JPC

SEPTEMBRE 1994

NUMERO 99

Le numéro 40 F

A PROPOS DU CLUB

Le Bureau	Editorial	*1*
J.Belin	Attention Danger !	*2*
	Courrier du Coeur	*2*

HP28

P. Jebeily	Fougères et Graphisme	*4*

HP48

Guy Toublanc	La HP48 en Liberté	*8*
T. Bestel	Séries de Fourier	*9*
G. Mimoun	Des chiffres et des lettres	*11*
Guy Toublanc	Trucs et Musique	*12*
F. Barbaise	Binome de Newton	*14*
L.V. Huynh	Réduction de programmes (Acte III)	*15*

HP95 / HP100 / HP200

J. Belin	Nouvelles Cartes Mémoire	*18*
G. Toublanc	SOS pour Lotus	*18*
J. Belin	Animations sur HP95/100 (acte II)	*19*
	Le coin des codes	*31*

EDITORIAL

Oui, vous pouvez le dire, ce numéro est encore en retard ! Si, comme précédemment (malheureusement), une partie de ce retard était dûe à un manque d'articles, sa principale cause est que, alors même que nous procédions à sa mise en page, nous avons eu le déplaisir de voir notre ordinateur tomber en panne !

Rappelons à ceux qui n'ont pas lu les comptes rendus de notre dernière assemblée générale, que nous utilisions depuis une cinquantaine de numéros un ordinateur Hewlett-Packard compatible avec la série 9000 (fonctionnant sous Unix, et incompatible avec les IBM PC) et un logiciel d'impression "fait maison".

Or, afin de nous simplfier notre travail, et améliorer notre mise en pages, nous avions décidé au début de l'année de passer sur IBM PC et d'utiliser un logiciel plus évolué. Ce passage devant se faire pour le JPC numéro 100 (Et c'est ici, cher lecteur, que vous devez jeter un coup d'oeil au numéro de ce journal...).

Et oui, notre ordinateur n'a rien touvé de mieux que de tomber en panne trois ou quatre heures seulement avant la retraite ! Si certains fatalistes ne font que mentionner une certaine Loi de Murphy, d'autre prétendent que cette machine aurait une âme et qu'elle n'aurait rien trouvé de mieux pour nous faire sentir combien elle était indispensable... Nous avons pu identifier la cause de la panne (dans l'alimentation), mais le fait que le composant mis en cause soit spécifique nous empêcha de le remettre en état par nos propres moyens. Et le faire réparer par HP alors qu'il ne devait plus nous servir qu'une fois était très discutable financièrement...

Il nous a donc fallu trouver une autre solution, permettant de faire ce journal, tout en essayant de garder le style des JPC précédents.

Pourquoi cette décision de ne pas changer la mise en pages à partir de ce numéro ? Peut-être parce que nous aimons les chiffres ronds, mais surtout parce que nous pensions au début que cela nous ferait moins de travail...

Mais, même si nous avons passé cinq ou six fois plus de temps à refaire la mise en page de ce numéro, ce temps n'est pas totalement perdu puisque en avons profité pour tester de nombreux programmes de mise en pages. Chose que nous aurions fatalement dû faire pour le numéro 100. Et force est de constater que de très nombreux logiciels de PAO qualifiés d'évolués sont restés sur le tapis dès qu'il fallait, par exemple, importer les listings HP48... Il nous a donc fallu recourir à deux programmes différents (un pour le texte et un autre pour les cadres) pour parvenir à nos fins.

En fait, si le JPC que vous avez entre les mains semble ressembler trait pour trait aux précédents, un comparaison attentive devrait vous montrer qu'il existe de nombreuses différences, quelques fois infimes (quelqu'un sera-t-il capable de nous les donner toutes ? Nous en avons compté une dizaine...).

Mais maintenant, une fois que nous aurons enfin choisi le programme de PAO qui nous servira pour les prochains JPC (les 100 prochains ?), notre plus gros problème sera de combler notre retard. Si nous avons déjà défini une partie du n°100 (avec un compte-rendu de la conférence d'Amsterdam et une présentation de nouveaux logiciels, entre autres), il faudra que vous nous aidiez un peu. Comment ? Tout simplement en nous envoyant le plus d'articles possible ! A propos, avez vous remarqué que nous avons trois nouveaux auteurs dans ce JPC ? Ce serait sympathique qu'il y en ait quatre dans le prochain...

ATTENTION DANGER !

Vous trouverez autre part dans ce journal mention de la mise hors service de la HP28S de notre ami Paul Jebeily lors du changement de ses piles.

Sa mésaventure est survenue car, depuis quelque temps, est apparu un nouveau type de pile ayant quasiment les mêmes dimensions que les LR1 de 1.5V (présentes dans les HP18B, HP19B, HP-28 et HP-41, aussi connues sous la référence 'N'), mais délivrant une tension de 12V au lieu de 1.5 V !

Inutile de vous expliquer le résultat obtenu lorsque l'on alimente une machine avec 36 Volts, au lieu de 4.5...

Si cela peut rassurer Paul, j'ai appris quasiment en même temps que cette mésaventure était arrivée à une autre personne, à l'étranger...

Le seul conseil que nous pouvons donc vous donner est donc de faire très attention au type de pile que vous donne le vendeur.

On peut tout de même s'étonner du laisser-aller des fabriquants de piles, décidant de mettre sur le marché un produit présentant un tel risque de confusion, surtout si l'on sait que ces piles sont la plupart du temps destinées à être placées dans des appareils coûtant plusieurs milliers de Francs...

Jacques Belin (123)

COURRIER DU COEUR

Jean ARNAUD
72, rue de Rochechouard
75009 PARIS
Tel : (1) 48 78 55 59

Vend (pour collectionneurs !) :

Une calculatrice HP67, une HP97 (HP67 avec imprimante incorporée), nombreux programmes. Le tout 2000 Fr.

Carles CRESPO
Felip II, 291 4-4
08016 BARCELONE
Espagne

Vend :

HP95LX 512Ko, 1300 F (port compris pour l'Europe).

Paul Jebeily
40, rue des Graves
31170 TOURNEFEUILLE
Tel :

Cherche :

Une HP28S :
- Soit neuve, à concurence de 650F.
- Soit d'occasion (excellent état) à moins de 500F.

André Chollier
28, rue de Trévise
75009 PARIS
Tel : (1) 42 46 33 84

Cherche :

Un module finance pour HP-41.



HP28

P. Jebeily Fougères et Graphisme *4*

FOUGERE DE BARNSLEY
et graphisme

NDLR : Paul Jebeily nous a informé de la ruine de sa HP28S en utilisant des piles de 12 volts au lieu de 1,5. Il a eu la gentillesse de nous transmettre certains de ses travaux dont quelques uns devaient être revus pour utiliser les nouvelles fonctions graphiques parues dans JPC 98.

Fougère de Barnsley

FOUGR est l'adaptation pour HP28S du programme de tracé de fougère fractale et paru pour HP95 dans *JPC 93* page 27 (NDLR: ce programme était lui-même une adaptation en Swift!Basic d'un programme pour HP48 en User RPL, mais paru dans *48Sxtant* et dont Paul n'a pu avoir connaissance. Ainsi la boucle est bouclée).

Il suffit de fournir un nombre entier positif (par exemple 500, 1000) pour obtenir le tracé d'une fougère fractale de Barnsley et la chaîne de l'image que vous pourrez sauvegarder pour réaffichage ou impression.

Le programme FOUGR exécutant de nombreuses itérations, deux programmes sont appelés :

- le programme SPEED pour gagner en rapidité.
- le programme ECROFF pour éteindre l'affichage et économiser les piles au-delà de 500 itérations. Cet affichage devant être rétabli en fin d'exécution par le programme ECRON.

Pour SPEED voir *JPC 85* ou version équivalente. Pour ECROFF et ECRON voir JPC 90 ou versions équivalentes.

Le PPAR doit être :

```
{ ( 0,0) (255,127) constant 1 ( 0,0) }
```

FOUGR

```
«
  0 64 0 0 0 → n x y x1 y1 a
  «
    CLLCD SPEED 1 n
    FOR i i 500 ==
      IF
      THEN ECROFF
      END
      0 RDZ RAND 'a' STO
      IF a .03 <
      THEN 57.6 'x1' STO
           .27 y * 'y1' STO
      ELSE
        IF a .17 <
        THEN -.139 x * .263 y * +
             65.644 + 'x1' STO
             .246 x * .224 y * +
              8.424 - 'y1' STO
        ELSE
          IF a .3 <
          THEN .17 x * .215 y * -
               47.016 + 'x1' STO
               .222 x * .176 y * +
               20.896 + 'y1' STO
          ELSE
               .781 x * .034 y * +
               12.384 + 'x1' STO
               -.032 x * .739 y * +
               63.18 + 'y1' STO
          END
        END
      END
      x1 'x' STO y1 'y' STO
      y 128 x -
      RèC PIXEL
    NEXT
    440 .5 BEEP n 500 ≥
    IF
    THEN
         ECRON
    END
    LCD→
  »
»
```

Pour obtenir un affichage 2,5 fois plus rapide on peut utiliser le programme PIXON de Guy Toublanc.

....A vos claviers....

Applications de Line

MVREC permet de déplacer un rectangle sur l'écran en effaçant l'image qu'il recouvre de l'épaisseur de son trait.

Utiliser les touches des 4 flèches pour ce déplacement et [+] et [-] pour augmenter ou diminuer le pas d'incrémentation.

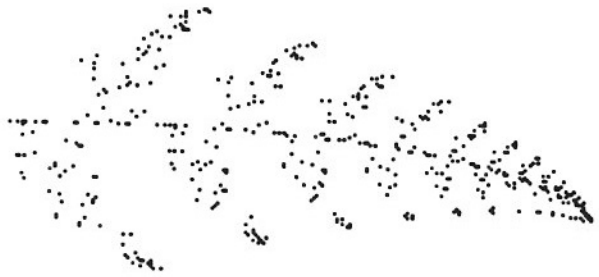
Ce programme ainsi que RECT, utilise le fameux programme LINE de Guy Toublanc (JPC 98).

RECT demande en argument deux nombres complexes correspondant aux coins diagonalement opposés d'un rectangle.

RECT

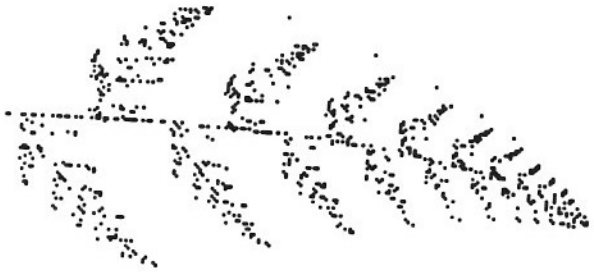
```
«
  DUP2 SWAP - RE  p1 p2 dx
```

```
  «
    p1 DUP dx + LINE
    p2 LINE
    DUP dx - LINE
    p1 LINE 5 DROPN
  »
»
```



MVREC

```
«
  1 0 0 0 0 → stp x1 y1 x2 y2
  «
    DO 1 2
      FOR i 9
        {
          « SF x1 y1 »
          « CF x2 y2 »
        }
        i GET EVAL R→C (55,15)
        OVER + (75,20) ROT + RECT
      NEXT
      x2 'x1' STO y2 'y1' STO
      {
        « stp 1 + 'stp' STO »
        « stp 1 - 'stp' STO »
        « stp x2 + 'x2' STO »
        « x2 stp - 'x2' STO »
        « stp y2 + 'y2' STO »
        « y2 stp - 'y2' STO »
      }
      { "+" "-" "RIGHT" "LEFT" "DOWN" "UP" }
      WAITK POS DUP 0 ≠
      IF
      THEN
        GET EVAL 0
      ELSE
        NOT SWAP DROP 440 .1 BEEP
      END
    UNTIL
    END
  »
»
```





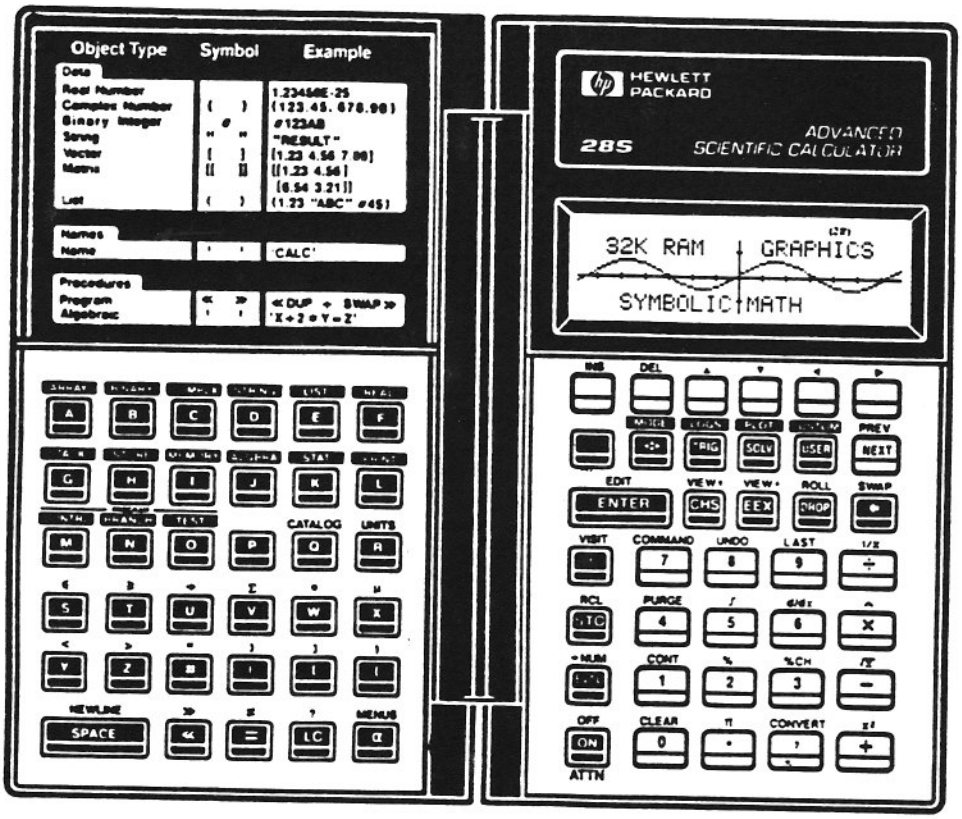
WAITK

```
« DO UNTIL KEY END »
```

Merci à Guy d'avoir créé de telles possibilités graphiques, en espérant que j'aurai la possibilité de les utiliser plus profondément (si je trouve une autre HP28S).



