JPC

MARS 1993
NUMERO 84

Le numéro 40 F

A PROPOS DU CLUB

HP-28

HP-48

HP-95

EDITORIAL

Tout d'abord, excusez nous pour le retard survenu pour la parution de ce numéro du journal. Sachez seulement qu'il est dû uniquement au fait que certains articles des auteurs habituels n'ont pas pu être prêts à temps, fautes de problèmes de dernière minute. N'ayant plus de stock d'articles nous permettant de modifier au dernier moment le contenu du journal, ce retard devenait inévitable. Bien que nous vous ayons alerté maintes fois sur le fait que nous ne pouvions pas assurer à nous seuls le contenu du journal, nous n'avons pas été suivis. Il est fort regrettable de penser que les prochains journaux auront sûrement un retard équivalent, à moins que de nombreux articles arrivent dès les prochains jours. De toutes façons nous ne pourrons plus assurer des articles aussi longs qu'auparavant, car nous avons tous en projet des réalisations qui nous demanderons de beaucoup de travail, et qui paradoxalement ne pourrons sûrement pas être publiés, car trop importants. En attendant de voir les résultats, il faudra que vous occupiez le terrain !

Le grand évenement de ce mois-ci concerne le retour en force de la rubrique HP28. Nous ne pouvons qu'être satisfaits de ce fait. Qui a dit que cette machine était morte ?

Un autre point qui nous satisfait pleinement concerne l'arrivée de nouveaux auteurs dans les colonnes de ce *JPC*. Il s'agit d'utilisateurs dont l'adhésion est relativement récente, ce qui montre qu'il n'est pas nécessaire d'avoir de longues années d'experience pour faire des articles intéressants !

Vous constaterez aussi que dans ce numéro, les pages d'informations générales sont inexistantes. Ne vous inquiétez pas, elles reviendront en force le mois prochain, dès que nous aurons vérifié les nombreuses (et surprenantes) informations que nous avons entendu ces jours ci...

Pour l'instant, nous vous souhaitons une Heureuse Programmation...

Le bureau


Ne vous battez plus avec votre calculateur: ACHETEZ un HP!
Make love, not war!
Donnez-moi mon JPC sinon je freppe!

HP-28

TRAITEMENT DE DONNEES STATISTIQUES

L'objet de cet article est de décrire un ensemble de petits programmes de traitement numérique et graphique de matrices de points en dimension 2 pour HP28S.

Mode d'emploi:

Le programme principal s'appelle 'GO' (dans GRAPH/STAT). Avant de le lancer, créer une matrice de points (dans le plan 2D). Vous la stockerez dans le répertoire GRAPH/STAT/Σ, pour la préserver de ce qui suit.

La syntaxe que j'ai adopté devant chaque nom d'objet est la même que celle présentée par L. Grand dans le n°82.

Le programme GO va vous permettre de choisir l'échelle de concentration affine de votre matrice de points pour ensuite visualiser le nuage de points sous la forme de points ou de segments.

Les programmes XY→CM et CM→XY permettent de connaître la position d'un point d'une échelle réelle à une échelle fixée (en centimètre par exemple, s'il s'agit de tracer des points d'une mesure physique sur une feuille de papier millimétré). Les données de calcul d'échelle sont données dans GO ou DEFXY.

Le programme XY→AR est identique à XY→CM, mais renvoie le résultat en format complexe, en sommant le point dans ΣDAT si elle existe.

Les programmes AC→AC et AC→AX transforment une matrice entière de points d'une échelle de points à l'autre.

DEFXY permet la saisie des paramètres de calculs d'échelle :

- XMIN : Valeur minorant l'ensemble des abscisses de points de la matrice réelle.
- YMIN : Valeur minorant l'ensemble des ordonnées de points de la matrice réelle.
- XMAX : Valeur majorant l'ensemble des abscisses de points de la matrice réelle.
- YMAX : Valeur majorant l'ensemble des ordonnées de points de la matrice réelle.
- CMX : Largeur en centimètres de votre feuille de papier millimétré (si OCMX=0).
- CMY : Hauteur en centimètres de votre feuille de papier millimétré (si OCMY=0).
- OCMX : Offset de CMX (mis souvent à 0).
- OCMY : Offset de CMY (mis souvent à 0).

Pour sortir du menu (sachant que des valeurs par défaut sont calculées), tapez [shift][1].

LCONCA : Programme permettant la juxtaposition des *j* matrices de *ni* points en 1 matrice de *n1+...+nj* points.

BORNS : Sauvegarde la matrice (ou liste de matrices) dans la matrice ΣDAT, et calcule les valeurs XMIN, YMIN, XMAX, YMAX par défaut.

CONCA : Comme LCONCA, mais uniquement à partir de 2 matrices et non d'une liste de matrices.

DRLIN : Le programme après traçage des points est en attente d'un appui sur une des touches :
- [E] : Effacer cet ensemble de points.
- [S] : Tracé segment par segment entre points en utilisant LINE4 du répertoire GRAPH.
- [L] : Comme [S] ou l'appui de tout autre touche pour continuer.

DRAW : Trace l'ensemble des points de la matrice.

A→LXY : convertit le format 'matrice de points' en format 'liste de points'.

Mon programme utilise LINE4 qui est une adaptation quasi-identique de LINE du livre de J.M. Ferrard sur la HP28S. L'affichage n'étant pas rapide, existe-t-il une autre version plus performante (en language machine par exemple) ?

Liste des programmes:

Dans le répertoire HOME

WAITK (→)
```
« DO  UNTIL KEY  END  »
```

BEEP2 (→)
```
« 1 2
  FOR i i 220 * .1 BEEP
  NEXT
»
```

LSTEX (3:Liste d'objets à exécuter
2:Dimension liste
1:Objet quelconque →)
```
« → lst nl prg
  «
    1 nl
    FOR i lst i GET prg
      IFERR EVAL
      THEN
```

```
      DROP
    END
  NEXT
 »
»
```

Dans le répertoire GRAPH

LINE4 (4:Réel 3:Réel 2:Réel 1:Réel →)

```
« → x1 y1 x2 y2
 «
  x1 y1 R→C x2 y2 R→C → p1 p2
  «
   IF p1 p2 ==
   THEN
    p1 PIXEL
   ELSE
    p1 p2 - C→R ABS SWAP ABS > → flag
    «
     x2 y2 x1 y1 { 2 2 } →ARRY  flag
     «
      [[ 0 1 ]
       [ 1 0 ]]
      *
     »
     IFT STOΣ LR  DROP2 PPAR 2 GET PPAR 1
     GET - flag  « IM 32 » « RE 137 »
     IFTE / → pas
     «
      ΣDAT [ 1 0 ] * ARRY→ DROP SWAP DUP2
      IF >
      THEN
       pas NEG 'pas' STO
      END
      FOR v v DUP PREDV
       flag « SWAP » IFT R→C PIXEL
       pas
      STEP
     »
    »
   END
  »
 »
»
```

PPAR

```
{ (0,0) (136,31) X 1 (0,0) }
```

Dans le sous-sous répertoire GRAPH/STAT

XY→CM (2:Abscisse Réelle
1:Ordonnée Réelle →
2:Abscisse en cm
1:Ordonnée en cm)

```
« → x y
 «
  2 SCI x OFFEX - EX * OCMX + RND y OFFEY - EY *
  OCMY + RND STD
 »
»
```

CM→XY (2:Abscisse en cm
1:Ordonnée en cm →
2:Abscisse réelle
1:Ordonnée réelle)

```
« → cx cy
 «
  2 SCI cx OCMX + EX / OFFEX + RND cy OCMY
  + EY / OFFEY + RND STD
 »
»
```

XY→AR (2:Abscisse Réelle
1:Ordonnée Réelle →
1:(Abscisse en cm, 1:Ordonnée en cm))

```
« XY→CM DUP2 R→C 3 ROLLD 2 →ARRY Σ+ »
```

AX→AC (1:Matrice de points réels de dim. 2 →
1:Matrice de points en cm)

```
« DUP SIZE → m sz
 «
  m STOΣ m sz 1 CON OFFEX 0 0 OFFEY { 2 2 }
  →ARRY * - EX 0 0 EY { 2 2 } →ARRY * sz
  1 CON OCMX 0 0 OCMY { 2 2 } →ARRY * + 2 FIX
 »
»
```

AC→AX (1:Matrice de points en cm →
1:Matrice de points réels)

```
« DUP SIZE → m sz
 «
  m STOΣ m sz 1 CON OCMX 0 0 OCMY { 2 2 }
  →ARRY * + EX INV 0 0 EY INV { 2 2 }
  →ARRY * sz 1 CON OFFEX 0 0 OFFEY { 2 2 }
  →ARRY * + 2 FIX
 »
»
```

GO (1:Matrice ou liste de matrices de points réels →)

```
« → Σd
 «
  Σd BORNS DEFXY Σd DRAWΣ
 »
»
```

DRAWΣ

```
« DUP TYPE
 5 ==
 « DUP SIZE »
 « 1 →LIST 1 »
 IFTE CLLCD DRAX LCD→ CLLCD → lΣ n i1
 «
  lΣ 1 GET
  IF n 2 ≥
  THEN
   2 n
```

```
    FOR k
      lΣ k GET CONCA
    NEXT
  END
  DRWA LCD→ i1 XOR →LCD DGTIZ CLLCD i1 →LCD lΣ n
  « DRLIN »
  LSTEX DGTIZ lΣ STOΣ
 »
»
```

ECHL (→)

```
« CMX XMAX XMIN - / 'EX' STO XMIN 'OFFEX' STO
  CMY YMAX YMIN - / 'EY' STO YMIN 'OFFEY' STO XMIN
  YMIN R→C XMAX YMAX R→C PMAX PMIN
»
```

DEFXY (→)

```
« { STO XMIN YMIN XMAX YMAX CMX CMY OCMX OCMY }
  MENU HALT ECHL
  {
    XY→CM CM→XY XY→AR AX→AC AC→AX GO
    DRAWΣ ΣDAT PPAR ECHL DEFXY Σ
  }
  DUP VARS 1 12 SUB SAME NOT
  « ORDER »
  « DROP  »
  IFTE
»
```

Remarque : Créer le répertoire `GRAPH/STAT/Σ` pour y stocker vos matrices de points.

LCONCA (1:Liste de matrices de points →
1:Matrices concaténées)

```
« DUP SIZE → lΣ n
  «
    lΣ 1 GET IF n 1 >
    THEN
      2 n
      FOR i
        lΣ i GET CONCA
      NEXT
    END
  »
»
```

BORNS (1:Liste (ou matrice) de points réels →)

```
« DUP TYPE 5 ==
  «
    LCONCA
  »
  IFT STOΣ NΣ
  «
    MINΣ MAXΣ 1 2
    START
      ARRY→ DROP R→C SWAP
    NEXT
    DUP2 PMAX PMIN C→R 'YMAX' STO 'XMAX'
    STO C→R 'YMIN' STO 'XMIN' STO
  »
  IFT
»
```

CONCA (2:Une matrice de n points
1:une matrice de p points →
1:une matrice de $(n+p)$ points)

```
« → a b
  «
    a ARRY→ 1 GET → n
    «
      b ARRY→ 1 GET n + 2 2 →LIST →ARRY
    »
  »
»
```

DRLIN (1:Une matrice de points →)

```
« DUP SIZE 1 GET LCD→ → Σd n
  im
  «
    IF n 2 MOD
    THEN Σd
      n 1 - GET Σd n GET { 1 2 } →ARRY
      Σd SWAP CONCA 'Σd' STO
    END
    Σd DRWA LCD→ { "E" "S" "L" }
    WAITK POS DUP 0 ==
    « DROP →LCD »
    «
      SWAP DROP 2 ≥ im →LCD
      «
        BEEP2 1 Σd SIZE 1 GET 2 * 2 -
        FOR j
          j j 3 +
          FOR jj
            Σd jj GET
          NEXT
          LINE4 2
        STEP
      »
      IFT
    »
    IFTE
  »
»
```

DRWA (1:Une matrice de points réels →)

```
« → Σdat
  «
    Σdat A→LXY « PIXEL » LSTEX
  »
»
```

A→LXY (1:Une matrice de points →
2:Liste de points en notation complexe
1:nombre de points)

```
« → a
  «
    a SIZE 1 GET → n
    «
      { } 1 n 2 *
      FOR i
        a i GET a i 1 + GET R→C +
        2
      STEP
      n
    »
  »
»
```

Paul Jebeily (571)

HP48S RETRO

Lors de la sortie de la HP28C beaucoup furent surpris par les possibilités de cette petite machine avec son nouveau langage RPL, mais après l'émerveillement arriva rapidement la déception : moins de 1700 octets à la disposition de l'usager. Vint après la HP28S dont les 32 K de RAM furent les bienvenus. Donc des possibilités réelles pour les programmeurs et ceux qui utilisent les outils des autres. Mais toujours pas de communication avec l'extérieur hors la possibilité d'imprimer le contenu des objets usuels. Aussi pour les machines saturées de programmes et autres objets, il faut faire le ménage pour introduire de nouveaux venus, quitte à réintroduire par la suite les objets purgés. Pire, pour les programmeurs en assembleur, les `Memory Lost` les guettent à tout instant et réduisent à néant de longues heures de travail.

Tout ceci a poussé certains à abandonner, ces petites machines non dépourvues de qualités par ailleurs. Le double clavier présente des avantages et le format les rend discrètes dans une poche. Une fois refermées leur écran est bien protégé et enfin n'oublions pas qu'avec le programme `SPEED` elles sont plus rapides que les HP48.

