JPC

AVRIL 1993

NUMERO 85

Le numéro 40 F

A PROPOS DU CLUB

HP28

HP48

HP95

EDITORIAL

Comme le mois précédent, ce JPC est très en retard, et nous nous en excusons. Afin que cela ne se reproduise plus, nous avons déjà commencé à prendre des dispositions qui, si tout se passe bien, devraient nous permettre de résorber ce retard dès le prochain numéro. Sachez aussi que nous étudions aussi des méthodes visant à rétablir une participation, même symbolique, de la part de tous les adhérents. Ceci vous sera annoncé plus en détail dans le prochain numéro.

Puisqu'il est (enfin !) arrivé, parlons maintenant de ce JPC. La rubrique HP28 fera-t-elle concurence à la HP48 ? En tous cas, elle a maintenant tout ce qu'il lui faut grâce à l'article de Guy Toublanc. Nous vous rappelons que la durée de vie des HP peut dépasser allègrement les 10 ans, alors pensez à réveiller de temps en temps votre 28, ce lui fera plaisir !

Depuis le mois dernier, nous avons en notre possession le dernier opus des Goodies Disks de notre ami Joseph K. Horn. Nous aurions pu vous annoncer cette nouvelle dans le denier numéro, mais nous avons préféré attendre que Jean-François Garnier aie eu le temps de l'explorer, pour vous en faire une présentation complète, ce qui est fait dans ce numéro. Bien sûr tous ceux qui nous ont commandé les Goodies n°7 n'ont pas attendu la sortie de ce numéro pour recevoir le nouvelle disquette !

Les programmeurs sur HP95 ne trouverons dans ce numéro qu'un seul article (hormis l'habituel correctif !). Il est cette fois-ci accompagné d'un important listing en C. Faites nous savoir si vous jugez que sa taille est trop importante, ou si au contraire nous pouvons en publier de plus importants (si, bien sûr, il présente suffisement d'elements "originaux"). Rappelez vous que si nous nous efforcons de publier ces listings, c'est pour que vous puissiez profiter de peu d'experience que nous avons et que vous puissiez réutiliser tout ou partie de ces listings pour vos propres applications. Que les débutants se rassurent, un bon programme C ne fait pas obligatoirement la taille de celui présenté aujourd'hui !

En attendant le prochain numéro, nous vous souhaitons bonne lecture...

Le bureau

P. S. : Le dessin en page 2 montre le résultat que l'on peut obtenir avec le programme d'Alexandre Buchmann qui est décrit quelques pages plus loin...

NOUVELLES FRAICHES

Les HP95 version 512ko on vu leur prix baisser. Vous le trouverez maintenant en dessous de 3000 Francs dans vos boutiques préférées.

Il semblerait que HP ait prévu, pour des futures machines, d'implanter un mode *EXAM*, permettant de "brider" de façon sûre (peut être matérielle), les machines au cours des examens.

Courez chez votre libraire préféré pour savoir lequel de ces livres sortira en premier : *Voyage au centre de la HP48, Tome III* ou *Les secrets de la HP48, Tome II*. De toute façons, je suis sûr que vous prendrez lez deux !

Roland Sivipos, un ancien membre du club qui possède une petite société d'électronique, se propose d'étendre les possibilités vos HP48 et HP95. Pour la HP48, il ajoute une interface paralèlle, permettant de commander une imprimante ou d'effectuer des transferts de données rapides (drivers fournis). Pour le HP95, il augmente la mémoire interne en placant l'électronique d'une carte PCMCIA dans le faible espace situé entre le clavier et les cartes "normales". Elle est alors reconnue comme un disque "E:". Bien sûr, ces modifications font perdre toute garantie, mais la qualité du travail est assurée. Contactez le club pour les détails pratiques.

