargement, réinitialisation complète de la carte après l'avoir laissée sans pile pendant deux semaines. C'est en vain car il y a ou refus de stocker les objets ou non conservation de ceux-ci dans le quatrième banc. Qui a eu ce problème (peut-être aussi avec une carte de 256 Ko) et a trouvé une solution soit logicielle ou matérielle ?

P. Morganti
4, rue Marguerite Bervoets
78280 GUYANCOURT
Tel : (1) 30 40 79 23

Cherche :

Tous programmes d'électronique pour HP48.



HP28

G. Toublanc
P. Jebeily

Désactivation de l'écran *4*
Graphismes et animations *4*

Le coin des codes *33*

ECRAN MINIMAL

L'apparition des fonctions ECRON et ECROFF dans l'article suivant me permet d'en profiter pour en apporter des versions améliorées. Rapelons que ces fonctions sont apparues des le livre *Voyage au centre de la HP28* (page 240).

ECROFF permet d'éteindre l'écran sans interrompre un programme. L'affichage doit être rétabli par ECRON avant la fin du programme. Ceci économise les piles lors de l'exécution de programmes longs dans leur déroulement ou par exemple lors des transferts d'objets HP28 <-> PC ou de tout type de transmission de données avec des périphériques ce qui est assez gourmand en énergie.

Le principe est de modifier le quartet en # FFF03h et représentant l'état de l'écran. Pour cela il y deux possibilités : faire un POKE à cette adresse ou faire la modification par un programme en langage machine (assembleur). Ce sont ces deux solutions qui sont exposées dans le livre cité et qui coûtent pour chaque programme :

36,5 octets avec le POKE
19,5 octets avec l'assembleur

Il existe une meilleure solution car plus économe en mémoire : l'utilisation des deux programmes en Rom qui font cela. Les voici désassemblés :

Pour éteindre l'écran c'est le programme d'adresse # 01023h:

```
01023   cd1ex              00fd8   p=      0
        rstk=c                     lchex   2
        gosub   00fd8              d1=(5)  fff03
00131   c=rstk                     dat1=c  1
        d1=c                       d1=(5)  ffb03
        a=dat0  a                  dat1=c  1
        d0=d0+  5                  rtncc
        pc=(a)
```

Pour allumer l'écran c'est le programme d'adresse # 01011h :

```
01011   cd1ex              00fbb   p=      0
        rstk=c                     lchex   a
        gosub   00fbb              d1=(5)  ffb03
        goto    01031              dat1=c  1
                                   d1=(5)  fff03
                                   dat1=c  1
                                   rtncc
```

Donc il suffit d'utiliser ces deux routines pour constituer deux mini-programmes de 7,5 octets, ce qui économise 24 octets pour les deux programmes.

Il n'est même pas besoin de mettre les codes dans la rubrique habituelle.

ECROFF
7,5 octets cksum # AF07h

```
76C2 0320 1009 F20
```

ECRON
7,5 octets cksum # 6D5Eh

```
76C2 0110 1009 F20
```

Donc ces deux programmes devront être utilisés par paire dans un programme sinon après ECROFF il faut faire un arrêt système faisant perdre tout ce qui est sur la pile.

```
« ECROFF ..... ECRON »
```

Pour les champions de l'anti-sèche il est possible de se faire des programmes qui efface l'écran lors du passage du prof' et permettent de poursuivre normalement l'utilisation de l'anti-sèche après. Comme je suis côté prof' je ne vais pas plus loin dans mes suggestions !

Guy Toublanc (276)

GRAPHIQUES ET ANIMATIONS SUR HP28S

Le graphique sur HP28S est limité par la taille de l'écran (quoi que...: je vous proposerai la prochaine fois la preuve du contraire), mais surtout par la lenteur du traçage. A moins d'avoir recours au language machine et au talent de Guy Toublanc (que je salue amicalement), programmer en RPL demande de la patience avant d'aboutir au résultat graphique attendu, ou inatendu !! Mais la surprise est souvent agréable et récompense largement le temps passé.

Voici donc quelques petits et simples programmes d'application graphique : ceux du registre { HOME GRAPH } permettent quelques convertions utiles pour le traitement d'images; ceux du registre { HOME GRAPH OBJETS } permettent de créer quelques animations avec des objets géométrique simples :

cercles, ellipses, carrés, rectangles polygones circulaires ou elliptiques, tracés en lignes, points ou surface, avec diverses méthodes. En répertoire { HOME } se trouvent quelques utilitaires généraux, nécessaires au bon déroulement des autres applications plus spécifiques.

Chemin { HOME }

WAITK
Retourne la chaine de la touche précédente après attente de l'appui de celle ci.
Sortie
1: "Chaine de la touche enfoncée"

```
« DO UNTIL KEY END »
```

MENUER
Exécution d'un menu personalisé dont on a défini, dans deux listes, les routines à évaluer pour chaque touche de clavier pressées.
Entrées
4: { Liste d'objets programmes }
3: { "Liste" "des" "touches" "définies" "dans" "le" "même" "ordre" }
2: "chaine graphique à afficher en menu"
1: "chaine graphique à afficher lors de l'appui sur une touche non définie.

```
«
  → ltch imenu imerr
  « DO imenu →LCD ltch MAITK POS DUP UNTIL
      « 1 »
      « imerr →LCD 250 .3 BEEP » IFTE
    END
    GET EVAL CLMF
  »
»
```

OFF
Mise en arrêt de la machine de la machine jusqu'à appui sur [ON] : la machine reprend là ou elle s'était arrêtée.

```
« # 18E58h SYSEVAL »
```

LSTX
Traite tous les objets d'une liste avec le même programme en pile.
Entrées :
3: { liste d'objets divers }
2: Taille de la liste : nl (algébrique)
1: Programme ou autre objet évalué sur chaque objet de la liste, dans l'ordre croissant si nl ≥ 0, dans l'ordre inverse sinon.

```
«
  → lst nl prg
  « 1 nl nl 0 <
    « ABS SWAP »
    IFT
    FOR i
      lst i GET prg
      IFERR EVAL THEN
        DROP
      END
    nl SIGN STEP
  »
»
```

PAUSE
Attend l'appui sur clavier.

```
« WAITK DROP »
```

SWPCH
Remplace un morceau de la chaîne 1 par la chaîne 2 à partir de la position du $p^{ème}$ caractère de la chaine 1.
Entrées :
3: "Chaine 1 à modifier"
2: "Chaine 2 à supperposer"
1: Position p (entier >0 ou <0)
Sorties
1: "Chaine 1 modifiée"
Remarquons que la chaîne modifiée peut être de taille supérieure à la taille initiale de la chaine 1, si |1+|taille chaine 1|-p| > |taille chaine 2| pour p>0, et le sera toujours si p≤0 (sauf si la chaine 2 est vide).

```
«
  → ch1 ch2 p
  « ch1 SIZE ch2 SIZE → s1 s2
    « p 1 >
      « ch1 0 p 1 - SUB »
      « "" »
      IFTE ch2 + ch1 p s2 + s1 SUB +
    »
  »
»
```

UP
Permet de revenir au niveau de répertoire supérieur (vers HOME), et affiche le path actuel dans le menu CUSTOM.

```
«
  PATH → p
  « p MENU p DUP SIZE 1 - 1 MAX GET EVAL 23 MENU »
»
```

Chemin { HOME GRAPH }

VARS :

```
{ PBLKBIN POINT? PXBAR XY→P.N P.N→XY
XY→C137L4N C137L4N→XY P→137L4
137L4→P COLI→PE PE→COLI COLI→P05
RDZXY PPAR }
```

Notations utilisées pour la recherche de la position du pixel sur l'écran :
- (X,Y) est point de l'écran de PPAR.
- (P,N) : P=Position du caractère de la chaine graphique de 548 octets (137*4). N=Position du pixel dans ce caractère.
- (C,L,N) C=Position de la colonne (équivalent à X). L=Position de la ligne, parmi 4, de 1 à 4 de haut en bas. N=(8-hauteur) du pixel de l'octet du caractère positionné en (C,L) de la chaine graphique.

XY→P.N
Convertit des coordonées XY en position dans la chaine graphique.
Entrées :
2: x ∈ [1..137]
1: y ∈ [1..32]
Sorties
2: P ∈ [1..548]
1: N ∈ [1..8]

```
« XY→C137L4N 3 ROLLD C137L4→P SWAP »
```

P.N→XY
Fonction réciproque à XY→P.N.
Entrées
2: P ∈ [1..548]
1: N ∈ [1..8]
Sorties
2: x ∈ [1..137]
1: y ∈ [1..32]

```
«
  → cp n
  « cp P→C137L4 n C137L4N→XY »
»
```

XY→C137L4N
Convertit des coordonées XY en position dans la chaine graphique.
Entrées
2: x ∈ [1..137]
1: y ∈ [1..32]
Sorties
3: C ∈ [1..137]
2: L ∈ [1..4]
1: N ∈ [1..8]

```
«
  → x y
  « x 4 y 8 / IP - y 8 MOD 0 == + 9 y 8 MOD DUP 0 ==
    8 * - - 8 MOD DUP 0 == 8 * +
  »
»
```

C137L4N→XY
Fonction réciproque à XY→C137L4N.
Entrées
3: C ∈ [1..137]
2: L ∈ [1..4]
1: N ∈ [1..8]
Sorties
2: x ∈ [1..137]
1: y ∈ [1..32]

```
«
  → c l n
  « c 4 l - 8 * 9 + n - »
»
```

P→C137L4
Conversion du n° de position du caractère de la chaine graphique en n° de colonne et n° de ligne.
Entrées
1: P ∈ [1..548]
Sorties
2: C ∈ [1..137]
1: L ∈ [1..4]

```
«
  → cp
  « cp 137 MOD DUP 0 == 137 * + cp 137 / IP
    cp 137 MOD 0 ≠ +
  »
»
```

C137L4→P
Fonction réciproque de P→C137L4.
Entrées
2: C ∈ [1..137]
1: L ∈ [1..4]
Sorties
1: P ∈ [1..548]

```
«
  → c l
  « l 1 - 137 * c + »
»
```

COLI→PE
Conversion des coordonées en mode "texte".
Entrées
2: Co ∈ [1..23]
1: Li ∈ [1..4]
Sorties
1: Pe ∈ [1..92]

```
« → co li '(li-1)*23+co' »
```

PE→COLI
Fonction réciproque à COLI→PE.
Entrées
1: Pe ∈ [1..92]
Sorties
2: Co ∈ [1..23]
1: Li ∈ [1..4]

```
«
  → p
  « p 23 MOD DUP 0 == 23 * + p 23 / IP
    p 23 MOD 0 ≠ +
  »
»
```

COLI→P05
Conversion de coordonnées "Textes" en position graphiques sur la chaine graphique de l'écran (utile pour récupérer la chaine graphique d'un caractère affiché en position (Co,Li).
Entrées
2: Co ∈ [1..23]
1: Li ∈ [1..4]
Sorties
2: P1 ∈ [1..544]
1: P2 ∈ [1..548]

```
«
  → co li
  « co li COLI→PE 6 * li 1 - -  DUP 5 -
    SWAP co 23 == -
  »
»
```

RDZXY
Donne un nombre entier aléatoire 'z' compris entre deux nombres 'x' et 'y' entiers.
Entrées
2: x (entier)
1: y (entier)
Sorties
1: z (entier)

```
«
  DUP2 SWAP - → x y d
  « d 0 ≠
    « x y MIN d ABS RAND * .5 + IP + »
    « x RAND .5 + IP * »
    IFTE
  »
»
```

PPAR

```
{ (1,1) (137,32) X 1 (0,0) }
```

XY→CARCL
Conversion de format Ecran Graphique (x,y) en format Ecran Texte (Colonne,Ligne).
Entrées :
2: X ∈ [1..137]
1: Y ∈ [1..32]
Sorties
2: Co ∈ [1..23]
1: Li ∈ [1..4]

```
«
   → X Y
   « X DUP 0 > * DUP NOT + DUP 137 ≤ * DUP 0 ==
     138 * + 6 / CEIL 5 Y DUP 0 > * DUP NOT + DUP 32
     ≤ * DUP 0 == 32 * + 8 / CEIL -
   »
»
```

POINT?
Teste la présence d'un point de coordonées (x,y) sur l'écran de PPAR={(1,1) (137,32)...}
Entrées :
2: X ∈ [1..137]
1: Y ∈ [1..32]
Sorties
1: 0 ou 1

```
«
  XY→P.N LCD→ → p n im0
  « im0 p p SUB NUM BIN R→B 2 n 1 - ^ / DUP
     2 / 2 * - B→R
  »
»
```

PXBAR
Affiche à l'écran la colonne graphique (haute de 8 pixels, large de 1) codée par son nombre binaire équivalent, à la position (colonne,ligne) de l'écran.
Entrées :
3: Bin : #....b
2: C ∈ [1..137]
1: L ∈ [1..4]

```
«
  C137L4→P LCD→ → bin p str
  « str bin B→R CHR p SPWCH →LCD »
»
```

Path : { HOME GRAPH OBJETS }

VARS :

```
{ LIMG ROTIMG FAMROT DELLIPS ELLIPS DCERCL CERCL
EPOLYG POLYG RECTG CARRE VERTIC HORIZ SEGMT LELLIPS
PELLIPS LCERCL PCERCL PPOLYG PEPOLYG LBLOK PRECT DSURF
PARL PARP PAS PPAR ΣPAR ΣDAT EQ LPGROT STP.W
PWAIT LTCHROT TEST INIT }
```

INIT
Initialise les modes nécessaires aux application de ce répertoire OBJETS. Il met aussi PWAIT à une valeur rapide pour ROTIMG.
Ce programme est à exécuter en premier : il fixe les divers modes.

```
«
  .01 'PWAIT' STO 2 FIX 31 SF 35 CF 36 SF 59 CF
  RAD 64 CF
»
```

ROTIMG
Anime les images de la liste en pile, à la cadence choisie interactivement par l'utilisateur au cours du déroulement du programme, en appuyant sur les touches définies par LTCHROT.
Entrées :
1: { Liste d'images graphiques }

```
«
  DUP SIZE DUP 0 ==
  « DROP2 ABORT »
  IFT → lst nl
```

```
  « DO
      lst nl PWAIT SIGN DUP 0 == + *
      « →LCD PWAIT ABS WAIT »
      LSTX
    UNTIL KEY
      «
        LTCHROT SWAP POS DUP 0 == + LPGROT SWAP GET
        EVAL
      » IFT 0
    END
  »
»
```

FAM1ROT

Trace une famille d'objets graphiques, stocke les images dans LIMG et lance l'animation avec ROTIMG (voir son fonctionnement dans TEST1 et TEST2).

Entrées :

4: Objet-Programme
3: { Liste des paramètres graphiques }
2: Le Pas d'incrémentation numérique
1: Nombre d'images à tracer

```
«
  SWAP DUP DUP ABS / ROT ABS * LCD→ →
  obj lparam pang n im0
  « { } n 0 ≠
    « n SIGN n
      FOR i
        lparam LIST→ DROP pang i * obj EVAL LCD→
        + im0 LCD→ 880 .1 BEEP
      n SIGN STEP
    »
    IFT DUP 'LIMG' STOP ROTIMG
  »
»
```

FAM2ROT

Identique à FAM1ROT mais utilise les programmes suplémentaires d'économie de temps SPEED et de piles (ECRON et ECROFF) du livre de Sébastien Lalande et Paul Courbis *Voyage au Centre de la HP28C/S*. (Le SPEED de Guy Toublanc est équivalent, et la taille est plus courte).

NDLR : Voir l'article précédent pour ECRON/ECROFF et le Coin des Codes pour SPEED.