Paul Jebeily (571)


Object Type
Symbol
Example
Data
Real Number
Complex Number
Binary Integer
String
Vector
Matrix
List
Names
Name
'CALC'
Procedures
Program
Algebraic
HEWLETT PACKARD
28S
ADVANCED SCIENTIFIC CALCULATOR
32K RAM
GRAPHICS
SYMBOLIC
MATH
ENTER
SPACE
ATTN

HP48

G. Toublanc	La HP48 en Liberté	*8*
T. Bestel	Séries de Fourier	*8*
G. Mimoun	Des chiffres et des lettres	*11*
G. Toublanc	Trucs et Musique	*12*
F. Barbaise	Binome de Newton	*14*
L.V. Huynh	Réduction de programmes (Acte III)	*15*
	Le coin des codes	*31*

LITTERATURE HP48

La HP48 en liberté
Vagabondages à travers 48 applications

Voici un livre qui sort des sentiers battus et qui devrait inciter les possesseurs de HP48 à programmer plus souvent, soit pour résoudre de petits problèmes pratiques ou pour se faire plaisir. En effet l'auteur nous montre que le champ d'utilisation de nos machines n'est limité que par notre imagination. Les sujets abordés nous offrent un choix très diversifié de thèmes parfois inattendus. L'auteur expose l'objet des différents programmes et leur solution en se rendant accessible au lecteur.

Tous les programmes sont écrits en User Rpl et accompagnés d'exemples d'utilisation. Contrairement à ce que l'on rencontre habituellement, c'est l'environnement explicatif qui l'emporte sur la longueur des listings des programmes. Quelques références bibliographiques accompagnent souvent les textes des thèmes abordés. Les 48 applications sont regroupées en 8 chapîtres :

- Suites (40 pages)
- Graphismes (18 pages)
- Fractales (22 pages)
- Heure et dates (18 pages)
- Astronomie (24 pages)
- Trains (42 pages)
- Jeux (32 pages)
- Divers (46 pages)

Si cette énumération fait déjà ressortir une bonne diversité de sujets abordés, le contenu des chapîtres vous surprendra encore plus.

L'auteur n'est pas un inconnu des lecteurs de JPC car c'est Robert Pulluard. lui-même, qui dirige *48Sxtant*, journal entièrement consacré aux HP48 et dont les articles ont souvent été repris mais aussi revus pour les applications HP48 contenues dans ce livre.

Les lecteurs à court d'idées de programmation devraient trouver leur bonheur en plus de programmes prêts à l'emploi.

La HP48 en Liberté
par Robert Pullard
256 pages
145 F TTC
D3i Diffusion
ISBN 2-908791-16-1

Bonne lecture à travers ces vagabondages.

Guy Toublanc (276)

TRACE DE SERIES DE FOURIER

Ce programme simple en RPL n'a pas d'autre prétention que de visualiser ou d'imprimer des séries de Fourier dont les coefficients ont déjà été calculés. A l'allure des courbes, on saura si les coefficients ont une chance d'être corrects. Ecrit pour des besoins personnels lors de mes études en mathématiques, c'est après avoir lu l'article "*Réflexions...*" de Robert Amram et Jacques Belin (*JPC 97*) que je me suis décidé à vous faire parvenir ce programme. Le RPL est assez puissant pour résoudre tout un tas de petits problèmes, tel celui-ci que je soumets à vos utilisateurs débutants qui, comme moi, sont moins à l'aise avec la programmation en System-RPL ou en assembleur.

Les modèles G et GX de la HP48 permettent la création de masques de saisie bien pratiques pour améliorer l'ergonomie d'un programme (commande `INFORM`). C'est particulièrement utile en examen où on a d'autres soucis que de se rappeler comment tel ou tel module fonctionne... Les possesseurs de modèles S et SX adapteront le source en utilisant la commande `INPUT`. Ceci dit, `INPUT` dispose d'un mode de saisie directe en alphanumérique, mode qui ne peut pas être programmé avec la commande
`INFORM` (il faut tout d'abord appuyer sur la touche `[ALPHA]` - et encore j'ai dû utiliser un artifice pour ne pas avoir à entrer les "", et qui interdit l'utilisation de certains noms comme noms de variables indépendantes : utilisation de noms globaux, Cf. le source). Je regrette aussi l'absence de la possibilité de programmer des cases à cocher dans la commande `INFORM`... Cela est sans doute possible en System-RPL ou en assembleur...

Il s'agit donc de tracer des séries de Fourier, connaissant les coefficients. Je rappelle qu'une série de Fourier peut s'écrire *f(x)=a0+somme(a(n)*cos(nx)+b(n)*sin(nx))* pour *n* allant de 1 à l'infini, où : *a0* est la valeur moyenne de la fonction *f(x)*, *a(n)* et *b(n)* les coefficients de la série. Cette définition est très incomplète, mais suffisante pour établir un lien avec les notations du programme. Après avoir calculé ces coefficients (ce n'était pas le but du programme, je le répète), il suffit de remplir ou de valider les différents champs qui sont présentés, à savoir :
- Le nom de la variable indépendante (les lettres O, N, E, I, R, G et D vous sont interdites pour les raisons évoquées ci-dessus);
- Le nom de l'équation de départ (peut être simplement une chaîne de caractères pour l'impression);
- Si l'équation de départ doit être tracée avec la série;
- L'intervalle de tracé;
- La valeur moyenne et les coefficients de la série;
- Le mode angulaire;

- Une liste contenant la ou les valeurs de n pour le tracé de la série;
- Si les différentes valeurs de n doivent être tracées simultanément;
- Enfin si la sortie doit s'effectuer à l'écran ou sur l'imprimante HP.

Vous placerez ce programme dans un répertoire vide, de préférence. En effet, il crée de nombreuses variables qui servent de valeurs par défaut lors des utilisations suivantes et il serait bon de ne pas mélanger tous vos fichiers !

Quelques remarques sur le source :

- Lors de la première utilisation, vous devrez patienter quelques secondes de plus à la création des trois masques de saisie : le programme crée ses variables, si elles n'existent pas (utilisation de la structure de contrôle `IFERR`...);

- La structure de contrôle `IF THEN ... ELSE ... END` vous paraîtra quelque peu étonnante : sachez qu'on peut placer le `IF` n'importe où (en fait il ne fait rien et c'est `THEN` qui analyse le contenu du sommet de la pile de données. Habitué au langage Forth (c'est peut-être pour cela que je préfère la logique HP !), il me semble plus cohérent d'écrire cette structure ainsi.

- Sur l'imprimante, toute la largeur du papier est utilisée (pourquoi s'en priver ?).

Quelques remarques concernant l'utilisation du programme :

- Lorsqu'on demande un affichage successif pour plusieurs valeurs de *n*, l'environnement `PICTURE` devient actif à chaque valeur. On appuiera sur `[CANCEL]` (dans le menu) pour passer au tracé suivant.
- Pour entrer un choix alphanumérique, il suffit d'appuyer sur la touche alpha suivie du choix;
- Le mot `FOURIER` crée beaucoup de variables à la première utilisation et se retrouve ainsi en dernière position dans le menu (pour l'utilisateur). Il est plus pratique de l'avoir en tête de liste. Pour ceci, vous utiliserez la séquence suivante :

```
VARS OBJ→ SWAP OVER 1 + ROLLD →LIST ORDER
```

Pour exemple et pour terminer, voici quelques fonctions et leurs coefficients à donner au programme (les ai-je bien calculés ?) :

1) $f(x) = x^2$ pour $-\pi <= x <= \pi$:
   $a0 = \pi^2/3$
   $a(n) = 4*(-1)^n/n$
   $b(n) = 0$

2) $f(x) = 0$ pour $-\pi <= x <= 0$,
   $f(x) = x$ pour $0 <= x < \pi$:
   $a0 = \pi/4$
   $a(n) = -2/(\pi*n^2)$
   $b(n) = -(-1)^n/n$

3) $f(x) = \pi-x$ pour $0 <= x < \pi$,
   $f(x) = -\pi+x$ pour $-\pi <= x < 0$:
   $a0 = -\pi/2$,
   $a(n) = 2/(\pi*n^2)*(1-(-1)^n)$,
   $b(n) = 2/n*(1-(-1)^n)$.

Source du programme :

```
«
  RCLF 'flags' STO -28 SF
  { O N E I R G D } PURGE
  IFERR 'varind' RCL
  THEN
    'X' SWAP STO
  ELSE
    PURGE
  END
  IFERR 'trafx' RCL
  THEN
    N SWAP STO
  ELSE
    DROP
  END
  IFERR 'équdép' RCL
  THEN
    'X^2' SWAP STO
  ELSE
    DROP
  END
  IFERR 'débint' RCL
  THEN
    '-π' SWAP STO
  ELSE
    DROP
  END
  IFERR 'finint' RCL
  THEN
    'π' SWAP STO
  ELSE
    DROP
  END
  "Tracé d'une série de Fourier -1-"
  {
    { "f(x):" "Equation de départ" }
    { }
    { "x:" "Variable indépendante" }
    { "f(x)?:" "Tracer f(x)? (O ou N)" }
    { "a:" "Début de l'intervalle" }
    { }
    { "b:" "Fin de l'intervalle" }
    { }
```

```
}
{ 2 0 }
{ 'X^2' X N '-π' 'π' }
équdép varind trafx
débint finint 5
→LIST INFORM
IF
THEN
  OBJ→ DROP
  'finint' STO
  'débint' STO
  'trafx' STO
  'varind' STO
  'équdép' STO
  IFERR 'valmoy' RCL
  THEN
    'π^2/3' SWAP STO
  ELSE
    DROP
  END
  IFERR 'coefan' RCL
  THEN
    '4*(-1)^n/n' SWAP STO
  ELSE
    DROP
  END
  IFERR 'coefbn' RCL
  THEN
    0 SWAP STO
  ELSE
    DROP
  END
  R -17 FC?
  IF
  THEN DROP -18 FC?
    IF
    THEN
      D
    ELSE
      G
    END
  END
  'modang' STO
  "Tracé d'une série de Fourier -2-"
  {
    { "a0:" "Valeur moyenne de la série" }
    { "an:" "Coefficient an de la série" }
    { "bn:" "Coefficient bn de la série" }
    { "Mode angulaire:" "(R(ad), D(eg) ou G(rad)" }
  }
  { 1 0 }
  { 'π^2/3' '4*(-1)^n/n' 0 modang }
  valmoy coefan coefbn modang 4 →LIST
  INFORM
  IF
  THEN
    OBJ→ DROP
    'modang' STO
    'coefbn' STO
    'coefan' STO
    'valmoy' STO modang
    DUP R SAME
    IF
    THEN
      DROP RAD
    ELSE
      D SAME
      IF
      THEN
        DEG
      ELSE
        GRAD
      END
    END
    IFERR 'valn' RCL
    THEN
      3 SWAP STO
    ELSE
      DROP
    END
    IFERR 'trasim' RCL
    THEN
      N SWAP STO
    ELSE
      DROP
    END
    IFERR 'sortie' RCL
    THEN
      E SWAP STO
    ELSE
      DROP
    END
    "Tracé d'une série de Fourier -3-"
    {
      { "n:" "Valeurs de n à tracer" }
      { "nn?:" "Tracé simultané des n? (O ou N)" }
      { "Sortie:" "(E)cran ou (I)mprimante" }
    }
    { 1 0 }
    { 3 N E }
    valn trasim sortie 3 →LIST
    INFORM
    IF
    THEN
      OBJ→ DROP 'sortie' STO
      'trasim' STO DUP
      TYPE 5 ≠
      IF
      THEN
        1 →LIST
      END
      'valn' STO sortie E SAME
      IF
      THEN
        PICT PURGE
      ELSE
```

```
  # 166d # 81d PDIM
 END
 trasim 0 SAME
 IF
 THEN
   1 valn SIZE
   FOR i
     valmoy 1 valn i GET
     FOR j j
       'n' STO coefan j
       varind * COS *
       coefbn j varind *
       SIN * + + EVAL
     NEXT
   NEXT
   valn SIZE trafx 0 SAME
   IF
   THEN
     équdép SWAP 1 +
   END
   →LIST STEQ FUNCTION
   'varind' RCL INDEP
   débint →NUM finint
   →NUM XRNG 0 RES 1
   ATICK ERASE AUTO
   DRAX DRAW sortie E
   SAME
   IF
   THEN
     PICTURE
   ELSE
     CR
"Série de Fourier équivalente à la fonction: "
     équdép + PR1 DROP
     "pour les valeur de n suivantes: "
     PR1 DROP valn PR1
     DROP CR PICT RCL
     PR1 DROP
   END
 ELSE
   1 valn SIZE
   FOR i
     valmoy 1 valn i GET
     FOR j j
       'n' STO coefan j
       varind * COS *
       coefbn j varind *
       SIN * + + EVAL
     NEXT
     STEQ trafx 0 SAME
     IF
     THEN
       EQ équdép 2 →LIST STEQ
     END
     FUNCTION 'varind'
     RCL INDEP débint
     →NUM finint →NUM
     XRNG 0 RES 1 ATICK
       ERASE AUTO DRAX
       DRAW sortie E SAME
       IF
       THEN
         PICTURE
       ELSE
         CR
"Série de Fourier équivalente à la fonction: "
         équdép +
         " pour n = " + n +
         PR1 DROP CR PICT
         RCL PR1 DROP
       END
       'varind' RCL PURGE
     NEXT
   END
  END
 END
 flags STOF
»
```

Thierry Bestel (601)

DES CHIFFRES ET DES LETTRES

Comme vous pourrez le constater ce programme est en RPL Uutilisateur. J'ai eu l'idée de le créer durant une soirée...

Ce jeux est inspiré des *Chiffres et des Lettres*. Afin de jouer à ce jeu il faut au minimum 2 joueurs. Il est possible soit de jouer au chiffre ou soit de jouer aux lettres.

Inspectons le contenu du programme CHIFFRE :

Il crée une variable BC permettant de stocker les chiffres choisit le candidat de manière aléatoire. Vous n'avez plus que 40 secondes pour tenter de trouver le "bon compte". Serait-il possible qu'une personne puisse completer ce programme en trouvant la solution du programme CHIFFRE (cela n'est pas dans mes capacités).