Actuellement beaucoup d'étudiants possèdent une HP28S, du fait de la décote par l'arrivée des HP48, et donc l'alimentation de la rubrique HP28 dans JPC se justifie.

Récemment, j'ai transformé mes deux HP28S pour qu'elles puissent communiquer avec l'extérieur. J'avais toujours refusé de me lancer dans cette aventure, craignant le pire, car inexpérimenté. Or le résultat est là: mes deux machines communiquent très bien avec un PC ou entre elles et encore avec l'imprimante à infrarouge HP. Tout ceci me pousse à relancer la rubrique HP28 dans JPC. En effet on peut envisager les choses sous deux angles :

- Ceux qui ne veulent pas transformer leur machine mais voudraient encore profiter des programmes en assembleur qui étendraient les possibilités de celle-là. Il est possible de faire aussi le *Coin des codes* comme pour les HP48, HP-71, PC avec une méthode d'introduction des codes fiable.
- Ceux qui ont transformé leur machine ont la possibilté de founir et de recevoir des programmes sur disquette et utilisables directement dans les HP28S.

Les possibilités actuelles

Comme pour les HP48 et l'assembleur, l'ouvrage de base en français est: *Voyage au centre de la HP28* de P. Courbis et S. Lalande. Ceux qui ne l'ont pas doivent consulter les rubriques des occasions ou se le faire prêter car il est épuisé.

Depuis la parution de ce livre les connaisances de la Rom HP28S se sont élargies grâce au travail des courageux et acharnés qui ont étendu le nombre des adresses de progammes et routines internes. Ceux qui possèdent aussi une HP48 peuvent s'aider de ce qui est connu de cette Rom pour rechercher des adresses de choses identiques dans les HP28S.

Concernant la méthode de transformation des HP28 une méthode bien documentée est fournie dans le livre *Voyage au centre....* Une méthode qui en dérive, mais plus sécurisante pour l'ouverture de la machine, est celle que préconise Patrice Godard dans sa notice intitulée: *Ma HP28S Mutante*. Cette méthode évite tout sciage de la machine mais par contre oblige à faire sauter toutes les têtes de rivets en plastique. C'est cette méthode que j'ai utilsée et que je recommanderais. Il est fourni des programmes de communication avec une méthode de contrôle. L'auteur mentionne plusieurs noms (pseudonymes) qui ont apporté leur contribution. Il est contactable sur divers BBS en bal `"PAQWAFER"`. En ce qui concerne l'interface série à réaliser j'ai été contraint d'abandonner celle décrite dans la notice et fonctionnant sans alimentation (ce qui me séduisait) pour adopter celle de *Au centre...* mais avec alimentation. L'ami qui m'a fourni la notice a été dans le même cas que moi.

Côté PC il faut adapter ce qui a été fait pour Apple dans *Au centre..* ou faire un programme ou en trouver un. En ce qui me concerne j'avais ébauché un programme en turbo-basic qui fonctionnait mais entre temps il m'en a été donné un réalisé en C et plus performant. Je compte me remettre à l'ouvrage et perfectionner le mien.

Une petite recommandation : dessoudez soigneusement la diode à infrarouge car elle est réutilisable.

Pour conclure je pense que les HP28S peuvent encore rendre de grands services en les améliorant côté matériel ou soft.

Sauvegarde déjà ?

Si vous (ou un ami) avez aussi une HP48 et un PC vous pouvez déjà sauvegarder vos objets HP28 sur disquette pour une réintroduction future si vous envisagé de transformer votre machine ou tout simplement pour **votre participation à JPC!!!**.

En effet, dans le Kit de communication pour HP48 existe un programme INPRT permettant de recevoir dans une HP48 des objets HP28 via les diodes à infrarouge. La HP48 se comporte alors comme une imprimante et reçoit les objets usuels sous forme de chaîne. Il suffit de faire OBJ→ dans la HP48 pour retrouver l'objet original mais avec les codes HP48 (donc souvent utilisable tel quel par la HP48) et qui peut être transféré en mode ASCII dans le PC pour un listing et enfin **peut être un article pour JPC!!!** ce qui ne serait pas mal du tout.

Cette façon d'opérer n'est valable que pour des listings ou la conversion d'objets HP28 en objets HP48. En effet pour pouvoir réutiliser à nouveau à l'avenir ces objets dans une HP28 il faut, dans celle-ci, les transformer en chaîne de codes hexa (pour les programmes en assembleur cela n'est possible que de cette manière) avant de les envoyer dans la HP48. A ce stade vous aurez reçu dans la HP48 la chaîne grossie d'un certain nombre de CHR 4 que vous aurez à éliminer soit dans la HP48, le PC ou la HP28 au retour. Je préfère les éliminer tout de suite dans la HP48. Vous transférez cette chaîne dans le PC et en mode binaire comme pour tout objet HP48. Quand votre HP28 communiquera avec l'extérieur vous pourrez récupérer les codes hexa. De retour dans la HP28 la chaîne sera à épurer de l'en-tête "HPHP.." (provenant de Kermit) et d'un CHR 26 en queue. Donc faire :

```
14 OVER SIZE 1 - SUB
```

pour obtenir la chaîne de codes à assembler et ainsi retrouver l'objet original HP28.

Comment envoyer des programmes au club

Pour ceux qui possèdent un PC et une HP28 avec connecteur :

- donner le listing des objets usuels sous forme de chaîne transférée de la HP28 au PC
- fournir si possible le fichier binaire des objets HP transférés dans le PC
- pour les programmes en assembleur fournir les listings et le type de compilateur utilisé: SASM, AREUH, HP48 de Laurent Grand etc... Donner évidemment les fichiers objets générés par ces compilateurs.

Pour ceux qui n'ont pas de HP28 avec connecteur essayer si possible de fournir les listings et les chaînes de codes hexa quels que soient les types d'objets et cela par l'intermédiaire d'une HP48 comme cela a été expliqué ci-dessus.

Enfin ceux qui ont travaillé et veulent en faire bénéficier le club mais n'ont pas les possibilités énoncées ci-dessus peuvent aussi nous envoyer leur travail sous forme listée par l'imprimante HP à infrarouge ou par leur propre main.

Comme les autres

Nous arrivons maintenant à la fourniture d'outils pour assembler les chaînes de codes des programmes en *system rpl* et assembleur. J'utiliserai quelques adresses non fournies dans *Au centre...* et qui pourront être des vieilles connaissances pour les uns et des nouveautés pour les autres.

Rappelons qu'une commande en *user rpl* vérifie toujours l'existence et la nature des arguments à fournir à la différence des commandes équivalentes en *system rpl*. Aussi pour ne pas les confondre celles-ci seront inscrites en minuscules alors que les commandes en *user rpl* sont toujours en lettres majuscule. D'autre part les commandes usuelles sont souvent polyvalentes. Par exemple:

"+" en *user rpl* permet:
- de faire la somme:
  - de nombres réels
  - de nombres complexes
  - de nombres binaires
  - de matrices et vecteurs
- de concaténer:
  - des chaînes
  - des listes
- etc..

alors qu'en *system rpl* on fait appel à des adresses différentes pour chaque type d'argument et il n'y a pas droit à l'erreur.

En première ligne nous retrouvons évidemment le classique assembleur de codes. C'est l'équivalent de ASC→ pour HP48 (JPC 78 et 79). Le listing du fichier source est compilable par *SASM*. A ce propos certains ont fait le reproche que j'emploie des mnémoniques en minuscules or je ne vois pas pourquoi que cela me serait interdit, cela était déjà pratiqué lors des beaux jours du HP71, et d'autre part les trois compilateurs cités ne sont pas incompatibles avec cette forme de listing.

Les adresses utilisées ci-dessous n'ont rien d'officiel mais dans l'état actuel des choses nous savons qu'il n'est pas besoin que ce soit supporté par HP puisque la fabrication des HP28 est finie. J'ai donné, lorsque cela était possible, les mêmes mnémoniques que pour HP48 mais en minuscules pour ne pas faire de confusions entre les deux machines.

ASC→

```
ASSEMBLE
        con(5)  #02c67          * début programme
        con(5)  #04f3d          * newob
        con(5)  #0204a          * dup
        con(5)  #040c2          * size
        con(5)  #02c96          * code
        rel(5)  fin             * offset fin code
        gosbvl  #04f27          * pop#
        gosbvl  #05081          * savptr
        a=a-1   a
        b=a     a               * compteur codes
        a=dat1  a               *
        a=a+con a,10            *
        d0=a                    * @ début codes
        d1=a                    *
        lchex   39
loop                            * conversion
        a=dat1  2               * ASCII -> HEXA
        ?a<=c   b               *
        goyes   inf10           *
        a=a-con b,7             *
                                *
inf10   dat0=a  1               *
        d1=d1+  2               *
        d0=d0+  1               *
        b=b-1   a               *
        gonc    loop            *
        govlng  #125e5          * getptrloop
fin
        con(5)  #3ceaa          * 2 get
        con(5)  #04f3d          * newob
        con(5)  #02f90          * fin programme
```

Certains trouveront peut-être étrange de ne pas trouver d'abord un assembleur en *user rpl*. Nous allons adopter une méthode connue qui consiste à stocker les codes hexa correspondant à ce source dans des entiers binaires que l'on va concaténer comme des chaînes grace au programme interne dont l'adresse est #3B82h. On remarquera que les codes sont stockés à l'envers par rapport à leur place définitive pour respecter la structure des entiers binaires et des chaînes. La boucle du programme aura donc permis de constituer un entier binaire avec l'ensemble des codes hexa ce qui évite de faire la conversion ASCII→Hexa. Le programme d'adresse #20238h fait un 2 GET de l'objet ce qui correspond à la partie code seulement sans le prologue ni la longueur. Il ne reste plus qu'à faire un NEWOB pour recréer en mémoire l'objet correspondant au code.

ASC→.rpl

```
«
  # 20204A04F3D02C67h # F80004F02C96040Ch
  # 8DCC05081F804F27h # 313103190F818341h
  # 681808AE91B51391h # 45DC061171085168h
  # F3D3CEAA125E5D8Eh # 2F9004h 28 STWS #0 OR
  64 STWS 1 7
      START # 3B82h SYSEVAL
      NEXT # 20238h SYSEVAL # 4F3Dh SYSEVAL
»
```

Pour obtenir le programme assembleur en assembleur il suffit de faire EVAL. Puisque ce programme se présente d'abord en *user rpl* nous allons pouvoir l'intégrer dans le programme ci-dessous qui a pour objet de faire rentrer les codes hexa de façon fiable car contrôlés par série de 16 caractères, évitant ainsi, en cas d'erreur de tout vérifier, ce qui est assez rébarbatif.

Tapez les deux programmes ASSCOD.28 et INPUT.28. Lorsque vous aurez le programme SPEED vous pourrez inclure cette commande dans ASSCOD comme ci-dessous. Ces deux programmes sont à entrer avec la plus grande attention car leur mauvais fonctionnement peut entraîner un désordre fatal pour les objets contenus dans votre machine. L'effort d'attention qui vous est demandé ici, est le seul et par la suite vous ne regretterez pas cet investissement qui vous procurera beaucoup de confort et de sécurité.

ASSCOD.28

```
«
  SPEED RCLF HEX 64 STWS 1 SF "" 'tmpcod' STO
  "nombre d'octets                    @ chaîne se
                                      @ terminant
  "                                   @ 2 par NEWLINE
  17 INPUT 1 CF STR→ 2 * 16 DUP2 / IP 3 ROLLD MOD
```

```
DUP2
  IF
  THEN 1 +
  END 3 ROLLD 1 + 4 ROLL # 0h DUP → n r f s o
  « 1 SWAP
    FOR i
      DO "ligne " "00" i 1 - R→B →STR 3 OVER
         SIZE 1 - SUB + DUP SIZE DUP 2 - SWAP SUB
         + DUP "                          @ chaîne
               chaine                     @ encadrée par
               " +                        @ 2 NEWLINE
               "---- ---- ---- ----"
               n i <
               IF
               THEN r DUP 4 / IP + SWAP OVER 1
               SWAP OVER - SUB SWAP 18 +
               ELSE 38
               END 'o' STO +
               "                          @ NEWLINE
               " + 1 FS?C
               IF
               THEN ROT SWAP CLLCD 1 DISP HALT SPEED
               ELSE o INPUT
               END DUP
               WHILE DUP " " POS DUP
               REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD
                  1 + 25 SUB
                  IF DUP " " ==
                  THEN DROP
                  ELSE +
                  END
               END DROP 0 OVER SIZE 1 SWAP
               FOR j OVER j DUP SUB NUM j * +
               NEXT s + DUP # FFFh AND
               "                    @ NEWLINE
               somme de controle    @ NEWLINE
                ---                 @ NEWLINE
               #"
               6 ROLL SWAP + 34 INPUT STR→ ==
               IF
               THEN SWAP DROP 1
               ELSE DROP2 1000 1 BEEP 1 SF 0
               END
      UNTIL
      END 's' STO
      WHILE DUP " " POS DUP
      REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 +
         19 SUB +
      END DROP 'tmpcod' DUP RCL ROT + SWAP STO
    NEXT f STOF
    tmpcod
  # 20204A04F3D02C67h # F80004F02C96040Ch     @@
  # 8DCC05081F804F27h # 313103190F818341h     @@
  # 681808AE91B51391h # 45DC061171085168h     @@
  # F3D3CEAA125E5D8Eh # 2F9004h 28 STWS #0 OR @@
  64 STWS 1 7                                 @@
     START # 3B82h SYSEVAL                    @@
     NEXT # 20238h SYSEVAL # 4F3Dh SYSEVAL    @@
     EVAL "fin" CLLCD 1 DISP                  @@
»
```

Si vous avez déjà un programme assembleur en assembleur vous pouvez remplacer les lignes avec @@ jusqu'à EVAL par le nom correspondant à votre programme. De même si vous préférez créer d'abord le programme ASC→ à partir du programme en *user rpl*, vous pouvez opérez de même.