Entrées :

4: Objet-Programme
3: { Liste des paramètres graphiques }
2: Le Pas d'incrémentation numérique
1: Nombre d'images à tracer

```
«
  ECROFF 'LIMG' PURGE SPEED FAM1ROT ECRON
  220 .1 BEEP 880 .1 BEEP 440 .1 BEEP OFF
»
```

LGPROT

C'est la liste des programmes de définition de chaque touche correspondante du clavier, en paralèlle (même ordre, même nombre) avec la liste de touche LTCHROT. LGPROT est utilisé par ROTIMG. Vous pouvez définir d'autres touches, à condition bien sûr de faire correspondre à chaque programme introduit à la p^{ème} position dans LGPROT la chaîne de caractère de la touche à introduire à la même p^{ème} position dans la liste LTCHROT. Si vous ne connaissait pas le nom de la touche, exécutez WAITK... Je signale tout de même que la 1ère position est réservée au traitement des erreurs. Ainsi, les touches [0] à [9] donnent un temps (en 10e de secondes) à STP.W. Pour rendre effectif la valeur STP.W de temps d'arrêt entre chaque image on tape [=]. La touche [.] divise par 10 STP.W (non effectif, donc cummulable). Les touches [+] [-] [*] [/] changent directement le temps d'arrêt effectif, d'après la valeur STP.W actuelle. [CHS] inverse le sens de rotation des images. [S] pour SPEED, [Q] pour quitter. [O] pour OFF de la machine (et reprise du programme après un [ON]).

```
{
  « 220 .1 BEEP 440 .1 BEEP 220 .1 BEEP »
  « .001 'STP.W' STO »
  « .1   'STP.W' STO »
  « .2   'STP.W' STO »
  « .3   'STP.W' STO »
  « .4   'STP.W' STO »
  « .5   'STP.W' STO »
  « .6   'STP.W' STO »
  « .7   'STP.W' STO »
  « .8   'STP.W' STO »
  « .9   'STP.W' STO »
  « 'STP.W' 10 STO/ »
  « STP.W 'PWAIT' STO »
  « STP.W 'PWAIT' STO+ »
  « 'PWAIT' STP.W STO- »
  « STP.W 'PWAIT' STO* »
  « 'PWAIT' STP.W STO/ »
  « PWAIT NEG 'PWAIT' STO »
  « SPEED »
  « CLEAR ABORT »
  « OFF »
}
```

LTCHROT

C'est la liste précédement définie, utilisée par ROTIMG (La première chaîne "ERROR" est réservée au traitement des erreurs dûes à l'appui sur des touches non définies.

```
{
  "ERROR" "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
  "." "=" "+" "-" "*" "/"  "CHS" "S" "Q" "O"
}
```

STP.W (variable)
.001

PWAIT (variable)
.001

DELLIPS
Trace une ellipse par la fonction DRAW.
Entrées :
4: Abscisse du centre de l'ellipse
3: Ordonnée du centre de l'ellipse
2: Demi largeur
1: Demi hauteur

```
«
  → a b rx ry
  « 1 'X' a - rx / SQ - √ ry * DUP b + SWAP NEG b
    + = STEQ DRAW
  »
»
```

ELLIPS
Trace une ellipse en 36 segments.
Entrées :
Voir DELLIPS

```
«
  → a b rx ry
  « a b rx ry 36 EPOLYG »
»
```

DCERCL
Trace un cerle avec DRAW.
Entrées :
3: Abscisse du centre du cercle
2: Ordonnée du centre du cercle
1: Centre du cercle

```
«
  → a b r
  « r SQ 'X' a - SQ - √ DUP b + SWAP NEG
    b + = STEQ DRAW
  »
»
```

CERCL
Trace un cerle "continu".
Entrées :
Voir DCERCL

```
«
  .5 π / OVER / 6 * → a b r p
  « IF r 0 ≠ THEN
      a r 't' COS * + b r 't' SIN * + 0 2 π * p PARP
    ELSE
      a b R→C PIXEL
    END
  »
»
```

EPOLYG
Trace un polygone elliptique. On peut rajouter après les 5 paramètres nécessaires, un paramètre facultatif : un angle de déphasage à l'origine (utilisé en animation pour faire des rotations successives).

Entrées :
5: Abscisse du centre du polygone
4: Ordonnée du centre du polygone
3: Demi largeur
2: Demi hauteur
1: Nombre d'arrêtes
0: Facultativement : la phase à l'origine (en radians)

```
«
  DEPTH 5 ==
  « 0 »
  IFT 4 PICK 3 PICK * 0 ≠ * → a b rx ry n offt
  « IF n 0 ≠ THEN
      IF rx ry + 0 ≠ THEN
        a rx 't' offt + COS * + b ry 't' offt + SIN * +
        0 2 π * DUP n / PARL
      ELSE
        a b R→C PIXEL
      END
    END
  »
»
```

POLYG
Trace un polygone (inscrit dans un cercle -non tracé- de rayon *r*).
Entrées :
4: Abscisse du centre du polygone
3: Ordonnée du centre du polygone
2: Rayon extérieur du polygone
1: Nombre d'arrêtes
0: Facultativement : la phase à l'origine (en radians)

```
«
  DEPTH 4 ==
  « 0 »
  IFT → a b r n offt
  « IF n 0 ≠ THEN
      IF r 0 ≠ THEN
        a r 't' offt n 2 MOD 1 + * + COS * + b r 't'
        offt + SIN * + 0 2 π * DUP n / PARL
      ELSE
        a b R→C PIXEL
      END
    END
  »
»
```

RECTG
Trace un rectangle (inscrit dans une ellipse de rayons *rx* et *ry*.
Entrées :
4: Abscisse du centre du rectangle
3: Ordonnée du centre du rectangle
2: Demi largeur elliptique
1: Demi hauteur elliptique

```
« SWAP 2 √ * SWAP 2 √ * 4 π 4 / EPOLYG »
```

CARRE
Trace un carré (inscrit dans un cercle -non tracé- de rayons *r*.
Entrées :
3: Abscisse du centre du rectangle
2: Ordonnée du centre du rectangle
1: Rayon circulaire externe

```
« 2 √ * 4 π 4 / POLYG »
```

VERTIC
Trace un segment vertical entre les points (l,y1) et (l,y2).
Entrées :
3: Ordonnée de départ
2: Ordonnée d'arrivée
1: Abscisse fixe

```
«
  → y1 y2 l1
  « y1 y2
    IF DUP2 ≠ THEN
      FOR y
        l1 y R→C PIXEL
      y2 y1 - SIGN STEP
    ELSE
      DROP l1 SWAP R→C PIXEL
    END
  »
»
```

HORIZ
Trace un segment horizontal entre les points (x1,h) et (x2,h).
Entrées :
3: Abscisse de départ
2: Abscisse d'arrivée
1: Ordonnée fixe

```
«
  → x1 x2 h1
  « x1 x2
    IF DUP2 ≠ THEN
      FOR x
        x h1 R→C PIXEL
      x2 x1 - SIGN STEP
    ELSE
      DROP h1 R→C PIXEL
    END
  »
»
```

SEGMT
Ce programme trace un segment entre deux points. La présence de PPAR est obligatoire. C'est la version 'LINE' de J.M. Ferrard de son livre *La maitrise de la HP28S*, T1, 3eme édition.
Entrées :
2: (x1,y1)
1: (x2,y2)

```
«
  IF DUP2 == THEN
    PIXEL
  ELSE
    OVER - DUP C→R PPAR LIST→ 4 DROPN -
    C→R ROT 32 * SWAP / ABS 3 ROLLD / 137 * ABS
    MAX INV PPAR 4 GET * → v pas
    « 0 1
      FOR k
        DUP v k * + PIXEL
      pas STEP
    »
  END
  DROP
»
```

PELLIPS
Trace une ellipse pleine, par le tracé des ellipses successivement concentriques.
Entrées :
Voir DELLIPS

```
«
  → x y a b
  « 0 a b MIN
    FOR u
      x y a u - b u - ELLIPS
    NEXT
  »
»
```

LELLIPS
Trace une ellipse pleine, avec un remplissage segment par segment : plus rapide !
Entrées :
Voir DELLIPS

```
«
  → a b rx ry
  « 1 'x' a - rx / SQ - √ ry * DUP b + SWAP
    NEG b + a rx - a rx + DSURF
  »
»
```

LCERCL
Trace un disque, avec un remplissage du même type que LELLIPSE : rapide.
Entrées :
3: Abscisse du centre du disque
2: Ordonnée du centre du disque
1: Centre du disque

```
«
  → a b r
  « r SQ 'x' a - SQ - √ DUP b + SWAP NEG
   b + a r - a r + DSURF
  »
»
```

PCERCL
Trace un disque, par un tracé successif de cercles concentriques (simple, mais lent !).
Entrées :
Voir LCERCL

```
«
  → x y a
  « 0 a
    FOR u
      x y a u - CERCL
    NEXT
  »
»
```

PPOLYG
Trace un polygone plein, par un tracé successif de polygones. on peut rajouter en dernier lieu un déphasage en radians.
Entrées :
Voir POLYG

```
«
  DEPTH 4 ==
  « 0 »
  IFT → x y r n offt
  « 0 r
    FOR u
      x y r u - n offt POLYG
    NEXT
  »
»
```

PEPOLYG
Comme PPOLYG, mais cette fous-ci : polygoone elliptique (plein). Là aussi, on peut introduire un déphasage en radians.
Entrées :
5: Abscisse du centre du polygone
4: Ordonnée du centre du polygone
3: Rayon en abscisse de l'ellipse externe au polygone.
2: Rayon en ordonnée de l'ellipse externe au polygone.
1: Nombre d'arrêtes
0: Facultativement : la phase à l'origine (en radians)

```
«
  DEPTH 5 ==
  « 0 »
  IFT → x y a b n offt
  « 0 a b MIN
    FOR u
      x y a u - b u - n offt EPOLYG
    NEXT
  »
»
```

LBLOK
Trace un bloc (ou rectangle plein) avec des segments horizontaux ou verticaux (le choix fait gagner du temps).
Entrées :
4: Abscisse du coin inférieur gauche
3: Ordonnée du coin inférieur gauche
2: Largeur
1: Hauteur

```
«
  DUP2 ≥ → x y a b fl
  « 0
    IF fl THEN
      b 1 -
      FOR u
        u x x a + 1 - y u + HORIZ
      NEXT
    ELSE
      a 1 -
      FOR u
        u y y b + 1 - x u + VERTIC
      NEXT
    END
  »
»
```

PRECT
Trace un rectangle plein, par une succession de rectangles concentriques.
Entrées :
4: Abscisse du centre du rectangle
3: Ordonnée du centre du rectangle
2: Demi largeur elliptique externe
1: Demi hauteur elliptique externe

```
«
  → x y a b
  « 0 a b MIN
    FOR u
      x y a u - b u - RECTG
    NEXT
  »
»
```

DSURF
Trace une surface entre 2 fonctions : f(x)+g(x)=0.
Entrées :
4: Equation cartésienne f(x)
3: Equation cartésienne g(x) (généralement -f(x))
2: Abscisse de départ
1: Abscisse d'arrivée

```
«
  → eq neq d f
  « d f
    FOR z
      z 'x' STO neq EVAL eq EVAL x VERTIC
    NEXT
    'x' PURGE
  »
»
```

PARL
Trace une fonction paramètrique par des segments (ou lignes) f(x(t), y(t)).
Entrées :
5: Equation paramétrique x(t)
4: Equation paramétrique y(t)
3: Début du paramètre 't'
2: Fin du paramètre 't'
1: Pas d'incrémentation de 't'

```
«
  0 → xt yt deb fin pas old
  « deb →NUM fin pas + →NUM
    FOR i
      i 't' STO
      IFERR
        xt EVAL RDN yt EVAL RDN R→C
        IF DUP old ≠ i deb ≠ AND THEN
          old OVER SEGMT
        END
        'old' STO
      THEN
        CLEAR
      END
      IF KEY THEN
        DROP fin pas + →NUM 'i' STO
      END
    pas STEP
    't' PURGE
  »
»
```

PARP
Trace la fonction paramétrique, point par point f(x(t),y(t)).
Entrées :
Voir PARL

```
«
   3 ROLLD DUP 2 ≥
   « SWAP »
   IFT →NUM → x y p d f
   « d →NUM 't' STO
     DO
       x EVAL y EVAL R→C PIXEL p 't' STO+
     UNTIL
       t f →NUM ≥ KEY
       « DROP2 1 »
       IFT
     END
     't' PURGE
   »
»
```

PAS
Calcule la longueur curviligne de la fonction paramétrique :

$$s = \int_{t0}^{t1} \sqrt{dx^2(t) + dy^2(t)}\, dt$$

Entrées :
4: x(t)
3: y(t)
2: t0
1: t1
Sortie :
1: s

```
«
  .001 → x y d f e
  « x 't' ∂ SQ y 't' ∂ SQ + √ 't' d f 3
    →LIST e ∫ DROP
  »
»
```

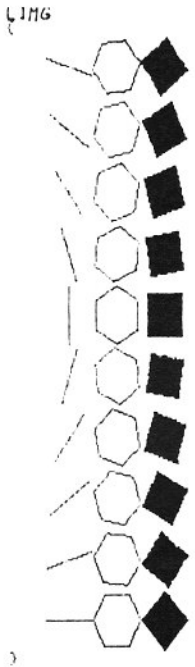
TEST1
Trace 10 images successives, puis les anime. Sur chacune de ces images sont tracés deux polygones : un carré et un pentagone. L'un des deux tourne par rapport à l'autre immobile. Devinez lequel !

```
«
   « 5 DUPN DROP2 4 0 POLYG POLYG »
   { 69 16 15 6 } 2 π * 11 / 10 CLLCD FAM1ROT ROTIMG
»
```

TEST2
Trace aussi 10 images et les met en animation. Le résultat des objets tracés est d'ailleurs affiché en exemple sous LIMG.
A vous de tracer vos animations !

```
«
  « 4 SPEED 5 DUPN 5 DUPN 5 ROLL 30 - 5 ROLLD SWAP
    6 * 2 / NEG POLYG 5 ROLL 30 + 5 ROLLD SWAP DROP
    4 SWAP 6 * 4 / NEG PPOLYG POLYG
  »
  { 69 16 15 6 } 2 π * 6 / 11 / 10 CLLCD
  FAM1ROT ROTIMG
»
```

Paul Jebeily (571)



HP48

G. Toublanc	Les cartes Ram 1Mo Hewlett-Packard	*14*
G. Toublanc	Programme bizarre et dérives litéraires	*14*
J.F. Garnier	Calculs astronomiques	*18*
	Le coin des codes	*33*

BIG RAM pour GX

Ceux qui se trouvent à l'étroit avec un maximum de 288 Ko de Ram pour une HP48SX, ont la possibilité d'utiliser les cartes Multiram TDS de 256 Ko et 512 Ko. Deux de ces cartes de 512 Ko permettent donc d'étendre de 1 Mo la Ram des HP48SX mais la gestion de cette Ram ne peut se faire que par bancs de 128 Ko dont un seul banc est actif à la fois pour les librairies et objets qui y sont stockés.

Pour la HP48GX la solution est différente car le port 2 est le seul à pouvoir recevoir une carte de plus de 128 Ko qui ne sera pas mergeable mais dont la capacité maxi est repoussée à 4 Mo. La Ram du port 2 est partitionnée en bancs de 128 Ko mais qui sont tous actifs grâce à la technique dite de *bank-switching*. Ainsi avec une carte de 1 Mo en faisant `[LIBRARY]` `[PORTS]` on pourra avoir les deux cas de figure :

- Sans carte dans le port 1 :
  :0: :2: :3: ..... :9:
- Avec une carte dans le port 1 :
  :0: :1: :2: :3: ..... :9:

Et on peut faire `:n:TOTO RCL` pour rappeler l'objet `TOTO` du banc *n* (*n* pouvant avoir ici la valeur 2 à 9 pour le port 2). De même une librairie, stockée dans le banc *n* sera active.

Comment se présente ce type de carte ?

Je ne connais actuellement que la carte de 1 Méga octets et diffusée par HP :

HP 82216A 1-MB RAM Card

On pouvait se la procurer, en France, pour 2690 francs, quand je l'ai achetée.