PROG.1 et PROG.2 vont permettre de choisir les consomnes ou les voyelles de manière aléatoire. PROG.3 permet de choisir les chiffres de maniere aléatoire.

Inspectons le programme LETTRE :

Là aussi il crée une variable 'BC' permettant de stocker les consomnes ou les voyelles. Là aussi vous n'avez plus que 40 secondes pour trouver un mot.

```
DIR
  CHIFFRE
    «
      "" 'BC' STO 1 6
      FOR i
        "[1]ERE [2]EME [3]ERE [4]EME"
        { "" } INPUT OBJ→
        IF TYPE 0 ==
        THEN
          PROG.3
        END
        BC + 'BC' STO
      NEXT
      1 3
      FOR j
        RAND 10 * FLOOR →STR
      NEXT
      + + " <CR>" + "       " SWAP +
      BC + CLLCD 3 DISP 40 WAIT 1800 2 BEEP
    »
  LETTRE
    «
      -57 CF -56 CF "" 'BC' STO 1 9
      FOR i
       "[C]onsomne
          ou
        [V]oyelle ?"
        { "" α } INPUT OBJ→
        IF V SAME
        THEN
          PROG.1
        ELSE
          PROG.2
        END
        BC SWAP + 'BC' STO BC 3 DISP 2 WAIT
      NEXT
      "Attention:" CLLCD 1 DISP .75 WAIT BC
      CLLCD 4 DISP 40
      WAIT 1450 1.75 BEEP
    »
  EXIT
    «
      HOME
    »
  PROG.1
    «
      { "A" "E" "I" "O" "U" "Y" }
      DO
        RAND 10 * FLOOR 'k' STO
      UNTIL
        k 6 < k 1 > AND
      END
      k GET
    »
  PROG.2
    «
      {
        "B" "C" "D" "F" "G" "H" "J" "K" "L" "M"
        "N" "P" "Q" "R" "S" "T" "V" "W" "X" "Z"
      }
      DO
        RAND 10 * FLOOR 'k' STO
      UNTIL
        k 20 < k 1 > AND
      END
      k GET
    »
  PROG.3
    «
      {
        "1 " "2 " "3 " "4 " "5 " "6 " "7 " "8 " "9 "
        "10 " "25 " "20 " "50 " "75 " "100 " "200 "
      }
      DO
        RAND 100 * FLOOR 'k' STO
      UNTIL
        k 1 ≤ k 16 ≥ AND
      END
      k GET
    »
  k 12 BC ""
END
```

Grégory Mimoun (576)

TRUCS, ASTUCES ET PETITS UTILITAIRES

Listes et musique

Faire de la musique sur HP48 (ou HP28) est très simple à programmer puisqu'il suffit d'exécuter une suite de :

```
f1 d1 BEEP f2 d2 BEEP ..... fn dn BEEP
```

où f1, f2...fn sont des fréquences correspondant aux notes du morceau de musique et d1, d2...dn sont les durées de ces notes.

Une solution élémentaire consiste à établir une liste de couples fréquence/durée :

```
LMUS :     { fn dn ..... f2 d2 f1 d1 }
```

donc la dernière note occupant la première position dans la liste, et d'exécuter le programme :

PLAY

```
« LIST→ 1 SWAP START BEEP 2 STEP »
```

Il pourra sembler plus naturel d'établir la liste de notes avec l'ordre suivant:

```
{ f1 d1 f2 d2 ..... fn dn }
```

puis de l'inverser. Ici la commande REVLIST des HP48G/GX n'est pas utilisable car il faut inverser la position des couples fréquence/durée. On pourra utiliser le programme pour tous :

REVMUS

```
«
  DUP SIZE → L N
  «
    N 1 - 1
    FOR I
      L I GET L I 1 + GET
    -2 STEP
    N →LIST
  »
»}
```

Nouvelle solution pour HP48G/GX

Ces machines offrent de nouvelles possibilités de traitement des listes (voir le chapître 17 du manuel d'utilisation). En ce qui concerne notre problème musical on peut opérer avec deux listes:

```
LF :     { f1 f2 ..... fn }
LD :     { d1 d2 ..... dn }
```

LF donc pour la liste des fréquences des notes et LD pour la liste des durées correspondantes. Avec ces deux listes comme arguments (LF au niveau 2 et LD au niveau 1) il suffira tout simplement de taper BEEP pour exécuter la musique.

Exemple:

BEEP avec les deux listes, ci-dessous mises sur la pile et dans cet ordre, vous jouera une musique de Jean-Michel Jarre.

LFjarre

```
{ 622 698 783 698 622 783 622 698 783 698
  622 783 587 622 698 622 587 698 587 622
  698 622 587 698 523 587 622 587 523 622
  523 587 622 587 523 622 587 523 466 415
  466 415 391 493 587 622 698 783 698 622
  783 622 698 783 698 622 783 587 622 698
  622 587 698 587 622 698 622 587 698 523
  587 622 587 523 622 523 587 622 587 523
  622 587 523 466 415 466 415 391 493 587 }
```

LDjarre

```
{ .125 .125 .125 .125  .125 .375 .125 .125 .125 .125
  .125 .375 .125 .125  .125 .125 .125 .375 .125 .125
  .125 .125 .125 .375  .125 .125 .125 .125 .125 .375
  .125 .125 .125 .125  .125 .5   .125 .375 .999 .375
  .375 .25  .999 .999 1.998 .125 .125 .125 .125 .125
  .375 .125 .125 .125  .125 .125 .375 .125 .125 .125
  .125 .125 .375 .125  .125 .125 .125 .125 .375 .125
  .125 .125 .125 .125  .375 .125 .125 .125 .125 .125
  .5   .125 .375 .999  .375 .375 .25  .999 .999 1.998
}
```

Récupération pour HP48G/GX et S/SX

Les possesseurs de HP48G/GX pourront récupérer des listes du type LMUS ci-dessus et les transformer en deux listes LF et LD en inversant d'abord LMUS avec le programme REVMUS ci-dessus puis en obtenant deux listes de types LF et LD avec le programme S→D.

3S→3D

```
«
  DUP SIZE → M N
  «
    { } DUP 1 N
    FOR I
      SWAP M I GET +
      SWAP M I 1 + GET +
    2 STEP
  »
»
```

A l'inverse les possesseurs de HP48S/SX pourront récupérer la musique ci-dessus, de Jean-Michel Jarre, en rassemblant les deux listes LF et LD en une seule pour être utilisée par le programme PLAY ci-dessus. Le programme D→S ci-dessous fera l'assemblage des deux listes en une seule qu'il faudra inverser avec REVMUS.

D→S

```
«
  DUP SIZE → F D N
  «
    1 N
    FOR I
      F I GET D I GET
    NEXT
    N 2 * →LIST
  »
»
```

Remarques:

Ce qui a été exposé ci-dessus est la façon la plus simple de faire jouer de la musique par nos machines mais cela n'est pas sans inconvénients. Ces listes sont gourmandes en mémoire (pour le morceau ci-dessus qui ne comporte que 90 notes chaque liste occupe 950 octets soit au total 1900 octets sans les noms). Il existe des méthodes de codage permettant de réduire notablement l'encombrement mémoire et d'éviter certains problèmes relatifs aux grosses listes lors de l'exécution *musicale*.

Il reste d'autre part le problème des silences qui se traduisent dans les programmes par :

```
..... durée WAIT ....
```

et qu'on ne peut remplacer dans les listes par:

```
..... fréquence=0 durée ....
```

car par exemple `0 5 BEEP` ne fera pas un silence de 5 secondes mais de 0 seconde.

Enfin les programmes ci-dessus sont livrés sous une forme très rustique pour être compréhensibles par les très débutants (voir article *BODE* du JPC-98).

Guy Toublanc (276)

BINOME DE NEWTON EN SYSTEM RPL

Pour ma première participation au club, voici un petit programme qui calcule de manière assez simple et rapide, les coefficients du Binôme de Newton. Ce programme vous sera utile si vous êtes en Terminale par exemple.

Note :

Les mnémoniques et la terminologie utilisées sont conformes à `RPLCOMP.EXE` distribué par HP dans les disquettes HORN, ici la 4.

Les points d'entrée sont donc officiels et ce programme marchera sur toute HP48. Dans le cas contraire et par la suite, ce sera indiqué en commentaire.

Tout d'abord, voici le source du programme :

```
( BINO: Computes the coefficients of )
( Newton's BINOmial )
( Written by F. BARBAISE for the HP-48 )
( Copyright Bouli Corporation 1994 )
( Bytes: 37.5     Checksum: # 87E6 )

::
  CK1NOLASTWD  CKREAL  DUP  COERCE  #1+
  ZERO_DO (DO)
    DUPINDEX@  UNCOERCE  %COMB
  SWAPLOOP (LOOP)
  COERCE  #1+  {}N
;
```

Puis le commentaire :

Une fois compilé avec `RPLCOMP.EXE`, vous devriez obtenir le checksum et le nombre d'octets indiqués en commentaire dans le source.

En langage système (Sys-RPL), il faut savoir que les instructions sont dépourvues de protections. C'est au programmeur de les faire ! Ici, pour calculer les coefficients, il nous faut un réel au niveau 1 de la pile. En effet, si les Entiers Système sont parfaitement accessibles en Sys-RPL, ils n'en sont pas moins en User-RPL (langage utilisateur, classique) ! La séquence `CK1NOLASTWD CKREAL` teste si il y a quelque chose dans la pile (`CK1NOLASTWD`) si oui, alors teste si c'est bien un nombre réel (`CKREAL`).

Les tests de présence et de nature d'argument étant faits, nous allons pouvoir commencer le programme.

`DUP COERCE #1+` va sauvegarder le nombre réel et le transformer en Entier Système puis lui ajouter 1. Par exemple, si on met le réel 4 au niveau 1 de la pile, on aura :

```
2:         4
1:      <5h>
```

Ceci sert à définir la boucle de calcul des coefficients du binôme. Cette boucle va de 0 à n, soit n+1 termes à calculer. Il faudra donc utiliser une boucle de type `ZERO_DO (DO) ... LOOP`.

Dans cette boucle, calculons ces fameux coefficients tout simplement à l'aide de la fonction `%COMB` qui nécessite 2 réels en argument.

`DUPINDEX@ UNCOERCE %COMB` va calculer ces coefficients. `DUPINDEX@` fait une copie (`DUP`) de l'objet situé au niveau 1 (ici un réel) rappelle le compteur de la boucle (`INDEX@`) qui est un entier système, le transforme en un nombre

réel (UNCOERCE) et fait une combinaison de n éléments choisis parmi INDEX@ c'est-à-dire : $C_{n,i}$

SWAPLOOP (LOOP) termine la boucle en replaçant le réel au niveau 1 de la pile (qui avait été sauvegardé, vous vous en rappelez ?).

COERCE #1+ {}N va transformer à nouveau le réel de départ en Entier Système, lui ajouter 1 et créer une liste ({}N) soit une liste de n+1 objets car on a fait une boucle de 0 à n.

Voila c'est fini. Pour clore le tout voici tout de même un exemple (simple) d'utilisation :

On veut développer $(a+b)^6$... Pas facile de calculer les coefficients de a^6, a^5*b, a^4*b^3 ... Et bien mantenant, vous le pouvez ! Entrez le réel 6 dans la pile et lancez le petit programme, vous devriez obtenir:

```
1: { 1 6 15 20 15 6 1 }
```

C'est plus précisément la sixième ligne du triangle de Pascal qui sert au développement de $(a+b)^n$.

Enfin sachez que si vous n'avez pas de PC et RPLCOMP.EXE, ne désesperez pas ! Ce logiciel existe sur HP48, cherchez bien...

Frédéric Barbaise (624)