Jacques Belin (123)

COURRIER DU COEUR

Asdin Aoufi
Appt 2637
9, rue Charles Garnier
95140 GARGES LES GONNESSE
Tel (pro) : (1) 49 40 34 34

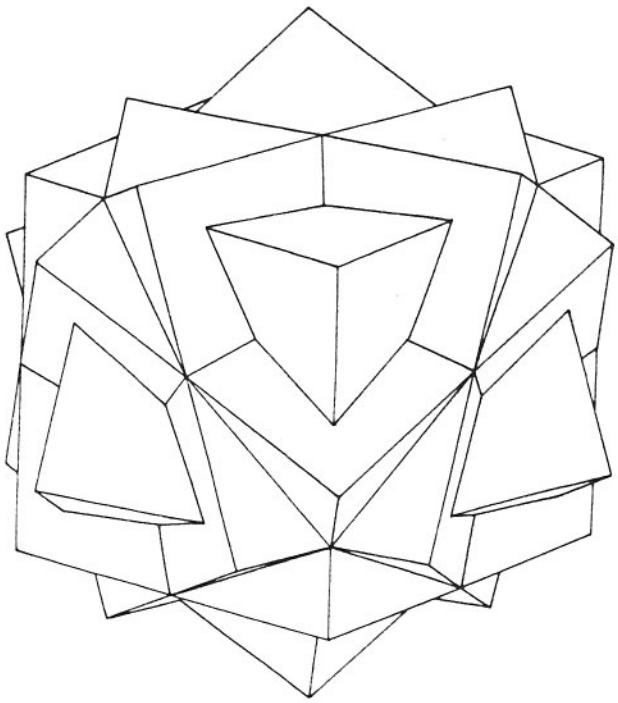
Vend :

HP48SX, HP42S, HP28S, prix à débattre.

Philippe Tenand
2 bis, rue de l'Egalité
94300 Vincennes
Tel : (1) 43 74 14 34
Vend :

HP48SX (sans manuel), livres d'application : HP-65 Commande numérique, HP33 Applications, HP33 Student Ingeniering, HP19/29 Finance, HP67/97 Game of Chance; 1 module Rom HP41 Real Estate. Prix à débattre.

D'autre part, Maubert Electronic nous a récement fourni une liste de matériel d'occasion concernant les HP-41 et HP-71 (machines, modules et périphériques). Contactez nous pour plus d'informations.



HP28

GESTIONNAIRE DE FICHIERS

Voici quelques programmes assez simples d'intérêt général pour la HP28S. Ils sont tous placés dans le répertoire HOME afin d'être facilement appelés.

SDIR

SDIR (S comme Super et DIR comme Directory) va lister de façon hiérachique l'ensemble des noms de répertoire et d'objets quelconques de chaque répertoire. Il n'utilise pas d'arguments placés dans la pile.

```
« PATH DUP SIZE DUP
SUB VARS DUP SIZE →
lv nv
  « { }
    IF nv 1 ≥
    THEN 1 nv
      FOR i lv i
        GET DUP
        IFERR RCL
        THEN EVAL
          DROP SDIR UP 1
          →LIST +
        ELSE DROP
        END
        IFERR +
        THEN
        END
      NEXT
    END
  »
»
```

XDIR

XDIR (X comme Execute) donne le même résultat final que SDIR mais permet en plus d'exécuter un programme (de façon systématique ou non) prg1 (donné dans la pile) et ce après chaque changement de (où retour à un) répertoire et d'un autre programme prg2 (de façon systématique ou non) pour chaque objet trouvé dans le répertoire actuel évalué.

XDIR utilise 4 arguments placés dans la pile de la manière suivante :

Niveau 4 : Prg1
Niveau 3 : f1
Niveau 2 : Prg2
Niveau 1 : f2

où Prg1 et Prg2 sont les objets à évaluer, sachant que juste avant l'évaluation de Prg1, la pile est vide (stockée par sécurité et vidée à nouveau après l'exécution de Prg1, puis restituée), et juste avant l'évaluation de Prg2, la pile contient le nom de l'objet actuel détecté dans le répertoire courant: 1:objet (la pile est aussi préservée de l'exécution de Prg2 par la même méthode). Les valeurs entières f1 et f2 sont comprises entre 0 et 3 et ont la signification suivante :

0 : pas d'évaluation momentanée de prg*i* (i=1 ou 2);
1 : évaluation momentanée de prg*i* (i=1 ou 2);
2 : pas d'évaluation de prgi pour tout répertoire (i=1) *ou bien* objet (i=2) de niveau hiérarchique (de répertoire) supérieur ou égal;
3 : évaluation systématique de prg*i* pour tout répertoire (i=1) *ou bien* objet (i=2) de niveau hiérarchique (de répertoire) supérieur ou égal.