Extérieurement elle se présente comme les autres cartes HP, auxquelles nous sommes habitués, si ce n'est que sa couleur est passée du gris souris au vert intense. Donc cette carte se partitionne en 8 bancs de 128 Ko qui ne sont pas mergeables. La possibilité de verrouillage existe toujours et permet de protéger cette Ram en devenant Rom. La HP48GX sait gérer ces bancs ce qui leur permet d'être actifs à tout moment contrairement à ce qui se passait avec les cartes TDS pour HP48SX dont un seul banc par port n'était actif. La commande `PINIT` doit être appelée pour initialiser tous les ports actifs. Celle-ci n'affecte pas ce qui est déjà stocké et d'après le manuel de référence des HP48G(X) cela se fait en stockant et purgeant un objet dans chacun des bancs accessibles.

J'ai une HP48GX version M et si j'ai par exemple des objets quelconques et des librairies dans les bancs 2 et 3, tout cela ne posant pas de problèmes car utilisables comme sur une HP48SX, j'ai pu constater des incompatibilités avec des librairies fonctionnant normalement en ports 0 et 1 mais non dans les ports suivants. Des rumeurs font état que `PINIT` et `STO` seraient boguées et qu'il existerait aux USA des programmes de remplacement.

Une anomalie assez bizarre :

Une librairie, qui travaille à adresse fixe et ne peut se mettre que dans le port 1, fait usage d'une table contenue dans une chaîne. Cette table étant aussi dans le port 1 j'en ai fait une copie sous le même nom dans le port 3. Il en est résulté que la librairie ne fonctionnait plus jusqu'à ce que je purge ou renomme cette table du port 3. Il y a un mystère que je n'ai pu élucider. Quelqu'un a peut-être une explication ?

La documentation accompagnant cette carte est plus que réduite puisque inexistante. il est indiqué sur la boite emballage qu'une carte d'installation est contenue dans le coffret, or il n'en est rien. Il y a chez HP depuis la sortie des séries G des choses déroutantes.

Malgré toutes ces lacunes et manques d'information cette carte me rend service et est plus pratique que les cartes Multiram pour HP48SX.

Une affaire à suivre...

Guy Toublanc (276)

PROGRAMME BIZARRE ET DERIVES LITTERAIRES

Les éditions du livre *Au centre de la HP48* se suivent mais se ressemblent par la conservation de programmes qui ne sont pas des modèles du genre.

Dans la dernière édition, que l'on peut considérer comme la quatrième pour une grande partie de cet ouvrage, on trouve le programme `R→Q√` (page 402) qui est le même que celui de la troisième édition (page 358), donc en principe devrait être sans gros défauts.

Ce programme est sensé transformer un réel en une fraction rationnelle ou en racine d'une fraction rationnelle, et ne fait que 213,5 octets donc peut se vérifier facilement.

Un rapide examen du listing laisse apparaître des défauts de forme évidents :

- 3 niveaux de [IF THEN ELSE] qui sont à remplacer par l'instruction [CASE] car cela est plus clair et moins octivore:

```
CASE ..... THEN ..... END
     ..... THEN ..... END
     ..... THEN ..... END
                ..... END
```

au lieu de la vieille programmation type HP-28 qui est encore utilisée très souvent dans ce livre pour HP-48:

```
IF
  .....
THEN
  .....
ELSE
  IF
    .....
  THEN
    .....
  ELSE
    IF
      .....
    THEN
      .....
    ELSE
      .....
    END
  END
END
```

Evidemment cette dernière solution est une aubaine pour gonfler un livre volumineux en apparence.

- 2 variables locales créées mais non utilisées par le programme (X et XX). Serait-ce un travail inachevé?

Après la forme passons au fond qui pose de graves problèmes. Si les 2 exemples donnés:

```
        10.5  R→Q√  →  '21/2'
3.49284983931  R→Q√  →  '√(61/5)'
```

sont irréprochables il n'en est plus de même pour les cas suivants :

```
       -10.5  R→Q√  →  SWAP Error:
                       Too Few Arguments
                                 '21/2'
```

```
-3.49284983931  R→Q√  →  SWAP Error:
                         Too Few Arguments
                           '2148665/615161'
```

Réponses incongrues données pour tout argument négatif.

```
√2  R→Q√  →  '√200000000001/
               100000000001
```

Bizarre! alors que →Q → '941664/665857' en mode STD

```
√5  R→Q√  →  5
```

encore plus bizarre mais totalement absurde!

Visiblement ce programme a été inclus très à la légère dans l'ouvrage et ce n'est pas sérieux pour une réédition qui ne devrait plus laisser place à de telles erreurs.

Je sais très bien qu'en matière d'approximation de réels par des rationnels le problème n'est pas des plus simples et qu'il n'existe pas de solution idéale pour tous les cas de figure qui dépendent, comme pour tout calcul numérique, de la précision de la machine et des arrondis.

exemple :

```
999997993279  1/X  1/X  →  999997993274
999997993270  1/X  1/X  →  999997993274
```

Je signale, pour la petite histoire, que lors de la création de la fonction en assembleur FRAC$ et pour HP-71 (JPC mars 87) j'avais innové par rapport à ce qui se faisait à cette époque. En effet tous les programmes imposaient une précision absolue ce qui donne des résultats parfois illogiques. Par exemple pour une précision de 10^{-6} on obtient sur HP48 en 6 FIX avec →Q :

```
4.000001 → 4000001/1000000
3.999999 → 4000003/1000001
4.0000003 → 4
```

alors que logiquement on devrait obtenir 3 fois 4 pour la précision demandée.

Ma fonctions permettait de calculer en mode par défaut avec une précision relative fonction du nombre à approximer et permettant d'obtenir des résultats cohérents. Une option pouvait imposer une précision absolue qui donnait 3 réponse égales à 4 pour l'exemple ci-dessus. Une fonction donc plus sophistiquée que →Q de nos HP48.

Parmi différents algorithmes celui des réduites de fractions continues semble être le plus fiable dans ce domaine et si certains bénéficient de la simplicité ils sont d'un portée plus limitée. Par exemple Sébastien Lalande publia dans un JPC de juillet 88 un programme →FRACTION pouvant donner des résultats aberrants hors de la fourchette 1E4 1E-3. C'est ce même programme qui fut repris dans *Au centre de la HP28.*

Pour remédier aux lacunes du programme R→Q√ je vous propose deux programmes épurés des défauts cités ci-dessus. Malgré les améliorations ils sont plus courts (145.5 et 182 octets contre 213.5) et peuvent être corrigés gratuitement si nécessaire.

Le premier renvoie un rationnel pour la racine carrée d'un nombre entier:

```
√2  R→Q√  →  '114243/80782'
-√5 R→Q√  →  '-(219602/98209)'
```

R→Q√1
145.5 octets cksum # 950Dh

```
« 10 FIX DUP SIGN SWAP ABS DUP →Q
  CASE DUP TYPE 9 ≠
    THEN END
       SWAP SQ →Q DUP TYPE 9 ≠
    THEN SWAP END
       DUP OBJ→ DROP2 SWAP DROP √ 3 PICK
       OBJ→ DROP2 SWAP DROP ≥
    THEN SWAP END
        √
  END SWAP DROP * STD
»
```

Le principe du programme est simple:

si →Q renvoie un réel on en reste là avec N
si →Q renvoie une expression algébrique 'ni1/di1' alors on teste si →Q(N²) renvoie une expression algébrique 'ni2\di2
  si non on garde 'ni1/di1'
  si oui on teste si √ni2 ≥ n/d
    si oui on garde 'ni1/di1'
    si non on prend '√(ni2/di2)'

Le second sait renvoyer la racine d'un entier pour la racine carrée d'un nombre entier:

```
√2  R→Q√  →  '√2'
-√5 R→Q√  →  '-√5'
```

R→Q√2
182 octets cksum # 7FC9h

```
« 10 FIX DUP SIGN SWAP ABS DUP →Q
  CASE DUP TYPE 9 ≠
    THEN END
       SWAP SQ →Q DUP TYPE 9 ≠
    THEN '√A' 'A' ROT 2 →LIST ↓MATCH DROP END
       DUP OBJ→ DROP2 SWAP DROP √ 3 PICK
       OBJ→ DROP2 SWAP DROP ≥
    THEN SWAP END
        √
  END SWAP DROP * STD
»
```

même principe que ci-dessus mais après le deuxième test on fait:

  si non on prend '√N'
    .....

Pour le reste ces deux programmes ont les mêmes fonctionnalités que leur inspirateur.

Donc à vous de choisir. Il est possible d'aller beaucoup plus loin dans la solution de ce type de problème. Ces programmes restent dans le domaine de la simplicité et n'ont pas la prétention de satisfaire tout le monde et en particulier les puristes matheux.

A propos du livre
Voyage au centre de la HP48 G/GX

Tout d'abord je signale que cette rapide analyse est basée sur un exemplaire que j'ai acheté et qui n'a pas été envoyé au club (il y a des principes qui se perdent).

A plusieurs reprises j'ai fait allusion, dans JPC, au livre *voyage au centre de la HP48* et à celui relatif à la HP28. Je recommandais même le premier dans un article d'initiation à l'assembleur, paru dans le numéro 21 de 48Sxtant.

Qu'en est-il aujourd'hui avec cette 4ième édition actualisée pour les HP48G/GX?

Maintenant, bien que cela était visible depuis longtemps, Paul Courbis fait cavalier seul comme auteur du livre.

Le passage de la première à la seconde édition se traduisait par la correction de coquilles ainsi que l'ajout d'une documentation sur les *linked arrays*, donc très peu de choses nouvelles. A l'époque j'avais trouvé assez scandaleux qu'un erratum n'ait pas été glissé dans les exemplaires de la première édition car les coquilles étaient connues dès les premières semaines

de la sortie du livre. Ce qui avait été fait pour le livre sur la HP28 n'a pas été renouvelé et donc on était en régression du côté honnêteté.

De la première à la troisième édition le livre a grossi de 114 pages. Cela est dû principalement à l'ajout de :
- Une extension du chapitre *programmer en langage machine* avec une tentative d'initiation au langage assembleur.

- Une extension du chapitre *les routines utiles* qui ont rapport avec l'assembleur.

- Création du chapitre *"Externals" utiles* dont la grosse partie est constituée de listes d'adresses avec mnémoniques pas toujours évidents.

- Création d'environ seize programmes répartis dans les différentes catégories.

Ces rappels concernant l'évolution de l'ouvrage me semblaient nécessaires pour faire le point sur la quatrième édition et en particulier par rapport à la troisième édition.

Celle-là nous donne un livre encore grossi de 83 pages qui apparemment devient impressionnant par son volume de 601 pages mais ... (cf ci-dessous).

Pour actualiser cette quatrième édition qui est devenue le *Voyage au centre de la HP48 G/GX* il a fallu tenir compte des changements apportés par ces nouvelles séries de machines :

- Rom occupant maintenant tout l'espace adressable par le micro-processeur c'est-à-dire 512 Ko au lieu de 256 Ko et donc avec de nouvelles instructions.

- Ram interne passant de 32 Ko à 128 Ko avec adresse de début déplacée de # 70000h à # 80000h.

- Port 2 pouvant recevoir des cartes d'extension de Ram pouvant aller jusque 4 Mo et que la HP-48GX sait gérer grâce à la technique du bank-switching.

Ces modifications ou possibilités nouvelles par rapport aux HP-48S(X) nécessitaient donc une mise à jour ou création de chapîtres.

En plus des modifications ou ajouts obligatoires les apports principaux sont les suivants :

- 21 nouveaux programmes se répartissant dans les différentes catégories. Parmi eux 7 ne sont pas de l'auteur. On notera la disparition logique de SOLV programme recherchant les racines d'un polynome, ce dernier étant avantageusement remplacé par la nouvelle instruction PROOT.

- Une liste des librairies standard de la HP48.

- Une présentation et une liste des clubs HP dans le monde. L'auteur de ceci étant Wlodek Mier-Jedrzejowicz, le président du club anglais HPCC et qu'un certain nombre ont eu le plaisir de faire la connaissance en janvier lors de la réunion anniversaire de PPC Paris.

- Enfin un chapître qui faisait toujours défaut dans les livres de Paul Courbis et Sébastien Lalande. En effet on trouvait dans les livres relatifs aux calculatrices HP et au HP71 à la fois références bibliographiques et listes de clubs (par exemple dans les livres publiés par Jean-Daniel Dodin alors président du club de Toulouse ou livres publiés par le président de HPCC). D'ailleurs PLC et SL ont fait leurs premières armes dans notre club. Donc ici un chapître dans la bonne tradition.

On peut constater que la liste des objets (avec les adresses) contenus dans la Rom a disparu par enchantement alors que celle des messages a été maintenue. Est-ce un oubli ou le temps manquait-il pour vérifier cette liste ou manque de place dans le livre ?

En tant que fidèle acheteur des *Voyage au centre de la HP...* (HP28 et 48) j'ai quelques remarques à faire sur le contenu du dernier "Voyage".

Sortir de l'imprimerie, début septembre, un livre sur les HP48G/X relevait de la gageure. Evidemment que cet ouvrage était attendu avec impatience mais peut-être qu'un peu moins de précipitation aurait permis de réaliser un livre plus satisfaisant de certains points de vue :

- On aurait aimé avoir un petit aperçu du contenu de la nouvelle partie de la Rom. On aurait pu nous dire que la carte d'équations a pris place dans la partie la plus haute de la Rom et le moyen le plus simple d'utiliser certaines routines dans le cas de recouvrement de cette partie de Rom par la Ram additionnelle. Y a-t-il des choses utiles dans la Rom entre # 80000h et # BFFFFh pour le programmeur en langage machine?

- Si les *externals* ont enfin été utilisés dans les programmes en langage machine (dans la troisième édition on était encore à faire DUP DROP en début de programme) ce ne sont pas toujours les plus adéquats. Dans ces programmes les entiers système en Rom ne sont pas utilisés bien que leurs points d'entrée soient supportés par HP.

- Un parti pris vis-à-vis de la documentation HP et de ses outils de développement qui existent et que l'auteur ne peut pas ignorer car étant sur la liste officielle des développeurs HP, il a plus que la plupart de nous des facilités pour accéder à cette documentation. Cette attitude peut être considérée comme une rétention d'informations. Au lieu des 390 *externals* du livre aux mnémoniques souvent peu explicites il aurait été préférable d'utiliser ceux de chez HP qui sont connus dans le monde entier et permettent aux programmeurs, d'un pays à l'autre, de se comprendre avec plus de 2300 points d'entrée supportés par HP. Grand silence aussi sur les très utiles *goodies disks* où l'on peut trouver un bon échantillonage (documenté et classé) de ce qui se fait hors de l'hexagone. Si dans certains domaines la France peut s'énorgueillir d'avoir de très bons programmeurs sur HP48, il ne faut pas négliger ce qui se fait ailleurs par des gens souvent plus disciplinés qui documentent leurs créations.

- Des programmes à revoir qu'on ne devraient pas retrouver sous la même forme d'une édition à la suivante. J'en ai donné un exemple flagrant au début de cet article. Il y aurait bien d'autres remarques à faire, moins graves mais génantes, sur des programmes à la fois en assembleur et en User Rpl mais je ne peux pas monopoliser ces colonnes avec ces problèmes. Je dirais seulement que pour le langage machine cet ouvrage est considéré comme une référence et en tant que tel les programmes peuvent servir de modèles et donc les débutants programmeront de la même manière ce qui leur fera prendre de mauvaises habitudes. S'il est normal de commencer une initiation sans affinements, passé ce stade les programmes donnés en exemple ne doivent pas avoir de faiblesses de forme. De ce point de vue l'auteur ne semble pas avoir fait beaucoup de progrès depuis la sortie de *Voyage ... HP28*. Au risque de choquer les inconditionnels je pose la question : programme-t-il encore beaucoup ? J'ai fait allusion à de nouveaux programmes faits par d'autres. Certains ont des faiblesses de forme de programmation et je me garderai bien de critiquer leurs auteurs qui ont eu le grand mérite de faire des choses intéressantes, l'un d'eux m'a avoué en toute humilité qu'il requérais l'indulgence. Ce que je reproche c'est au *professionnel* de ne pas avoir aidé les auteurs à redresser les maladresses ce qui aurait permis au lecteur d'avoir des modèles à imiter.