INPUT.28

Ce programme est fait pour remplir le rôle de la commande *INPUT* de la HP48 qui n'existe pas chez son ancêtre. Evidemment ce sous programme est plus spécialisé pour les besoins de la cause.

```
«
  SWAP 1
  WHILE OVER CLLCD 1 DISP
  REPEAT
    DO
    UNTIL KEY
    END
    IF DUP "ENTER" ==
    THEN DROP 0
    ELSE
      IF DUP "BACK" ==
      THEN DROP 1 OVER SIZE 1 - SUB
      ELSE + DUP SIZE 3 PICK - 2 + 5 MOD NOT 1 FC?
        AND
        IF
        THEN " " +
        END
      END 1
    END
  END SWAP 60 SUB
»
```

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

1- lancez le programme ASSCOD.28
2- donnez le nombre d'octets (1/2 oct. compris) puis validez avec ENTER.
3- tapez chaque ligne de codes, correspondant au numéro de ligne à 3 chiffres, sans les espaces et validez.
4- tapez la somme de contrôle et validez. S'il y a erreur la ligne de codes sera demandée à nouveau après émission d'un BEEP. L'appui sur EDIT fera apparaître la ligne des codes qui pourra être corrigée. Relancez avec CONT.
5- stockez le programme assemblé dans la variable donnée en tête.

Nous avons donc maintenant nos outils pour le coin des codes HP28S comme pour les autres machines. A l'avenir ces deux programmes seront dans la rubrique qui les concerne mais juste avec le minimum d'indications pour leur utilisation.

Guy Toublanc (276)

FAST-ORIELLES HP28S

Dans l'article *HP28S RETRO* j'ai fait allusion à la possibilité de coopération HP48<->HP28. Ce qui a été fait pour l'une des machines peut très bien être adapté à l'autre. Ici j'applique cette possibilité à propos du calcul des factorielles en multi-précision. Dans JPC 81 Laurent Grand nous avait donné FFACT, programme en *user rpl* et assembleur. Dans JPC 82 je donnais une version optimisée. C'est cette version que j'adapte à la HP28S. Donc les adresses changent mais les calculs en *system Rpl* et assembleur sont identiques.

Ce qui a été dit dans JPC 81 et 82 est toujours valable. Il existe dans la ROM une routine pour générer une chaîne mais son utilisation m'a valu de nombreux plantages. Son adresse est #1E9BAh et le programme principal est identique à MAKEN$ de la ROM HP48. Quand j'aurai le temps j'analyserai les sous-programme de cette routine. Si quelqu'un l'a déjà utilisé qu'il ait la gentillesse d'en donner la solution. Normalement il faut donner le nombre de quartets de la chaîne à créer (prologue compris) dans C(A) et au retour D0 pointe sur le premier caractère, R0(A) = adresse prologue et R1(A) = adresse longueur. Avec cette routine mon programme aurait été plus court.

En ce qui concerne les mnémonique en *system rpl* j'utilise le même principe exposé dans l'article *HP28S RETRO*. A coté des mnémoniques j'ai indiqué la nature des arguments à fournir au programme par exemple: *re* pour un réel. Voici donc le fichier source et vous trouverez la liste des codes hexa dans la rubrique *le coin des codes*.

FFACT.28S

```
ASSEMBLE
        con(5)  #02c67      * docol
        con(5)  #09bdd      * IP
        con(5)  #11c83      * abs re
        con(5)  #0204a      * dup
        con(5)  #02933      * reel 15
        nibhex  1000000000000510
        con(5)  #08b26      * < re re
        con(5)  #0e3e2      * IF
        con(5)  #0e3f8      * THEN
        con(5)  #02c67      * docol
        con(5)  #125f6      * fact re
        con(5)  #1743f      * ->str re
        con(5)  #02f90      * semi
        con(5)  #0e459      * ELSE
        con(5)  #02c67      * docol
        con(5)  #0204a      * dup
        con(5)  #0c53b      * r->sb  n
        con(5)  #020e5      * swap
        con(5)  #0204a      * dup
        con(5)  #0204a      * dup
        con(5)  #3eeff      * e
        con(5)  #11d82      * / re re
        con(5)  #11eda      * log re
        con(5)  #11d48      * * re re
        con(5)  #020e5      * swap
        con(5)  #0204a      * dup
        con(5)  #11ceb      * + re re
        con(5)  #11443      * pi
        con(5)  #11d48      * * re re
        con(5)  #11e5e      * sqrt
        con(5)  #11eda      * log
        con(5)  #11ceb      * + re re
        con(5)  #12485      * ceil re
        con(5)  #0c53b      * r->sb  long. de n!
        con(5)  #02c96      * docode
        rel(5)  fin         * offset pour fin code
        gosbvl  #04f27      * pop#
        gosbvl  #05081      * savptr
        d0=(5)  #c015f
        c=0     a
        dat0=c  a
        st=0    10
        a=a+a   a
        a=a+con a,10
        r1=a
        gosub   load
L3ch    c=r1
        gosbvl  #053ae      * res_room
        gonc    Ok3ch
        gosub   load        * resptr
        ?st=0   10
        goyes   Retry
        govlng  #0393e      * setmemerr
Retry   gosbvl  #04a94      * garbagecol
        st=1    10
        goto    L3ch
Ok3ch   cd0ex
        r3=c
        d0=c
        lchex   02a4e       * docstr
        dat0=c  a
        a=r1
        a=a-con a,5
        d0=d0+  5
```

```
        dat0=a  a
        d0=d0+  5
        a=a-con a,5
        c=0     w
        c=a     a
        csrb
        rstk=c          * long. n!
        cd0ex
        rstk=c
        r1=c
        d1=c
        c=a     a
        gosbvl  #5042   * write_zeros
        gosub   load
        a=dat1  a
        d1=a
        d1=d1+  5
        c=0     w
        r0=c
        c=dat1  a
        r2=c
        c=rstk
        d1=c
        c=0     a
        c=c+1   a
        p=      11
        dat1=c  wp
loop
        c=r1
        d0=c
        c=r0
        d=c     a
        c=0     a
l1      rstk=c
        setdec
        a=0     w
        a=dat0  wp
        c=r2
        b=0     w
mult1   sb=0
        csrb
        ?sb=0
        goyes   mult2
        b=b+a   w
mult2   a=a+a   w
        ?c#0    a
        goyes   mult1
        c=rstk
        c=c+b   w
        dat0=c  wp
        d0=d0+  12
        c=0     wp
        cslc
        cslc
        cslc
        cslc
        sethex
        d=d-1   a
        gonc    l1
        ?c=0    a
        goyes   l2
        dat0=c  a
        a=r0
        a=a+1   a
        r0=a
l2      c=r2
        c=c-1   a
        r2=c
        ?c#0    a
        goyes   loop
out1
        p=      0
        c=rstk
        c=c-1   a
        b=c     a
        csrb
        d=c     a
        c=r1
        c=c+b   a
        d0=c
        rstk=c
loop2   a=dat0  1
        c=dat1  1
        dat1=a  1
        dat0=c  1
        d1=d1+  1
        d0=d0-  1
        d=d-1   a
        gonc    loop2
        c=rstk
        d1=c
        c=c+b   a
        d0=c
        lchex   30
l3      c=dat1  1
        dat0=c  2
        d0=d0-  2
        d1=d1-  1
        b=b-1   a
        gonc    l3
        gosub   load
        a=r3
        dat1=a  a
        govlng  #125ec  * retour rpl
load    govlng  #050b8  * resptr
fin
        con(5)  #02f90  * semi
        con(5)  #0e479  * END
        con(5)  #02f90  * semi
```

Guy Toublanc (276)

HP-48

OPTIMISATION D'ENREGISTREMENTS

Puisque vous manquez cruellement d'articles de "nouveaux", je me lance à l'eau avec ce petit programme de mon cru.

Il m'arrive souvent de réaliser des copies de compacts sur cassettes et je déplore à chaque fois le "blanc" laissé à la fin de mes enregistrements. Il existe pourtant un choix de morceaux qui minimise l'espace magnétique perdu.

Le trouver à la main est un travail titanesque pour peu que le nombre de morceaux dépasse 4 ou 5 !

Programmer cette application n'est pas, du moins à mon niveau, franchement facile non plus. J'ai fini par réaliser que seule la récursivité pouvait me sortir d'affaire (c'est un des rares cas ou elle me semble incontournable). En langage algorithmique, la partie récursive de mon programme peut s'écrire shématiquement comme suit :

RECM :

```
prend une liste de temps {t1..tn} au niveau 1
calcule la somme S=t1+..+tn
Si S > temps maximum autorisé TM
   de i=1 à n fait
      enlève l'élément i de la liste
      applique RECM
   fin de boucle
Sinon
  Si S > approximation précédemment trouvée
      S devient la meilleure approximation trouvée
      Affichage de S
  Fin de test
Fin de test
```

Le programme RECORD.MAS permet, quand à lui, d'initialiser les variables et de démarrer la recherche. Le mode d'emploi est très simple, il vous suffit d'entrer au niveau 2 une liste de temps sous la forme mm.ss et au niveau 1 le temps maximal admissible sous la forme mm.ss, d'exécuter RECORD.MAS. Celui-ci vous rappelle le mode d'emploi et s'arrête. Vous démarrez alors la recherche par [CONT].

S'affichent alors la meilleure approximation trouvée, le nombre de morceaux impliqués dans cette approximation, un rappel du temps maximal admissible et la procédure à effectuer pour arréter la recherche. Quand l'approximation vous convient, tapez [ON], [CLR] et [LS] pour obtenir la liste "gagnante".

Un petit exemple (réel) de recherche:

Niveau 2 : {19.59 7.22 6.34 16.25 6.35 9.01}
Niveau 1 : 30

On obtient rapidement :

29.34 pour {6.34 16.25 6.35}

soit un blanc de 26 secondes!

RECM

#C79Fh
310.5 octets

```
« DUP SIZE
  IF 1 ≠
  THEN DUP OBJ→ 2 SWAP
    START HMS+
    NEXT → L T
    «
      IF T M >
      THEN 1 L SIZE
        FOR I L I
          SWAP OBJ→ 2 + DUP
          ROLL OVER SWAP -
          ROLL DROP 3 - →LIST
          RECM
        NEXT
      ELSE
        IF T TS >
        THEN T DUP
          →STR
          " : Temps trouve" +
          1 DISP 'TS' STO L
          DUP SIZE →STR
          " : n de morceaux"
          + 2 DISP 'LS' STO
        END
      END
    »
  ELSE DROP
  END
»
```

RECORD.MAS

#B8E0h
312.5 octets

```
« CLLCD { LS TS }
  TMENU
  "*** RECORDMASTER ***      @
                             @
                             @    ATTENTION,
                             @ Une seule chaîne !
                             @
```

```
  Liste de temps          @
  temps max ... svp"      @
  1 DISP 2 WAIT HALT
  0 'TS' STO { } 'LS'
  STO 'M' STO CLLCD
  "pour un temps maxi de "
  M →STR +
  " pour arreter → ON       @ Une seule chaîne
    puis CLR puis LS"       @
  + 3 DISP RECM LS
  2000 .2 BEEP { TS LS M }
  PURGE
»
```

Il y a bien entendu de nombreuses améliorations à apporter à cette application, notamment dans les directions suivantes :

- remplacer les HMS+ par des + plus rapides en calculant d'abord le nombre de secondes que représente chaque temps.
- Se passer totalement de variables globales et, si possible de variables locales.
- Réécrire l'application en *System Rpl* pour un gain en temps et en place.

J'espère vraiment lire prochainement dans JPC des versions plus performantes de cette application.

Jean-Michel MERMET (568)

NDLR : Autre amélioration possible, pourrait-t-on accepter des paramètres entiers, pour gérer les disquettes informatiques ?

RECUPERATION DE PORT

Ce programme recherche dans le port specifié toutes les librairies et tous les backups validés, et les dépose sur la pile.

Il doit pouvoir retrouver après un Invalid Card Data tout ce qui peut encore l'être, car il recherche systématiquement dans toute la carte.