Mode d'emploi:

Mettre initialement (en pile) f*i* (i=1 ou 2) égal à 1 ou 0 est indifférent; Mettre f*i* (i=1 ou 2) égal à 3 ou 2 initialement (en pile) provoque l'évaluation ou non (suivant f*i* (i=1 ou 2)) de prg*i* (i=1 ou 2) systématiquement pour tous les niveaux de répertoire.

La syntaxe des listes obtenues est la suivante :
2 : Répertoire initial
1 : { objet 1,objet 2,..,objet k, Répertoire1 {objet 1, objet 2,..,objet k, Répertoire 2 { objet 1,objet 2,.., objet k,..},..,Répertoire k{..}..}.

Chaque liste incluse est précédée du nom de répertoire qui contient les objets et répertoires de cette liste.

```
«
  0 0 → prg1 f1 prg2 f2 pil obj
  «
    xPATH DUP SIZE DUP SUB CLLCD DUP
    "Repertoire" SWAP →STR + 1 DISP
    IF f1 0 == f1 1 == OR
    THEN
      "EVAL. [O]ui,[N]on,[T],[J]amais de "
      prg1 →STR + 2 DISP DEPTH →LIST 'pil'
      STO
      {
        « prg1 EVAL »
        « prg1 EVAL 3 'f1' STO »
        « 2 'f1' STO »
      }
      DO
        { "O" "N" "T" "J" } WAITK UNTIL POS
      END
      LASTARG GET EVAL CLEAR pil LIST→ DROP
    ELSE
      IF f1 3 ==
```

```
  THEN
    DEPTH →LIST 'pil' STO prg1 EVAL CLEAR pil
    LIST→ DROP
  END
END
VARS DUP SIZE → lv nv
«
  { }
  IF nv 1 ≥
  THEN
    1 nv
    FOR i
      lv i GET DUP
      IFERR RCL
      THEN
      EVAL DROP prg1 f1 prg2 f2 XSDIR UP 1 →LIST
      + CLLCD "Repertoire" PATH DUP SIZE DUP SUB
      →STR + 1 DISP
        IF f1 1 == f1 0 == OR
        THEN
          "EVAL. [O]ui,[N]on,[T]oujours,[J]amais de "
          prg1 →STR DUP SIZE 1 - 2 SWAP SUB + 2
          DISP DEPTH →LIST 'pil' STO
          {
            « prg1 EVAL »
            « »
            « prg1 EVAL 3 'f1' STO »
            « 2 'f1' STO »
          }
          DO
            {
              "O" "N" "T" "J"
            } WAITK
            UNTIL POS
          END
          LASTARG GET EVAL CLEAR pil LIST→
          DROP
        ELSE
          IF f1 3 ==
          THEN
            DEPTH →LIST 'pil' STO
            prg1 EVAL CLEAR
            pil LIST→ DROP
          END
        END
      ELSE
        DROP DUP "Objet " SWAP
        →STR DUP SIZE 1 -
        2 SWAP SUB + 2 DISP
        IF f2 1 == f2 0 == OR
        THEN
          "EVALuation [O][N][T][J] du prog. "
          prg2 →STR DUP SIZE 1 -
          2 SWAP SUB + 3
          DISP 'obj' STO DEPTH
          →LIST 'pil' STO
          {
            « obj prg2 EVAL »
            « »
            « obj prg2 EVAL 3 'f2' STO »
            « 2 'f2' STO »
          }
          DO
            { "O" "N" "T" "J" } WAITK
            UNTIL POS
          END
          LASTARG GET EVAL CLEAR pil LIST→
          DROP obj
        ELSE
          IF f2 3 ==
          THEN
            'obj' STO DEPTH →LIST
            'pil' STO obj
            prg2 EVAL CLEAR pil
            LIST→ DROP obj
          END
        END
      END
      IFERR +
      THEN
      END
    NEXT
  END
»
```

CHDIR

Il transforme une liste obtenue par SDIR ou XDIR en une chaîne indentée ou structurée et donne le nombre de lignes (une ligne étant formée par le caractère 10 inséré), ce nombre est utile à PRDIR et SCTXT.

```
« →STR DUP SIZE 0 0
  → ch n tab ncr
  «
    IF ch "{" POS n 3 ≥ AND
    THEN
      "" 1 n
      FOR i
        ch i i SUB DUP { "{" "}" }
        SWAP POS 1 +
        {
          « + »
          «
            DROP 10 CHR + ncr 1 + 'ncr'
            STO tab SPC + "{" + tab 2 + 'tab' STO
          »
          «
            DROP 10 CHR + ncr 1 + 'ncr' STO tab 2 -
            'tab' STO tab SPC + "}" +
          »
        }
        SWAP GET EVAL
      NEXT
```

```
    END
    ncr 1 + 2 *
  »
»
```