- Un gonflement excessif du volume du livre par des artifices typographiques :
-- Trop de pages avec peu de texte. *Voyage au centre de la HP28* qui faisait 272 pages atteindrait haut la main les 400 pages avec les artifices utilisés dans les éditions pour HP48, de même les ouvrages de J.M Ferrard pourraient augmenter en volume de 50% en utilisant les mêmes techniques. A la place de grands blancs on aurait préféré des commentaires pour les programmes en User Rpl.
-- Les chapîtres du début n'ont pas leur place dans un ouvrage s'adressant à des lecteurs qui veulent aller au-delà de ce que le constructeur met à leur disposition. Je ne suis pas le premier à le dire.

Ce que je retiens de cet ouvrage c'est la partie qui est en relation directe avec son titre. Pour le reste il y a souvent mieux ailleurs.

Donc une bonne compilation d'informations, utile et nécessaire dans ce domaine car actuellement non totalement concurencée ailleurs mais l'auteur qui se transforme progressivement en homme d'affaire (qui marche bien) pourrait lasser ses fidèles lecteurs par un cumul de pis aller et un manque de fini.

Guy Toublanc (276)

CALCULS ASTRONOMIQUES

Les calculs astronomiques sont un domaine de prédilection pour les calculateurs scientifiques. On en trouve de nombreux exemples pour différents machines (y compris dans les *Goodies Disks* pour la HP48), mais aucun ne correspondant à mes souhaits, j'utilise une adaptation des formules utilisées dans le module Navigation de la regrettée HP41, elles mêmes basées sur des équations développées au United States Naval Observatory. La précision atteinte est moyenne, de l'ordre de quelques minutes d'arc (la puissance de calcul de la HP-41 n'était pas très élevée), mais très suffisante pour mes besoins, c'est à dire identifier une planète dans le ciel.

Ce premier article décrit les routines de calcul utilisées. Elles sont à considérer comme une boîte à outils permettant de réaliser diverses applications selon vos besoins. Dans un prochain article, je fournirai les routines pour obtenir une représentation graphique des planètes dans le système solaire, ainsi qu'une vue de leurs positions dans le ciel tel qu'il est vu par nous, pauvres terriens.

Quelques rappels

Je n'entrerai pas dans le détail des théories de modélisations des mouvements des astres dits errants. Je rappelerai seulement que les planètes tournent toutes autour du soleil approximativement dans le même plan (plan de l'ecliptique) et dans le même sens. Le référenciel commode pour décrire ces

mouvements est le système de coordonnées écliptiques dont l'origine est le soleil, les axes X et Y étant dans le plan de l'écliptique, et l'axe Z étant normal à ce plan. L'axe X est dirigé dans la direction de l'équinoxe d'automne. Un objet est repéré par les angles lamda (angle de la projection du point sur le plan de l'écliptique avec l'axe X) et beta (angle du point avec le plan), ainsi que (pour les planètes) par la distance R au soleil. Les planétes décriraient des ellipses parfaites (et immuables) autour du soleil (loi de Képler) si elles ne s'influençaient pas entre elles. Toute la difficulté de la prévision du mouvement précis des planètes réside dans la détermination des variations périodiques des orbites. Plus la précision souhaitée augmente, plus la série de termes périodiques à prendre en compte est importante.

Organisation du répertoire ASTRO :

J'ai suivi le principe d'organisation suivant :

ASTRO contient les programmes d'application, ainsi qu'un sous-répertoire OBJ contenant tous les objets utilitaires, non directement appelés par l'utilisateur. Cela évite d'encombrer le répertoire de travail d'un tas objets et de variables (c'est un des inconvénients du RPL, à mon avis). Les programmes d'applications comporte donc la séquence OBJ ... UPDIR pour accéder aux routines utilitaires de OBJ.

Dans le répertoire ASTRO de ce premier article, vous trouverez :

CALC : Lance le calcul des positions planétaires,
POSPL : liste contenant le résultat du calcul,
DT : dernières date et heure utilisées pour le calcul (pour mémoire), et OBJ.

Je compte enrichir ce répertoire dans de prochains articles (j'accepterais volontiers toute contribution).

Mode d'emploi de CALC

Entrées:
2: date au format jj.mmaaaa ou mm.jjaaaa selon flag -42 (ou 0 si date courante).
1: heure au format hh.mmss (ou 0 si heure courante)
Sorties:
Positions des planetes dans POSPL.

Les positions sont exprimées sous forme de vecteurs en unités astronomiques (1 U.A. = distance moyenne terre-soleil = 149.6E6 km, dixit HP48) L'affichage est mis sous forme de coordonnées sphériques, avec les angles exprimés en degrées, mais peut être ensuite changé par l'utilisateur. CALC fait appel à FA (calcul des arguments fondamentaux commun à tous), puis aux formules propres à chaque corps céleste : Vénus, Terre, Mars, Saturne, Jupiter.

Note : Les coordonnées sphériques ne sont pas tout à fait identiques aux coordonnées écliptiques habituellement utilisées en astronomie, la différence réside dans la 3ème coordonnée qui est l'angle avec l'axe Z en coordonnées sphériques, alors qu'il s'agit de l'angle avec le plan XY en coordonnées écliptiques, les 2 angles étant complémentaires (au sens mathématique: leur somme vaut 90 degrés).

L'utilisation d'un système de coordonnées connu de la 48 permet de disposer de la puissance de calcul sur les vecteurs : par exemple, pour obtenir la distance entre deux planètes, il suffit de faire la différence entre les deux vecteurs.

Exemple :

Positions des planètes le 17 ~~novembre~~ octobre 1993 à 21h48mn45s.

Faire :

```
17.101993 21.4845 CALC
```

POSPL contient alors :

```
{
  :Venus:   [  .719   ∠  150.753   ∠ 86.736 ]
  :Terre:   [  .996   ∠   24.540   ∠ 90     ]
  :Mars:    [ 1.533   ∠ -123.082   ∠ 90.236 ]
  :Jupiter: [ 5.449   ∠ -154.999   ∠ 88.735 ]
  :Saturne: [ 9.796   ∠  -31.189   ∠ 91.417 ]
}
```

Comme prévu, la Terre est pratiquement à 1 U.A. du soleil, et dans le plan de l'écliptique, les autres planètes sont plus ou moins éloignées du plan, en particulier Venus.

Calculons maintenant la distance terre-mars à cette date :
- Extraire la terre et mars :

```
POSPL 2 GET POSPL 3 GET
```

- Puis faire la différence, et en prendre la norme :

```
- ABS
```

On obtient :
2.434 U.A., soit 364 millions de km.

Listings

```
@ ASTRO.SRC
@ J-F Garnier (242)

DIR
@--------------------------------------------------
@CALC
@  calcul des positions
@ entrees:
@  niveau2: date (ou 0 si date courante)
@  niveau1: heure (ou 0 si heure courante)
@ sorties:
@  sauvegarde da date et heure dans DT
@  positions dans POSPL
@--------------------------------------------------
CALC
«
  SWAP
  IF DUP 0 == THEN          @ si date=0,date actuelle
    DROP DATE
    SWAP
    IF DUP 0 == THEN DROP TIME END @ si heure=0...
    SWAP                          @ ..heure actuelle
  END
  SWAP
  DUP2 2 →LIST 'DT' STO @ copie date et heure en
                        @  DT pour info
  OBJ                   @ va dans répertoire OBJ
  JD                    @ calcul temps julian
  FA                    @ arguments fondamentaux
  VENUS                 @ planètes
  TERRE
  MARS
  JUPITER
  SATURNE
  UPDIR                 @ retour dans ASTRO
  5 →LIST 'POSPL' STO   @ construit résultat
»                       @   dans POPSL

DT          @ sauvegarde des dernières date et heure
{ 0 0 }

POSPL   @ résultats
{ }

@ sous répertoire OBJ
OBJ
DIR

@--------------------------------------------------
@JD
@  conversion de la date et de l'heure
@  en temps exprimé en jours Julian (Tjd)
@ entrees:
@  niveau2: date
@  niveau1: heure
@ sorties:
@  niveau1: temps en jour julian
@--------------------------------------------------
JD
«
  0 0 0 → d t j m a    @ date time jour mois année
  «
    IF -42 FS? THEN
      d DUP IP 'j' STO     @ format jj.mmaaaa
      FP 100 * DUP IP 'm' STO
    ELSE
      d DUP IP 'm' STO     @ format mm.jjaaaa
      FP 100 * DUP IP 'j' STO
    END
    FP 1E4 * 'a' STO


    a 367 *
    m 9 + 12 / IP a + 7 * 4 / IP -
    m 9 - 7 / IP a + 100 / IP 1 + 3 * 4 / IP -
    m 275 * 9 / IP +
    j +
    730516.5 -
    t HMS→ 24 / +
  »
»



@--------------------------------------------------
@FA
@ calculs des arguments fondamentaux
@ entrees:
@  niveau1: temps en jours Julian (Tjd)
@ sorties:
@  Ts
@  Lm,Gm,Fm,Ls,Gs,L2,G2,F2,L4,G4,F4,L5,G5,L6,G6
@  Dms,Om,Eps,Dl
@--------------------------------------------------
FA
«
  → Tjd
  «
    DEG

    Tjd 36525 / 1 + 'Ts' STO
    218.31624 Tjd 13.17639643 * + 'Lm' STO
    134.96292 Tjd 13.06499295 * + 'Gm' STO
    93.27276  Tjd 13.22935027 * + 'Fm' STO
    280.46592 Tjd .985647348  * + 'Ls' STO
    -2.47464  Tjd .9856       * + 'Gs' STO
    181.97928 Tjd 1.602169    * + 'L2' STO
    50.40828  Tjd 1.60213022  * + 'G2' STO
    105.29928 Tjd 1.60214407  * + 'F2' STO
    -4.55292  Tjd .524071181  * + 'L4' STO
    19.38816  Tjd .52402078   * + 'G4' STO
    305.88984 Tjd .524050085  * + 'F4' STO
    32.25888  Tjd .083091215  * + 'L5' STO
    20.35116  Tjd .08309121   * + 'G5' STO
    47.9862   Tjd .033459736  * + 'L6' STO
    317.87532 Tjd .033459736  * + 'G6' STO
```

```
    Lm Ls - 'Dms' STO
    Lm Fm - 'Om' STO
    Om COS 400 / Ts 1.3056E-2 * - 23.4522222 +
    'Eps' STO
    Om SIN 4.7222E-3 * 'Dl' STO
  »
»

@-------------------------------------------------
@TERRE, VENUS, MARS, SATURNE, JUPITER
@ entrees:
@  arguments calculés par FA
@ sorties:
@  niveau1: position en coordonnées sphériques,
@           et en unités U.A.
@-------------------------------------------------
TERRE
«
  0 0 0 → r b l          @ R beta lambda
  «
    DEG                       @ calculs en degrés

    Gs SIN 1.919 *
    Gs 2 * SIN .02 * +
    Gs SIN Ts * 4.72E-3 * -
    Gs G5 - COS 1.94E-3 * -
    Ls 360 MOD + Dl - 180 +
    'l' STO

    0 'b' STO

    1.00014 Gs COS .01675 * -
    Gs 2 * COS 1.4E-4 * -

    'r' STO r l 90 b -        @ transforme coord.
                              @  ecliptiques -> sphérique
    -15 SF -16 SF             @ vect. en format coord.
                              @  sphériques
    →V3 "Terre" →TAG
  »
»

VENUS
«
  0 0 0 → r b l
  «
    DEG

    G2 SIN .78166667 *
    F2 2 * SIN .05027778 * -
    G2 SIN Ts * 180 / -
    G2 2 * SIN 300 / +
    Gs G2 - 2 * COS 360 / -
    Gs G2 - 3 * COS 1.944E-3 * +
    L2 360 MOD + Dl - 'l' STO
    F2 SIN 3.393056 *
    G2 SIN F2 COS * .0461 * +
    'b' STO

    G2 COS -.00493 * .72335 +
    'r' STO

    r l 90 b - -15 SF -16 SF
    →V3 "Venus" →TAG
  »
»

MARS
«
  0 0 0 → r b l
  «
    DEG

    G4 SIN 10.68083333 *
    G4 2 * SIN .6216666667 * +
    G4 3 * SIN .050277778 * +
    F4 2 * SIN .014444444 * -
    G4 SIN Ts * .010277778 * +
    G4 G5 2 * - COS 6.111111E-3 * -
    L4 360 MOD + Dl -
    'l' STO

    F4 SIN 1.834166667 *
    G4 F4 - SIN .17277778 * +
    G4 F4 + SIN .17083333 * +
    G4 2 * F4 + SIN 56.25 / +
    'b' STO

    G4 3 * COS -4.7E-4 *
    G4 2 * COS -6.6E-3 * +
    G4 COS .1417 * - 1.53031 +
    'r' STO

    r l 90 b - -15 SF -16 SF →V3
    "Mars" →TAG
  »
»

SATURNE
«
  0 0 0 → r b l
  «
    DEG

    G6 SIN 6.40139 *
    Ts 1.39278 * +
    G5 2 * G6 5 * - COS .746944 * -
    .696389 + G5 2 *
    G6 5 * - SIN .32694 * +
    G5 2 * G6 4 * - COS .22944 * -
    G6 2 * SIN .222778 * +
    G5 G6 2 * - SIN .118056 * +
    G6 COS Ts * 15.7205 / -
    G5 2 * G6 6 * - COS .0425 * -
    G6 SIN Ts * .039444 * -
    G6 COS .031667 * -
    G5 2 * G6 5 * - SIN Ts * .028056 * +
```

```
    L6 2 * DUP COS .01944 *
    SWAP SIN .01861 * - -
    G5 2 * G6 6 * - SIN .018333 * +
    G5 2 * G6 5 * - COS Ts * 60 / +
    G5 G6 3 * - SIN 87.80 / +
    G6 3 * SIN 92.308 / +
    G5 G6 - SIN 8.6111E-3 * +
    G5 G6 - 2 * SIN 8.6111E-3 * +
    G5 2 * G6 3 * - COS 8.056E-3 * -
    L6 + 360 MOD Dl -
    'l' STO

    G6 DUP COS -.92944 *
    SWAP SIN 2.30472 * +
    G6 2 * DUP COS .0525 *
    SWAP SIN .128333 * - -
    .051389 +
    G6 COS Ts * .021944 * +
    G5 2 * G6 4 * - COS 50.704 / -
    G5 2 * G6 6 * - SIN .012778 * +
    G5 2 * G6 6 * - COS 80 / -
    G6 3 * SIN .008056 * + 'b' STO
    9.55774 G6 COS .53252 * -
    G5 2 * G6 4 * - SIN .01878 * -
    G6 2 * COS .01482 * -
    G5 G6 - SIN .00817 * +
    'r' STO

    r l 90 b - -15 SF -16 SF →V3
    "Saturne" →TAG
  »
»


JUPITER
«
  0 0 0 → r b l
  «
    DEG

    G5 SIN 5.537222 * Ts 1.3952778 * +
    .6975 + G5 2 * G6 5 * - COS .303611 * +
    G5 2 * SIN 5.99 / +
    G5 2 * G6 5 * - SIN .133056 * -
    G5 G6 - 2 * SIN .051389 * -
    G5 3 * G6 5 * - SIN .038056 * +
    G5 G6 2 * - SIN .036389 * -
    G5 G6 - COS .02194 * +
    G5 G6 - 2 * COS .02111 * -
    G5 COS Ts * .020556 * -
    G5 SIN Ts * 52.94 / +
    G5 2 * G6 3 * - COS 54.55 / +
    G5 3 * G6 5 * - COS .0175 * +
    L5 + 360 MOD Dl -
    'l' STO

    G5 COS -1.3033 *
    G5 SIN 13.9 / + 1
    G5 2 * COS - .063056 * +
    'b' STO

    G5 2 * COS -.00604 *
    G5 COS .25122 * -
    5.20883 +
    'r' STO

    r l 90 b - -15 SF -16 SF →V3
    "Jupiter" →TAG
  »
»
```

Variables auxilliaires à créer en les initialisant à 0 :

```
Ts Lm Gm Fm Ls Gs L2 G2 F2 L4
G4 F4 L5 G5 L6 G6 Dms Om Eps Dl

END  @ fin du DIR OBJ

END  @ fin du DIR ASTRO
```

Jean-François GARNIER (242)

HP95

J. Belin — Paneau de paramètrage du HP95 *24*

Le coin des codes *33*

PARAMETRAGE FACILE

Le HP95 possède de nombreux paramètres de configuration non présents sur les IBM PC, tels que le volume sonore ou le contraste. Malheureusement, si certains sont accessibles par le SETUP du System Manager, certains autres doivent être modifiés par la ligne de commande du Dos. Pas très ergonomique... De plus, certains paramètres tels que le temps d'auto-extinction, ne peuvent pas être modifiés car HP n'a pas prévu d'utilitaires pour cela.