Il s'inspire de la librairie RECOVER de HPeed. L'avantage étant que si la librairie est dans le port perdu, on n'y a pas accès, alors qu'il suffit de retaper ce programme et de l'assembler pour espérer retrouver le port.

```
::
   CK1&Dispatch
   ONE
   ::
      COERCE
CODE
        GOSBVL   #06641    * POP.A
        GOSBVL   #0679B    * SAVPTR
        D=0      A         * nb d'objets trouvés
        D0=(5)   #70400
        D1=(5)   #70400
        ?A=0     A
        GOYES    PORT0
        A=A-1    A
        ?A=0     A
        GOYES    PORT1
        A=A-1    A
        ?A=0     A
        GOYES    PORT2

         * PORT NON VALIDE
         * ON SORT

FIN     C=D A
        R0=C A
        GOSBVL   #06537    * PUSH nb d'objets
                           * trouvés
        GOVLNG   #00D84    * Retour au RPL

PORT0   D0=(4)   #0597     * @ DEBUT
        D1=(2)   #4D       * @ FIN
        GOTO     SUITE2

PORT1   D0=(2)   #2C       * @ Info Crate
        GOSUB    PRESENTE?
        GOTO     SUITE

PORT2   D0=(2)   #37       * @ Info Carte
        GOSUB    PRESENTE?
SUITE   D0=D0+   1
        C=DAT0   A         * Taille carte
        D0=D0+   5         * Saute a l'@
                           * de début de port
        A=DAT0   A
        D0=A
        A=A+C    A
        B=A A              * @ Fin de recherche
        GOTO     RECHERCHE

SUITE2  A=DAT0   A
        D0=A
        A=DAT1   A
        B=A A
RECHERCHE
        LCHEX    02
R1      A=DAT0   B         * Cherche la valeur 02
                           * (fin de prologue
                           * commun au LIB et BACKUP)
```

```
        ?A=C    B
        GOYES   TROUVE?
        DO=DO+  1
        ADOEX
        ?A>=B   A       * Arrivé à la fin
                        * du port
        GOYES   FIN
        ADOEX
        GOTO    R1

TROUVE? DO=DO-  3
        LCHEX   02B40
        A=DATO  A
        ?C=A    A       * Est-ce une LIB ?
        GOYES   TROUVE
        LCHEX   02B62
        ?A=C    A       * Est-ce un BACKUP ?
        GOYES   TROUVE
        DO=DO+  4
        GOTO    RECHERCHE

TROUVE  DO=DO+  5       * Saute prologue
        A=DATO  A       * A=Taille
        A=A-CON A,4     * moins CRC
        CDOEX
        RO=C A          * Sauvegarde DO
        GOSBVL  #0597E  * Calcul du CRC
        C=0 A
        C=DATO  4
        ?A=C    A       * Est-ce égal au CRC ?
        GOYES   BON     * Oui => Trouvé !!!
MAUVAIS A=RO    A       * Continue la recherche
        DO=A
        GOTO    RECHERCHE

BON     D=D+1   A       * Un de plus
        DO=(5)  #70579
        D1=(5)  #70574
        A=DATO  A
        C=DAT1  A
        C=A-C   A
        C=C-CON A,5     * Reste-t-il
                        * de la mémoire ?
        GOC     INSUF.MEM
        A=A-CON A,5
        D1=A            * Augmente la taille
                        * de la pile
        DATO=A  A
        A=RO A
        A=A-CON A,5
        DAT1=A  A       * Met l'objet sur
                        * la pile
        D1=(5)  #0010B
        LCHEX   8F      * Allume les
        DAT1=C  B       * indicateurs d'état
        LCHEX   FFF
B1      C=C-1   X       * ATTEND
        GONC    B1
        LCHEX   80      * Eteint les
        DAT1=C  B       * indicateurs
        GOTO    MAUVAIS

INSUF.MEM
        C=0     A
        LCHEX   1
ERREUR  A=C     A
        GOSBVL  #067D2  * GETPTR
        GOVLNG  #05023  * ERROR

PRESENTE?
        A=DATO  P
        ?ABIT=0 3       * Carte présente?
        GOYES   NOCARTE
        ?ABIT=1 1       * Carte merge?
        GOYES   CMERGE
        RTN

NOCARTE
CMERGE  C=0     A
        LCHEX   A
        GOTO    ERREUR

ENDCODE
   ;
;
```

Pierre Silvestre De Sacy (572)

NOUVELLE VERSION DU KIT

Nous vous avons dit récement qu'il était dans nos projets de faire évoluer notre kit de développement pour HP48. Une nouvelle étape a été franchie ce mois ci.

La disquette contient maintenant :

- Les utilitaires et documentations développés par HP, connus aussi sous le nom de *Goodies 4*, qui étaient présents sur la derniere version de notre disquette.
- Le compilateur de Laurent Grand, permettant de mixer de l'*User RPL* et de l'assembleur (programme distribué en Shareware).
- Le programme d'exploration de la Rom HP48, écrit par Jean-François Garnier.

La Documentation a bien entendu été mise à jour.

Jacques Belin (123)

AH ! VOUS ECRIVEZ

Vous vous sentez en verve, mais vous ne savez pas sous quelle forme "l'équipe de rédaction" souhaite recevoir votre prose. C'est ici que se trouvent les réponses à vos questions.

Dans la mesure du possible, vous devez nous envoyer vos écrits sur support magnétique (carte, cassette ou disquette) dans le format lisible directement pour votre machine. Vous pouvez taper vos articles sur IBM PC, mais n'utilisez pas de traitement de texte (WORD...). Utilisez plutôt un éditeur simple manipulant des fichiers en ASCII pur, sans codes d'enrichissement. Soyez sans crainte, nous vous retournerons vos biens après copie.

Si vous n'avez pas accès à un IBM, et seulement dans ce cas là, vous pouvez à la rigueur nous envoyer des disquettes provenant de MacIntosh.

Si vous ne pouvez pas utiliser de support magnétique, ou ne pouvez vous rendre aux réunions, alors et alors seulement faites le sur papier.

Que ce soit sur une feuille de papier, ou sur support magnétique, ne dépassez pas 50 caractères par ligne.

Pour nous épargner du travail, insérez dans votre texte les commandes de formattage suivantes (et non les commandes du formatteur HP-71) :

"^" centre un titre, par exemple :
^TITRE

"\" (CHR$(92)) marque le début et la fin d'un paragraphe. Par exemple :

\Début de paragraphe exprimant le contenu de vos idées qui, même si vous en doutez, intéressera certains des membres du Club. Surtout si vous vous sentez débutant. Les articles pour débutants écrits par des débutants sont ceux qui manquent le plus. Fin de paragraphe.\

Pour écrire une accolade ({ ou}), il faut doubler l'accolade, une accolade simple ayant une signification spéciale dans l'édition de JPC.

Les utilisateurs de HP-71 utiliseront le fichier CHARLEX, qui a été souvent listé dans le coin des Lhex, pour utiliser les caractères accentués du jeu Roman8.

Les utilisateurs de HP-48 doivent nous envoyer les programmes RPL sous deux formes : Binary, plus ASCII avec *Translate Code* : 3. Les programmes composés de plusieurs variables doivent être dans un répertoire à transférer sur disquette et dans les deux modes. Si vous insérez dans le corps des articles le texte des programmes, veuillez faire précéder ce texte de `{rpl` et le faire suivre de `{endrpl`.

Les utilisateurs de HP-95 et MS-DOS nous transmettront les sources de leurs programmes accompagés des fichiers compilés. Il nous indiquerons le nom et la version du compilateur ou de l'interpréteur qu'ils ont utilisé.

Le Bureau

PPC PARIS SE REUNIT
UNE FOIS PAR MOIS

Comme vous le savez peut être déjà, PPC Paris se réunit une fois par mois, en plein coeur de Paris. Amenez votre matériel, votre bonne volonté et vos idées ! Plus vous en apporterez, et plus vous en trouverez chez vos collègues de PPC.

Ces réunions se déroulent de manière très libre, aucun ordre du jour, discussion ou autre n'étant imposé. Un membre du bureau est toujours présent. Ainsi, si vous désirez remettre votre article tout frais au Journal, si vous avez des suggestions à faire, si vous voulez vous procurer des anciens numéros de JPC, ce sera en principe toujours possible.

Si donc cela vous intéresse, n'hésitez plus un seul instant, venez nous rejoindre tous les premiers samedis de chaque mois (sauf en période de vacances scolaires) au :

Centre de Jeunesse et de Loisirs Jean Verdier
11 rue de Lancry
75010 Paris

et en montant au deuxième étage, vous entendrez des éclats de rire et des discussions passionnées vers la salle 215. Attention, toutefois, de venir entre 16 et 19h.

Pour l'accès en métro, trois possibilités s'offrent à vous :
- Métro Strasbourg Saint Denis :
Sortie porte St Martin / Bd St Denis, coté pairs
- Métro République :
Sortie Bd St Martin, coté pairs
- Métro Jacques Bonsergent :
Sortie Bd Magenta, coté impairs.

Ah, j'oubliais ! JPC est (souvent) distribué en avant première lors de ces réunions... A bon entendeur, salut !

Les dates des prochaines réunions sont :

Samedi 6 Mars 1993
Samedi 3 Avril 1993
Samedi 5 Juin 1993

NOUS EN AVONS

La coopérative du Club vous propose :

- Des **anciens numéros** de JPC (Préciser les numéros).
- Des **années complètes** de numéros de JPC (février à janvier).
- La **Programathèque HP-71**, regroupant tous les Lex et programmes pour HP-71 et HP-75 parus à ce jour dans *JPC*. Elle vous est livrée avec un catalogue décrivant brièvement tous les programmes.
- La **JPC Rom** pour HP-71, comportant plus d'une centaine de fonctions en assembleur (mathématiques, programmation structurée, utilitaires systèmes...). Joindre une Eprom vierge (64 Ko) à votre règlement. Il est possible d'y ajouter vos propres programmes.
- Les **Goodies Disks HP48**, disquettes compressées : Disk1 = G1 + G2 + G3, Disk2 = G5 + G6, Disk3 = G7.
- Les **utilitaires de développement** pour HP-48 fonctionnant sur IBM PC. La disquette contient les utilitaires développés mais non supportés par HP connus aussi sous le nom de **goodies 4**. C'est à dire un assembleur et un Linker pour le microprocesseur Saturn, un compilateur RPL et un générateur de librairies ainsi qu'une très abondante documentation sur le fonctionnement en détail de la HP48. La disquette contient aussi un compilateur écrit par Laurent Grand, permettant de mixer de l'*User Rpl* et de l'assembleur. Le dernier programme est un désassembleur/explorateur de la Rom de la HP48, écrit par Jean-François Garnier. Vous pouvez obtenir la disquette avec la documentation imprimée et reliée par nos soins.

Le port, et eventuellement la disquette (précisez le format), sont inclus dans les prix.

Si vous souhaitez des renseignements complémentaires, n'hésitez pas à nous contacter.

VOUS EN VOULEZ

Nom :
Prénom :
No d'adhérent :
Adresse :

	Quantité	Prix Unitaire Adhérent	Autre	Prix Total
Anciens numéros de JPC		x 40 FF	45 FF	
Année complète de JPC		x 300 FF	350 FF	
JPC Rom + Manuel		x 500 FF	600 FF	
JPC Rom + Manuel + vos propres programmes		x 700 FF	800 FF	
Actualisation Eprom		x 150 FF	200 FF	
Goodies Disks 1, 2 et 3 compressés sur disque 1		x 40 FF	50 FF	
Goodies Disks 5 et 6 compressés sur disque 2		x 40 FF	50 FF	
Goodies Disks 7 compressés sur disque 3		x 40 FF	50 FF	
Kit de développement pour HP-48 (disquette seule)		x 40 FF	50 FF	
Kit de développement pour HP-48 (disquette + doc imprimée)		x 170 FF	200 FF	
			Total	FF

Préciser éventuellement les numéros de JPC commandés :

PPC PARIS

Association régie par la loi de 1901, enregistrée
à Paris le 2 décembre 1982 sous le numéro 82/3240

BULLETIN D'ADHESION

Nom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Prénom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| Date de naissance |_|_|/|_|_|/|_|_|

Adresse |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Code Postal |_|_|_|_|_| Ville |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Téléphone Domicile |_|_|_|_|_|_|_|_|_|_|_|_|_| Bureau |_|_|_|_|_|_|_|_|_|_|_|_|_|

Profession ______________________________

Intérêts ______________________________

____

Matériel HP en votre possession ______________________________

____

Autre matériel informatique ______________________________

____

Comment avez-vous connu PPC Paris ? ______________________________

____

Que recherchez-vous au sein de PPC Paris ? ______________________________

____

____

La Loi No 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés, garantit à toute personne justifiant de son identité un droit d'accès et de rectification auprès des services ou organismes chargés de mettre en oeuvre des traitements informatiques comportant des informations nominatives le concernant.

Je souhaite adhérer au club PPC Paris conformément aux statuts de l'Association. Au mieux de ma connaissance, je déclare avoir le droit de fournir tous les programmes et informations que je vous enverrai (sans enfreindre des obligations de secret à l'égard d'autres personnes ou organismes) pour publication dans le Journal de liaison, sans obligations ni responsabilité d'aucune sorte (en cas d'utilisation frauduleuse) de la part des dirigeants de PPC-Paris.

Date |_|_|/|_|_|/19|_|_|

Signature, précédée de la mention "Lu et Approuvé" ______________________

Le montant de la cotisation s'élève à 350.00 F pour un an.
Etudiants: 300.00 F (justificatif indispensable)

Paiement à l'ordre de "PPC Paris" par chèque bancaire ou virement postal (CCP No 18 823 40 C à Paris). Ne pas utiliser d'Eurochèques.

Veuillez envoyer toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01, France

HP-95

LE GRAPHISME SUR HP95

INTRODUCTION

Une grandes différences du HP95 par rapport à un IBM PC classique concerne le graphisme. En effet, alors que le mode texte est parfaitement supporté, le HP95LX ne reconnait aucun mode standard du PC, ne serait-ce que le CGA.

Cependant, il comporte des fonctions graphiques beaucoup plus puissantes que sur la majorité des cartes VGA disponibles sur le marché, puisqu'il est possible de tracer, en un seul appel système, des lignes, des rectangles ou des images complètes. Rappelons que les fonctions VGA standard ne se résument qu'à l'affichage du pixel !

Cet article a donc pour but de vous décrire en détail les caractéristiques et les fonctions de cette machine.

CARACTERISTIQUES GENERALES

L'écran du HP95 est divisé en un espace de 240×128 pixels. Les deux seules couleurs admises sont le blanc et le noir (pas de niveau intermédiaire).