PDIR

Il donne à l'utilisateur le choix de visualiser la chaîne sortie par CHDIR sur imprimante ou sur écran. Le choix [E] (touche E appuyée) exécute le programme SCTXT.

```
« → ch ncr
  «
  ch
    {
      « DUP 1 DISP PR1 »
      « ch ncr SCTXT »
    }
    DO
    { "I" "E" }
      DO
        "IMPRESSION          @
        [I]mprimante ou      @  Chaine Commune
        [E]cran        "     @
        1 DISP
      UNTIL KEY
      END
      UNTIL POS
    END
    LASTARG GET EVAL
  »
»
```

SCTXT

Il demande les mêmes arguments en pile que PDIR, un menu d'interruption s'affiche, attendant l'appui sur l'une des 6 touches programmées :
- [INS] : affichage de la page suivante (scrolling de 4 lignes vers le haut).
- [DEL] : affichage de la page précédente (scrolling de 4 lignes vers le bas).
- [UP] : affichage de la ligne suivante (scrolling de 1 ligne vers le haut).
- [DOWN] : affichage de la ligne précédente (scrolling de 1 ligne vers le haut).
- [LEFT] : affichage et positionnement sur la première page.
- [RIGHT] : affichage et positionnement sur la dernière page.
- L'appui sur toute autre touche fait sortir du programme principal (PRDIR ou SCTXT).

```
«
  SWAP DUP SIZE 1 → ncr ch n i
  «
    DO
      {
        « 1 »
        « 4 i 4 + MIN 'i' STO 0 »
        « i 4 - ncr NEG 4 + MAX 'i' STO 0 »
        « 4 i 1 + MIN 'i' STO 0 »
        « i 1 - ncr NEG 4 + MAX 'i' STO 0 »
        « 1 'i' STO 0 »
        « ncr NEG 4 + 'i' STO 0 »
      }
      ch n i NDISP
      { "DEL" "INS" "DOWN" "UP" "LEFT" "RIGHT" }
      DO
        UNTIL KEY
      END
      CLLCD POS 1 + GET EVAL
      UNTIL
    END
    CLMF
  »
»
```

CUT2

C'est un sous-programme de SCTXT. Il se charge de couper la chaîne 'ch' de longueur 'n' après le 'nième' caractère ou chaîne 'car' de longueur 'v' rencontré dans la chaîne 'ch'. CUT2 ne retourne alors que la partie droite de chaîne 'ch' ainsi coupée. L'exécution de CUT2 est proportionnelle à la valeur de 'nième'.

```
«
  0 → ch u niem car v k
  «
    ""
    IF u 1 ≥ v 1 ≥ u v 1 - ≥ AND AND
    THEN
      1 u v - 1 +
      FOR i
        ch i i v + 1 - SUB car SAME
        «
          k 1 + 'k' STO k niem ≥
          «  ch i v + u SUB u v - 1 + 'i' STO »
          IFT
        »
        IFT
      NEXT
    END
    IFERR +
    THEN
    END
  »
»
```

NPOS

Cet utilitaire n'a pas été utilisé ici mais peut rendre service. Il retourne la liste et le nombre de toutes les positions de la petite chaîne 'car' de longueur 'nc' rencontrée dans la chaîne 'ch' de longueur 'nch'.

```
«
  0 0 → ch nch car nc i np
  «
    ch 1 nch SUB car 1 nc SUB → ch car
    «
      { }
      IF nch nc * nch nc ≥ * nch nc == +
      THEN
        WHILE ch car POS DUP 'i' STO
        REPEAT
          ch 1 i 1 - SUB ch i nc + nch nc + SUB + 'ch'
          STO i nc np * + + np 1 + 'np' STO
        END
      END np
    »
  »
»
```

NDISP

Cet utilitaire est utilisé par SCTXT. En entrant dans l'ordre :

3 : chaîne à afficher
2 : longueur de chaîne
1 : n, valeur algébrique entière (positive ou négative) correspondant au numéro de ligne de l'écran (réel ou virtuel), sachant que la numérotation est la même (mais étendue) que celle utilisée pour la commande DISP.

```
«
  → ch n d
  «
    d 1 ≥
    « ch d DISP »
    « ch n 1 d - 10 CHR 1 CUT2 1 DISP »
    IFTE
  »
»
```