Pour palier à ces problèmes, il existe dans le domaine public de nombreux utilitaires tels que HP95CTRL ou TIMEOUT (inclus dans notre disquette n°1).

Je vous présente cependant un utilitaire (appelé 95PANEL) permettant de regrouper en un seul programme interactif tous ces utilitaires. Je me suis efforcé, en prime, de limiter sa taille à moins de 1536 octets, soit la taille de 3 secteurs des disques du HP95.

Ce programme ne fonctionne pas sur le HP100LX.

Présentation de l'écran :

```
                    HP95 Panel

Volume          : 2          Io          : OFF
Contrast        : 6          Port        : RS232

Light sleep     : ON         Baud        : 115200
Time out        : 3  mn      bitS        : 7
                             paRity      : EVEN
cUrsor track    : OFF        stOps       : 1
Alt arrows      : ON
-------------------------------------------------------
                Total    Free    Batteries
Mem          :   508K    360K   2.5V (Main)
Disk C       :  1021K     25K   2.9V (Backup)
Card (RO)    : 20000K   5832K
```

La première partie de l'écran affiche les données qui peuvent être modifiées. Pour chaque paramètre, il suffit de presser la lettre majuscule correspondante (pour certaines touches, il est possible de décroître la valeur en "Shiftant" la touche).

Volume
Cette ligne permet de régler le niveau du bip sonore. Notez que certains programmes sonores travaillant à un très bas niveau (SND.EXE) travaillent toujours à un niveau constant. D'autre part, le System Manager stocke sa propre valeur du volume dans le fichier SETUP.ENV de façon indépendante de celle du DOS.

Contrast
Cette ligne affiche le contraste courant. Elle est plus présentée ici à titre d'information que par but purement utilitaire, puisque vous savez sûrement que le contraste est modifiable en permanence par les touches [ON] [+] et [ON] [-].

Light sleep et *Timeout*
Pendant l'utilisation de certains programmes (de communication, par exemple) il est nécessaire de désactiver l'auto-extinction du HP95. La première de ces lignes permet donc cela. La seconde ligne permet de régler ce temps par pas d'une minute (de 1 à 59). Une valeur de '0' désactive l'extinction automatique.

cUrsor tracking
Cette ligne permet d'activer ou désactiver la possibilité qu'à le HP95 de déplacer la fenêtre d'affichage pour suivre le curseur. Je vous rapelle que ce paramètre (ainsi que le suivant) peut être modifié dans un fichier batch par la commande DISPCTL.

Alt arrows
Cette ligne permet d'activer ou désactiver la possibilité qu'à le HP95 de déplacer la fenêtre d'affichage à l'aide des touches ALT + Flèches.

Io
Cette ligne permet d'activer la liaison série pour l'utilisation des programmes de communication. N'oubliez pas de la désactiver ensuite, car une liaison série active en permanence est le premier facteur d'usure des piles.

Port
Cette ligne permet de selectionner le port de sortie des données : InfraRouge ou RS232.

Baud, bitS, paRity et stOps
La commande MODE n'étant pas implantée sur le HP95, j'ai inclus les fonctions de configuration de la RS232 :
- Baud : selectionne la vitesse de transmission. Les valeurs reconnues sont les mêmes que celles admises par le System Manager : 300, 1200, 2400, 4800, 9600, 19200, 57600 et 115200 bauds. Notez que, comme le volume, les applications System Manager stockent leurs propres valeurs de configuration.
- Bits : Sélectionne longueur donnees 5, 6, 7 ou 8 bits.
- Parité : Sélectionne la parité : NONE, EVEN, ODD.
- Stops : Nombre de bits de stops : un ou deux.

En plus de ces paramètres, le programme affiche d'autres valeurs courantes, telles que la taille mémoire disponible ou la capacité des batteries.

Notez que la taille totale indiquée par le disque C tient compte de la présence des fichiers inclus dans la Rom. J'aurai préféré y voir la valeur sélectionnée dans le Setup, mais c'est ce que nous envoie le DOS... Il n'était malheureusement pas possible de soustraire une constante, car la taille et le nombre de ces fichiers et dépendante de la version du HP95.

Le programme affiche les informations des cartes mémoire. La mention (RO) (Read Only) indique que l'on est en presence d'une Rom ou d'une carte Ram protégée en écriture. Le programme reconnait les Cartes SunDisk, à la condition que le driver soit présent.

Les valeurs indiquées pour les tensions des batteries sont indicatives, mais relativement proches de la réalité. Il est a noter que le HP95 n'offre pas la possibilité de déterminer avec précision la tension de la pile incluse dans la carte.

Pour sortir du programme, il suffit de presser la touche ENTER.

Informations sur le fonctionnement du programme

Tout d'abord, vous avez pu noter que j'ai indiqué au début de cet article que ce programme est incompatible avec le HP100LX. Ceci est dû au fait que j'ai dû court-circuter (en accédant directement au matériel) les routines standard communes aux deux machines, pour lire certaines valeurs clés. Or, si les deux machines sont compatibles à un haut niveau, il faut savoir que les deux machines sont *complètement* diférentes au niveau du matériel... Suite à un prochain numéro !

Un des gros problèmes à résoudre dans ce programme était l'existence des cartes SunDisk. En effet, lorsqu'il n'y a pas de driver, le HP95 se trouve dans une situation imprévue, puisque le port PCMCIA l'informe qu'une carte est présente mais qu'il ne peut pas la lire. Cela aboutit au fameux message :

```
General Failure error reading drive A
  Abort, Retry, Ignore ?
```

Il est inutile de dire que l'apparition d'un tel message rend l'utilisation du programme impossible. Heureusement, le Dos nous permet de résoudre ce problème. Pratiquement, cela se fait en créant une petite routine d'interception de l'interruption 24h, renvoyant la réponse "Fail".

Un autre problème posé par les cartes, moins prévisible celui là, est que les fonctions d'accès aux informations de la cartes se sont révélées particulièrement lentes (près d'une demi seconde !). Et cela quelque soit le type de la carte. Ceci rendant l'affichage effroyablement lent... Ceci m'a donc ammené à réécrire le programme de façon à ne réafficher les données des cartes qu'après rallumage de la machine, avec une recherche du code de la touche [ON] dans le buffer du clavier.

Afin de réduire la taille du programme, j'ai dû utiliser quelques astuces et bousculer un peu mes habitudes programmation qui consistent écrire du code très structuré :
- Réduction du code de l'écran de présentation, en remplacant les espaces par des tabulations (code ASCII 9) et optimisation de la position des colonnes en fonction de ces tabulations.
- Utilisation d'un code compact (même si il n'est pas "esthétique"), plutôt que la rapidité ou la logique (voir la routine DiskSpce !).
- Suppression des sauvegardes des registres lors des appels aux fonctions. Le gain en octets est indéniable, mais nécessite une grande prudence.
- Si un sous-programme se termine par un appel a un autre sous-programme suivi d'un ret, il est préférable de faire un branchement direct à ce sous programme.
- On peut même supprimer le Jump, si on place le sous programme appelant juste avant le sous programme appelé (voir dspon/printf).
- Enfin, ayant besoin d'une chaine vide de quatre caractères pour effacer la chaine "(RO)", j'ai profité du fait que j'utilise une chaine vide de même longueur dans le tableau s_PAR (elle est vide car la valeur d'index 2, n'est pas utilisée par la parité). En y placant s_RW, j'ai donc économisé 5 octets. C'est toujours ça de gagné.

Bien sûr, ces dernières astuces doivent être très distinctement indiquées, afin de ne pas créer de bugs en cas de modifications du programme.

La dernière astuce utilisée consiste à utiliser l'option d'optimisation "-m2" du *Turbo Assembler*, permettant d'éliminer les nombreux NOP que l'assembleur met après certaines instructions de saut.

D'autre part, et dans un autre registre, vous pouvez noter que le texte de la première ligne de l'écran de présentation ne comporte pas de tabulations en replacement des espaces. Cela est dû au fait que j'ai constaté que celles-ci étaient affichées sur une position différente à chaque exécution du programme, comme si les tabulations correspondaient à un nombre aléatoire de caractères ! Ce dysfonctionnement ne se produit que pour la première ligne. Est-ce un bug des fonctions d'affichages de la Rom ?

Vous trouverez les codes hexadécimaux de ce programme dans le *Coin des Codes*.