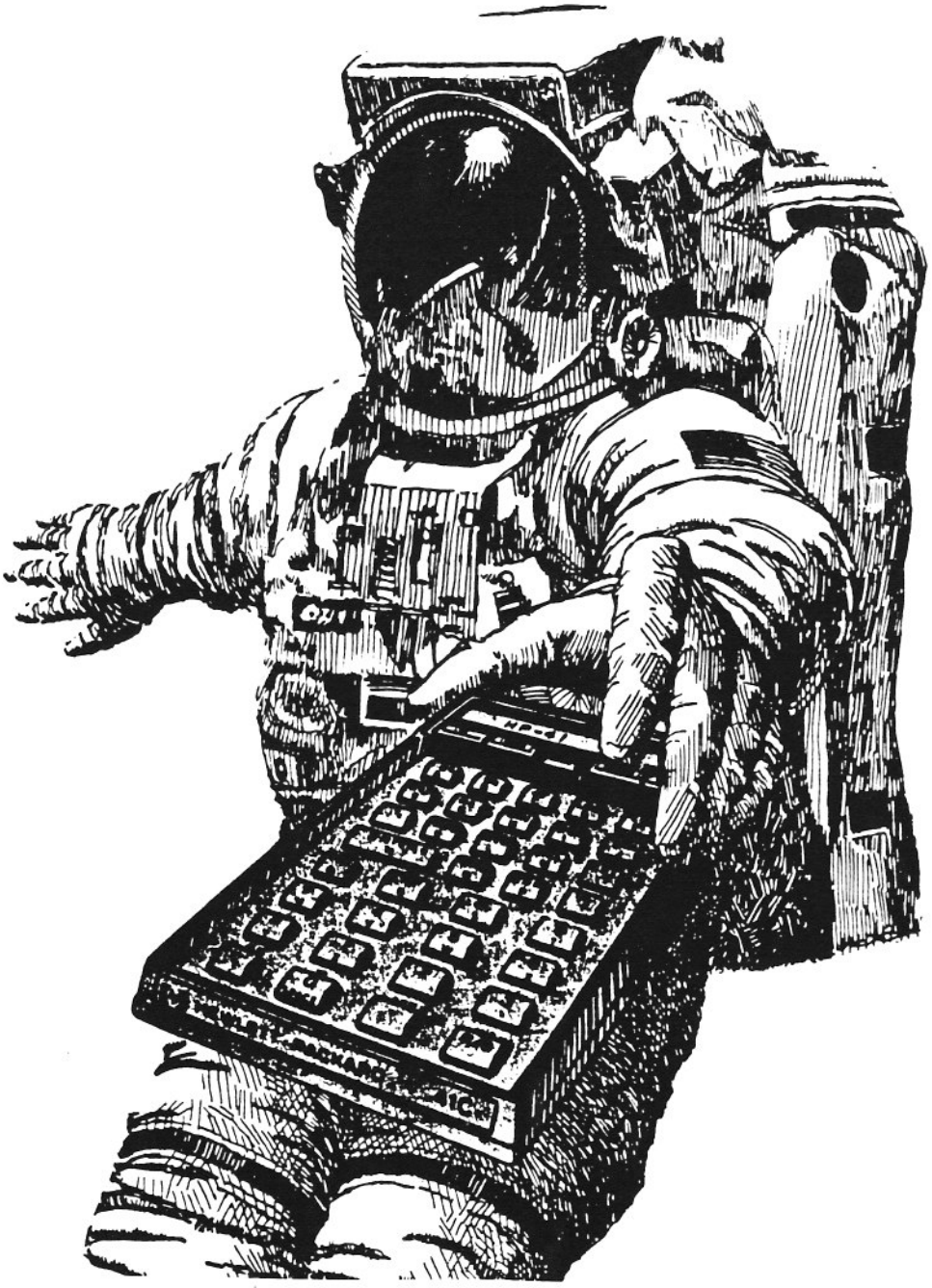
REDUCTION DE PROGRAMMES

Acte III

Merci à Guy Toublanc de m'avoir signalé les inperfections dans mon programme, je rectifie mon tir et voici un "STRIP" qui prend moins de place (312 octets, contre 326) et pas de bugs. STRIP fonctionne également sur les listes comportant les commandes en User-RPL, par exemple pour créer les menus exécutables (voir *JPC 85* p15).

```
ASSEMBLE
        nibasc  \HPHP48-P\
RPL
::
  ' ::
      INHARDROM?    ?SEMI
      DUPNULL{}?  casedrop  NULL{{}}
      DUPNULLCOMP?  casedrop  'NOP
      INNERCOMP  OVER'  x>>
      EQIT
        :: DUP  #1+ROLL  ROT2DROP  #2-  ;
      DUP#0_DO  (DO)
        ROLL
        ::
          ::
            ::
              '  xSYSEVAL
              EQcase
                ::
                  OVER  TYPEHSTR?
                  NOTcase  ::  2RDROP  ;
                  DROP  TWO  NTHCOMPDROP
                ;
              DUP' xIF      EQ  caseDROP
              DUP' xCASE    EQ  caseDROP
              DUP' xUNTIL   EQ  caseDROP
              DUP' xENDTIC  EQ  caseDROP
              ' xIFEND
              EQcase
                ::
                  ISTOP-INDEX  #1=  caseDROP
                  RDROP
                ;
              RDROP
            ;
            INDEX@  #1-  INDEXSTO
            ISTOP@  #1-  ISTOPSTO
            RDROP
          ;
          ' x<<       EQcasedrop  'NOP
          ' xSILENT'  EQcasedrop  ::  ' '  ;
          ' x>>ABND   EQcasedrop  ::  ' ABND  ;
          DUPTYPELIST?  case  ID STRIP
          DUPTYPECOL?   IT    ID STRIP
        ;
        ISTOP@
      LOOP
      ::N
    ;
  >R
  OLASTOWDOB!  CK&DISPATCH1
  FIVE
    ::
      R>  EVAL  DUPTYPECOL?  NOT?SEMI
      DOBINT DOLIST  CHANGETYPE
    ;
  EIGHT
    NOP
;
```

Lu Vy Huynh (579)



HP95 / HP100 / HP200

J. Belin	Nouvelles Cartes Mémoire	*18*
G. Toublanc	SOS pour Lotus	*18*
J. Belin	Animations sur HP95/100 (acte II)	*19*
	Le coin des codes	*31*

NOUVELLES CARTES MEMOIRES

Les DoubleFlash PLus

La firme ACE vient d'introduire sur le marché de nouvelles cartes Flash, équivalentes aux Sundisk, sous la dénomination *DoubleFlash+*.

Il s'agit en fait de cartes fabriquées par *IBM Microelectronics*. La principale différence par rapport aux Sundisk est qu'elles sont équipées d'une mémoire cache de 32 ko, permettant d'accélérer notablement la lecture.

Elles ne fonctionnent que sur HP100/200.

Comme pour les cartes *DoubleFlash*, elles sont accompagnées de versions de drivers spécialement écrites par ACE, ainsi que d'autres utilitaires tel que `BATTMAN`.

Les prix sont inférieurs de 15 à 20% à ceux des Sundisk commercialisées sous la marque ACE (DoubleFlash), soit :

5Mo : 309$
10Mo : 459$
20Mo : 799$
30Mo : 1149$
40Mo : 1499$

Les capacités indiquées sont les valeurs réelles (hardware), et non les références données très discutablement par ACE (tenant compte de la présence de Stacker) pouvant laisser penser que la capacité est double.

Il est à noter qu'il a été constaté un problème lorsque des piles alcalines Toshiba ou Duracel sont installées dans le palmtop, entrainant l'affichage intempestif du message "Low Battery". Cela est dû du au fait que ces cartes provoquent un drain de courant plus élevé lorsque l'on allume le palmtop. Il est donc conseillé d'utiliser des accus (Nicad ou Nimh) ou des piles au lithium.

Pour ceux qui n'auraient pas la nouvelle adresse d'ACE, la voici :

ACE Technologies
592 Weddell Drive #6
Sunnyvale, CA 94089
USA

Des cartes Ram à bas Prix

Pour ceux qui ne peuvent pas s'offrir les cartes précédentes, mais qui ont un besoin pressant d'augmenter leur capacité de stockage, il a été mentionné que les cartes Ram 1Mo commercialisées à 500F par Apple pour le Newton fonctionnaient parfaitement sur les Palmtops HP (bien que certaines mauvaises langues aient déclaré qu'il fallait absolument retirer le très visible logo Apple pour obtenir un fonctionnement correct !).

Attention, ceci ne concerne que les cartes 1 Mo. les cartes Newton d'une capacité supérieure sont en fait des cartes Flash au format Intel, incompatibles avec les HP95/100/200.

Jacques Belin (123)

LOTUS HP : S.O.S

En ce qui concerne les nouveaux possesseurs de HP100 il y a un problème relatif à la documentation du Lotus intégré à la Rom. Celui-ci est une version 2.4 or les versions pour PC sont plus évoluées puisque maintenant elles fonctionnent sous Windows. Les livres relatifs à Lotus dans ses anciennes versions disparaissent des catalogues des éditeurs. Le manuel du HP100 consacre à peine quatre pages à Lotus, HP se retranchant devant le fait que Lotus est déjà familier aux utilisateurs de HP100 !

Ceci a pour conséquence qu'il est difficile de connaître actuellement ce qui distingue le Lotus des HP100 de celui qu'on nous présente dans les livres dans ses versions les plus récentes pour PC.

Un lecteur de *JPC* ayant une expérience du Lotus HP100 (ou même HP95) et de celui des PC, pourrait-il apporter quelques informations sur ces différences et ce qui peut apparaître comme des spécificités côté HP100/95. Il y a certainement des possibilités qui ne sont pas évidentes à priori pour un débutant.

Merci d'avance.

Guy Toublanc (276)

ANIMATIONS :

Acte II

Dans l'avant dernier JPC, je vous ai présenté une méthode et des outils permettant de générer des animations dans un format compatibles avec nos palmtops.

Ce deuxième article vous présente donc le programme `PLAYA95`, qui vous permettra d'afficher ces animations. Rappelons (et c'était un de mes imperatifs initiaux), qu'il fonctionne aussi bien sur HP95 que sur HP100/HP200.

Utilisation

La syntaxe du programme est la suivante :

```
playa95  [-l] [-d]  files  [files..]
```

Il accepte comme paramètre un ou plusieurs fichiers A95 (décrits dans le premier article). Les *wildcards* sont autorisés.

L'option '`-l`' indique que l'on veut créer une boucle réaffichant tous les fichiers.

L'option '`-d`' permet de désactiver l'affichage du message "`Scanning file`" apparaissant avant l'affichage de l'animation.

Une fois que le fichier est chargé, vous pouvez utiliser les touches suivantes :

- `[ESC]` : Fin d'exécution du programme.
- `[E]` : Passage au fichier suivant.
- `[S]` : Stoppe l'animation.
- `[R]` : Relance l'animation.

Lorsque l'animation se déroule, il est possible de contrôler la vitesse d'affichage à l'aide des touches suivantes :

- `[+]` : Accélère l'animation.
- `[-]` : Ralentit l'animation.
- `[*]` : Restitue la vitesse originale.

Lorsque l'animation est stoppée, les touches actives sont les suivantes :

- `[→]` : Affiche l'image suivante.
- `[←]` : Affiche l'image précédente.
- `[F]` : Affiche la première image.
- `[L]` : Affiche la dernière image.

Performances

Comme je l'ai indiqué la dernière fois, les performances sont dépendantes du mode de compression, mais aussi du support de stockage du fichier.

A titre indicatif, voici les performances maximum que l'on peut atteindre sur un HP100 (avec un fichier contenant des images non compressées) suivant le type de stockage :

- Carte SunDisk (version SDPL5) : 30 im/sec.
- Disque C et Carte Ram : 50 im/sec.

Avec un HP95, la vitesse est d'environ 20 images par seconde, quelque soit le support.

Il est à noter que pour les animations de type 'Ping Pong' la vitesse d'affichage en sens inverse peut être beaucoup plus lente que dans le sens normal, d'autant plus que le fichier gros. Cela est dû aux algorithmes de lecture de fichiers du DOS qui sont optimisés dans le sens normal de lecture (dans l'autre sens, et même dans le cas d'un accès "direct", le DOS doit retrouver la position de l'octet à lire en relisant la chaine des pointeurs des clusters incluse dans la FAT).

Utilisation avec Stacker

Si le programme fonctionne très bien avec des fichiers stockés sur les cartes de type SunDisk, il n'en n'est pas de même lorsque la carte est compressée avec un logiciel de type *Stacker*, qu'il s'agit d'une carte Ram ou Flash.

En effet, l'animation devient saccadée, en affichant extrèmement rapidement trois ou quatre images, puis s'arrétant longuement sur l'image suivante. Cela est dû au fait que lorsque l'on demande de lire un certain nombre d'octets dans un fichier situé dans une unitée compressée, Stacker lit un plus grand nombre d'octets et les décompresse dans un buffer intermédiaire. Si cette méthode est efficace dans le cas d'un disque dur (possédant un temps d'accès important), elle l'est moins dans le cas d'une carte mémoire.

Il est théoriquement possible de réduire la taille du buffer, (afin d'obliger Stacker à ne lire qu'une seule image à la fois), mais je n'ai pas constaté de différence notable.

La seule solution que je conseille donc aux utilisateurs des cartes stackées est de transférer (de manière temporaire) le fichier A95 sur le disque C. Malheureusement, cette solution sera inapplicable si vous êtes possesseur d'un HP100 1Mo (ou pire, d'un HP95 512 Ko !) et que vous désirez visualiser une très

grosse animation (à titre indicatif, une de mes dernières créations fait plus de 980 ko !).

Notez que ces observations concernent Stacker 3.0. Je n'ai pas eu la possibilitée d'effectuer des tests avec la nouvelle version.

Utilisation sous le System Manager

Le programme semble bien fonctionner lorsqu'il est lancé à partir du System Manager (par l'intermediaire d'un fichier batch, afin d'ajouer les paramètres). Il est en effet possible (sur le HP100) de revenir aux applications System Manager lorsque l'animation est visualisée (ce qui stoppe son déroulement, puisque le SM n'est pas un vrai multitache) puis de revenir sur l'animation sans encombre.

Cependant, le programme modifiant la fréquence du timer interne (elle est multipliée par 16), certains inconvénients peuvent apparaitre.

- L'heure affichée par les principales applications est erronnée et l'application 'Chronomètre' devient inutilisable, car les secondes semblent défiler à un rythme rapelant une séquence célébre du film *La Machine à remonter le temps*. Notez cependant que dès que vous terminerez l'exécution de PLAYA95, tout rentrera dans l'ordre et l'heure réelle sera correctement affichée.

- Si vous utilisez d'autres programmes effectuant des mesures des temps, ces valeurs auront des grands risques d'être inexacts. Il est à noter à ce sujet que la valeur indiquée par 100BUDDY pour le temps d'utilisation des batteries est irrémédiablement erronée, car elle est stockée dans une variable mémoire indépendante, hors de mon contrôle.

- Le HP100 utilisant le timer pour contrôler le temps de charges des batteries (il n'est en pleine charge que pendant les six premières heures), il y a un risque de recharge insufisante si vous lancez une animation pendant une phase de recharge.

- Plus généralement, certains programmes utilisant le timer pour d'autres fonctions (par exemple les jeux d'arcades) deviendrons impossibles à utiliser.

Par contre, les alarmes étant gérées par l'horloge temps réel (circuit indépendant du timer), il n'y aura pas de problème. Mieux, si une alarme survient pendant l'affichage d'une animation, celle ci sera interrompue et l'alarme traitée. Cependant, j'ai constaté que le HP100 affichait un message d'erreur tout de suite après, lorsque il devait sélectionner l'alarme suivante. N'ayant pas eu le temps d'effectuer de tests à ce sujet, je vous conseille d'être prudents si vous êtes un gros utilisateur de ces alarmes.

Pour terminer sur ce sujet, il est à noter enfin que le temps d'auto-extinction étant aussi géré par le timer, il a fallu aussi le modifier afin d'éviter que le palmtop s'éteigne au bout de 10.25 secondes (avec la valeur par défaut). Il aurait été possible de modifier cette valeur en tenant compte de celle choisie par l'utilisateur, mais j'ai préféré la fixer à 3 minutes. Cela permet d'être sûr, d'une part, de ne pas avoir une extinction intempestive si l'utilisateur a sélectionné une valeur trop faible, et d'autre part d'éviter une trop grande consommation des piles dans le cas contraire. Bien sûr, la valeur originale est restituée en fin d'exécution du programme.

Analyse du programme

Le principe général du programme consiste à lire une image à la fois sur le disque (car certaines animations sont suceptibles d'etre plus importantes que la mémoire disponible) puis l'afficher à un top donné régulièrement par l'horloge standard de l'ordinateur (rendant ainsi la vitesse d'affichage égale sur tous les ordinateurs, même si il possèdent des microprocesseurs différents).