L'origine des coordonnées se situe dans le coin supérieur/gauche de l'écran. La numérotation des points est comprise dans l'intervale 0-239 pour les abcsisses et 0-127 pour les ordonnées.

Contrairement au mode VGA, les pixels ne sont pas carrés. Ils suivent un rapport aproximatif de 4/3. Ce qui implique qu'une routine de tracé de cercle ne tenant pas compte de ce fait tracera une ellipse (plus large que haute). Une zone de quatre pixels de haut sur trois pixels de large apparaitra en revanche sensiblement carrée.

L'adresse de la mémoire vidéo est située en B000. Chaque segment de 8 pixels d'une ligne est stocké dans un octet. La taille de cette mémoire est donc de 240*128/8=3840 octets.

LISTE DES FONCTIONS

Alors que sur le PC les fonctions graphiques sont regroupées dans l'interruption 10h (avec les fonctions vidéo du mode texte), les fonctions graphiques du HP95 sont placées dans l'interruption 5Fh. Sauf cas spécifique, toutes les fonctions décrites dans cet article seront donc appelées à partir de cette interruption. Comme d'habitude le numéro de la fonction est placé dans le registre AH.

Après exécution de l'interruption, les registres semblent être remis à leur état initial (à l'exception du registre AX). Cela simplifierait la programmation en assembleur. Cependant, il est conseillé de vérifier, car je n'ai fait qu'un test rapide des retours d'interruptions.

Entrée et sortie du mode vidéo

Avant toute opération graphique, il est nécessaire de placer le HP95 en mode graphique. Cela est fait en exécutant la fonction 00h, sous-fonction 20h :

```
mov   AH,00h
mov   AL,20h
int   5Fh
```

Ceci permettra d'utiliser les fonctions graphiques. L'écran est effacé et les paramêtres courants (position du curseur...) seront mises à leur valeurs par défaut (voir chapitre suivant).

Si vous n'avez pas besoin de conserver les paramètres courants, vous pouvez utiliser cette fonction pour effacer l'écran.

En fin de programme, il est nécessaire de retourner en mode Texte. Cela peut être fait en utilisant les fonctions normales du BIOS vidéo. Par exemple en utilisant la fonction 0h sous fonction 7h de l'interruption 10h. L'écran sera effacé.

Lecture des informations vidéo

Au cours de l'exécution du programme, il peut être nécessaire de connaitre les informations concernant les différents modes et valeurs utilisés. Ces données sont accessibles après avoir appelé la fonction 02h, et placées dans une table que vous aurez déclaré dans votre programme.

Valeurs d'entrée :
AH = 02h
ES:DI : adresse de la table d'information.

Valeur retournée :
DX:AX : adresse de la table (= ES:DI)

Format de la table d'informations graphique :

Offset	Taille	Défaut	Description
00h	BYTE	20	Mode vidéo courant
01h	BYTE	3	Mode vidéo par défaut
02h	WORD	240	Largeur affichage en pixels

```
04h     WORD      128  Hauteur affichage en pixels
06h     WORD        0  Colonne curseur
08h     WORD        0  Ligne curseur
0Ah     WORD    FFFFh  Aspect ligne
0Ch     WORD        0  Méthode de remplacement
0Eh     WORD        1  Couleur de tracé
10h     WORD        0  Colonne gauche fenêtre
12h     WORD        0  Ligne haute fenêtre
14h     WORD      239  Colonne droite fenêtre
16h     WORD      127  Ligne basse fenêtre
18h     WORD        0  Colonne origine logique
1Ah     WORD        0  Ligne origine logique
1Ch  8 BYTES        *  Masque de remplissage
                       |
                       +-> * = FF FF FF FF FF FF FF
```

Pour des explications détaillées sur ces données, reportez vous aux paragraphes suivants.

Commandes de positionnement et d'initialisation

Ce chapitre contient la description de différentes commandes n'ayant pas d'effet visible, mais permettant de préparer ou d'ajuster le comportement des fonctions de tracé.

Positionnement du curseur

Cette fonction permet de positionner le curseur graphique à une coordonnée qui servira d'origine aux fonctions de tracé.

Valeurs d'entrée :
AH = 08h
CX = Colonne
DX = Ligne

Valeur retournée :
Aucune

Positionnement de l'origine logique

Afin de simplifier certains programmes, il peut être intéressant d'écrire des routines permettant d'effectuer des tracés "standard" à une position quelconque de l'écran. Cette fonction permet donc de choisir une nouvelle origine de coordonnées, autre que le coin supérieur gauche de l'ecran.

Valeurs d'entrée :
AH = 03h
CX = Colonne
DX = Ligne

Valeur retournée :
Aucune

La position du curseur graphique reste constante par rapport à l'origine. Ce qui veut dire que si l'on déplace cette origine, le curseur suivra physiquement le déplacement de la fenêtre.

Notons à ce sujet que pour toutes les fonctions de tracé, les valeurs sont données en coordonées absolues par rapport à l'origine logique.

Sélection d'une fenêtre graphique

Cette fonction permet de définir une fenêtre limitant les tracés à une zone prédéfinie. Les fonctions de tracé ayant des coordonnées finales extérieures à cette fenêtre s'interompront aux limites de celle-ci.

Valeurs d'entrée :
AH = 04h
CX = Colonne coin Supérieur/Gauche
DX = Ligne coin Supérieur/Gauche
SI = Colonne coin Inférieur/Gauche
DI = Ligne coin Inférieur/Gauche

Valeur retournée :
Aucune

L'origine et la position du curseur ne sont pas modifiées par cette fonction.

Avant l'appel de la fonction de tracé, le curseur graphique doit être situé à l'intérieur de la fenêtre, sinon le tracé sera totalement ignoré.

Sélection d'une couleur de trait

Cette fonction permet de définir la couleur utilisée par toutes les fonctions de tracé.

Valeurs d'entrée :
AH = 09h
AL = Nouvelle couleur (Blanc = 0, Noir = 1)

Valeur retournée :
Aucune

Notez que dans le cas des remplissages, par exemple, si on choisit la couleur blanche, la zone sera intégralement blanche, quelque soit le motif de remplissage.

Sélection du type de ligne

Cette fonction permet de décrire l'aspect de la ligne, telle qu'elle sera tracée au cours de l'appel des fonctions 05h (pour le tracé d'un cadre) et 06h.

Valeurs d'entrée :
AH = 0Bh
CX = Aspect de la ligne (sur 16 bits)

Valeur retournée :
Aucune

Le registre CX doit contenir l'image binaire de la ligne, qui sera répété autant de fois que nécessaire. Les valeurs courament admises sont :

```
FEDCBA9876543210  code  Type

----------------  FFFF  Ligne pleine (par défaut)
- - - - - - - -  AAAA  Trait interrompu court
--- --- --- ---  EEEE  Trait interrompu long
----- - ----- -  FAFA  Trait mixte
```

Définition du motif de remplissage

La fonction 05h (voir plus bas) permet de remplir une zone rectangulaire avec un motif prédéfini (hachures, motif cyclique...). La fonction suivante permet de sélectionner le motif courant.

Valeurs d'entrée :
AH = 01h
ES:DI : Adresse d'une table contenant le motif.

Valeur retournée :
Aucune

Le motif est décrit dans une table de 8 octets (8 * 8 bits). Par exemple, si nous désirons remplir une zone avec des hachures ayant une largeur de deux pixels, nous pourrons utiliser un motif tel que celui-ci :

```
Description motif  Valeur Hexa
-----------------  -----------
1 1 0 0 0 0 0 0    =    C0
1 0 0 0 0 0 0 1    =    81
0 0 0 0 0 0 1 1    =    03
0 0 0 0 0 1 1 0    =    06
0 0 0 0 1 1 0 0    =    0C
0 0 0 1 1 0 0 0    =    18
0 0 1 1 0 0 0 0    =    30
0 1 1 0 0 0 0 0    =    60
```

Ce motif sera répété verticalement et horizontalement autant de fois que nécessaire. Notez aussi qu'il aurait très bien pu être noté avec les codes suivants :

```
18 30 60 C0 81 03 06 0C
```

Ceci aurait créé le même motif, mais décalé de 3 pixels vers la droite.

Sélection de la loi de remplacement

Cette fonction permet de modifier la façon dont seront traitées les couleurs, au cours des opérations de tracé suivantes. Ceci peut permettre de créer certains effets intéressants, tels que la visualisation en négatif (au sens photographique du terme) d'une partie de l'écran.

Valeurs d'entrée :
AH = 0Ah
AL = Nouvelle loi de remplacement
00 : Force (défaut)
01 : AND
00 : OR
00 : XOR

Valeur retournée :
Aucune

Théoriquement, la fonction effectue une opération booléenne (AND, OR, XOR) entre le contenu de ce qui est déjà affiché et ce qui doit être tracé (trait interrompu, motife de remplissage...). Cependant, si la couleur noire est bien prise en compte, la couleur blanche (contenue dans le tracé) est ignorée dans le cas de l'opération FORCE, qui comme son nom l'indique devrait "forcer" l'affichage du nouveau tracé sur l'ancien.

Afin de rendre plus compréhensible le fonctionnement de cette fonction, voici deux tableaux montrant les résultats des tracés, suivant les paramètres d'entrée. La couleur courante est celle définie par la fonction 09h. La couleur du fond se rapporte à ce qui est déjà visible sur l'écran pour un pixel donné. La "couleur du trait" se rapporte à la couleur d'un pixel tel qu'il est défini (de façon constante) dans les paramètres des fonctions 01h et 0Bh. Le terme "Inchangé" indique que visuellement l'opération n'a eu aucun effet.

- Couleur courante : Noir

Operation	Couleur du Fond	Couleur du trait Blanc	Noir
FORCE	Blanc	Blanc	Noir
	Noir	Noir	Noir
AND	Blanc	Inchangé	
	Noir		
OR	Blanc	Blanc	Noir
	Noir	Noir	Noir
XOR	Blanc	Blanc	Noir
	Noir	Noir	Blanc

- Couleur courante : Blanc

Operation	Couleur du Fond	Couleur du trait Blanc	Noir
FORCE	Blanc	Blanc	Blanc
	Noir	Noir	Blanc
AND	Blanc	Blanc	Blanc
	Noir	Noir	Blanc
OR	Blanc	Inchangé	
	Noir		
XOR	Blanc	Inchangé	
	Noir		

Fonctions de tracé

Ce sont les fonctions ayant une action immédiatement visible.

Affichage d'un pixel

Cette fonction affiche un point, en utilisant la couleur courante.

Valeurs d'entrée :
AH = 07h
CX = Colonne
DX = Ligne

Valeur retournée :
Aucune

Après exécution, le curseur graphique est placé sur les coordonnées du point.

Lecture de la couleur d'un pixel

Cette fonction est l'inverse de la précédente.

Valeurs d'entrée :
AH = 0Ch
CX = Colonne
DX = Ligne

Valeur retournée :
AX = Couleur du pixel

La position du curseur graphique n'est pas affectée par cette fonction.

Affichage d'une ligne

Cette fonction trace une ligne entre la position courante du curseur et un point spécifié en CX,DX. Elle utilise le type de ligne, la couleur et le mode de remplacement courant.

Valeurs d'entrée :
AH = 06h
CX = Colonne finale
DX = Ligne finale

Valeur retournée :
Aucune

Après exécution, le curseur graphique est placé sur les coordonnées finales de la ligne.

Affichage d'un rectangle

Cette fonction permet de tracer des rectangles, de les entourer d'une bordure ou de les remplir à l'aide d'un motif prédéfini par la fonction 01h. Elle utilise le type de ligne, la couleur et le mode de remplacement courant.

Elle prend les coordonées du curseur graphique comme origine du tracé.

Valeurs d'entrée :
AH = 05h
AL = Type de tracé
00 : Cadre, utilisant le type de ligne défini par la fonction 0Bh.
01 : Remplissage uni, en utilisant la couleur définie par fonction 09h.
02 : Remplissage, en utilisant le

motif défini par la fonction 01h.
CX = Colonne finale
DX = Ligne finale
Départ : Position courante du curseur

Valeur retournée :
Aucune

Les coordonées du coin final peuvent être inférieures au coin d'origine.

Le curseur graphique n'est pas déplacé par cette fonction. Ceci permet de simplifier les tracés successifs (effacement, cadre, hachurage) sur un même rectangle.

L'origine du hachurage est toujours le coin supérieur gauche de l'écran, indépendement de la position du rectangle. Ceci implique que quelque soit la position de différents rectangles utilisant le même motif, les hachures coincideront parfaitement.

Fonctions de manipulation d'image

Ces fonctions permettent d'effectuer des manipulations sur une zone graphique. Il est ainsi possible de sauvegarder tout ou partie de l'écran dans un buffer, ou afficher (en un seul appel système) une image telle qu'une icone ou un logo.

Elles sont relativement équivalentes aux fonctions GROB de la HP48.

La première fonction permet de stocker dans un buffer le contenu d'une zone de l'écran graphique.

Valeurs d'entrée :
AH = 0Dh
CX = Colonne du premier coin
DX = Ligne du premier coin
BP = Colonne du deuxième coin
SI = Ligne du deuxième coin
ES:DI : adresse du buffer d'image.

Valeur retournée :
Aucune
(Buffer image initialisé)

Le buffer d'image a le format suivant :

Offset	Taille	Description
00h	WORD	Nombre de plans (toujours 1 sur le 95)
02h	WORD	Nombre de bits/pixel (1 sur le 95)
04h	WORD	Largeur de l'image en pixels
06h	WORD	Hauteur de l'image en pixels
08h	N BYTES	Image (N = (Largeur+7)/8 * Hauteur)

La zone image est organisée d'une façon équivalente à la fonction 01h, à savoir que 8 pixels d'une ligne sont stockés dans un octet.