Jacques Belin (123)

```
;*****************************************************************************
;*****************************************************************************
;**                          95PANEL.ASM                                    **
;**                      Configuration du HP95                              **
;**                                                                         **
;**   Dernière modification : 15/08/93        Auteur : Jacques Belin        **
;**       Création                                                          **
;**                                                                         **
;*****************************************************************************
;*****************************************************************************
        DOSSEG
code    segment para 'CODE'
        assume cs:code
        ORG     100h

start:  jmp     strt                    ; saut à la première inst. exécutée

K_ENTER EQU     000Dh                   ; code de la touche ENTER

Not_95  DB      "This program runs only on an HP95LX",13,10,0

                                        ; Ecran de présentation
Dispstr DB      13,9,"       HP95 Panel",13,10
        DB      10
        DB      "Volume       :",9,9,"Io",9,":",13,10
        DB      "Contrast     :",9,9,"Port",9,":",13,10
        DB      10
        DB      "Light sleep  :",9,9,"Baud",9,":",13,10
        DB      "Time out     :",9,"  mn",9,"bitS",9,":",13,10
        DB      9,9,9,"paRity",9,":",13,10
        DB      "cUrsor track :",9,9,"stOps",9,":",13,10
        DB      "Alt arrows   :",13,10
        DB      "-----------------------------------------",13,10
        DB      9,"     Total   Free   Batteries",13,10
        DB      "Mem",9,"  :",9,9,9,"(Main)",13,10
        DB      "Disk C",9,"  :",9,9,9,"(Backup)",13,10
s_CARD  DB      "Card      : Not Installed",0

P_VOL   EQU     0020Fh                  ; codes de position des données sur
P_CONT  EQU     0030Fh                  ;   l'écran, (voir fonction gotoxy)
P_SLEEP EQU     0050Fh
P_TIME  EQU     0060Fh
P_CURS  EQU     0080Fh
P_ALT   EQU     0090Fh
P_IO    EQU     00222h
P_PORT  EQU     00322h
P_BAUD  EQU     00522h
P_BITS  EQU     00622h
P_PAR   EQU     00722h
P_STOP  EQU     00822h

P_TMEM  EQU     00C0Bh
P_FMEM  EQU     00C12h
P_BMAIN EQU     00C1Bh
P_TDISK EQU     00D0Bh
P_FDISK EQU     00D12h
P_BBACK EQU     00D1Bh
P_CARD  EQU     00E00h
P_RW    EQU     00E05h
P_TCARD EQU     00E0Bh
P_FCARD EQU     00E12h

s_IR    DB      "IR   ",0
s_RS232 DB      "RS232",0
                                        ; Codes du diviseur de fréquence RS232
n_BAUD  DW      384                     ; 300    bauds
        DW      96                      ; 1200   bauds
        DW      48                      ; 2400   bauds
        DW      24                      ; 4800   bauds
        DW      12                      ; 9600   bauds
        DW      6                       ; 19200  bauds
        DW      2                       ; 57600  bauds
        DW      1                       ; 115200 bauds

s_PAR   DB      "NONE",0                ; Codes de parité
        DB      "ODD ",0
s_RW    DB      "    ",0                ; on utilise la chaine vide pour s_RW
        DB      "EVEN",0
L_PAR   EQU     5                       ; longueur chaines (y compris caract 0)

s_RO    DB      "(RO)",0

s_tmp   DB      "      ",0              ; stockage temporaire des nombres ASCII

keymap  DB      "v"                     ; table d'index des touches accessibles
        DB      "V"
        DB      "c"
        DB      "C"
        DB      "l"
        DB      "t"
        DB      "T"
        DB      "u"
        DB      "a"
        DB      "i"
        DB      "p"
        DB      "b"
        DB      "B"
        DB      "s"
        DB      "S"
        DB      "r"
        DB      "R"
lastkey DB      "o"

vectors DW      OFFSET incvol           ; Table des vecteurs vers les fonctions
        DW      OFFSET decvol           ;  de modification
        DW      OFFSET inccont
        DW      OFFSET deccont
        DW      OFFSET invslp
        DW      OFFSET inctime
        DW      OFFSET dectime
        DW      OFFSET invcurs
        DW      OFFSET invalt
        DW      OFFSET invio
        DW      OFFSET invport
        DW      OFFSET incbaud
        DW      OFFSET decbaud
        DW      OFFSET incbits
        DW      OFFSET decbits
        DW      OFFSET incpar
        DW      OFFSET decpar
        DW      OFFSET invstop


strt:   cld                             ; on assure l'effacement du flag D

        mov     ax,2524h                ; on remplace le vecteur de l'int 24h
        mov     dx,offset nocrd_0       ;   par celui de la routine nocrd_0
        int     21h

        mov     ah,4Ah                  ; on réjuste taille du bloc programme
        mov     bx,2048/16              ;  à 2048 octets (prgm + pile)
```

```
        int     21h
        mov     sp,2048                 ; on replace le pointeur de pile

        mov     ax,40h                  ; on positionne ES sur le segment des
        mov     es,ax                   ;   données du BIOS

        mov     ax,4dd4h                ; vérification d'exécution
        int     15h                     ;  sur HP95 (HP100 refusé)
        cmp     bx,'HP'
        jne     main_1
        cmp     cx,0101h
        je      main_2                  ; si hp95, on passe
main_1: mov     dx,OFFSET Not_95
        call    printf
        jmp     exit                    ; sortie du programme

main_2: and     byte ptr es:[17h],10111111b ; assurer l'effacement du CAPS LOCK
        mov     ax,0007h                ; effacement de l'écran
        int     10h
        mov     dx,OFFSET Dispstr       ; affichage écran de présentation
        call    printf
        call    dspvol                  ; affichages des données initiales
        call    dspcont
        call    dspslp
        call    dsptime
        call    dspcurs
        call    dspalt
        call    dspio
        call    dsppport
        call    dspbaud
        call    dspbits
        call    dsppar
        call    dspstop

                                        ; AFFICHAGES INFORMATIONS MEMOIRE
main_3: mov     ax,es:[013h]            ; affichage taille mémoire maxi
        mov     dx,offset P_TMEM
        call    dspspace

        mov     bx,0FFFFh               ; lecture mémoire disponible
        mov     ah,48h
        int     21h
        mov     ax,bx
        mov     cl,6                    ; conversion de paragraphes en Koctets
        shr     ax,cl                   ;  (shr ax,6 = ax*16/1024)
        inc     ax                      ; on ajoute les 2ko réservés par 95panel
        inc     ax
        mov     dx,offset P_FMEM        ; affichage de la mémoire dispo
        call    dspspace

        mov     al,0                    ; affichage tension bat. principales
        mov     dx,offset P_BMAIN
        call    dispbat

        mov     al,1                    ; affichage tension pile backup
        mov     dx,offset P_BBACK
        call    dispbat

        mov     dl,3                    ; AFFICHAGE INFOS DISQUE C
        call    diskspce                ; lecture capacités totale et disponible
        push    bx                      ; sauve capacité disponible
        mov     dx,offset P_TDISK       ; affichage capacité totale
        call    dspspace
        pop     ax                      ; on récupère capacité disponible dans AX
        mov     dx,offset P_FDISK       ; affichage
        call    dspspace

                                        ; AFFICHAGE INFOS CARTE
        mov     dx,0E304h               ; Lecture des infos du port PCMCIA
        in      al,dx
        test    al,01000000b            ; Carte présente ?
        jnz     main_5                  ; oui, saut vers affichage infos cartes
main_4: mov     dx,P_CARD               ; non, affichage "not installed"
        call    gotoxy
        mov     dx,offset s_CARD
        call    printf
        jmp     main_7
        ;-----------
main_5: mov     dx,P_RW                 ; positionnement curseur après "card"
        call    gotoxy
        mov     dx,offset s_RW          ; DX pointe sur chaine vide
        test    al,00010000b            ; écriture autorisée ?
        jnz     main_6
        mov     dx,offset s_RO          ; DX pointe sur "(RO)"
main_6: call    printf                  ; affichage
        mov     dl,1                    ; lecture capacités totale et disponible
        call    diskspce
        jc      main_4                  ; si carry, erreur -> aff "not installed"
        push    bx                      ; sauve capacité disponible
        mov     dx,offset P_TCARD       ; affichage capacité totale
        call    dspspace
        pop     ax                      ; on récupère capacité disponible dans AX
        mov     dx,offset P_FCARD       ; affichage
        call    dspspace

main_7: mov     dx,00F00h               ; positionne curseur sur derniere ligne
        call    gotoxy

                                        ; LECTURE DU CLAVIER
main_8: mov     ah,11h                  ; lecture état du clavier
        int     16h
        jnz     main_a                  ; si touche dans buffer -> traitement
                                        ; TEST DU TRAITEMENT DE LA TOUCHE "ON"
        mov     bx,es:[1Ah]             ; lecture addresse dernière touche
        dec     bx                      ;   placée dans buffer
        dec     bx
        cmp     bx,1Eh                  ; (ajustement sur le buffer circulaire)
        jae     main_9
        mov     bx,3Ch
main_9: cmp     word ptr es:[bx],0A000h ; La touche ON a-t-elle été pressée ?
        jne     main_8                  ; non, on reprend scrutation du clavier
        mov     word ptr es:[bx],0h     ; oui, on efface le code (car on
                                        ;  repassera dessus au cours de la
                                        ;  prochaine boucle)...
        jmp     main_3                  ; ...puis on réaffiche les infos carte

                                        ; TRAITEMENT DE LA TOUCHE PRESSEE
main_a: mov     ah,10h                  ; Lecture du buffer clavier
        int     16h
        cmp     al,0                    ; code étendu ?
        je      main_c                  ;   oui, pas touche pas utilisée
        cmp     al,K_ENTER              ; touche ENTER ?
        je      exit                    ;   oui, sortie du programme
        mov     si,offset keymap        ; SI = base de la table des touches
main_b: cmp     [si],al                 ; touche pressée = entrée table ?
        je      main_d                  ; oui, on traite
        inc     si                      ; non, on pointe sur entrée suivante
        cmp     si,offset lastkey       ; a-t-on dépassé la dernière entrée ?
        jbe     main_b                  ; non, on boucle
main_c: jmp     main_7                  ; oui: mauvaise touche -> on retourne
        ;-------------                  ;   a la scrutation
main_d: sub     si,offset keymap        ; on récupère l'index dans la table
```

```
        shl     si,1                    ; multiplication index par 2 (byte->word)
        call    ds:[si+offset vectors]  ; appel de la routine de traitement+aff
        jmp     main_7                  ; retour à la scrutation du clavier
        ;-------------

exit:   mov     ah,4ch                  ; retour au DOS
        int     21h
        ;---------

;************************ SOUS-PROGRAMMES ************************
                                        ; TRAITEMENT ET AFFICHAGE DU VOLUME
incvol: call    getE309                 ; lecture valeur courante
        inc     al                      ; incrémentation
        jmp     decvo_1
        ;-------------
decvol: call    getE309                 ; lecture valeur courante
        dec     al                      ; décrémentation
decvo_1: and    al,00000011b            ; on garde que les 2 bits de poids faible
        mov     ah,48h                  ; écriture nouvelle valeur
        int     15h
        mov     dl,7                    ; BEEP !
        call    putch
dspvol: call    getE309                 ; lecture valeur courante
        mov     cx,offset P_VOL
dspvo_1: xor    ah,ah                   ; on assure AH = 0
        mov     bx,2                    ; on n'affichera que deux caractères maxi
        xor     dx,dx                   ; seul AX est utilisé
        jmp     displeft                ; affichage
        ;-------------

                                        ; TRAITEMENT ET AFFICHAGE DU CONTRASTE
inccont: call   getE309                 ; lecture valeur courante
        inc     bl                      ; incrémentation
        jmp     decco_1
        ;-------------
deccont: call   getE309                 ; lecture valeur courante
        dec     bl                      ; décrémentation
decco_1: and    bx,00001111b            ; on garde que les 4 bits de poids faible
        mov     ax,4700h                ; sélection nouvelle valeur
        int     15h
dspcont: call   getE309                 ; lecture valeur courante
        mov     ax,bx                   ; contraste dans AX
        mov     cx,offset P_CONT        ; saut dans routine d'affichage du volume
        jmp     dspvo_1
        ;------------

                                        ; TRAITEMENT+AFFICHAGE DU LIGHT SLEEP
invslp: xor     byte ptr es:[0D6h],00000001b ; inversion valeur courante
dspslp: mov     al,es:[0D6h]            ; lecture valeur courante
        mov     dx,offset P_SLEEP       ; saut dans routine d'affichage Alt Arrows

        jmp     dspal_1
        ;-------------

                                        ; TRAITEMENT+AFFICHAGE DU TIME OUT
inctime: add    es:[0A3h],1092          ; incrémentation valeur courante
        jmp     dsptime                 ;   (1092 ticks = 60 secondes)
        ;-------------
dectime: sub    es:[0A3h],1092          ; décrémentation valeur courante
dsptime: mov    ax,es:[0A3h]            ; lecture valeur courante
        add     ax,20                   ; petit offset
        mov     cx,60                   ; multiplication valeur courante par 60
        mul     cx                      ;   (DX contiendra le temps en minutes)
        mov     ax,dx                   ; AX = temps en minutes
        mov     cx,offset P_TIME        ; saut dans routine d'affichage du volume
        jmp     dspvo_1
        ;------------

                                        ; TRAITEMENT+AFFICHAGE CURSOR TRACKING
invcurs: xor    byte ptr es:[93h],00000010b ; inversion du bit cursor tracking
dspcurs: mov    al,es:[93h]             ; lecture valeur courante
        shr     al,1                    ; décalage d'un bit
        mov     dx,offset P_CURS        ; saut dans routine d'affichage AltArrows
        jmp     dspal_1
        ;-------------

                                        ; TRAITEMENT+AFFICHAGE ALT ARROWS
invalt: xor     byte ptr es:[93h],00000001b  ; inversion du bit alt arrows
dspalt: mov     al,es:[93h]             ; lecture valeur courante
        mov     dx,offset P_ALT
dspal_1: and    al,1                    ; on ne garde que le bit de poids faible
        xor     al,1                    ; on inverse le bit (est inversé en mem)
        jmp     dispi_1                 ; affichage de ON ou OFF
        ;------------

                                        ; TRAITEMENT+AFFICHAGE ACTIVATION PORT
invio:  mov     dx,0E301h               ; lecture valeur courante
        in      al,dx
        xor     al,00000100b            ; inversion du bit
        out     dx,al                   ; réécriture nouvelle valeur
dspio:  mov     dx,0E301h               ; lecture valeur courante
        in      al,dx
        shr     al,1                    ; décalage de 2 bits
        shr     al,1
        and     al,1                    ; on ne garde que le bit de poids faible
        mov     dx, offset P_IO         ; affichage de ON ou OFF
dispi_1: jmp    dispon
        ;------------

                                        ; TRAITEMENT+AFFICHAGE SELECTION PORT
invport: mov    dx,0E30Ah               ; lecture valeur courante
        in      al,dx
        xor     al,00000100b            ; inversion du bit correspondant
        out     dx,al                   ; réécriture nouvelle valeur
dspport: mov    dx,0E30Ah               ; lecture valeur courante
        in      al,dx
        shr     al,1                    ; décalage de 2 bits
        shr     al,1
        and     al,1                    ; on ne garde que le bit de poids faible
        mov     dx,offset P_PORT
        call    gotoxy                  ; positionnement curseur
        mov     dx,offset s_IR
        cmp     al,1                    ; bit = 1 ?
        je      dspor_1
        mov     dx,offset s_RS232       ; non, on travaille en RS232
dspor_1: jmp    printf                  ; affichage
        ;------------

                                        ; TRAITEMENT+AFFICHAGE DE LA VITESSE
incbaud: call   getsrial                ; lecture valeur courante
        inc     bx                      ; incrémentation
        jmp     decba_1
        ;-------------
decbaud: call   getsrial                ; lecture valeur courante
        dec     bx                      ; décrémentation
decba_1: and    bx,00000111b            ; on garde que les 3 bits de poids faible
        call    setsrial                ; séléction nouvelle valeur
dspbaud: call   getsrial                ; lecture valeur courante
        shl     bl,1                    ; multiplication par 2 pour accès table
        mov     cx,ds:[bx+offset n_BAUD] ; diviseur dans CX
        mov     dx,1                    ; DX:AX = 115200
        mov     ax,0C200h
```

```
        cmp     cx,1                    ; on saute 115200/1 (pas possible
        je      dspba_1                 ;  sur 80x86)
        div     cx                      ; on divise 115200 par le diviseur
dspba_1: mov    cx,offset P_BAUD        ; affichage
        mov     bx,6
        jmp     displeft
        ;-------------


                                        ; TRAITEMENT+AFFICHAGE DU NBRE DE BITS
incbits: call   getsrial                ; lecture valeur courante
        inc     dx                      ; incrémentation
        jmp     decbi_1
        ;-------------
decbits: call   getsrial                ; lecture valeur courante
        dec     dx                      ; décrémentation
decbi_1: and    dl,00000011b            ; on garde que les 2 bits de poids faible
        call    setsrial
dspbits: call   getsrial                ; lecture valeur courante
        mov     al,dl
        add     al,5
        mov     cx,offset P_BITS        ; saut dans routine d'affichage du volume
        jmp     dspvo_1
        ;------------


                                        ; TRAITEMENT+AFFICHAGE DE LA PARITE
incpar:  call   getsrial                ; lecture valeur courante
incpa_1: inc    ax                      ; incrémentation
        cmp     al,00000010b            ; si al = 2..
        je      incpa_1                 ; ..on incrémente encore une fois
        jmp     decpa_2
        ;-------------
decpar:  call   getsrial                ; lecture valeur courante
decpa_1: dec    ax                      ; décrémentation
        cmp     ax,00000010b            ; si al = 2..
        je      decpa_1                 ; ..on décrémente encore une fois
decpa_2: and    al,00000011b            ; on garde que les 2 bits de poids faible
        call    setsrial                ; écriture nouvelle valeur
dsppar:  call   getsrial                ; lecture valeur courante (dans AL)
        mov     cl,L_PAR                ; cl = longueur chaine (y compris code 0)
        mul     cl                      ; AX = AL * CX = offset dans table
        add     ax,offset s_PAR         ; AX = adresse chaine
        mov     dx,offset P_PAR         ; positionnement curseur
        call    gotoxy
        mov     dx,ax                   ; affichage
        jmp     printf
        ;------------


                                        ; TRAITEMENT+AFFICHAGE DES BITS STOP
invstop: call   getsrial                ; lecture valeur courante
        xor     cl,00000001b            ; inversion du bit
        call    setsrial                ; écriture nouvelle valeur
dspstop: call   getsrial                ; lecture valeur courante
        mov     al,cl                   ; on ajoute 1 (pour afficher 1 ou 2)
        inc     al
        mov     cx,offset P_STOP        ; saut dans routine d'affichage du volume
        jmp     dspvo_1
        ;------------


                                        ; ROUTINE DE GESTION DE L'INT 24H
nocrd_0: mov    al,3                    ;  si activé, on retourne FAIL
        iret                            ; retour à la routine DOS ayant détecté
        ;---                            ;   l'erreur
```

```
;*************************************************
; GETe309 : décode les informations du port E309
; entrées : aucune
; sorties : AL = Volume
;           BX = Contraste
; Note    : AH perdu
; Note    : REGISTRES CX, DX NON CONSERVES
;*************************************************
getE309  PROC   NEAR
        mov     dx,0E309h               ; Lecture du port
        in      al,dx                   ; (AL = CCCCVVxx)
        mov     cl,2                    ; cl = valeur de décalage
        shr     al,cl                   ; AL = 00CCCCVV
        mov     bl,al                   ; BL = 00CCCCVV
        and     al,00000011b            ; AL = 000000VV
        shr     bl,cl                   ; BL = 0000CCCC
        xor     bh,bh
        ret
getE309  ENDP


;*************************************************
; GETSRIAL : donne la configuration de la RS232
; entrées : aucune
; sorties : AX = parité  : 0 =  None
;                          1 =  Impaire (ODD)
;                          3 =  Paire (EVEN)
;           BX = Vitesse : 0 =    300 bauds
;                          1 =   1200 bauds
;                          2 =   2400 bauds
;                          3 =   4800 bauds
;                          4 =   9600 bauds
;                          5 =  19200 bauds
;                          6 =  57600 bauds
;                          7 = 115200 bauds
;           CX = bits stops : 0 = 1 bit stop
;                             1 = 2 bits stops
;           DX = Données : 0 =   5 bits données
;                          1 =   6 bits données
;                          2 =   7 bits données
;                          3 =   8 bits données
;*************************************************
getsrial PROC   NEAR
        mov     dx,03FBh                ; lecture du port de configuration
        push    dx
        in      al,dx
        push    ax                      ; on sauve la configuration courantes

        mov     cl,al                   ; sauvegarde config dans CX
        or      al,10000000b            ; on arme le bit DLAB
        out     dx,al

        mov     dl,0F8h                 ; lecture du diviseur courant
        in      ax,dx
        xor     bx,bx                   ; on initialise l'index à 0
getsr_1: cmp    ds:[bx+offset n_BAUD],ax ; diviseur courant = entrée table ?
        jbe     getsr_2                 ; oui, on a trouvé le code
        inc     bx                      ; non, on passe à l'entrée courante
        inc     bx
        jmp     getsr_1                 ; on boucle...
        ;-------------
getsr_2: shr    bx,1                    ; conversion index en code

        pop     ax                      ; restauration contenu du port 3fb
        pop     dx
        out     dx,al
```

```
        mov     dl,cl                   ; nombre de bits de données dans DX
        and     dx,00000011b

        shr     cl,1                    ; nombre de bits stop dans CX
        shr     cl,1
        mov     al,cl
        and     cx,00000001b

        shr     al,1                    ; parité dans AX
        and     ax,00000011b
        ret
getsrial ENDP


;*****************************************************
; SETSRIAL : configure la RS232
; entrées : AX, BX, CX, DX = voir sorties GETSRIAL
; sorties : aucune
; Note    : REGISTRES AX, BX, CX, DX NON CONSERVES
;*****************************************************
setsrial PROC   NEAR
        push    dx
        push    ax

        mov     al,80h                  ; on arme le bit DLAB du port 3FB
        mov     dx,03fbh                ;  avant d'ecrire la vitesse en 3f8
        push    dx
        out     dx,al

        shl     bx,1                    ; ecriture du diviseur en 3f8
        mov     ax,ds:[bx+offset n_BAUD]
        mov     dl,0F8h
        out     dx,ax

        pop     dx                      ; on récupère 03FB dans dx
        pop     ax                      ; on récupère le AX initial
        shl     al,1                    ; AL = 00000PP0
        or      al,cl                   ; AL = 00000PPS
        shl     al,1                    ; AL = 0000PPS0
        shl     al,1                    ; AL = 000PPS00
        pop     cx                      ; on récupère le DX initial dans CX
        or      al,cl                   ; AL = 000PPSLL
        out     dx,al                   ; ecriture du nouveau mot de controle

        ret
setsrial ENDP


;*****************************************************
; DISKSPCE  : renvoie les capacités totale et disponible
;           d'un disque
; entrées : DL = numéro du disque (A=1, B=2, C=3..)
; sorties : AX = capacité totale (en ko)
;           BX = Capacité disponible (en ko)
;           Si carry armé : erreur -> données non valides
; Note : on assume que les secteurs font 512 octets
;      : capacités limitées à 32Mo
; Note    : REGISTRES CX, DX NON CONSERVES
;*****************************************************
diskspce PROC   NEAR
        mov     ah,36h                  ; appel de la fonction DOS de lecture
        int     21h                     ;   des capacités du disque
        cmp     ax,0FFFFh               ; si ax = FFFF -> erreur
        je      disks_3
                                        ; Rapel : La fonction 36h retourne :
                                        ;  AX = nbre de secteurs/cluster
                                        ;  BX = Capacité dispo (en clusters)
                                        ;  CX = Taille de secteurs en octets
                                        ;  DX = capacité totale du disque

                                        ; AX (clusters/sect) ne contient que
                                        ;  1, 2, 4, 8.. (1b, 10b, 100b, 1000b...)
                                        ;  on va en profiter...
disks_1: shr    ax,1                    ; dernier bit transféré dans Carry
        jc      disks_2                 ; si carry, plus de multiplications
        shl     bx,1                    ; on multiplie capacités par deux
        shl     dx,1
        jmp     disks_1                 ; et on boucle
        ;-------------

disks_2: shr    bx,1                    ; on divise les capacités par deux
        shr     dx,1                    ;  et on obtient directement des Ko
        mov     ax,dx
        clc                             ; pas d'erreur -> efface Carry
        jmp     disks_4
        ;-------------

disks_3: stc                            ; si erreur -> on arme le Carry
disks_4: ret
diskspce ENDP


;*****************************************************
; DISPLEFT : affiche un nombre entier long, justifié à gauche
;           (maxi 6 digits). Place des blancs en fin de
;           chaine
; entrées : DX:AX = nombre à afficher
;           BX = nombre de caractère de la chaine
;           CH,CL = position sur l'écran
; sortie  : Aucune
; Note    : Assume flag D à 0
; Note    : REGISTRES AX, BX, CD, DX, SI NON CONSERVES
;*****************************************************
displeft PROC   NEAR
        push    bx                      ; on sauve nombre de caractères désiré

        push    dx                      ; on sauve le mot de poids fort
        mov     dx,cx                   ; on positionne le curseur
        call    gotoxy
        pop     dx                      ; on récupère le mot de poids fort

        mov     bx,6
        mov     si,offset s_tmp         ; conversion en ascii dans s_tmp
        call    ltoa                    ;   (cadré à droite)

displ_1: lodsb                          ; lecture du caractère en DS:SI (+inc si)
        cmp     al," "                  ;   si espace, on boucle
        je      displ_1

        dec     si                      ; on annule le dernier décalage
        mov     dx,si
        call    printf                  ; affichage chaine
        call    strlen                  ; calcul longueur utile de la chaine

        pop     dx                      ; récupération nbre de caractères désiré
        sub     dx,cx                   ; cx = cx - longueur chaine
        mov     cx,dx
        jz      displ_3                 ; si résultat = 0, pas d'espaces en plus

        mov     ah,02h                  ; affichage de 'CX' caractère espace
        xor     bx,bx
```

```
        mov     dl," "
displ_2: int    21h
        loop    displ_2

displ_3:
        ret
displeft ENDP

;*****************************************************
; DISPON  : affiche ON ou OFF suivant le contenu de AL
; entrées : AL = 0 ou 1
;           DH,DL = position sur l'écran
; sortie  : Aucune
; Note    : REGISTRES DX NON CONSERVE
; NOTE    : LE CODE CONTINUE DANS PRINTF
;*****************************************************
s_ON    DB      "ON ",0
s_OFF   DB      "OFF",0

dispon  PROC    NEAR
        call    gotoxy                  ; on positionne le curseur
        mov     dx,offset s_ON          ; initialise dx sur "ON"
        cmp     al,1                    ; si al différent de 1..
        je      dispo_1
        mov     dx,offset s_OFF         ; ..on initialise dx sur "OFF"
dispo_1:                                ; le code continue fonction suivante !
dispon  ENDP


;*****************************************************
; PRINTF  : affiche une chaine de caractères dans
;           le flux standard stdout
; entrées : DS:DX = Chaine
; sortie  : Aucune
;*****************************************************
printf  PROC    NEAR
        push    ax
        push    bx
        push    cx
        call    strlen                  ; calcul de la longueur de la chaine
        mov     ah,40h                  ; affichage
        mov     bx,1
        int     21h
        pop     cx
        pop     bx
        pop     ax
        ret
printf  ENDP


;*****************************************************
; DISPBAT  : affiche la tension des batteries
; entrées : AL = baterie (main=0, backup=1, carte=2)
;           DX = position de la chaine à l'écran
; sortie  : Aucune
; Note    : REGISTRES AX, DX NON CONSERVES
; NOTE    : LE CODE CONTINUE DANS PUTCH
;*****************************************************
dispbat PROC    NEAR
        call    gotoxy                  ; positionnement du curseur

        mov     ah,50h                  ; lecture de la tension de la batterie
        int     15h
        xor     ah,ah                   ; conversion en dixièmes de volts
        mov     dl,5
        div     dl
        dec     ax
        cmp     al,19
        jnb     dispb_1
        inc     ah
        cmp     al,14
        jnb     dispb_1
        inc     ah
dispb_1: aam                            ; conversion en 2 codes ASCII
        add     ax,3030h
        mov     cx,ax
        mov     dl,ch
        call    putch                   ; affichage des volts
                                        ; note : on profite ici que les registres
                                        ;  ne sont pas restitués par putch
        mov     dl,"."                  ; affichage du point décimal
        int     21h
        mov     dl,cl                   ; affichage des dixièmes de volts
        int     21h
        mov     dl,"V"                  ; affichage du "V" final
        jmp     putch
dispbat ENDP


;*****************************************************
; DSPSPACE : affiche une capacité (mémoire ou disque)
;           justifiée à droite, suivie de la lettre 'K'
; entrées : AX = nombre à afficher
;           DH,DL = position sur l'écran
; sortie  : Aucune
; Note    : REGISTRES AX, BX, CX, DX, SI  NON CONSERVES
; NOTE    : LE CODE CONTINUE DANS PUTCH
;*****************************************************
dspspace PROC   NEAR
        call    gotoxy                  ; positionnement du curseur
        mov     si,offset s_tmp         ; si pointe sur chaine de stockage
        mov     bx,6                    ; conversion AX en chaine, dans s_tmp
        xor     dx,dx
        call    ltoa
        mov     dx,si                   ; affichage chaine
        call    printf
        mov     dl,"K"                  ; affichage du "K" final
dspspace ENDP                           ; on continue dans la fonction suivante !

;*****************************************************
; PUTCH   : affiche un caractère
; entrées : DL = Code du caractère
; sortie  : Aucune
; Note    : REGISTRES AX, BX  NON CONSERVES
;           (possibilité utilisée dans dispbat !)
;*****************************************************
putch   PROC    NEAR
        mov     ah,02h
        xor     bx,bx
        int     21h
        ret
putch   ENDP
```

```
;*****************************************************
; GOTOXY  : positionnement du curseur
; entrées : DH = ligne
;           DL = Colonne
; sortie  : Aucune
;*****************************************************
gotoxy   PROC   NEAR
         push   ax
         push   bx
         push   dx

         mov    ah,02h
         xor    bx,bx
         int    10h

         pop    dx
         pop    bx
         pop    ax
         ret
gotoxy   ENDP


;*****************************************************
; STRLEN  : calcule la longueur d'une chaine
; entrées : DS:DX adresse de la chaine
; retour  : CX = longueur
;*****************************************************
strlen   PROC   NEAR
         push   bx

         xor    cx,cx
         mov    bx,dx
strle_1: cmp    byte ptr [bx],0        ; fin de la chaine ?
         je     strle_2                ; oui -> fin de routine
         inc    cx                     ; incrémentation taille
         inc    bx                     ; on pointe sur caractère suivant
         jmp    strle_1                ; on boucle...
         ;-------------

strle_2: pop    bx
         ret
strlen   ENDP


;=================================================
; LTOA : Conversion d'un entier long en chaine
;        justifiée à droite
;        Pas de code de fin de chaine ('\0')
; Entrées: DX:AX = nombre à convertir
;          BX = nombre de digits total
;          DS:SI = Adresse du début de chaine
;=================================================
ltoa     PROC   NEAR
         push   ax
         push   bx
         push   cx
         push   dx
         push   di
         push   si
         push   bp

         mov    di,dx                  ; on sauve DX:AX dans DI:BP
         mov    bp,ax

         add    si,bx                  ; calcul position du dernier caractère
         dec    si                     ;  de la chaine (digit poids faible)

         mov    cx,10                  ; diviseur (base)

ltoa_1:  mov    ax,di                  ; on récupère le mot de poids fort
         xor    dx,dx                  ; la division ne se fait que sur AX
         div    cx                     ; division, le reste est en DX
         mov    di,ax                  ; on sauve le quotient

         mov    ax,bp                  ; mot de poids faible
         div    cx                     ; division DX:AX/CX, le digit est en DX
         mov    bp,ax                  ; sauve quotient

         add    dl,30h                 ; conversion du digit en ASCII
         mov    [si],dl                ; écriture dans chaine
         dec    si
         dec    bx
         jz     ltoa_3                 ; on arrete si nbre de caractères atteint

         mov    ax,bp                  ; si BP ou DI differents de 0..
         or     ax,di
         jnz    ltoa_1                 ; ..on continue le décodage

         mov    cx,bx                  ; on remplit les digits non utilisés
ltoa_2:  mov    byte ptr [si]," "      ; par des espaces
         dec    si
         loop   ltoa_2

ltoa_3:  pop    bp
         pop    si
         pop    di
         pop    dx
         pop    cx
         pop    bx
         pop    ax
         ret
ltoa     ENDP

Code     ends
         END    start                  ; That's All Folks !
```