WAITK

```
« DO UNTIL KEY END »
```

UP

```
« PATH DUP SIZE 1 - DUP 0 == + GET EVAL »
```

SPC

```
« → n
  «
    ""
    IF n 1 ≥
    THEN
      1 n
      START
        " " +
      NEXT
    END
  »
»
```

Paul JEBEILY (571)

SAGA HP28S
BYTES & TUTTI QUANTI

Dans l'article *HP28S RETRO* (JPC 84), je faisais allusion au problème dramatique des *Memory Lost* et fournissais une méthode fiable d'introduction des codes hexa à assembler. Le programme principal faisant ce travail comporte un assembleur en User Rpl. Cet assembleur ne fait pas de vérification de l'argument car cela est fait en amont. Il a aussi la possibilité pour ceux qui veulent un assembleur autonome d'assembler ce programme assembleur et dans le programme ASSCOD.28 d'appeler ce sous programme assembleur. Cette solution a l'avantage d'un encombrement mémoire moins important et aussi d'offrir la possibilité d'assembler des codes hexa non listés avec leurs sommes de contrôle. Cette dernière possibilité doit au moins se faire en vérifiant la validité de l'argument: une chaîne non nulle. Aussi voici une version de ASC→ qui fait cette vérification et donne les messages :

```
Too Few Arguments  si il n'y a rien sur la pile
Bad Argument Type  si ce n'est pas une chaîne
Bad Argument Value si c'est une chaîne nulle
```

En donnant cette nouvelle version de ASC→ cela me permet d'apporter les précisions suivantes :

Dans les listings on trouve à la fois les adresses des objets internes (primitives, programmes etc..) et les mnémoniques correspondantes pour la HP48 quand cela est possible. Mais ces appellations ne doivent pas être considérées comme définitive car pouvant être remises en cause.

Par exemple:
- DOCOL ou :: : début de structure programme. Met sur la pile des retours l'adresse de l'objet suivant à exécuter.

- CK1 vérifie si DEPTH >= 1 sinon renvoie Too Few Arguments.
- SAVPTR sauve D0 D1 B(A) et D(A).
- GETPTR restaure D0 D1 B(A) et D(A).
- Errjmp arrête tout en affichant le message de l'erreur dont le numéro est dans A(A).
- GETPTRLOOP restaure D0 D1 B(A) D(A) et retourne au RPL.
- TOTEMPOB recrée l'objet sur la pile.
- SEMI ou ; : fin de structure programme (enlève un niveau de pile de retours qui est l'adresse de l'objet suivant à exécuter.

Lorsque des commandes User Rpl sont utilisées, un *x* les précède.
Pour abréger les commentaires, dans ceux-ci "@" veut dire *pointe sur.*

ASC→

```
ASSEMBLE
        con(5)  #02c67      * DOCOL
        con(5)  #0c3e1      * CK1
        con(5)  #02c96      * DOCODE
        rel(5)  fin         * offset fin code
        gosbvl  #05081      * SAVPTR
        a=dat1  a           * adr. objet 1er niv.
        d0=a                * @ l'objet
        d1=a                * idem
        lchex   02a4e       * prologue chaine
        a=dat1  a           * prologue de l'objet
        ?a=c    a           * une chaine ?
        goyes   string      * oui on continue
        lahex   0202        * non : bad arg. type
err     gosbvl  #050b8      * GETPTR
        govlng  #0396a      * Errjmp
string  d1=d1+  5           * @ longueur
        a=dat1  a           * longeur + 5
        a=a-con a,6         * longueur - 1
        gonc    nonull      * si no carry : saut
        lahex   00203       * sinon bad arg. val.
        goc     err         * saut pour erreur
nonull  asrb.f  a           * nombre caract. - 1
        b=a     a           * compteur caractères
        d0=d0+  10          * @ début codes
        d1=d1+  5           * idem
        lchex   39
loop    a=dat1  2           * conversion
        ?a<=c   b           * ASCII -> HEXA
        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      * TOTEMPOB
        con(5)  #02f90      * SEMI
```

La fonction BYTES de la HP48 serait bien utile dans une HP28. Elle permet de déterminer à la fois l'encombrement mémoire d'un objet et la somme de contrôle qui caractérise un objet. Aussi je vous livre cette fonction. La rapidité du programme ne peut être comparée honnêtement avec celle de la HP48. En effet celle-ci a un circuit intégré lui permettant de faire ce calcul plus rapidement que par soft uniquement.