Sur tous les IBM PC, un top d'horloge est donné de façon standard 18.2 fois par seconde, par l'intermédiaire de l'interruptions 1Ch. Il suffirait donc d'écrire un petit gestionnaire d'interruption lançant l'affichage de l'image dès que cette interruption est appelée.

Cependant, il est nécessaire de modifier cette valeur, car elle n'est pas très adaptée pour notre application, pour les raisons suivantes :

- Elle est un peu trop importante, car les palmtops sont souvent incapables de lire, décoder et afficher dans ce laps de temps les images stockées sous le format PCX. Comme je l'ai dit dans le précédent article, la vitesse admise pour le HP95 se situe plutôt vers les 10-12 im/s.
- Il serait tout simplement intéressant de donner à l'utilisateur la possibilité de faire varier la vitesse d'affichage !

Théoriquement, la manière la plus simple pour modifier la vitesse aurait été de modifier directement la fréquence de sortie du circuit commandant les interruptions 1Ch. Malheureusement, si il est tout à fait possible de l'augmenter, nous ne pouvons pas la diminuer, car la fréquence de 18,2 hz est justement la valeur minimum autorisée par le hardware de l'IBM PC (voir plus bas pour plus de détails).

Afin d'obtenir une plage de vitesse intéressante, il est donc nécessaire de recourir à une astuce, consistant à augmenter la fréquence de sortie de l'horloge, puis la

diviser de façon logicielle, à l'aide d'un compteur n'autorisant l'accès au code de l'interruption qu'au bout d'un certain nombre d'appels.

Par souci de simplification, j'ai donc choisi de travailler à une fréquence "hard" fixe plus élevée, et de contrôler la vitesse à l'aide du compteur "soft". L'inverse (fréquence variable et compteur fixe) aurait cependant été possible. J'ai donc multiplié la fréquence "hard" par 16. Associé avec un diviseur "soft" acceptant 256 valeurs, ceci permet d'obtenir une plage de vitesses "utile" comprise entre 1.13 im/s et 291.2 im/s. La vitesse maximum étant évidement toute théorique, compte tenu des temps de lecture, de décompression et d'affichage d'une image.

Une fois le problème de la fréquence réglée (enfin, pas tout à fait, voir un peu plus bas...) nous pouvons déterminer à quel moment le programme peut activer la fonction `viewim`. Cet accès n'est autorisé que lorsque toutes les conditions suivantes sont validées :

- Le compteur "soft", (`intcnt`) est arrivé à la valeur zéro. Ce conmpteur est initialisé avec la valeur correspondant à la vitesse courante (en 'ticks'). La vitesse par défaut étant la valeur stockée dans le fichier.
- Si nous somme en mode "défilement" (flag `run` armé). Dans le cas contraire, le contrôle de l'affichage est géré par routines de gestion du clavier.
- Si le flag `buffpl` indiquant que le buffer contenant une image est plein. Ceci permet d'éviter de lancer l'affichage alors que le DOS est toujours en train de lire le fichier.

Notez que le fait d'avoir placé l'appel à la routine de lecture du fichier en dehors de la routine d'interruption complique un peu le programme (à cause de l'utilisation de la variable `buffpl`), mais cela permet de "durcir" le programme, en évitant la possibilité d'un débordement de pile ou d'une réentrance du DOS.

Une puce et un Bug

Au cours du développement du programme, j'ai été confronté à un problème assez étrange : si le programme fonctionnait parfaitement sur le HP100, la fréquence d'affichage était totalement instable sur le HP95, provoquant une animation totalement sacadée ! Bien sûr, après avoir éliminé la possibilité d'un bug dans le programme, mes soupçons se sont portés sur des hypothèses relatives à la différence de vitesses des microprocesseurs des deux palmtops (surcharge temporaire de la pile...). Cependant, quelques tests portants sur des gestionnaires d'interruption très simplifiés me montrèrent que ce n'était pas le programme qui était à mettre en cause, mais l'horloge elle même. En fait, après avoir fait part de mes difficultés à un des développeurs du HP95 à Corvallis, celui ci m'a appris que le problème ne venait pas du programme, mais d'un bug présent dans certaines versions du Hardware du HP95 !

Après que mon interlocuteur m'ait fourni quelques renseignements complémentaires sur la façon dont ce bug était géré par le Bios du HP95 (merci Everett !), il m'a été possible de corriger ce problème de façon à ce que le programme fonctionne aussi bien sur le HP95 que sur le HP100.

Mais avant de discuter de cela, je pense qu'il n'est pas inutile de décrire brièvement comment sont générés les signaux d'horloge sur IBM PC. D'autant plus que j'ai remarqué que ce sujet était de moins en moins abordé dans les livres traitant de la programmation système sur ces ordinateurs.

Lors de la sortie de l'IBM PC, la gestion de certaines fonctions utilisées automatiquement, à des périodes régulières (rafraichissement de la Ram, gestion de l'horloge...) était assurées par un circuit intégré Intel, portant l'appelation de 8253 (ou 8254 sur certains modèles ultérieurs). Depuis, vous ne trouverez plus ce circuit sur les cartes mères de votre ordinateur, car ses fonctions ont été intégrées dans des puces plus généralistes. Mais les méthodes d'accès ont été conservées.

Pratiquement, ce circuit se compose de 3 compteurs (ou plutôt 3 décompteurs) 16 bits. L'évolution de leur valeur est assurée par une horloge externe travaillant à la fréquence de 1193180 Hz, sur tous les compatibles IBM (correspondant au quart de la vitesse des microprocesseur utilisés sur les premiers PC). Lors du passage à 0 d'un des compteurs, une patte du circuit correspondant à celui-ci change d'état, ce qui provoque la génération d'une interruption sur l'IBM PC. Par exemple, le compteur 0 est réservé à la génération d'un signal d'horloge qui est lié matériellement à l'interruption Bios 08h gérant (rntre autres) le compteur incrémentant l'heure courante. Cette interruption possède en plus la particularité de générer une interruption 1Ch à chaque appel. Normalement, cette dernière interruption retourne immediatement au code de l'interruption 8h, mais il est possible (cela a été fait pour ça d'ailleurs) d'écrire un petit gestionnaire interceptant cette interruption pour effectuer, en plus, une action à chaque top d'horloge. C'est bien sûr ce j'ai fait dans mon programme.

Le circuit possède 6 modes de travail permettant de déterminer le type de comptage à effectuer. Par exemple, le mode 0 travaille en décompteur simple. Une fois un des compteurs chargé avec une valeur quelconque, il commence à la décompter. Lorsque il atteint la valeur 0, il provoque une interruption, puis continue le

décomptage, en repartant de sa valeur maximum, soit 65536. Dans le cas du mode 2, la différence est que c'est la valeur de départ que est prise en compte après l'arrivée sur le 0, et non 65536. Ceci permet de transformer le compteur en diviseur de fréquence.

Chaque compteur peut bien sûr travailler indépendament sur un mode différent.

Normalement, afin de modifier la fréquence d'activation de l'interruption 08h/1Ch (autrement dit, le compteur 0 du 8253), on utilise le code suivant :

```
mov     al,34h
out     43h,al
jmp     $+2
jmp     $+2
mov     ax,(1193180/FRQ)
out     40h,al
jmp     $+2
jmp     $+2
mov     al,ah
out     40h,al
```

La première valeur, stockée dans le port 43h, est une concaténation de bits représentant les différents modes de travail du 8253. En simplifiant, et en initialisant certains bits avec les valeurs classiquement utilisées, nous pouvons dire que son format binaire est *cc11ppp0*, avec *cc* indiquant le numéro du compteur et *ppp* indiquant le mode de travail. La valeur indiquée (34h) indique donc que l'on désire faire travailler le compteur 0 en en mode 2 (diviseur de fréquence). Le compteur lui même étant accessible par le port 40h.

Les lignes `jmp $+2` n'ont pour effet que de passer à l'instruction suivante. C'est en fait une petite astuce permettant de ralentir le programme, afin de laisser au 8253 le temps de traiter les ordres que nous lui envoyons.

`FRQ` est la fréquence désirée. Si on divise la fréquence d'entrée du 8253 par `FRQ`, nous obtenons la valeur du diviseur devant être stocké dans le registre du circuit. La taille de celui-ci étant de 16 bits, la valeur maximum admissible est de 65535. Si nous divisons la fréquence maximale (1193180 Hz) par 65535, nous obtenons environ 18.2 Hz, qui est la fréquence *minimum* pouvant être générée sur l'IBM PC.

Ceci expliqué, revenons au problème affectant le HP95. Notons au passage qu'il n'existe que pour les révisions A et B du *Hooper*, le circuit Intel/HP spécifique au HP95 intégrant la plupart de l'électronique logique (hors microprocesseur et mémoires). Une version C, corrigeant ces défauts à été introduite, mais très tardivement. Vous pouvez utiliser le programme `95INFO` (présent sur notre disquette n°4) pour connaitre quelle version de HP95 est en votre possession.

La solution à utiliser est d'utiliser le mode 0. La différence par rapport au mode 2 est qu'une fois la valeur 0 atteinte (et l'interruption armée), il ne recommence pas le décomptage à partir de la valeur que l'on a stockée, mais à partir de 65535. Ce qui implique que l'on devra, à chaque appel de l'interruption, recharger la valeur initiale dans le 8253. Ces instructions devront bien sûr être incluses dans le gestionnaire d'interruption.

Tout ceci serait encore simple (enfin, relativement !) si il ne falait pas prendre un autre élément en compte. En effet, si nous stockons la valeur correspondant à la fréquence désirée, nous constatons vite que cette fréquence est légèrement plus faible que ce que nous attendons. Cela est dû au fait nous ne réinitialisons pas le 8253 immédiatement après que l'interruption se soit produite, mais un peu plus tard. En effet, il ne faut oublier qu'entre la première instruction exécutée par l'interruption et celle commandant l'initialisation du 8253, il y a un certain nombre de lignes de code, situées tout au début de notre gestionnaire d'interruption (et incluant en partie la routine de réinitialisation elle même !), mais aussi un certain nombre des lignes (à priori inconnu) précédent l'appel à l'Int 1Ch, dans le gestionnaire de l'interruption 08h...

Il faut donc retrancher à la valeur initiale, une autre valeur correspondant au temps d'exécution de ce code.

Dernière difficulté, cette valeur est différente pour chaque machine. Ceci à cause de plusieurs facteurs :
- La vitesse d'horloge du microprocesseur.
- Son type (un 80386 exécute certaines instructions en moins de cycles d'horloge que le 8086).
- Le code du gestionnaire de l'interruption 08h, qui est différent pour chaque BIOS.

Afin de déterminer cette valeur, nous pouvons utiliser deux méthodes :
- Soit calculer le temps théorique d'exécution, en convertissant (à l'aide des tables fournies par Intel) chaque instruction en nombre de cycles d'horloge correspondant, puis en tenant compte de la vitesse du microprocesseur. Si cela vous parait facile, n'oubliez pas que cela nécessite de désassembler une partie du code le code de l'interruption 08h. C'est simple si l'appel à l'Int 1Ch est au début, beaucoup moins si il est à la fin et qu'il y a de complexes sous-programmes entre les deux..
- Soit en mesurant, à l'aide d'un chronomètre, le temps d'intervention d'un évenement dépendant de l'interruption 08h, puis en déterminant la valeur recherchée da façon à que cet évenement survienne au bon moment. Pour notre cas, je me suis tout simplement basé sur le temps d'autoexctintion des palmtops, que j'ai fixé, rapellons le, à trois minutes. Le résultat est peut

être un peu moins précis, mais c'est suffisant pour la plupart des applications.

Pour notre programme, la valeur théorique à charger dans le 8253 est de 1000h (puisque nous voulons multiplier la fréquence originale par 16). Après le test précédement cité, nous obtenons 0E24h pour le HP95 et 0FFC pour le HP100. La valeur correspondant au HP100 devrait convenir pour le HP200, à moins (et c'est peu probable) qu'il y ait eu un grand changement de la code de l'interruption 08h.

Une fois que la valeur recherchée est déterminée pour chaque machine, il n'y a plus qu'à la stocker dans une variable (que j'ai appelé PICval) initialisée au début du programme, ou dans la routine d'initialisation du gestionnaire d'interruption, comme ceci :

```
        mov     ax,4dd4h  ; test : HP95 ou HP100/200
        int     15h
        cmp     cl,2
        je      initi_1
        mov     word ptr [PICval],0E24h
        jmp     initi_2
        ;-------------
initi_1: mov    word ptr [PICval],0FFCh
initi_2: mov    al,34h
        out     43h,al
        jmp     $+2
        jmp     $+2
        mov     ax,01000h ; Nouv. fréq. = 18.2Hz*16
        out     40h,al
        mov     al,ah
        jmp     $+2
        jmp     $+2
        out     40h,al
```

En examinant ces lignes, vous pourrez remarquer que, malgré ce que j'ai dit dans mes explications, j'initialise le 8253, dans le mode normalement utilisé pour le HP100 : mode 2, abec valeur de comptage de 1000h. En fait, même si cette initialisation provoque le bug sur le HP95, cela n'est pas grave, car le délai exact que mettra la première image à s'afficher importe peu...

Ensuite, il ne reste plus qu'à placer au tout début de notre gestionnaire d'interruption les lignes permettant de réinitialiser le 8253 après chaque appel (ne pas oublier de sauvegarder le registre AX juste avant) :

```
        mov     al,0h
        out     43h,al
        jmp     $+2
        jmp     $+2
        mov     ax,cs:[PICval]
        out     40h,al
        jmp     $+2
        jmp     $+2
        mov     al,ah
        out     40h,al
```

Cette programmation peut paraitre un peu "hybride", puisque chaque machine utilise dans les deux routines d'initialisation et de traitement, du code destiné à l'autre (code HP100 pour l'initialisation, et code HP95 pour le traitement), mais elle a le mérite de fonctionner, sans trop compliquer le programme, donc sans ralentir l'exécution.