LE COIN DES CODES

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

Note importante :

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

Programmes HP48 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 les deux programmes correspondant à votre machine:
- pour HP-48 : ASSCOD48 et INPUT48
- pour HP-28S : ASSCOD28 et INPUT28

(*attention aux parties communes et spécifiques HP-48 ou HP-28S et aux commentaires*)
Ceci avec la plus grande attention car leur 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 [.].

ASSCOD48 996.5 octets
cksum # 5D1Bh

```
« RCLF HEX 64 STWS 1 SF "" 'tmpcod' STO
"nombre d'octets                          @ + NEWLINE
                                          @ NEWLINE
" 17 INPUT48 1 CF STR→ 2 * 16             @ INPUT28
DUP2 / IP 3 ROLLD MOD DUP2 0 > +          @ si HP-28S
3 ROLLD 1 + 4 ROLL # 0h DUP → n r f s o « 1 SWAP
FOR i
 DO "ligne " i 1 - R→B # 1000h + →STR 4 6 SUB + DUP
  "                                       @ NEWLINE
  chaine                                  @ + NEWLINE
  " + "---- ---- ---- ----" n i <
  IF THEN r DUP 4 / IP + SWAP OVER 1 SWAP OVER -
       SUB SWAP 18 + ELSE 38 END 'o' STO +
  "                                       @ NEWLINE
  " + 1 FS?C IF THEN ROT SWAP CLLCD 1 DISP HALT
             ELSE o INPUT48 END DUP       @ INPUT28
  WHILE DUP " " POS DUP                   @ si HP-28S
  REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD
   1 + 25 SUB IF DUP " " == THEN DROP ELSE + END
  END DROP 0 OVER SIZE 1 SWAP
  FOR j OVER j DUP SUB NUM j * + NEXT
  s + DUP # FFFh AND
  "                                       @ NEWLINE
  somme de controle                       @ + NEWLINE
   ---                                    @ + NEWLINE
  #" 6 ROLL SWAP + 34 INPUT48 STR→ ==     @ INPUT28
  IF THEN SWAP DROP 1                     @ si HP-28S
  ELSE DROP2 1000 1 BEEP 1 SF 0 END
 UNTIL END 's' STO
 WHILE DUP " " POS DUP
 REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 + 19 SUB +
 END DROP 'tmpcod' DUP RCL ROT + SWAP STO
NEXT f STOF » tmpcod
@ * HP-48 * @@ ********** HP-28S **********
"GROB 8 " @@ # 20204A04F3D02C67h # F80004F02C96040Ch
OVER SIZE @@ # 8DCC05081F804F27h # 313103190F818341h
2 / " "   @@ # 681808AE91B51391h # 45DC061171085168h
+ + SWAP  @@ # F3D3CEAA125E5D8Eh # 2F9004h 28 STWS
+ STR→    @@ #0 OR 64 STWS 1 7 START # 3B82h SYSEVAL
# 4017h   @@ NEXT # 20238h SYSEVAL # 4F3Dh SYSEVAL
SYSEVAL   @@ EVAL
# 56B6h   @@ toutes ces lignes marquées @@ peuvent
SYSEVAL   @@ être remplacées par ASC→ :
DROP      @@ soit pour HP-48 (JPC-79 page 11)
NEWOB     @@ ou pour HP-28S (JPC-85 page 8)
"fin" CLLCD 1 DISP »                      @ HP-28S dito
```

INPUT48 412.5 octets
cksum # 15C7h

INPUT28 228 octets
cksum # 9412h

```
« SWAP 1                                  @ partie commune
WHILE OVER CLLCD 1 DISP                   @ HP-48 et HP-28
REPEAT                                    @ puis séquence
 DO UNTIL KEY END                         @ HP-48 ou HP-28
@ ********** HP-48 ********** @@ ****** HP-28S ******
 IF DUP 51 == THEN DROP 0       @@ IF DUP "ENTER" ==
 ELSE IF DUP 55 == THEN DROP    @@ THEN DROP 0
   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  @@ suite voir HP-48
           92 == THEN 48 END    @@
                 46 END CHR     @@
  + DUP SIZE 3 PICK - 2 + 5 MOD NOT @ partie
  1 FC? AND IF THEN " " + END END 1 @ commune
END END SWAP 60 SUB »                     @ HP-48 et HP-28
```

Donc à partir de maintenant 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, correspondant au numéro de ligne à 3 chiffres, sans les espaces et validez.
4- tapez la somme de contrôle et validez. S'il y a erreur la ligne de codes sera demandée à nouveau après émission d'un BEEP. L'appui sur EDIT fera apparaître la ligne des codes qui pourra être corrigée. Alors relancez avec CONT.
5- après l'affichage de "fin" stockez le programme assemblé dans la variable donnée en tête.
6- si tout s'est bien déroulé vous pouvez purger tmpcod qui contient la chaîne de codes.

Programmes MS-DOS

Afin d'être utilisé par tous, ce programme est destiné à être écrit en GWBASIC. Il devrait cependant être facile de le convertir pour un autre programme (QBASIC, Turbo BASIC...).

Mode d'emploi :

1- Lancer le programme : GWBASIC MAKEDOS.BAS
2- Entrer le nom du fichier destination.
3- Entrer la taille du fichier.
4- Entrer les listes de codes puis le checksum (en prenant soin d'entrer les codes héxadécimaux en majuscules). En cas d'erreur corriger la ligne, en prenant soin de placer le curseur après le dernier caractère avant de taper sur la touche d'entrée.
5- Une fois que toutes les lignes sont entrées, sortir du GWBASIC en exécutant la commande SYSTEM. Le nouveau programme est immédiatement disponible.