Quelques adresses à commenter :
- ?TYPEIDNT renvoie le booléen TRUE si l'objet est un nom global sinon FALSE.
- DUP c'est le DUP interne.
- ITE si le booléen est TRUE alors on exécute l'objet suivant sinon on saute au 2ième objet. C'est l'équivalent de IF THEN ELSE.
- rcl_IDNT rappelle l'objet dont le nom est sur la pile après avoir fait un DUP. Si l'objet n'existe pas alors arrêt et affichage de Undefined Name sinon l'objet est rappelé sur la pile.
- SWAP c'est le SWAP interne.
- BINT_0 est l'entier binaire 0.
- SKIPOB routine interne qui fait pointer D0 sur l'objet suivant.
- #>% conversion d'un binaire en réel.
- %2 réel 2.
- %/ divisision de 2 réels.
- PUSH#LOOP pousse sur la pile l'entier binaire dont la valeur est en R0(A) puis retour au RPL.

Le principe du programme est le suivant :

On vérifie s'il y a un objet sur la pile avec CK1 on teste s'il y a sur la pile un nom global.
Si oui, on rappelle l'objet et on recrée l'entier binaire #0 en mémoire. On stocke dans celui-ci la place mémoire occupée par le nom : prologue (5 nibs) + 2 fois 2 nibs pour le nombre de caractères + le nombre de caractères * 2.
Sinon on ne crée que l'entier binaire #0.
Puis dans les deux cas on détermine la place en mémoire occupée par l'objet. Ce nombre de quartets sera ajouté à celui, éventuel du nom. Enfin ce sera le calcul de la somme de contrôle.

Syntaxe:

L'objet ou son nom sur la pile,

```
BYTES  ASC→
```

donnera :
  niveau 2 : cheksum
  niveau 1 : nombre d'octets

exemples:
sur la pile « 1 + 2 »

```
BYTES   ASC→
```

donnera :
  # A75Ah
  17.5

Le super programme ci-dessus est stocké dans TROIS

sur la pile 'TROIS'

```
BYTES   ASC→
```

donnera :
  # A75Ah
  27

Avec une HP48, si on obtient les mêmes nombres d'octets. En revanche le cheksum est différent, ce qui est normal car le codage en mémoire n'est pas le même pour les deux machines.