La position du curseur graphique n'est pas affectée par cette fonction.

La deuxième fonction permet d'afficher une image, à partir du contenu d'un buffer (de format équivalent à celui de la fonction précédente).

Valeurs d'entrée :
AH = 0Dh
AL = Bit 2 : Inverse les couleurs l'inclusion.
Bit 1-0 : Loi de remplacement
00 : Force (défaut)
01 : AND
00 : OR
00 : XOR
CX = Colonne du premier coin
DX = Ligne du premier coin
ES:DI : adresse du buffer d'image.

Valeur retournée :
Aucune

La position du curseur graphique n'est pas affectée par cette fonction.

Le comportement de la loi de remplacement est légèrement différent par rapport à celui de la fonction 0Ah. En effet, la couleur blanche (définie par un code 0 dans le contenu du buffer) est ici réellement prise en compte, dans le sens où elle peut effacer une zone noire. Les opérations logiques sont donc parfaitement effectuées et l'intruction FORCE "force" réellement le tracé. Par contre, la couleur courante définie par la fonction 09h n'a aucune influence sur le tracé.

Si l'image ne tient pas en totalité dans l'écran, l'appel est ignoré.

Ecriture de texte

En mode graphique, les fonctions standard du DOS (fonction 09h de l'interruption 21h, par exemple) ne sont pas utilisables pour écrire une ligne de texte. Il est donc nécessaire d'utiliser la fonction suivante pour tout affichage de caractères.

Valeurs d'entrée :
AH = 0F
AL = 0 : Affichage standard (horizontal).
# 0 : Affichage vertical (bas vers haut).
CX = Colonne du coin sup/gauche du 1er caract.
DX = Ligne du coin sup/gauche du 1er caract.

ES:DI : Pointeur sur chaine, terminée par un caractère de code ASCII 0.

Valeur retournée :
Aucune

La position du curseur graphique n'est pas affectée par cette fonction.

Si un caractère (et les suivants) est placé (même partiellement) en dehors de l'écran ou d'une fenêtre, il ne sera pas affiché.

Notez que vous devrez toujours effacer le texte (en traçant un rectangle vide par dessus) avant de réécrire un nouveau texte, sinon il y aura superposition de l'ancien et du nouveau texte.

Accès direct en mémoire vidéo

Certaines personnes seront probablement tentées de travailler en accédant directement en mémoire vidéo. Si cela ne devrait pas poser trop de problèmes (je n'ai pas encore essayé), il faut qu'elles soient conscientes qu'elles risquent d'avoir des problèmes de comptabilité avec les successeurs du HP95LX. En effet, si il semble assuré que l'interruption 5Fh sera conservée, il est probable que, la taille de l'écran augmentant, le calcul de l'adressage d'un point soit différent. Jusqu'à réception d'informations supplémentaires, je ne ferais donc que conseiller d'utiliser les fonctions standard, même si cela influe sur les performances.

UNE PETITE BIBLIOTHEQUE GRAPHIQUE

Afin d'exploiter de façon simple les fonctions que nous venons de voir, je vous présente une petite bibliothèque de fonctions graphiques accessibles à partir du langage C. Notez qu'ayant écrit le corps des routines en assembleur (pour des raisons de simplicité et de rapidité), elle nécessite les Turbo C *et* Turbo Assembleur.

Si vous avez bien suivi la partie précédente de l'article, vous n'aurez surement pas de problème pour l'étudier. Elle ne contient en fait que les fonctions décrites précédement, sans variations ni nouvelles fonctions plus évoluées (tracé de cercles...).

Afin de permettre de l'inclure facilement dans un programme composé de plusieurs modules, la bibliothèque est composée de deux fichiers : `grafx95.c` qui contient les fonctions et `grafx95.h` qui contient la déclaration de différentes structures et constantes.

Plutôt que de redécrire en détail les fonctions, je n'expliquerai ici que certains points spécifiques.

Constantes prédéfinies

Afin de ne pas être confondues avec les constantes utilisées habituellement par Turbo C, les codes de couleurs sont renomés `CWHITE` et `CBLACK`. Il faut donc utiliser celles ci, et non `WHITE` et `BLACK`.

Les constantes `OUTLINE`, `SOLID`, `PATERN`, `FORCE`, `AND`, `OR` et `XOR` sont utilisées par les fonctions de tracé.

Les constantes *xxxx*`_LINE` donnent quelques exemples d'aspects de ligne. Elles doivent être utilisées de la façon suivante :

```
set_line_type(CENTER_LINE);
```

Les constantes *xxxx*`_FILL` donnent quelques exemples de motifs de hachurage. Les douze premiers sont identiques à ceux qui sont définis dans Turbo C et Turbo Pascal. Comme pour les aspects de ligne, elles peuvent être utilisées de la façon suivante :

```
set_fill_mask(SLASH_FILL);
```

Déclaration et manipulation d'une image

Le fait que la taille d'un buffer Image soit variable complique (en langage C) la définition de la structure la décrivant. En effet, un tableau contenu dans une structure doit obligatoirement avoir sa longueur définie par une constante. J'ai donc eu recours à une astuce, consistant à créer une macro chargée de définir une structure ayant les bons paramètres. Cette structure servira plus tard à la déclaration du buffer lui-même. Par exemple, si nous désirons créer une structure décrivant un buffer de 32×25 pixels, nous la créerons ainsi (en l'appelant par exemple `IMAGE_32_25`) :

```
DESCRIPT_IMAGE(32, 25, IMAGE_32_25)
```

Une fois ceci fait, il nous faudra déclarer la variable de la même façon que nous l'aurions fait avec une structure définie en `typedef` :

```
IMAGE_32_25 image01;
```

Si nous désirons créer une image prédéfinie (un logo par exemple), nous procèderons ainsi :

```
IMAGE_32_25 logo = {
                    1, 1, 32, 25,
                    {
                    0x00, 0x00, 0x00, 0x00,
                    0x3f, 0xc0, 0x3f, 0xe0,
                          .
                          .
                          .
                    0x1f, 0x80, 0x3f, 0xc0,
                    0x00, 0x00, 0x00, 0x00,
                    },
                   };
```

Les premiers chiffres initialisent le contenu de l'entête. Les deux premières valeurs devront toujours être égales à 1.

Après ceci, le buffer sera utilisé de la façon suivante :

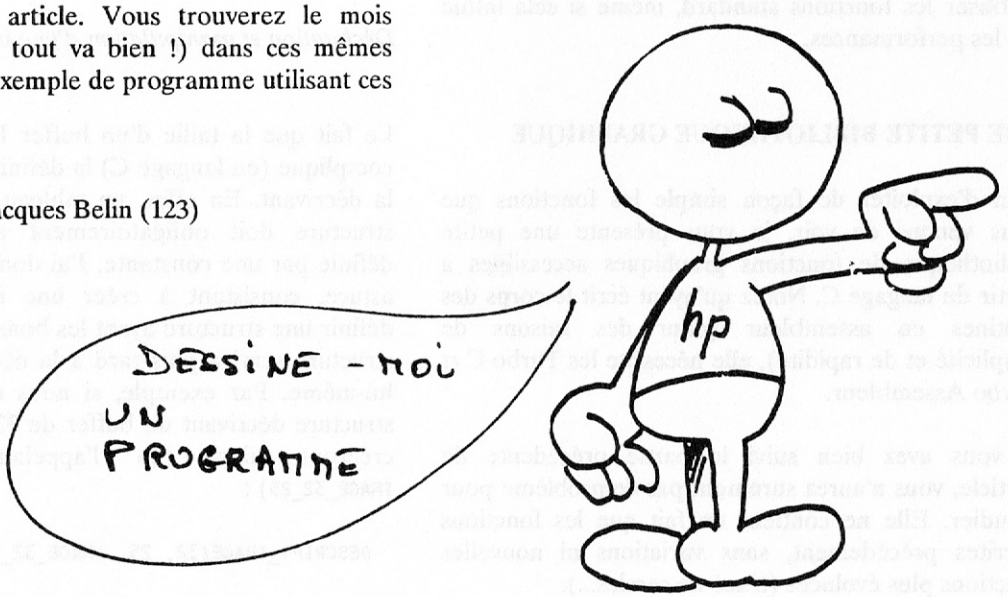
```
put_image((IMAGE *) &logo, 205, 99, INVERSE, AND);
```

Si vous utilisez des buffers de taille variable, il est plutôt conseillé de les allouer dynamiquement. la macro `LEN_IMAGE` vous indiquera la taille nécessaire à ce buffer, en fonction des dimensions de l'image et de l'entête du buffer :

```
buffer = farmalloc( (long) LEN_IMAGE(32, 17));
```

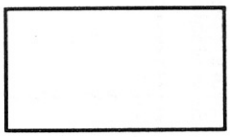
Ceci termine cet article. Vous trouverez le mois prochain (enfin, si tout va bien !) dans ces mêmes pages un premier exemple de programme utilisant ces fonctions.

Jacques Belin (123)

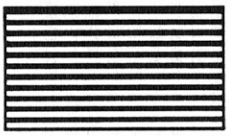


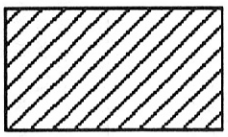
MOTIFS DE REMPLISSAGES DEFINIS DANS GRAFX95.H

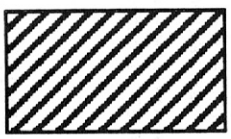
Motifs définis dans Turbo C et Turbo Pascal