Pour finir (et pour remercier ceux qui on bien voulu me lire jusqu'à la fin !), je voudrais faire un aveu. Si vous regardez attentivement le programme, vous vous appercevrez que j'ai omis de parler d'une autre fonctionnalité intéressante du programme : Il est capable d'afficher les fichiers PCX ! En fait si je laisse cette partie "non documentée", cela est dû au fait qu'elle est très incomplête, puisqu'elle ne permet d'afficher que des images 240×128, issues de programmes générant des fichiers utilisant la variante la plus simple du format PCX. Par exemple, il n'est pas possible d'afficher le fichier TOPCARD.PCX du HP95, qui a été probablement créé à l'aide du Paintbrush inclus dans Windows. Pour le rendre compatible avec PLAYA95 (tout en restant affichable par le System Manager), vous pouvez le convertir à l'aide de *Graphic Workshop*. Il aurait bien sûr été possible de tenir compte de ce problème dans le programme, mais étant donné que c'est la même routine qui affiche les images fixes ou animées, cela aurait été au détriment de la rapidité d'affichage des animations.

Jacques Belin (123)

```
;*****************************************************************************
;*****************************************************************************
;**                                PLAY95.ASM                               **
;**                          Affichage de fichiers A95                      **
;**                                                                         **
;**   Dernière modification : 21/08/94        Auteur : Jacques Belin        **
;**       'nettoyage' code initvect()                                       **
;*****************************************************************************
;*****************************************************************************

PATH_SIZ EQU    80                      ; Taille maxi d'un chemin
DTA_SIZE EQU    48                      ; Taille de la DTA

FA_RDNLY EQU    1                       ; Codes d'attributs des fichiers
FA_HIDD  EQU    2
FA_SYST  EQU    4
FA_VOL   EQU    8
FA_DIR   EQU    16
FA_ARCH  EQU    32

K_ESC    EQU    001Bh                   ; Définition des codes de touches
K_ENTER  EQU    000Dh
K_UP     EQU    4800h
K_DOWN   EQU    5000h
K_LEFT   EQU    4B00h
K_RIGHT  EQU    4D00h
K_PLUS   EQU    002Bh
K_MINUS  EQU    002Dh
K_MULT   EQU    002Ah
K_E      EQU    0045h
K_F      EQU    0046h
K_L      EQU    004Ch
K_R      EQU    0052h
K_S      EQU    0053h


MAGIC    EQU    5435h                   ; Code de reconnaissance de fichier A95

NOCOMP   EQU    0                       ; Code des modes de compression
PCXCOMP  EQU    1

ONETIME  EQU    0                       ; Affichage une seule fois
LOOPS    EQU    1                       ; Affichage en boucle
PINGPONG EQU    2                       ; Affichage en mode "Ping Pong"

PCXHDR   STRUC                          ; ENTETE FICHIER PCX
         PMANUF   DB      (?)           ; Code fabriquant (toujours 0Ah)
         PVERS    DB      (?)           ; Version
         PRLE     DB      (?)           ; Type de codage (toujours 1)
         PBITPX   DB      (?)           ; Nombre de bits par pixel
         PLEFT    DW      (?)           ; Coordonnées des coins de l'image
         PTOP     DW      (?)
         PRIGHT   DW      (?)
         PBOTTOM  DW      (?)
         PHRES    DW      (?)           ; Résolution Horizontale
         PVRES    DW      (?)           ; Résolution verticale de l'écran
         PRGB     DB 48 DUP (?)         ; Palette de couleurs de l'écran
         PVMODE   DB      (?)           ; Mode vidéo (non utilisé)
         PNPLANES DB      (?)           ; Nombre de plans de l'image
         PBPLIN   DW      (?)           ; Nombre d'octets pour une ligne
         PPALINFO DW      (?)           ; Si 0->image couleur, 1->Niveaux Gris
         PSHRES   DW      (?)           ; Résolution horizontale (scanner)
         PSVRES   DW      (?)           ; Résolution verticale (scanner)
         PFILLER  DB 54 DUP (?)         ; Pour faire 128 octets...
PCXHDR   ENDS

A95HDR   STRUC                          ; ENTETE FICHIER A95
         AMAGIC   DW      (?)           ; Code de contrôle
         AVERS    DB      (?)           ; Version du fichier
         ACOMP    DB      (?)           ; Mode de compression
         ANIMAGES DW      (?)           ; Nombre d'images
         AANIM    DB      (?)           ; Type d'animation (loop, pingpong...)
         ASPEED   DB      (?)           ; Vitesse d'affichage idéale
         AFILER   DB 120 DUP (?)        ; Réservés pour un usage futur
A95HDR   ENDS


         DOSSEG
code     segment para 'CODE'
         assume cs:code,ds:code
         ORG    100h

start:   jmp    main                   ; Première inst. exécutée
         INCLUDE getopt.inc             ; Routine d'analyse des options
                                        ; (sources parus dans JPC 87)
                                        ; DECLARATIONS DES VARIABLES
optstr   DB     "ld",0                 ; Chaine des options
errflg   DB     0                      ; Flag d'erreur
opti     DW     0                      ; Sauvegarde de optind
loopflg  DB     0                      ; Flag affichage en boucle des fichiers
nodisp   DB     0                      ; Flag non-affichage "Scanning..."
escape   DB     0                      ; Flag permettant la sortie definitive

dta      DB     DTA_SIZE dup (0)       ; DTA de travail
path     DB     PATH_SIZ dup (0)       ; Directory de travail
nameptr  DW     (?)                    ; Pointeur sur premier caractère nom
handle   DW     (?)                    ; Handle du fichier

oldtime  DW     (?)                    ; Ancienne valeur du Time-out

seghdr   DW     (?)                    ; Segment du buffer entête fichier
segbuff  DW     (?)                    ; Segment du buffer de lecture
segimg   DW     (?)                    ; Segment du buffer image

oldintsg DW     (?)                    ; Sauvegarde du vecteur de l'Int 1C
oldintof DW     (?)

PICval   DW     (?)                    ; Compteur 8253

imsize   DW     (?)                    ; Taille de l'image lue
numimage DW     (?)                    ; Numéro de l'image courante
nbimages DW     (?)                    ; Nombre d'images
run      DB     (?)                    ; Flag mode pas a pas
sens     DB     (?)                    ; Sens d'affichage (pour Ping-Pong)
intcnt   DB     (?)                    ; Compteur avant prochain tick valide
curspd   DB     (?)                    ; fréquence courante (en ticks)


viewvect DW     viewraw                ; Table de vecteurs pour viewimg
         DW     viewpcx

bufflg   DB     (?)


                                        ; MESSAGES d'erreur
notHP    DB     "This program runs omly on an HP palmtop",13,10,10,"$"
usage    DB     "usage : pcx95 [-l] [-d] file [file...]",13,10,10,"$"
notpcx   DB     "   Invalid file type",0
badvers  DB     "   Bad file version",0
invcomp  DB     "Unknown compression mode",0
filerr   DB     "   Error reading File",0
scan     DB     "   Scanning file...  ",0
endmsg   DB     "PLAYA95 version 1.01",13,10
         DB     "Written by Jacques Belin",13,10,10,"$"


                                        ; ICI COMMENCE LE CODE
main:    mov    ax,4dd4h                ; Exécution autorisée sur HP uniquement
         int    15h
         cmp    bx,04850h
         je     main_01
         mov    dx,offset notHP
         jmp    main_18
         ;-------------

main_01: mov    ah,4Ah                  ; On réjuste taille du bloc programme
         mov    bx,4096/16
         int    21h
         mov    sp,4096                 ; On replace le pointeur de pile

         call   parse                   ; Parsing dans argv[]

main_02: call   getopt                  ; ANALYSE DES OPTIONS
         cmp    al,EOF                  ; Fin d'analyse ?
         je     main_07                 ; Oui, on passe à la suite du prgm
main_03: cmp    al,"l"                  ; Si option "l"
         jne    main_04
         mov    byte ptr [loopflg],1    ; on arme le flag
         jmp    main_06                 ; ..et on continue l'analyse
main_04: cmp    al,"d"                  ; Si option "d"
         jne    main_05
         mov    byte ptr [nodisp],1     ; on arme le flag
         jmp    main_06                 ; ..et on continue l'analyse
```

```
main_05: cmp    al,"?"                  ; Si erreur, on arme le flag
         jne    main_06
         inc    byte ptr [errflg]
main_06: jmp    main_02                 ; On boucle tant qu'on a pas EOF
         ;-------------

main_07: cmp    byte ptr [errflg],0     ; Si une erreur a été détectée...
         je     main_09
main_08: mov    dx,OFFSET usage         ; ...on affiche le message d'erreur...
         jmp    main_18
         ;-------------

main_09: mov    ax,[optind]             ; argc == optind ? (pas de path ?)
         cmp    ax,[argc]
         je     main_08                 ; non, erreur.

         mov    [opti],ax               ; On initialise opti sur 1er param

                                        ; ALLOCATION DES BUFFERS
         push   cs                      ; On repositionne la DTA
         pop    ax
         mov    bx,offset dta
         call   setdta

         mov    ah,48h                  ; Allocation du buffer contenant entête
         mov    bx,(128/16)+1           ;  128 octets
         int    21h
         mov    [seghdr],ax

         mov    ah,48h                  ; Allocation buffer image non traitée
         mov    bx,((3840*2)/16)+1      ;  deux fois la taille image
         int    21h
         mov    [segbuff],ax

         mov    ah,48h                  ; Allocation tableau pointeurs/images
         mov    bx,((500*6)/16)+1       ;  500 elements  de 6 octets
         int    21h
         mov    [segimg],ax


         call   initgrph                ; Passage en mode graphique


                                        ; ANALYSE PARAM. & RECHERCHE FICHIERS
main_10: mov    ax,cs:[opti]            ; On repositionne optind sur 1er param
         mov    cs:[optind],ax

main_11: push   cs                      ; Préparation paramètre
         pop    ds
         mov    es,[psp]
         mov    bx,[optind]             ; bx = optind
         shl    bx,1
         mov    di,OFFSET argv          ; si = argv
         mov    di,[di+bx]              ; di ^ argv[optind]

         mov    si,offset path          ; Copie de l'argument dans "path"
         call   strcpy

         push   ds                      ; es=ds
         pop    es

         mov    dx,si                   ; cherche dernier caractère de la chaine
         call   strlen
         add    dx,cx
         dec    dx
         mov    bx,dx

main_12: cmp    si,bx                   ; cherche (en partant depuis la fin de
         ja     main_13                 ;   le chaine) un délimiteur ":" ou
         cmp    byte ptr [bx],"\"       ;   "\" ,séparant le path et le nom
         je     main_13                 ;   du fichier
         cmp    byte ptr [bx],":"
         je     main_13
         dec    bx
         jmp    main_12

main_13: inc    bx                      ; on pointe sur nom fichier, sans path
         mov    [nameptr],bx            ; on sauve ce pointeur

         mov    ah,4Eh                  ; on lance la recherche des fichiers
         mov    cx,FA_ARCH              ;   correspondant au paramètre (ayant
         mov    dx,offset path          ;   peut-être des "wildcards"
         int    21h
main_14: jc     main_15                 ; si carry, pas trouvé -> param suivamt

         mov    di,offset dta           ; Copie du nom du fichier trouvé
         add    di,1Eh                  ;  (situé à l'offset 1E de la DTA)
         mov    si,[nameptr]            ;   a la fin du path du paramètre
         call   strcpy

         mov    dx,offset path          ; Traitement du fichier graphique
         call   viewfil
         jc     main_17

         mov    ah,4Fh                  ; Cherche fichier suivant
         int    21h
         jmp    main_14                 ; et on boucle
         ;-------------

main_15: cmp    word ptr [opti],0       ; Y a t-il des paramètres ?
         je     main_17                 ; Non, on va plus bas
         inc    word ptr cs:[optind]    ; Sinon, on jncrémente optind...
         mov    ax,[optind]             ; et on boucle tant qu'il y a des params
         cmp    [argc],ax
         je     main_16
         jmp    main_11

main_16: cmp    byte ptr [loopflg],0    ; Si option '-l' non active..
         je     main_17                 ; ...on termine le programme
         jmp    main_10                 ; ...sinon, on recommence

main_17: call   exitgrph                ; On repasse en mode texte

         mov    es,[segimg]             ; Libération du buffer
         mov    ah,49h
         int    21h

         mov    es,[segbuff]            ; Libération du buffer
         mov    ah,49h
         int    21h

         mov    es,[seghdr]             ; Libération du buffer
         mov    ah,49h
         int    21h

         mov    dx,offset endmsg        ; on affichera un message de fin

main_18: mov    ah,09h                  ; Affichage message
         int    21h
         mov    ah,4ch                  ; Retour au DOS
         int    21h

;*************************************************
; initgrph : passe en mode graphique
; entrées : aucune
; sortie : aucune
;*************************************************
initgrph PROC   NEAR
         push   ax
         mov    ax,00A0h
         int    5Fh
         pop    ax
         ret
initgrph ENDP

;*************************************************
; exitgrph : sort du mode graphique
; entrées : aucune
; sortie : aucune
;*************************************************
exitgrph PROC   NEAR
         push   ax
         mov    ax,0087h
         int    5Fh
         pop    ax
         ret
exitgrph ENDP
```

```
;*****************************************************
; VIEWFIL : affiche un fichier graphique
; entrées : DX:DX Nom du fichier
;           (terminée par caractère 0)
; Sortie : si carry = 1 -> touche ESC pressée
; NOTE : Le mode graphique doit être activé
;*****************************************************
Viewfil  PROC   NEAR
         push   ax
         push   bx
         push   cx
         push   dx
         push   es
         push   di
         push   ds
         push   si

         mov    byte ptr [escape],0     ; Effacement du flag Escape

         mov    ax,3d02h                ; Ouverture du fichier
         mov    dx,offset path
         int    21h
         jnc    view_01
         jmp    view_35                 ; Si carry -> erreur lecture

view_01: mov    bx,ax
         mov    [handle],bx             ; On sauve le handle du fichier

         push   ds                      ; On sauve DS

         mov    ah,3Fh                  ; Lecture entete du fichier
         mov    cx,128
         mov    ds,[seghdr]
         xor    dx,dx
         int    21h
         jnc    view_02
         jmp    view_35                 ; Si carry -> erreur lecture

view_02: push   ds                      ; ES pointe sur l'entete
         pop    es

         pop    ds                      ; On récupère l'ancien DS

         cmp    word ptr es:[AMAGIC],MAGIC  ; Est-ce Un fichier A95 ?
         je     view_08                 ; Oui, on va plus bas

                                        ; AFFICHAGE D'UN FICHIER PCX
         cmp    byte ptr es:[PMANUF],0ah    ; controle validité fichier PCX
         jne    view_03
         cmp    byte ptr es:[PRLE],1
         jne    view_03
         cmp    byte ptr es:[PBITPX],1
         jne    view_03
         cmp    word ptr es:[PRIGHT],239
         jne    view_03
         cmp    word ptr es:[PBOTTOM],127
         je     view_04
view_03: mov    dx,offset notpcx        ; Si erreur, sortie avec message
         jmp    view_36
         ;-------------

view_04: push   ds
         mov    ah,3Fh                  ; Lecture fichier dans buffer
         mov    cx,9999
         mov    ds,[segbuff]
         xor    dx,dx
         int    21h
         pop    ds
         jnc    view_05
         jmp    view_35                 ; Si carry -> erreur lecture
         ;-------------

view_05: call   viewpcx                 ; Affichage du fichier

         mov    ah,0
         int    16h                     ; Lecture du clavier
         cmp    al,0                    ; Code étendu ?
         je     view_06
         and    ax,5fh                  ; On efface le scan code .
view_06: cmp    ax,K_ESC                ; Touche ESC ?
         jne    view_07                 ;    Non, on va plus bas
         mov    byte ptr [escape],1     ; On arme le flag
view_07: jmp    view_38                 ;
         ;-------------

view_08:                                ; TRAITEMENT FICHIER A95
         cmp    byte ptr es:[AVERS],1h  ; Controle version fichier A95
         je     view_09
         mov    dx,offset badvers
         jmp    view_36
         ;-------------

view_09: cmp    byte ptr es:[ACOMP],1   ; Controle mode de compression maxi
         jbe    view_10
         mov    dx,offset invcomp
         jmp    view_36
         ;-------------

                                        ; STOCKAGE POSITION IMAGES DANS TABLEAU
view_10: mov    word ptr [numimage],1   ; Initialisation numéro d'image courante
         mov    dx,es:[ANIMAGES]        ; Lecture du nombre d'images
         mov    [nbimages],dx

         push   es

         mov    es,[segimg]             ; ES:DI = segimg:0
         xor    di,di

         cmp    byte ptr [nodisp],1     ; si option '-d' activée...
         je     view_11                 ;    ...on n'affiche pas ce message
         mov    dx,offset scan          ; Affichage message d'attente
         call   dispmsg

view_11: mov    ax,4200h                ; Positionnement sur 1er mot longueur
         mov    dx,128
         xor    cx,cx
         int    21h
         jnc    view_12
         jmp    view_35                 ; Si carry -> erreur lecture

view_12: add    ax,2                    ; Stockage position dans tableau
         stosw
         mov    ax,dx
         stosw
         mov    ah,3fh                  ; Lecture longueur
         mov    cx,2
         mov    dx,offset imsize        ; (stockage dans 'imsize')
         int    21h
         jnc    view_13
         jmp    view_35                 ; Si carry -> erreur lecture

view_13: mov    ax,[imsize]
         stosw                          ; Stockage longueur dans tableau

         mov    dx,[numimage]           ; nbimages==numimage ?
         cmp    dx,[nbimages]
         je     view_14                 ;    ; oui, on sort de la boucle
         inc    word ptr [numimage]     ; Incrémentation numéro d'image courant

         add    ax,2                    ; On ajoute 2 (taille éléments
         mov    dx,ax                   ;  indiquant taille image) à imsize
         xor    cx,cx
         mov    ax,4201h                ; On pointe sur l'entête de l'image
         int    21h                     ;  suivante
         jmp    view_12                 ; et on boucle.
         ;-------------

view_14: pop    es
         mov    word ptr [numimage],0   ; réinitialisation des valeur de départ
         mov    byte ptr [run],1
         mov    byte ptr [sens],1
         mov    byte ptr [bufflg],0
         mov    al,es:[ASPEED]
         mov    [curspd],al
         mov    [intcnt],al

         call   initint                 ; init interruption 1Ch

                                        ; BOUCLE PRINCIPALE
view_15: cmp    byte ptr [bufflg],1     ; Si le buffer est plein...
         je     view_17                 ;    ...on va plus bas
         cmp    byte ptr es:[AANIM],ONETIME ; si c'est pas une animation ONETIME
         jne    view_16                 ;    ...on va plus bas
         mov    ax,[numimage]           ; si la dernière image a été affichée...
         cmp    ax,[nbimages]
         jne    view_16
         jmp    view_37                 ;    ...on passe au fichier suivant.
         ;-------------
```

```
view_16: call   nextim                  ; .. on pointe sur l'image suivante
         call   readim                  ; .. et on charge l'image.
         mov    byte ptr [bufflg],1     ; Le buffer est maintenant plein

view_17: mov    ah,11h                  ; Lecture si touche prete
         int    16h
         jz     view_15                 ; si pas de touche, on boucle

         mov    ah,10h                  ; lecture du code de la touche
         int    16h
         cmp    al,0                    ; on efface le scan code + on normalise
         je     view_18                 ;   en MAJuscules.
         and    ax,7fh
         cmp    al,60h
         jb     view_18
         sub    al,20h
view_18: cmp    ax,K_E                  ; touche 'E' pressée