BYTES.28S

```
ASSEMBLE
        con(5)  #02c67      * DOCOL
        con(5)  #0c3e1      * CK1
        con(5)  #0204a      * DUP
        con(5)  #3e7f1      * ?TYPEIDNT
        con(5)  #3dca3      * ITE
        con(5)  #02c67      * DOCOL
        con(5)  #0bffd      * rcl_IDNT
        con(5)  #020e5      * SWAP
        con(5)  #10c2f      * BINT_0
        con(5)  #04f3d      * TOTEMPOB
        con(5)  #020e5      * SWAP
        con(5)  #02c96      * DOCODE
        rel(5)  fin1
        c=dat1  a           * adresse du nom
        d1=d1+  5           * drope le nom
        d=d+1   a           *
        cd1ex               * @ le nom
        d1=d1+  5           * @ nombre de caract.
        a=0     a
        a=dat1  b           * nombre de caract.
        a=a+a   a           * nombre de quartets
        a=a+con a,9         * total pour le nom
        d1=c                * @ adr. entier bin.
        c=dat1  a           * adr. entier binaire
        cd1ex               * @ prologue
        d1=d1+  10          * @ valeur entier bin
        dat1=a  a           * sauve long. du nom
        d1=c                * restaure D1
        govlng  #125ec      * retour RPL
fin1
        con(5)  #02f90      * SEMI
        con(5)  #02c67      * DOCOL
        con(5)  #10c2f      * BINT_0
        con(5)  #04f3d      * TOTEMPOB
        con(5)  #02f90      * SEMI
        con(5)  #020e5      * SWAP
        con(5)  #02c96      * DOCODE
        rel(5)  fin
        a=dat1  a           * adr. objet
        d1=d1+  5           * @ adr. ent. binaire
        d=d+1   a           * et drope adr. objet
        cd1ex               *
        rstk=c              * sauve cette adresse
        d1=c                * restaure D1
        gosbvl  #5081       * GETPTR
        d0=(5)  #fff00      * adresse pour SPEED
        lchex   f           * pour vitesse maxi
        dat0=c  1           * stockage
        c=a     a           * adresse objet
        d0=c                * @ l'objet
        rstk=c              * sauve cette adresse
        gosbvl  #02e94      * SKIPOB
        c=rstk              * récupère adr. objet
        d1=c                * @ l'objet
        ad0ex               * adr. objet suivant
        c=a-c   a           * longueur objet
        r0=c                * sauve la longueur
        d=c     a           *
        d=d-1   a           * compteur quartets
        c=rstk              * adr. entier binaire
        d0=c                * @ adr. entier bin.
        a=dat0  a           * adresse entier bin.
        d0=a                * @ l'entier binaire
        d0=d0+  10          * @ la valeur
        c=0     a
        lchex   f
        r2=c                * R2(A) := 0000F
        c=0     a           * initialise cheksum
loop    rstk=c              * sauve cheksum
        a=dat1  1           * quartet de l'objet
        gosub   xor         * XOR avec le cheksum
        a=r2.f  a           * A(A) := 0000F
        a=a&c   a           * 0000F AND cheksum
        c=a     a           *
        p=      6           *
doubl   a=a+a   a           * boucle pour obtenir
        p=p-1               * cheksum * 2^7
        gonc    doubl       *
        c=a+c   a           * chksum * ( 2^7 + 1)
        p=      4
Doubl   a=a+a   a           * boucle pour obtenir
        p=p-1               * cheksum * ( 2^12 )
        gonc    Doubl
        a=a+c   a           * chks*(2^12 +2^7 +1)
        c=rstk              * récupère l'ancien
        csr     a           * et divise par 16
        gosub   xor         * puis XOR avec nouv.
        d1=d1+  1           * @ quartet suivant
        d=d-1   a           * actualise compteur
        gonc    loop        * no carry alors saut
```

```
        cr0ex               * sinon sauve cheksum
        a=dat0  a           * longueur du nom si
        a=a+c   a           * long. objet + nom
        ar0ex               * sauve en R0
        dat0=a  a           * chks -> entier bin.
        gosbvl  #050b8      * GETPTR
        c=0     a           * enfin on pousse sur
        p=      0           * la pile 1 entier
        govlng  #17523      * binaire := R0(A)
xor     b=c     a           * duplique C(A)
        b=b!a   a           * fait A(A)  OR B(A)
        c=a&c   a           * fait C(A)  AND A(A)
        c=-c-1  a           * complément - 1
        c=c&b   a           * fait C(A)  AND B(A)
        rtn
fin     con(5)  #31e22      * #>%
        con(5)  #112de      * %2
        con(5)  #11d82      * %/
        con(5)  #02f90      * SEMI
```

Pour identifier et comparer les différents types d'objets utilisateur nous possédons maintenant un moyen fiable. Le *Coin des codes* pour HP-28S sera l'égal de celui de la HP-48 avec cette nouvelle fonction.

Le compère obligé de ASC→ c'est évidemment →ASC qui fait l'inverse en transformant un objet en son code hexadécimal. Le principe de ce programme est simple :

Calcul du nombre de quartets occupés par l'objet en soustrayant l'adresse de l'objet de celle de l'objet suivant via SKIPOB. On réserve un espace chaîne, à l'aide de res_str, et de longueur égale au double du nombre de quartets du code de l'objet plus 10 pour le prologue et la longueur. Enfin le code hexadécimal de chaque quartet est converti en ASCII.

Une adresse non listée ci-dessus :

- res_str (dénomination non officielle) réserve un espace chaîne. La longueur totale en quartets avec le prologue compris doit être en C(A). D0 D1 B(A) D(A) seront sauvegardés par res_str en début de routine donc bien les restituer avant d'appeler res_str. Au retour D0 pointe sur le futur premier caractère, R0(A) contient l'adresse du prologue et R1(A) l'adresse de la longueur de la chaîne qui est *à déterminer par le programmeur* donc ne pas se faire piéger comme moi (voir JPC 84 p 11).