EMPTY_FILL


SOLID_FILL

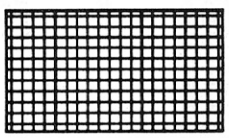

LINE_FILL

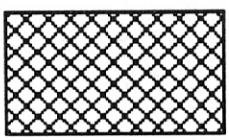

LTSLASH_FILL

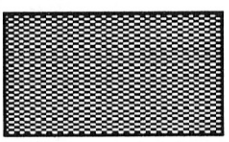

SLASH_FILL

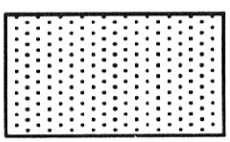

BCKSLASH_FILL

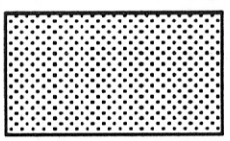

LTBKSLASH_FILL


HATCH_FILL

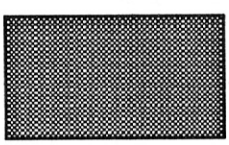

XHATCH_FILL


INTERLEAVE_FILL

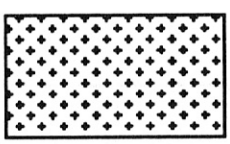

WIDE_DOT_FILL

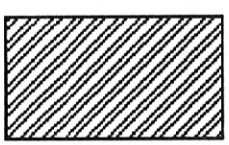

CLOSE_DOT_FILL

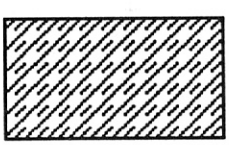
Autres Motifs

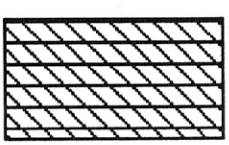

GRAY_FILL

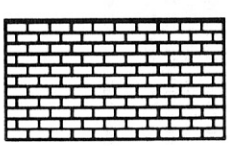

ZIGZAG_FILL

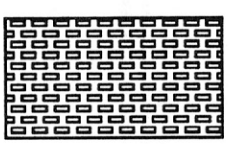

CROSS_FIL

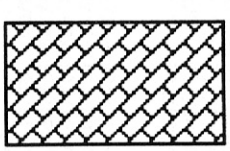

ALU_FILL

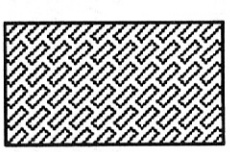

BRASS_FILL

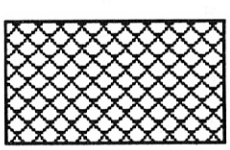

PLASTIC_FILL

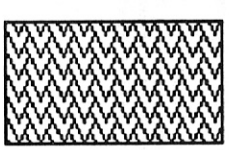

BRIKS1_FILL

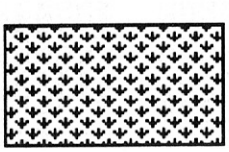

BRIKS2_FILL

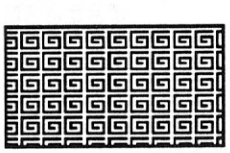

BRIKS3_FILL

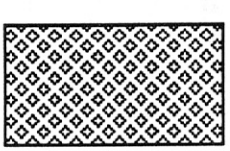

BRIKS4_FILL

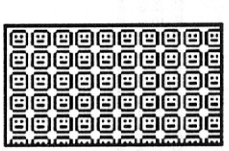

TILE1_FILL

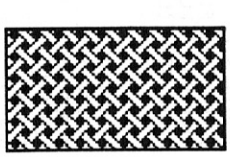

TILE2_FILL


GRASS_FILL

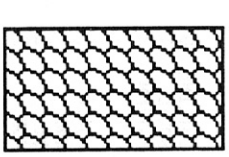

SPIRAL_FILL

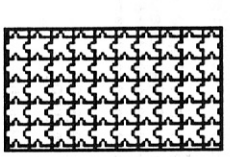

DIAMS_FILL


FACES_FILL


CLOTH_FILL


CUBES_FILL


SCALES_FILL


PUZZLE_FILL

```
/*****************************************************************************/
/*****************************************************************************/
/**                          GRAFX95.H                                      **/
/**          Fonctions graphiques pour le HP95 (fichier header)             **/
/**                                                                         **/
/**   Création : 25/02/93                          Auteur : Jacques Belin   **/
/**   Dernière modification : Ajout des motifs de remplissage               **/
/**                                                                         **/
/**  Note : Développé et testé uniquement sur Turbo C++ 1.01                **/
/**         TASM nécessaire (pour l'assemblage en ligne de grafx95.c)       **/
/*****************************************************************************/
/*****************************************************************************/

#ifndef  GRAFX95
#define  GRAFX95 1

#include <dos.h>

#define  CWHITE    0                          /* définition des couleurs */
#define  CBLACK    1

                                  /* Codes de fonctions pour draw_rectangle */
#define  OUTLINE   0                                    /* tracé de cadre */
#define  SOLID     1                                  /* rectangle plein */
#define  PATERN    2                                      /* hachurage */

                                              /* Lois de remplacement */

#define  FORCE     0
#define  AND       1
#define  OR        2
#define  XOR       3

#define  NORMAL    0                         /* flag d'inversion d'une image */
#define  INVERSE   1

#define  SOLID_LINE    0xFFFF                    /* types de lignes standard */
#define  DOTTED_LINE   0xAAAA
#define  CENTER_LINE   0xEEEE
#define  DASHED_LINE   0xFAFA

                          /* motifs de remplissages définis dans turbo C */
#define  EMPTY_FILL        "\x00\x00\x00\x00\x00\x00\x00\x00"
#define  SOLID_FILL        "\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF"
#define  LINE_FILL         "\xFF\xFF\x00\x00\xFF\xFF\x00\x00"
#define  LTSLASH_FILL      "\x01\x02\x04\x08\x10\x20\x40\x80"
#define  SLASH_FILL        "\xE0\xC1\x83\x07\x0E\x1C\x38\x70"
#define  BCKSLASH_FILL     "\xF0\x78\x3C\x1E\x0F\x87\xC3\xE1"
#define  LTBKSLASH_FILL    "\xA5\xD2\x69\xB4\x5A\x2D\x96\x4B"
#define  HATCH_FILL        "\xFF\x88\x88\x88\xFF\x88\x88\x88"
#define  XHATCH_FILL       "\x81\x42\x24\x18\x18\x24\x42\x81"
#define  INTERLEAVE_FILL   "\xCC\x33\xCC\x33\xCC\x33\xCC\x33"
#define  WIDE_DOT_FILL     "\x80\x00\x08\x00\x80\x00\x08\x00"
#define  CLOSE_DOT_FILL    "\x88\x00\x22\x00\x88\x00\x22\x00"

                                                  /* Autres motifs */
#define  GRAY_FILL         "\xAA\x55\xAA\x55\xAA\x55\xAA\x55"
#define  ZIGZAG_FILL       "\x88\x41\x22\x14\x88\x41\x22\x14"
#define  CROSS_FILL        "\x40\xE0\x40\x00\x04\x0E\x04\x00"
#define  ALU_FILL          "\x09\x12\x24\x48\x90\x21\x42\x84"
#define  BRASS_FILL        "\x11\x22\x44\x88\x10\x20\x40\x80"
#define  PLASTIC_FILL      "\xFF\x40\x20\x10\x08\x04\x02\x01"
#define  BRIKS1_FILL       "\xFF\x80\x80\x80\xFF\x08\x08\x08"
#define  BRIKS2_FILL       "\xFC\x84\xFC\x00\xCF\x48\xCF\x00"
#define  BRIKS3_FILL       "\x01\x02\x06\x09\x90\x60\x40\x80"
#define  BRIKS4_FILL       "\x44\x0A\x11\x22\x44\x88\x11\xA0"
#define  TILE1_FILL        "\x80\x41\x22\x1C\x08\x14\x22\xC1"
#define  TILE2_FILL        "\x82\x82\x44\x44\x2A\x2A\x11\x11"
#define  GRASS_FILL        "\x20\xA8\x70\x20\x02\x8A\x07\x02"
#define  SPIRAL_FILL       "\xFE\x80\xBE\xA2\xBA\x82\xFE\x00"
#define  DIAMS_FILL        "\x22\x50\x88\x50\x22\x05\x88\x05"
#define  FACES_FILL        "\x7C\x82\xAA\x82\xBA\x82\x7C\x00"
#define  CLOTH_FILL        "\xB8\x11\x8B\xC7\xA3\x11\x3A\x7C"
#define  CUBES_FILL        "\x3E\x46\xFE\x8E\xAE\x8C\xF8\x00"
#define  SCALES_FILL       "\xC1\x32\x0C\x04\x02\x02\x01\x01"
#define  PUZZLE_FILL       "\x01\x03\x04\x04\x23\x51\x51\xDF"


typedef struct                                  /* Table d'informations graphique */
        {
         char vmode;                                        /* mode vidéo courant */
         char def_vmode;                                      /* mode par défaut */
         int  disp_width;                        /* largeur de l'écran en pixels */
         int  disp_eight;                        /* hauteur de l'écran en pixels */
         int  pen_col;                  /* colonne courante du curseur graphique */
         int  pen_row;                    /* ligne courante du curseur graphique */
         int  line_type;                             /* type de ligne courant */
         int  rep_rule;                        /* loi de remplacement courante */
         int  color;                                    /* couleur courante */
         int  left_clip;                /* limite gauche de la fenêtre graphique */
         int  top_clip;                  /* limite haute de la fenêtre graphique */
         int  right_clip;               /* limite droite de la fenêtre graphique */
         int  bottom_clip;                /* limite base de la fenêtre graphique */
         int  col_orig;                        /* colonne de l'origine logique */
         int  row_orig;                          /* ligne de l'origine logique */
         char fill_mask[8];                    /* motif de remplissage courant */
        } GRAPHINFO95;


typedef struct                                                  /* buffer image */
        {
         int  planes;                           /* nombre de plans (1 sur HP95) */
         int  bits_by_pixel;            /* nombre de bits par pixel (1 sur HP95) */
         int  width;                             /* nombre de pixels en largeur */
         int  height;                            /* nombre de pixels en hauteur */
         char image[2];                            /* premier octet de l'image */
        } IMAGE;


                /* Macro permettant de calculer le la taille du buffer image */
#define LEN_IMAGE(width, height)    (8+(((width+7)/8)*height))

/*****************************************************************************/
/* DESCRIPT_IMAGE : descripteur d'un buffer Image                            */
/* paramètres : wdth, hght = largeur, hauteur de l'image en pixels           */
/*              NAME = type de structure                                     */
/* ATTENTION : Cette macro ne décrit que le type de structure (typedef).     */
/*             Elle doit être suivie d'une déclaration de variable utilisant */
/*             ce type de structure.                                         */
/*****************************************************************************/
#define DESCRIPT_IMAGE(wdth, hght, NAME)  typedef struct \
        {                                                \
         int  planes;                                    \
         int  bits_by_pixel;                             \
         int  width;                                     \
         int  height;                                    \
         char image[(((wdth)+7)/8)*(hght)];              \
        } NAME;
```

```
                        /* prototypes des fonctions incluses dans grafx95.c */
void init_graph(void);
void exit_graph(void);
void get_graph_info(GRAPHINFO95 *graphinfo);
void set_logorigin(int x, int y);
void set_clip_region(int left, int top, int right, int bottom);
void move_pen(int x, int y);
void set_color(char color);
void set_rule(char rule);
void set_line_type(int type);
void set_fill_mask(char *mask);
void plot(int x, int y);
int  get_pixel(int x, int y);
void draw_line(int x_dest, int y_dest);
void draw_rectangle(int x_dest, int y_dest, char fill_type);
void get_image(IMAGE *image, int x, int y, int nb_cols, int nb_lines);
void put_image(IMAGE *image, int x, int y, int invert, int rule);
void write_text(int x, int y, char *text, int rotate_flag);

#endif
```

```
/*****************************************************************************/
/*****************************************************************************/
/**                            GRAFX95.C                                    **/
/**                Fonctions graphiques pour le HP95                        **/
/**                                                                         **/
/**   Création : 25/02/93                          Auteur : Jacques Belin   **/
/**   Dernière modification : Modifications paramètres de put_image         **/
/**                                                                         **/
/**  Note : Développé et testé uniquement sur Turbo C++ 1.01                **/
/**         TASM nécessaire (pour l'assemblage en ligne de grafx95.c)       **/
/*****************************************************************************/
/*****************************************************************************/
#include "grafx95.h"

#pragma inline                          /* autorisation assemblage en ligne */


/*****************************************************************************/
/* INIT_GRAPH : Positionnement du HP95 en mode graphique                     */
/* Entrées : Aucune                                                          */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void init_graph(void)
   {
    asm   mov   ax,00h
    asm   mov   al,0A0h
    asm   int   5fh
   }


/*****************************************************************************/
/* EXIT_GRAPH : Retour en mode texte                                         */
/* Entrées : Aucune                                                          */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void exit_graph(void)
   {
    asm   mov   ah,00h
    asm   mov   al,7
    asm   int   10h
   }


/*****************************************************************************/
/* GET_GRAPH_INFO : Lecture de la table d'entrée vidéo                       */
/* Entrées : graphinfo = pointeur sur la table d'entrée                      */
/* Sortie  : graphinfo initialisé                                            */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void get_graph_info(GRAPHINFO95 *graphinfo)
   {
    int seg, off;

    seg = FP_SEG((void far *) graphinfo);
    off = FP_OFF((void far *) graphinfo);

    asm   mov   ah,02h
    asm   mov   es,seg
    asm   mov   di,off
    asm   int   5fh
   }


/*****************************************************************************/
/* SET_LOGORIGIN : positionnement de l'origine logique                       */
/* Entrées : x, y = coordonées de la nouvelle origine logique                */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void set_logorigin(int x, int y)
   {
    asm   mov   ah,03h
    asm   mov   cx,x
    asm   mov   dx,y
    asm   int   5fh
   }


/*****************************************************************************/
/* SET_CLIP_REGION : Positionnement de la fenêtre de travail                 */
/* Entrées : left, top = coordonées du coin supérieur gauche                 */
/*           right, bottom = coordonées du coin inférieur droit              */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void set_clip_region(int left, int top, int right, int bottom)
   {
    asm   mov   ah,04h
    asm   mov   cx,left
    asm   mov   dx,top
    asm   mov   si,right
    asm   mov   di,bottom
    asm   int   5fh
   }


/*****************************************************************************/
/* MOVE_PEN : Positionne le curseur graphique                                */
/* Entrées : x, y = coordonées du point                                      */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void move_pen(int x, int y)
   {
    asm   mov   ah,08h
    asm   mov   cx,x
    asm   mov   dx,y
    asm   int   5fh
   }
```

```
/*****************************************************************************/
/* SET_COLOR : sélection de la couleur courante                              */
/* Entrées : color = nouvelle couleur (1:Noir, 0:Blanc)                      */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void set_color(char color)
   {
    asm   mov   ah,09h
    asm   mov   al,color
    asm   int   5fh
   }


/*****************************************************************************/
/* SET_RULE : Sélection de la loi de remplacement                            */
/* Entrées : rule = loi de remplacement (FORCE, AND, OR, XOR)                */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void set_rule(char rule)
   {
    asm   mov   ah,0ah
    asm   mov   al,rule
    asm   int   5fh
   }


/*****************************************************************************/
/* SET_LINE_TYPE : sélection de l'aspect des lignes                          */
/* Entrées : type = aspect de la ligne (sur 16 bits)                         */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void set_line_type(int type)
   {
    asm   mov   ah,0bh
    asm   mov   cx,type
    asm   int   5fh
   }


/*****************************************************************************/
/* SET_FILL_MASK : Sélection du masque de remplissage                        */
/* Entrées : mask = pointeur sur le masque de remplissage (8 octets)         */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void set_fill_mask(char *mask)
   {
    int seg, off;

    seg = FP_SEG(mask);
    off = FP_OFF(mask);

    asm   mov   ah,01h
    asm   mov   es,seg
    asm   mov   di,off
    asm   int   5fh
   }


/*****************************************************************************/
/* PLOT : Affiche un pixel                                                   */
/* Entrées : x, y =coordonées du point                                       */
/* Utilise : couleur courante                                                */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void plot(int x, int y)
   {
    asm   mov   ah,07h
    asm   mov   cx,x
    asm   mov   dx,y
    asm   int   5fh
   }


/*****************************************************************************/
/* GET_PIXEL : Lecture de la couleur d'un pixel                              */
/* Entrées : x, y =coordonées du point                                       */
/* Valeur retournée : couleur = 1-> Noir, 0->Blanc                           */
/*****************************************************************************/
int get_pixel(int x, int y)
   {
    asm   mov   ah,0ch
    asm   mov   cx,x
    asm   mov   dx,y
    asm   int   5fh
    return(_AX);
   }


/*****************************************************************************/
/* DRAW_LINE : tracé d'une ligne à partir du point courant                   */
/* Entrées : x_dest y_dest = coordonées du point terminal                    */
/* Utilise : couleur, type de ligne et loi de remplacement courantes         */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void draw_line(int x_dest, int y_dest)
   {
    asm   mov   ah,06h
    asm   mov   cx,x_dest
    asm   mov   dx,y_dest
    asm   int   5fh
   }


/*****************************************************************************/
/* DRAW_RECTANGLE : tracé d'un rectangle à partir de la position courante    */
/*                  du curseur, tenant compte de la couleur et de la         */
/*                  loi de remplacement courante.                            */
/* Entrées : x_dest, y_dest = coordonnées du coin de fin                     */
/*           fill_type = Type de tracé                                       */
/*                       OUTLINE = tracé d'un cadre, utilisant le type de    */
/*                                 ligne courant.                            */
/*                       SOLID = Tracé d'un rectangle plein, utilisant       */
/*                               la couleur courante                         */
/*                       PATERN = Tracé d'un hachurage, tenant compte        */
/*                                du masque courant                          */
/* Utilise : couleur, type de ligne, masque de remplissage et loi de         */
/*           remplacement courantes                                          */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void draw_rectangle(int x_dest, int y_dest, char fill_type)
   {
    asm   mov   ah,05h
    asm   mov   cx,x_dest
    asm   mov   dx,y_dest
    asm   mov   al,fill_type
    asm   int   5fh
   }
```

```
/*****************************************************************************/
/* GET_IMAGE : Lecture d'une zone graphique dans un buffer                   */
/* Entrées : image = pointeur sur un buffer image                            */
/*           x, y = coordonnées du coin supérieur gauche de l'image          */
/*           nb_cols = nombre de colonnes de l'image                         */
/*           nb_lines = nombre de lignes  de l'image                         */
/* Sortie : buffer image initialisé                                          */
/* Valeur retournée : Aucune                                                 */
/* NOTE IMPORTANTE : Turbo C utilisant le registre BP pour la pile des       */
/*                   paramètres, il est nécessaire de le sauvegarder avant   */
/*                   de l'utliser pour 'x1'. L'interruption 5F semblant      */
/*                   restituer l'intégralité des registres après exécution,  */
/*                   j'ai utilisé BX pour la sauvegarde. Prions pour         */
/*                   que HP ne change rien dans le futur !                   */
/*****************************************************************************/
void get_image(IMAGE *image, int x, int y, int nb_cols, int nb_lines)
   {
    int seg, off;
    int x1, y1;

    x1 = x+nb_cols-1;
    y1 = y+nb_lines-1;
    seg = FP_SEG((void far *) image);
    off = FP_OFF((void far *) image);

    asm   mov   ah,0dh
    asm   mov   cx,x
    asm   mov   dx,y
    asm   mov   es,seg
    asm   mov   di,off
    asm   mov   si,y1
    asm   mov   bx,bp                          /* sauvegarde de BP dans BX */
    asm   mov   bp,x1
    asm   int   5fh
    asm   mov   bp,bx                                 /* restitution de BP */
   }


/*****************************************************************************/
/* PUT_IMAGE : Affichage d'une image                                         */
/* Entrées : image = pointeur sur un buffer image (déjà initialisé)          */
/*           x, y = coordonnées du coin supérieur gauche de l'image          */
/*           invert = si 1, l'image est inversée avant d'appliquer la loi de */
/*                    remplacement                                           */
/*           rule = loi de remplacement (FORCE, AND, OR, XOR)                */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void put_image(IMAGE *image, int x, int y, int invert, int rule)
   {
    int seg, off;
    int c;

    c = (invert<<2)+rule;

    seg = FP_SEG((void far *) image);
    off = FP_OFF((void far *) image);

    asm   mov   ah,0eh
    asm   mov   al,c
    asm   mov   cx,x
    asm   mov   dx,y
    asm   mov   es,seg
    asm   mov   di,off
    asm   int   5fh
   }
```

```
/*****************************************************************************/
/* WRITE_TEXT : Affichage d'une ligne de texte                               */
/* Entrées : x, y = coordonnées du coin supérieur gauche du 1er caractère    */
/*           texte = texte à afficher                                        */
/*           rotate_flag = si différent de 0, affichage en vertical          */
/* Valeur retournée : Aucune                                                 */
/*****************************************************************************/
void write_text(int x, int y, char *text, int rotate_flag)
   {
    int seg, off;

    seg = FP_SEG((void far *) text);
    off = FP_OFF((void far *) text);

    asm   mov   ah,0fh
    asm   mov   al,rotate_flag;
    asm   mov   cx,x
    asm   mov   dx,y
    asm   mov   es,seg
    asm   mov   di,off
    asm   int   5fh
   }
```