         je     view_19
         cmp    ax,K_ESC                ; si pas touche ESC...
         jne    view_20                 ;   ..on continue plus bas
         mov    byte ptr [escape],1     ; on arme 'escape' (sortie définitive)
view_19: jmp    view_37                 ; fin du traitement du fichier
         ;-------------

view_20: cmp    byte ptr [run],1        ; si mode "run"..
         jne    view_21                 ;   .. on va plus bas
         jmp    view_28                 ; saut long (cause distance)
         ;---------

view_21: cmp    ax,K_R                  ; si touche 'R' pressée...
         jne    view_22
         mov    byte ptr [run],1        ; on repasse en mode "run"
         jmp    view_15
         ;------------

view_22: cmp    ax,K_RIGHT              ; si touche '->' pressée...
         jne    view_23
         call   nextim                  ; ...on affiche la prochaine image
         call   readim
         call   viewim
         jmp    view_15
         ;------------

view_23: cmp    ax,K_LEFT               ; si touche '<-' pressée...
         jne    view_24
         call   previm                  ; ...on affiche l'image précédente
         call   readim
         call   viewim
         jmp    view_15
         ;------------

view_24: cmp    ax,K_F                  ; si touche 'F' pressée...
         jne    view_25
         mov    byte ptr [sens],1       ; ...on affiche la première image
         mov    word ptr [numimage],1
         call   readim
         call   viewim
         jmp    view_15
         ;------------

view_25: cmp    ax,K_L                  ; si touche 'L' pressée...
         jne    view_27
         mov    ax,[nbimages]           ; ...on affiche la dernière image
         mov    [numimage],ax
         call   readim
         call   viewim
         jmp    view_15
         ;------------

view_27: jmp    view_15                 ; on boucle.
         ;-----------

                                        ; MODE "RUN"
view_28: cmp    ax,K_S                  ; si touche 'S' pressée...
         jne    view_29
         mov    byte ptr [run],0        ; ...on passe en mode "pas-a-pas"
         call   viewim
         jmp    view_15
         ;-----------

view_29: cmp    ax,K_PLUS               ; si touche '+' pressée...
         jne    view_31
         cmp    byte ptr [curspd],1     ; ...on augmente la vitesse
         je     view_30                 ;   (en fait, on diminue le nombre de
         dec    byte ptr [curspd]       ;   boucles d'attente)
view_30: jmp    view_15
         ;-----------

view_31: cmp    ax,K_MINUS              ; si touche '-' pressée...
         jne    view_33
         cmp    byte ptr [curspd],255   ; ...on diminue la vitesse
         je     view_32                 ;   (en fait, on augmente le nombre de
         inc    byte ptr [curspd]       ;   boucles d'attente)
view_32: jmp    view_15
         ;-----------

view_33: cmp    ax,K_MULT               ; si touche '*' pressée...
         jne    view_34
         mov    al,es:[ASPEED]          ; ...on restitue la vitesse originelle
         mov    [curspd],al
         jmp    view_15
         ;-----------

view_34: jmp    view_15                 ; on boucle.
         ;-----------

view_35: mov    dx,offset filerr        ; affichage message d'erreur fichier
view_36: call   dispmsg                 ; affichage message d'erreur
         mov    ah,0h                   ;   avec attente de pression de touche
         int    16h
         jmp    view_38

view_37: call   restint                 ; désinstallation gestionnaire INT 1C
view_38: mov    bx,[handle]             ; Fermeture du fichier
         mov    ah,3eh
         int    21h

         cmp    byte ptr [escape],0     ; si touche ESC
         je     view_39
         stc                            ; on arme le Carry
view_39: pop    si
         pop    ds
         pop    di
         pop    es
         pop    dx
         pop    cx
         pop    bx
         pop    ax
         ret
viewfil  ENDP

;*************************************************
; intvect : gestionnaire d'interruption
; entrées : aucune
; sorties : aucune
; NOTE : ROUTINE D'INTERRUPTION, NE PAS APPELER !
;*************************************************
intvect  PROC   FAR
         push   ax

         mov    al,0h                   ; reinitialisation du compteur du 8253
         out    43h,al                  ;  en mode 0 (cause bug Hopper)
         jmp    $+2                     ; On attend un peu que le 8253 ait fini
         jmp    $+2
         mov    ax,cs:[PICval]          ; On remet la valeur de comptage
         out    40h,al
         jmp    $+2
         jmp    $+2
         mov    al,ah
         out    40h,al

         dec    cs:[intcnt]             ; on décrémente le compteur "soft"
         jnz    intve_2                 ; si different de 0, on ignore l'int

         mov    al,cs:[curspd]          ; reinitialisation du compteur Soft
         mov    cs:[intcnt],al

         cmp    byte ptr cs:[run],0     ; si mode pas a pas...
         je     intve_2                 ; ... on ignore l'interruption

         cmp    byte ptr cs:[bufflg],0  ; si le buffer est vide...
         je     intve_2                 ; ... on ignore l'interruption

         push   bx                      ; sauvegarde des registres
         push   cx
         push   dx
         push   es
```

```
        push    di
        push    ds
        push    si
        push    bp

        push    cs                      ; DS = CS  (=segment données prgm)
        pop     ds

        mov     byte ptr [run],0        ; on désarme le flag, pour éviter une
                                        ;  réentrance.

        mov     es,[seghdr]             ; ES = segment entete
        call    viewim                  ; Affichage image

        mov     byte ptr cs:[buffflg],0 ; le buffer est maintenant virtuellement
                                        ;    vidé.

        mov     byte ptr [run],1        ; on remet le flag en place

intve_1: pop    bp                      ; on restitue les registres et on sort
        pop     si
        pop     ds
        pop     di
        pop     es
        pop     dx
        pop     cx
        pop     bx
intve_2: pop    ax
        iret
intvect ENDP

;*****************************************************
; initint : initialise l'interruption timer
; entrées : aucune
; sorties : aucune
;*****************************************************
initint PROC    NEAR
        push    ax
        push    bx
        push    dx
        push    ds
        push    es

        mov     ax,40h                  ; sauvegarde anciene valeur timeout
        mov     es,ax
        mov     bx,es:[0A3h]
        mov     [oldtime],bx

        mov     ah,46h                  ; modification valeur timeout pour
        mov     bx,0CCC0h;              ;   corriger le changement de la
        int     15h                     ;   fréquence d'horloge.

                                        ; Acceleration de la fréquence du timer
        mov     ax,4dd4h                ; Suivant la machine (HP95 ou HP100)
        int     15h                     ;  on choisit la valeur du comptage
        cmp     cl,2                    ;  (cause Bug du 'Hopper')
        je      initi_1
        mov     word ptr [PICval],0E24h ; valeur de comptage pour HP95
        jmp     initi_2
;-------------
initi_1: mov    word ptr [PICval],0FFCh ; valeur de comptage pour HP100
initi_2: mov    al,34h
        out     43h,al
        jmp     $+2                     ; On attend un peu que le 8253 ait fini
        jmp     $+2
        mov     ax,01000h               ; Nouvelle fréquence =  18.2 hz * 16
        out     40h,al
        jmp     $+2                     ; Quelques instructions nulles, afin de
        jmp     $+2                     ;  perdre du temps (synchronisation 8253)
        mov     al,ah
        out     40h,al


        mov     ax,351Ch                ; sauvegarde ancien vecteur INT 1C.
        int     21h
        mov     [oldintsg],es
        mov     [oldintof],bx

        mov     dx,offset intvect       ; initialisation nouveau vecteur
        mov     ax,251Ch
        int     21h

        pop     es
        pop     ds
        pop     dx
        pop     bx
        pop     ax
        ret
initint ENDP

;*****************************************************
; resttint : ré initialise l'interruption timer
; entrées : aucune
; sorties : aucune
;*****************************************************
restint PROC    NEAR
        push    ax
        push    bx
        push    dx
        push    ds

        push    ds
        mov     dx,[oldintof]           ; reinitialisation ancien vecteur
        mov     ds,[oldintsg]
        mov     ax,251Ch
        int     21h
        pop     ds

        mov     ax,4dd4h                ; reinitialisation timer à 18.2 hz
        int     15h
        cmp     cl,2
        je      resti_1
        mov     al,30h
        jmp     resti_2
resti_1: mov    al,36h
resti_2: out    43h,al
        jmp     $+2
        jmp     $+2
        mov     ax,0h
        out     40h,al
        jmp     $+2
        jmp     $+2
        mov     al,ah
        out     40h,al

        mov     ah,4bh                  ; resynchronisation Horloge RTC -> timer
        int     15h

        mov     ah,46h                  ; rechargement valeur timeout originale
        mov     bx,[oldtime]
        int     15h

        pop     ds
        pop     dx
        pop     bx
        pop     ax
        ret
restint ENDP


;*****************************************************
; READIM  : Lit une image
; entrées : numimage (globale)
; sorties : données image (compressée) dans segbuf:0
;*****************************************************
readim  PROC    NEAR
        push    ds
        mov     bx,[handle]             ; récupération handle du fichier

        mov     ax,[numimage]           ; ---> DS:SI = tableau[numimage]
        dec     ax
        mov     cx,6
        mul     cx
        mov     si,ax
        mov     ds,[segimg]

        lodsw                           ; lecture position image dans fichier
        mov     dx,ax
        lodsw
        mov     cx,ax
        mov     ax,4200h                ; positionnement dans fichier
        int     21h

        lodsw                           ; lecture taille image dans tableau
        mov     cx,ax
```

```
        mov     ah,3Fh                  ; lecture image
        mov     ds,cs:[segbuff]
        xor     dx,dx
        int     21h

        pop     ds
        ret
readim  ENDP


;*****************************************************
; VIEWIM  : affiche une image
; entrées : données à afficher en segbuff:0
; NOTE : Le mode graphique doit être activé
;*****************************************************
viewim  PROC    NEAR
        push    bx
        xor     bh,bh                   ; calcul offset dans table des vecteurs
        mov     bl,es:[ACOMP]
        shl     bx,1
        call    [bx+offset viewvect]    ; appel routine
        pop     bx
        ret
viewim  ENDP


;*****************************************************
; VIEWPCX : affiche une image PCX
; entrées : données à afficher en segbuff:0
; NOTE : Le mode graphique doit être activé
;*****************************************************
viewpcx PROC    NEAR
        push    es
        push    di
        push    ds
        push    si
        push    cx
        push    dx

        mov     ds,[segbuff]            ; DS:SI pointe sur le buffer compresse
        xor     si,si

        mov     cx,0b000h               ; ES:DI pointe sur la RAM Vidéo
        mov     es,cx
        xor     di,di

        xor     ch,ch                   ; On efface CH (CX utilisé par 'REP')

viewp_1: lodsb                          ; on charge l'octet à décoder dans AL
        mov     dl,al                   ; sauvegarde dans DL
        and     dl,0C0h                 ; Si AL = 11xxxxxxh (code RLE)...
        cmp     dl,0C0h
        je      viewp_2                 ;  ...alors plusieurs octets à dupliquer
        stosb                           ; on recopie dans la ram vidéo
        jmp     viewp_3
        ;-------------

viewp_2: mov    cl,al
        and     cl,3fh                  ; CX = nombre d'octets à dupliquer
        lodsb                           ; AL = octet à dupliquer
        rep     stosb                   ; on copie dans la ram vidéo

viewp_3: cmp    di,3840                 ; si fin de la mémoire vidéo non atteinte
        jb      viewp_1                 ;  ..on boucle

        pop     dx
        pop     cx
        pop     si
        pop     ds
        pop     di
        pop     es
        ret
viewpcx ENDP


;*****************************************************
; VIEWRAW : affiche une image non compressée
; entrées : données à afficher en segbuff:0
; NOTE : Le mode graphique doit être activé
;*****************************************************
viewraw PROC    NEAR
        push    es
        push    di
        push    ds
        push    si
        push    cx
        push    dx

        mov     ds,[segbuff]            ; DS:SI pointe sur le buffer
        xor     si,si

        mov     cx,0b000h               ; ES:DI pointe sur la RAM Vidéo
        mov     es,cx
        xor     di,di

        mov     cx,3840                 ; on recopie les 3840 octets en
        rep     movsb                   ;  ram vidéo

        pop     dx
        pop     cx
        pop     si
        pop     ds
        pop     di
        pop     es
        ret
viewraw ENDP


;*****************************************************
; NEXTIM  : détermine le numéro de l'image suivante
; entrées : aucune
; valeur retournée : aucune
;*****************************************************
nextim  PROC    NEAR
        push    ax

        cmp     byte ptr es:[AANIM],PINGPONG ; si animation 'Ping-Pong'..
        je      nexti_2                 ; ..on va plus bas

        inc     word ptr [numimage]     ; on incrémente le numéro d'image
        mov     ax,[numimage]           ; si on n'est pas arrivé à la dernière..
        cmp     ax,[nbimages]
        jng     nexti_4                 ;   ...on se dirige vers la sortie

        mov     word ptr [numimage],1   ; sinon, on revient à la première
nexti_1: jmp    nexti_4                 ; on se dirige vers la sortie
        ;-------------
                                        ; Cas de l'animation PING-PONG
nexti_2: cmp    byte ptr [sens],0       ; Si sens courant 'inversé'..
        je      nexti_3                 ;   ..on va plus bas
        inc     word ptr [numimage]     ; incrémentation numéro image
        mov     ax,[numimage]           ; si on n'est pas arrivé à la dernière..
        cmp     ax,[nbimages]
        jng     nexti_4                 ;   ...on se dirige vers la sortie

        mov     ax,[nbimages]           ; sinon, on revient l'avant dérnière
        dec     ax
        mov     [numimage],ax
        mov     byte ptr [sens],0       ; et on change de sens de défilement
        jmp     nexti_4                 ;   ...on se dirige vers la sortie
        ;-------------

                                        ; Sens Ping-Pong 'retour'
nexti_3: dec    word ptr [numimage]     ; Décrémentation numéro d'image
        jnz     nexti_4                 ; si ce n'était pas la première image
                                        ;   qui était affichée, on sort
        mov     word ptr [numimage],2   ; .sinon, on repasse à la deuxième image
        mov     byte ptr [sens],1       ; ..et on change de sens
nexti_4: pop    ax
        ret
nextim  ENDP


;*****************************************************
; PREVIM  : détermine le numéro de l'image précédente
; entrées : aucune
; valeur retournée : aucune
;*****************************************************
previm  PROC    NEAR
        push    ax

        cmp     byte ptr es:[AANIM],PINGPONG ; si animation 'Ping-Pong'...
        je      previ_2                 ; ...on va plus bas

        cmp     word ptr [numimage],1   ; si on est pas sur la première image..
        jne     previ_1                 ; ..on va plus bas
        mov     ax,[nbimages]           ; ..sinon, numimage=nbimages
```

```
        mov     [numimage],ax
        jmp     previ_4                 ; on se dirige vers la sortie
        ;-------------

previ_1: dec    word ptr [numimage]     ; on décrémente le numéro d'image
        jmp     previ_4                 ; on se dirige vers la sortie
        ;-----------

                                        ; Cas de l'animation PING-PONG
previ_2: cmp    byte ptr [sens],1       ; Si sens courant 'normal'..
        je      previ_3                 ;   ..on va plus bas
        inc     word ptr [numimage]     ; incrémentation numéro image
        mov     ax,[numimage]           ; si on n'est pas arrivé à la dernière..
        cmp     ax,[nbimages]
        jng     previ_4                 ;   ...on se dirige vers la sortie

        mov     ax,[nbimages]           ; sinon, on revient l'avant dérnière
        dec     ax
        mov     [numimage],ax
        mov     byte ptr [sens],1       ; et on change de sens de défilement
        jmp     previ_4                 ;   ...on se dirige vers la sortie
        ;-------------

                                        ; Sens 'Ping-Pong 'aller'
previ_3: dec    word ptr [numimage]     ; Décrémentation numéro d'image
        jnz     previ_4                 ; si ce n'était pas la première image
                                        ;   qui était affichée, on sort
        mov     word ptr [numimage],2   ; .sinon, on repasse à la deuxième image
        mov     byte ptr [sens],0       ; ..et on change de sens
previ_4: pop    ax
        ret
previm  ENDP

;*************************************************
; READLEN : lit la longueur d'une image
; entrées : bx = handle du fichier
;             (pointant sur données à lire)
; Sortie : CX = taille en octets
;*************************************************
readlen PROC    NEAR
        push    ax
        push    bx
        push    dx

        mov     ah,3Fh                  ; lecture taille image -> 'imsize'
        mov     cx,2
        mov     dx,offset imsize
        int     21h

        mov     cx,[imsize]             ; transfert contenu 'imsize' dans CX

        pop     dx
        pop     bx
        pop     ax
        ret
readlen ENDP

;*************************************************
; DISPMSG: affiche un message au centre de l'écran graphique
; entrées : DX = offset chaine message
; NOTE : Le mode graphique doit être activé
;*************************************************
dispmsg PROC    NEAR
        push    ax
        push    bx
        push    cx
        push    dx
        push    es
        push    di

        push    dx

        mov     ah,08h                  ; positionnement curseur
        mov     cx,42
        mov     dx,59
        int     5fh

        mov     ax,0900h                ; selection couleur blanche
        int     5fh

        mov     ax,0501h                ; affichage rectangle blanc
        mov     cx,188
        mov     dx,68
        int     5fh

        mov     ax,0901h                ; selection couleur noire
        int     5fh

        mov     ax,0F00h                ; affichage du texte
        mov     al,0
        mov     cx,48
        mov     dx,60
        push    cs
        pop     es
        pop     di
        int     5fh

        pop     di
        pop     es
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
dispmsg ENDP

;*************************************************
; SETDTA  : Positionne l'adresse de la DTA
; entrées : AX:BX = adresse DTA
; sortie  : aucune
;*************************************************
setdta  PROC    NEAR
        push    ax
        push    bx
        push    dx
        push    ds

        mov     ds,ax                   ; on positionn les paramètres
        mov     dx,bx
        mov     ah,1Ah                  ; selection de la DTA
        int     21h

        pop     ds
        pop     dx
        pop     bx
        pop     ax
        ret
setdta  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 différent suivant le programme d'entrée et la machine de destination. Chaque listing est prévu pour être entré sur sa machine de destination. Par exemple, ne tentez pas d'entrer un programme HP48 dans un fichier MS-DOS à l'aide du programme MAKEDOS. Vous obtiendriez un fichier intransférable dans la HP48.

Programmes HP48 et HP28S

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

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

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

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

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

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

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