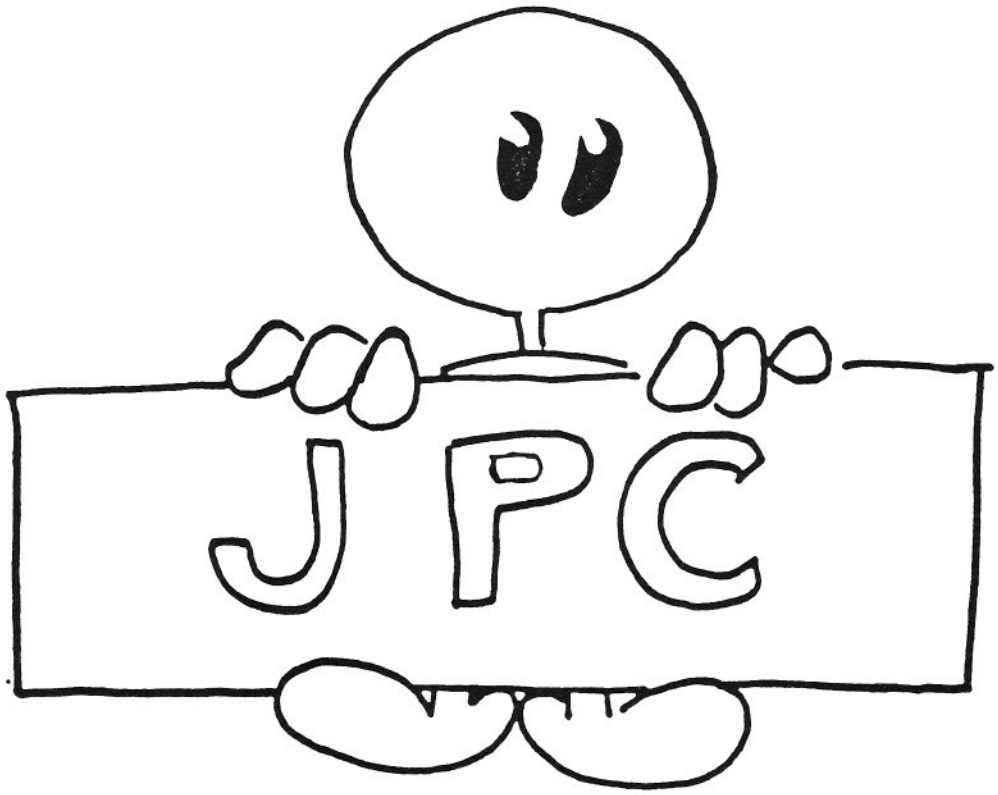
Note : La taille du fichier résultant peut être supérieure d'un octet à ce qui est affiché dans le listing. Cela n'est pas un problème.

Programme MAKEDOS.BAS

```
 10 INPUT "Nom du fichier : ",NOM$ : INPUT "Nombre d'octets : ",N : N=N*2
 20 OPEN NOM$ FOR OUTPUT AS #1 : CLOSE #1 : KILL NOM$
 30 OPEN "bin.tmp" FOR OUTPUT AS #1 : S=0 : P$="---- ---- ---- ----"
 40 NLINES=N\16 : LENLAST=(N MOD 16)+((N MOD 16)\5)
 50 IF (N MOD 16)=0 THEN NLINES=NLINES-1 : LENLAST=LEN(P$)
 60 FOR X=0 TO NLINES
 70   IF X=NLINES THEN P$=LEFT$(P$, LENLAST)
 80   C$=P$
 90   X2$="00"+HEX$(X) : PRINT RIGHT$(X2$,3);":";
100   Y=CSRLIN : LOCATE Y,6 : PRINT C$; : LOCATE Y,6 : INPUT "",C$ : IF Y=24 THEN Y=Y-1
110   LOCATE Y,27 : PRINT " sm = ---" : LOCATE Y,33 : INPUT "",D$
120   M=S
130   FOR Z=1 TO LEN(C$)
140     IF MID$(C$,Z,1)<>" " THEN M=(M+((Z-(Z\5))*ASC(MID$(C$,Z,1)))) MOD 4096
150   NEXT Z
160   D2$="00"+HEX$(M) : D2$=RIGHT$(D2$,3)
170   IF D2$<>D$ THEN PRINT "Erreur de somme" : BEEP : GOTO 90
180   FOR Z=1 TO LEN(C$) STEP 2
190     IF MID$(C$,Z,1)=" " THEN Z=Z-1 : GOTO 230
200     CH=ASC(MID$(C$,Z,1))-48 : IF CH>9 THEN CH=CH-7
210     CL=ASC(MID$(C$,Z+1,1))-48 : IF CL>9 THEN CL=CL-7
220     PRINT#1,CHR$((16*CH)+CL);
230   NEXT Z
240   S=M
250 NEXT X
260 CLOSE #1 : NAME "bin.tmp" AS NOM$ : END
```

```
SPEED                          (HP28)
# A389h                    17 octets

     0123 4567 89AB CDEF    sm

000: 69C2 0D10 0013 2340    B07
001: 0FFF 1341 5428 D108    7E5
002: 30                     878


95PANEL.COM                 (MS-DOS)
                        1535 octets

     0123 4567 89AB CDEF    sm

000: E9D4 0154 6869 7320    C13
001: 7072 6F67 7261 6D20    8E0
002: 7275 6E73 206F 6E6C    7D3
003: 7920 6F6E 2061 6E20    4D8
004: 4850 3935 4C58 0D0A    29E
005: 000D 0920 2020 2020    D12
006: 2020 4850 3935 2050    838
007: 616E 656C 0D0A 0A56    651
008: 6F6C 756D 6520 2020    222
009: 2020 2020 3A09 0949    E3D
00A: 6F09 3A0D 0A43 6F6E    D4E
00B: 7472 6173 7420 2020    81C
00C: 2020 3A09 0950 6F72    4FD
00D: 7409 3A0D 0A0A 4C69    363
00E: 6768 7420 736C 6565    033
00F: 7020 203A 0909 4261    BE7
010: 7564 093A 0D0A 5469    964
011: 6D65 206F 7574 2020    542
012: 2020 203A 0920 206D    195
013: 6E09 6269 7453 093A    E82
014: 0D0A 0909 0970 6152    A5A
015: 6974 7909 3A0D 0A63    7E5
016: 5572 736F 7220 7472    425
017: 6163 6B20 3A09 0973    0AB
018: 744F 7073 093A 0D0A    E79
019: 416C 7420 6172 726F    B7C
01A: 7773 2020 203A 0D0A    871
01B: C4C4 C4C4 C4C4 C4C4    7D1
01C: C4C4 C4C4 C4C4 C4C4    731
01D: C4C4 C4C4 C4C4 C4C4    691
01E: C4C4 C4C4 C4C4 C4C4    5F1
01F: C4C4 C4C4 C4C4 C4C4    551
020: 0D0A 0920 2020 2020    FE1
021: 546F 7461 6C20 2020    B87
022: 4672 6565 2020 2042    660
023: 6174 7465 7269 6573    2C8
024: 0D0A 4D65 6D09 2020    F34
025: 3A09 0909 284D 6169    C3F
026: 6E29 0D0A 4469 736B    A12
027: 2043 0920 203A 0909    5FD
028: 0928 4261 636B 7570    27F
029: 290D 0A43 6172 6420    E46
02A: 2020 2020 203A 204E    A8B
02B: 6F74 2049 6E73 7461    746
02C: 6C6C 6564 0049 5220    2E9
02D: 2020 0052 5332 3332    DC6
02E: 0080 0160 0030 0018    83E
02F: 000C 0006 0002 0001    262
030: 004E 4F4E 4500 4F44    FD8
031: 4420 0020 2020 2000    9BA
032: 4556 454E 0028 524F    6D4
033: 2900 2020 2020 2020    0E0
034: 0076 5663 436C 7454    D92
035: 7561 6970 6242 7353    94C
036: 7252 6F14 041B 043B    647
037: 0442 0459 0468 0471    1E1
038: 048B 049C 04AF 04C6    058
039: 04EA 04F0 0419 051F    D69
03A: 0536 0540 0563 05FC    AA6
03B: B824 25BA 7905 CD21    89E
03C: B44A BB80 00CD 21BC    7AD
03D: 0008 B840 008E C0B8    5CC
03E: D44D CD15 81FB 5048    3E3
03F: 7506 81F9 0101 7409    FD5
040: BA03 01E8 3F03 E907    D88
041: 0126 8026 1700 BFB8    B89
042: 0700 CD10 BA29 01E8    98C
043: 2B03 E80E 01E8 2F01    72E
044: E83C 01E8 5201 E868    4A9
045: 01E8 7601 E887 01E8    25B
046: 9B01 E8C5 01E8 F101    FDF
047: E813 02E8 2E02 26A1    CEB
048: 1300 BA0B 0CE8 4003    A2B
049: BBFF FFB4 48CD 218B    A00
04A: C3B1 06D3 E840 40BA    826
04B: 120C E82B 03B0 00BA    630
04C: 1B0C E8F1 02B0 01BA    462
04D: 1B0D E8E9 02B2 03E8    2A2
04E: 6F02 53BA 0B0D E80F    212
04F: 0358 BA12 0DE8 0803    F4D
050: BA04 E3EC A840 750E    DBD
051: BA00 0EE8 1603 BA4B    C5F
052: 02E8 B102 EB28 BA05    AC1
053: 0EE8 0803 BA8B 02A8    929
054: 1075 03BA 9502 E89C    7BE
055: 02B2 01E8 3302 72D8    4CD
056: 53BA 0B0E E8D1 0258    2C2
057: BA12 0EE8 CA02 BA00    117
058: 0FE8 E002 B411 CD16    EFA
059: 751E 268B 1E1A 004B    CE6
05A: 4B83 FB1E 7303 BB3C    BFF
05B: 0026 813F 00A0 75E4    967
05C: 26C7 0700 00E9 56FF    806
05D: B410 CD16 3C00 7412    44C
05E: 3C0D 741C BEA1 0238    223
05F: 0474 0946 81FE B202    FB8
060: 76F5 EBBA 81EE A102    EBC
061: D1E6 FF94 B302 EBAE    F2E
062: B44C CD21 E865 01FE    E40
063: C0EB 05E8 5E01 FEC8    E37
064: 2403 B448 CD15 B207    BCB
065: E872 02E8 4E01 B90F    A36
066: 0232 E4BB 0200 33D2    70B
067: E9C4 01E8 3E01 FEC3    65E
068: EB05 E837 01FE CB83    5A4
069: E30F B800 47CD 15E8    45A
06A: 2A01 8BC3 B90F 03EB    3BC
06B: D826 8036 D600 0126    F67
06C: A0D6 00BA 0F05 EB41    DC3
06D: 2681 06A3 0044 04EB    B0F
06E: 0726 812E A300 4404    702
06F: 26A1 A300 0514 00B9    339
070: 3C00 F7E1 8BC2 B90F    2C2
071: 06EB A626 8036 9300    EDC
072: 0226 A093 00D0 E8BA    DA5
073: 0F08 EB0D 2680 3693    ABE
074: 0001 26A0 9300 BA0F    88E
075: 0924 0134 01EB 14BA    69D
076: 01E3 EC34 04EE BA01    56D
077: E3EC D0E8 D0E8 2401    327
078: BA22 02E9 7201 BA0A    117
079: E3EC 3404 EEBA 0AE3    0FE
07A: ECD0 E8D0 E824 01BA    F99
07B: 2203 E8C7 01BA 6502    CD1
07C: 3C01 7403 BA6B 02E9    B1E
07D: 5B01 E8A0 0043 EB04    878
07E: E89A 004B 83E3 07E8    6B6
07F: C600 E890 00D0 E38B    4DC
080: 8F71 02BA 0100 B800    11F
081: C283 F901 7402 F7F1    EDF
082: B922 05BB 0600 E9E6    D23
083: 00E8 7100 42EB 04E8    B15
084: 6B00 4A80 E203 E897    8DE
085: 00E8 6100 8AC2 0405    544
086: B922 06E9 FBFE E854    569
087: 0040 3C02 74FB EB09    454
088: E84A 0048 3D02 0074    049
089: FA24 03E8 7200 E83C    E2D
08A: 00B1 05F6 E105 8102    A68
08B: BA22 07E8 4601 8BD0    823
08C: E9E2 00E8 2700 80F1    4FE
08D: 01E8 5400 E81E 008A    2AC
08E: C1FE C0B9 2208 E9B8    19F
08F: FEB0 03CF BA09 E3EC    23A
090: B102 D2E8 8AD8 2403    FC9
091: D2EB 32FF C3BA FB03    FF2
092: 52EC 508A C80C 80EE    FB1
093: B2F8 ED33 DB39 8771    DEF
094: 0276 0443 43EB F6D1    C9B
095: EB58 5AEE 8AD1 83E2    C04
096: 03D0 E9D0 E98A C183    AD9
097: E101 D0E8 2503 00C3    760
098: 5250 B080 BAFB 0352    535
099: EED1 E38B 8771 02B2    2A8
09A: F8EF 5A58 D0E0 0AC1    17D
09B: D0E0 D0E0 590A C1EE    12A
09C: C3B4 36CD 213D FFFF    28A
09D: 7413 D1E8 7206 D1E3    07D
09E: D1E2 EBF6 D1EB D1EA    1D3
```

```
09F: 8BC2 F8EB 01F9 C353   085
0A0: 528B D1E8 9E00 5ABB   02B
0A1: 0600 BE9A 02E8 B100   DA9
0A2: AC3C 2074 FB4E 8BD6   DA2
0A3: E82A 00E8 9300 5A2B   B40
0A4: D18B CA74 0AB4 0233   865
0A5: DBB2 20CD 21E2 FCC3   81A
0A6: 4F4E 2000 4F46 4600   430
0A7: E869 00BA 3006 3C01   0B5
0A8: 7403 BA34 0650 5351   C7E
0A9: E866 00B4 40BB 0100   8BB
0AA: CD21 595B 58C3 E84B   7D1
0AB: 00B4 50CD 1532 E4B2   5C0
0AC: 05F6 F248 3C13 7308   287
0AD: FEC4 3C0E 7302 FEC4   1CF
0AE: D40A 0530 308B C88A   01C
0AF: D5E8 2100 B22E CD21   E2E
0B0: 8AD1 CD21 B256 EB15   CA6
0B1: E819 00BE 9A02 BB06   AF1
0B2: 0033 D2E8 2B00 8BD6   97F
0B3: E8AA FFB2 4BB4 0233   762
0B4: DBCD 21C3 5053 52B4   483
0B5: 0233 DBCD 105A 5B58   325
0B6: C353 33C9 8BDA 803F   260
0B7: 0074 0441 43EB F75B   12D
0B8: C350 5351 5257 5655   D27
0B9: 8BFA 8BE8 03F3 4EB9   CEC
0BA: 0A00 8BC7 33D2 F7F1   BE6
0BB: 8BF8 8BC5 F7F1 8BE8   C5E
0BC: 80C2 3088 144E 4B74   9E0
0BD: 0E8B C50B C775 E38B   944
0BE: CBC6 0420 4EE2 FA5D   8FE
0BF: 5E5F 5A59 5B58 C3     0C7
```



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 EXchange), 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.

Les informations et programmes parus dans ce journal sont publiées "Tels quels" et ne peuvent en aucun cas engager la responsabilité de Hewlett-Packard ou de PPC Paris. Hewlett-Packard se réserve le droit de ne pas répondre aux questions concernant le sujet de certains articles.

Les programmes publiés peuvent être utilisés librement. Cependant, ils ne peuvent être vendus ou fournis dans un ensemble commercialisé, sous quelque forme que ce soit, sans l'accord écrit de l'auteur ou de PPC Paris.

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