LE COIN DES CODES

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

Note importante :

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

Programmes HP48

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

Ceux qui n'ont pas encore de programme assembleur pourront procéder de la manière suivante :
- par mesure de sécurité sauvegardez vos programmes et fichiers, éventuellement verrouillez vos cartes RAM pour devenir ROMs.
- tapez le programme ASSCOD.

ASSCOD

1277h
884 octets

```
« RCLF HEX 64 STWS -2 SF 1 CF "" 'tmpcod' STO
  "nombre d'octets"  "" INPUT OBJ→ 2 * 16
  DUP2 / IP 3 ROLLD  MOD DUP2
  IF
  THEN 1 +
  END 3 ROLLD 1 + 4 ROLL
  # 0h → n r f s
    « 1 SWAP
      FOR i
        DO
          "ligne        "
          "00" i 1 - R→B →STR 3 OVER SIZE 1 -
          SUB + DUP SIZE DUP 2 - SWAP SUB + DUP
          "                    @ chaîne commençant
          chaine" +            @ par ← (newline)
          "                    @ chaîne commençant
          ___ ___ ___ ___" @ par ← (newline)
                                 @ et composée de 4
                                 @ groupes de 4
                                 @ CHR 95 obtenus par
                                 @ α shift bleu ×
                                 @ 1 espace séparant
                                 @ 2 groupes
          n i <
          IF
          THEN 1 r DUP 4 / IP + SUB
          END + 1 FC?C
          IF
          THEN { "" α }
          ELSE ROT
          END INPUT 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 s + DUP # FFFh AND "#"
          "                    @ "somme de controle"
          somme de controle    @ précédée et suivie
          ___"                 @ de ← (newline)
                               @ puis 3 CHR 95
                               @ (α shift bleu ×)
          7 ROLL SWAP + { "" α }
          INPUT + OBJ→ ==
          IF
          THEN SWAP DROP 1
          ELSE DROP2 1000 .5 BEEP 1 SF 0
          END
        UNTIL
        END 's' STO 'tmpcod' DUP RCL
        ROT + SWAP STO
      NEXT f STOF
    »
  tmpcod
  WHILE DUP " " POS DUP
  REPEAT DUP2 1 SWAP 1 - SUB 3 ROLLD 1 + MAXR
     SUB +
  END DROP
  "GROB 8 "                    @ si vous avez ASC→
    OVER SIZE 2 / " " +        @ JPC 79 page 14
    + SWAP + STR→          @ vous pouvez remplacer
    # 4017h SYSEVAL            @ ces lignes
    # 56B6h SYSEVAL            @ par ASC→
    DROP NEWOB                 @ (plus rapide)
»
```

Donc à partir de maintenant pour tout assemblage de chaîne de codes procédez de manière suivante :
1- lancez le programme ASSCOD.
2- donnez le nombre d'octets (1/2 oct. compris).
3- tapez chaque ligne de codes avec un espace entre chaque groupe de 4 caractères.
4- tapez la somme de contrôle. S'il y a erreur la ligne sera réaffichée pour correction après émission d'un

BEEP.
5- stockez le programme assemblé dans la variable donnée en tête.
6- si tout s'est bien déroulé vous pouvez purger tmpcod qui contient la chaîne de codes.

Programmes MS-DOS

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

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

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

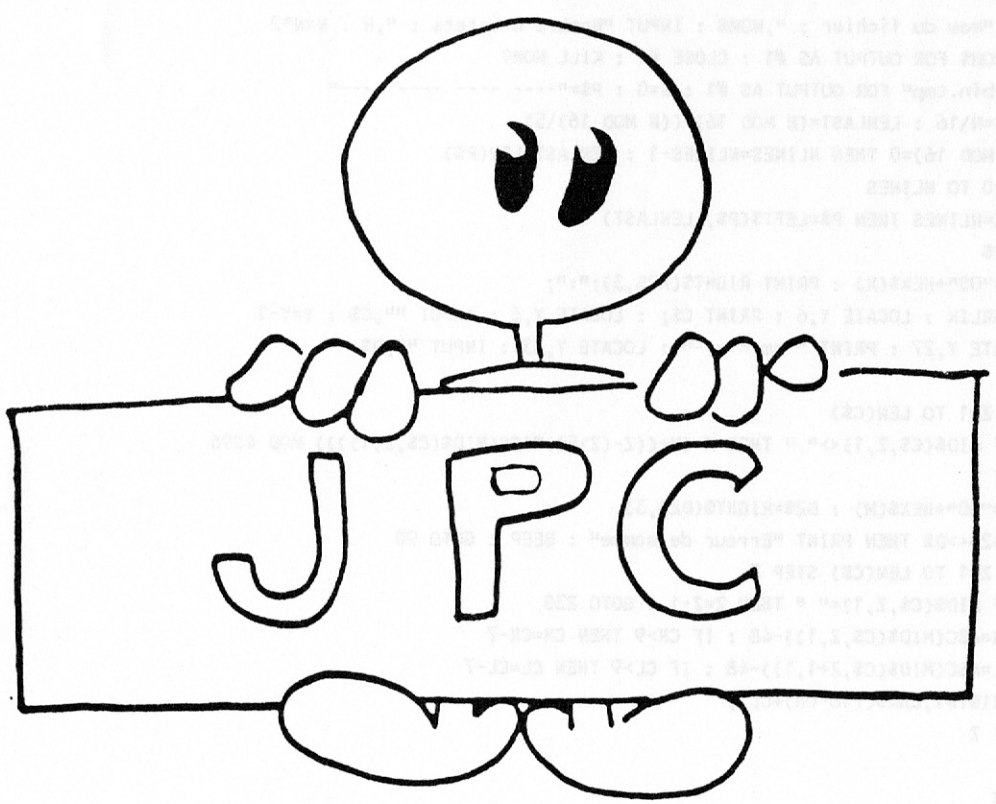
Programme MAKEDOS.BAS

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

Programmes HP28

Voir en page 9 de ce numéro.

```
FFACT28S.BIN                (HP28)
                          314 octets

      0123 4567 89AB CDEF   sm

000: 76C2 0DDB 9038 C11A   E5E
001: 4020 3392 0100 0000   862
002: 0000 0051 062B 802E   50D
003: 3E08 F3E0 76C2 06F5   33F
004: 21F3 4710 9F20 954E   0E0
005: 076C 20A4 020B 35C0   DE3
006: 5E02 0A40 20A4 020F   A95
007: FEE3 28D1 1ADE 1184   8CC
008: D115 E020 A402 0BEC   730
009: 1134 4118 4D11 E5E1   4D2
00A: 1ADE 11BE C115 8421   200
00B: B35C 069C 20BA 1008   F1F
00C: F72F 408F 1805 01BF   D2E
00D: 510C D214 484A C481   ABB
00E: 8F09 1017 3711 198F   7B2
00F: EA35 0502 7261 86A9   4A0
010: 08DE 3930 8F49 A408   26B
011: 5A68 DF13 610B 1343   F06
012: 4E4A 2014 4111 818F   BC8
013: 8416 4140 1648 18F8   8C1
014: 4AF2 D681 E061 3606   55F
015: 1091 35D6 8F24 0507   208
016: BF01 4313 1174 AF21   EDE
017: 0814 710A 0713 5D2E   C6E
018: 62B1 5511 1913 4118   7DB
019: D7D2 0605 AF01 5211   40A
01A: 1AAF 1822 81E8 3250   0B2
01B: A78A 748A ECE0 7A79   04C
01C: 1541 16BA 9281 2812   C98
01D: 8128 1204 CF59 B8AA   C0C
01E: D014 4110 E410 011A   7F1
01F: CE10 A8AE F820 07CE   750
020: D581 ED71 19C9 1340   437
021: 615A 015F 0159 015C   121
022: 0170 180C F57E 0713   E67
023: 5C91 3431 0315 F015   A57
024: C118 11C0 CD5F E7D0   9F0
025: 0113 1418 DCE5 218D   85A
026: 8B05 009F 2097 4E00   52C
027: 9F20                  747

RECOVERP                    (HP48)
# 379Ah                  219 octets

      0123 4567 89AB CDEF   sm

000: D9D2 0ECE 819F F30D   FDA
001: 9D20 AEC8 1CCD 20E8   F86
002: 1008 F146 608F B976   DAE
003: 0D31 B004 071F 0040   972
004: 78A8 32CC 8A8A 2CC8   96B
005: A8F2 DB10 88F7 3560   6D2
006: 8D48 D001 A795 01DD   512
007: 46F2 019C 2721 16B0   1B2
008: 0197 3760 1165 1421   CD2
009: 3034 0000 CCAD 86E0   B79
00A: 0142 1301 43D8 3120   6FF
00B: 14A9 6241 1601 328B   359
00C: 8391 3269 EF18 2340   017
00D: 4B20 1428 A251 3426   BFD
00E: B208 A290 1636 0CF1   965
00F: 6414 2818 F831 3610   554
010: 88FE 7950 D215 E38A   3D0
011: 2C01 1013 0629 FE71   11B
012: B975 071F 4750 7142   D2F
013: 147E E818 FA44 E381   B90
014: 8F84 1311 4011 0818   6CC
015: F841 411F B010 031F   399
016: 814D 32FF FA3E 5CF3   436
017: 1081 4D6C 9FD2 301D   2B8
018: A8F2 D760 8D32 0501   EFA
019: 5208 0863 B080 8714   B28
01A: 001D 230A 66DF B213   93A
01B: 0B21 30               D67
```



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** (Handhelds European Clubs EXchange), la fédération des principaux clubs Européens d'utilisateurs de calculateurs HP.

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

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

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

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