```
BINO                          (HP48)
# 87E6h                  37.5 octets

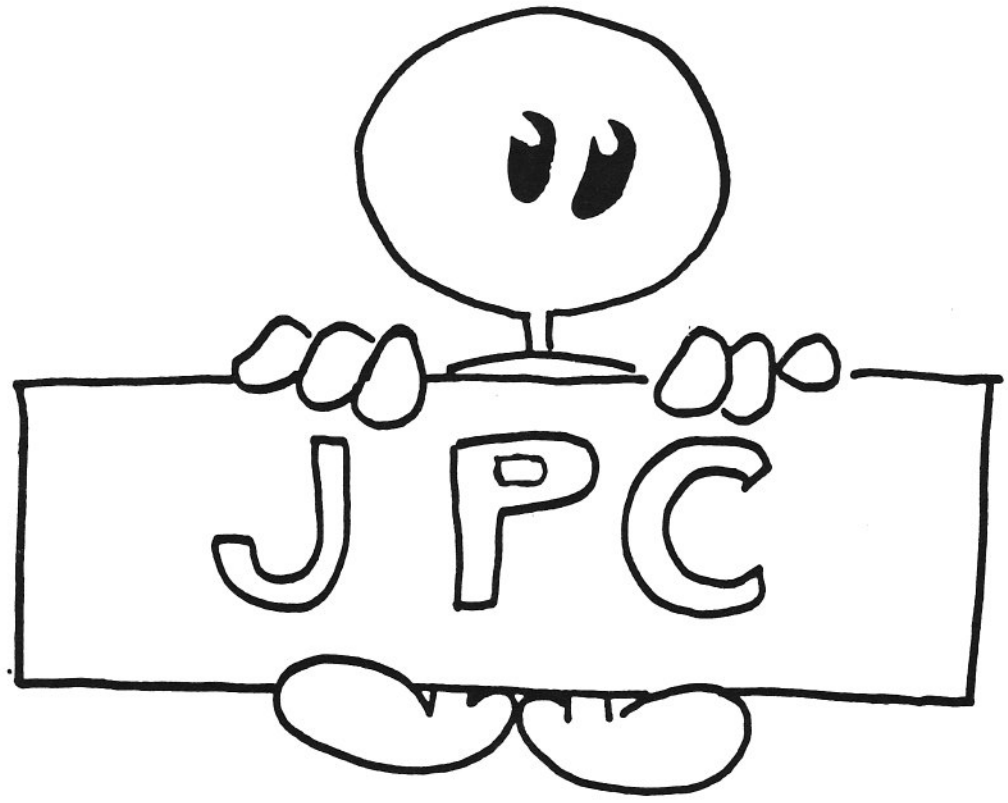
     0123 4567 89AB CDEF    sm

000: D9D2 02BA 81D2 B368    E30
001: 8130 AEC8 1FED 303C    D9B
002: 3701 1436 FBD8 126E    C1B
003: A2D4 436A EC81 FED3    C53
004: 0954 50B2 130          9B2


STRIP                         (HP48)
# 024Bh                  312 octets

     0123 4567 89AB CDEF    sm

000: D9D2 079E 60D9 D20A    EA9
001: D2A1 B3A1 6F6A 367F    DC7
002: 8169 E550 D123 67F8    BA3
003: 16DB 236F A450 1693    8BA
004: 6936 32F2 E36D 9D20    6F0
005: 8813 03F2 1662 726E    413
006: 4E30 B213 0F74 3652    08E
007: 330D 9D20 D9D2 0D9D    FFB
008: 2079 E60E 25A1 3391    CDE
009: 6D9D 202C 2304 4126    8C7
00A: DA91 6D9D 20E4 116B    6CF
00B: 2130 4423 0300 40C9    2AD
00C: B26B 2130 5293 63CE    06F
00D: 22E2 B30B 4916 5293    D1D
00E: 6D87 32E2 B30B 4916    A76
00F: 5293 6DE0 32E2 B30B    8B1
010: 4916 5293 6976 32E2    5A1
011: B30B 4916 79E6 05DF    49C
012: 2233 916D 9D20 F281    225
013: 57A2 26B4 9167 BF60    028
014: B213 07BF 60B2 1301    C80
015: 2270 E0E3 0072 7094    8B6
016: 270E 0E30 5927 07BF    6EE
017: 60B2 1307 9E60 E163    40F
018: 2AB8 16DB 2367 9E60    209
019: C943 2AB8 16D9 D207    011
01A: 9E60 79E6 0B21 3079    CE8
01B: E60E F532 AB81 6D9D    C6C
01C: 2079 E607 9470 B213    8F7
01D: 0112 2639 9168 4E20    584
01E: 5035 4525 9405 7E12    1F3
01F: 6CB9 1684 E205 0354    E3C
020: 5259 405B 2130 9427    A37
021: 0433 7054 450B 2130    5BD
022: F9F6 0D48 812B F811    380
023: 2040 D9D2 0F10 70E8    141
024: F607 E126 C2A1 6119    E2D
025: 2047 A203 BA50 B213    B18
026: 0F30 40E8 E60B 2130    7DA
```



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.

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