L'explication de ceci est que cette routine est utilisée pour convertir un réel en chaîne et que l'espace chaîne réservé est d'office: # 30h. Puis la routine déterminera la longueur exacte suivant le format (STD, FIX, SCI).

Syntaxe:

```
l'objet sur la pile →ASC → la chaîne
```

exemple:

```
« + »   →ASC   →   "76C200D9E0EEE806E9E009F20"
```

inversement

```
"76C200D9E0EEE806E9E009F20"   ASC→   →   « + »
```

→ASC

```
ASSEMBLE
        con(5)  #02c67      * DOCOL
        con(5)  #0c3e1      * CK1
        con(5)  #02c96      * DOCODE
        rel(5)  fin
        gosbvl  #05081      * SAVPTR
        c=dat1  a           * adresse objet
        d0=c                * @ l'objet
        rstk=c              * sauve adr. sur RSTK
        gosbvl  #02e94      * SKIPOB
        c=rstk              * récupère adr. objet
        ad0ex               * adresse next objet
        a=a-c   a           * longueur objet
        r3=a                * sauve cette long.
        a=a+a   a           * double longueur
        a=a+con a,10        * +10 : prol. + long.
        r2=a                * sauve ce résultat
        gosub   load        * appelle GETPTR
        c=a     a           * espace à réserver
        gosbvl  #1e9ba      * res_str
        d0=d0-  5           * @ longueur chaîne
        a=r2                * longueur totale
        a=a-con  a,5        * - 5 : du prologue
        dat0=a   a          * charge la longueur
        d0=d0+   5          * @ 1er caractère
        ad0ex               * adresse 1er caract.
        gosub   load        * appelle GETPTR
        d0=a                * @ 1er caractère
        a=dat1  a           * adresse objet
        d1=a                * @ objet
        c=r3                * longueur objet
        d=c     a           * en quartets
        d=d-1   a           * compteur quartets
        lchex   39          * pour conversion
loop    a=c     b           * conversion
        a=dat1  1           * HEXA -> ASCII
        ?a<=c   b           *
        goyes   inf10       *
        a=a+con b,7         *
inf10   dat0=a  2           * charge le caractère
        d1=d1+  1           * @ quartet suivant
        d0=d0+  2           * @ caract. suivant
        d=d-1   a           * actualise compteur
```

```
        gonc    loop        * si no carry : saut
        gosub   load        * sinon restaure ptr
        c=r0                * adresse chaîne dans
        dat1=c  a           * adresse 1er niveau
        govlng  #125ec      * retour RPL
load    govlng  #050b8      * GETPTR
fin
        con(5)  #02f90      * SEMI
```

La commande RCL de la HP28 ne permet pas de rappeler sur la pile un répertoire contrairement à ce qui est possible avec une HP48. Aussi, si vous devez transférer un répertoire il faut transférer une à une les variables du répertoire. Pour éviter cela voici la commande RCLDIR.

Donc pour rappeler un répertoire sur la pile il suffit de mettre son nom puis RCLDIR

Juste une nouvelle adresse:

- rcl_gn/ln qui rapelle une variable globale ou locale si elle existe et renvoie un booléen : TRUE si cette variable existe sinon FALSE.

Donc un petit programme de 15 octets seulement.

RCLDIR

```
ASSEMBLE
        con(5)  #02c67      * DOCOL
        con(5)  #0c3e1      * CK1
        con(5)  #1e737      * rcl gn/ln
        con(5)  #02106      * DROP
        con(5)  #02f90      * SEMI
```

La commande PURGEDIR fait défaut sur la HP-28. Si l'on veut purger un répertoire il faut aller dans le sous-répertoire le plus profond, purger une à une ses variables et refaire le même travail avec le répertoire parent et ainsi de suite. Cela est fastidieux. Pour remédier à cela voici encore une nouvelle commande: PURGEDIR.

Son emploi est le même que PURGE.

Quatre adresses à commenter :

- Dispatch1 vérifie que l'argument est bien du type précisé par le system binary (entier court) qui suit cette adresse. Sinon recherche une autre une autre option. Si l'argument ne convient pas alors arrêt avec message Bad Argument Type.
- SIX system binary correspondant au type nom global
- rcl_gn&err (dénomination non officielle) rappelle la variable précisée par le nom. S'il y a erreur arrêt avec message: *Undefined Name*.
- purge_gn purge l'objet correspondant au nom.

Le principe du programme est le suivant :

Après verification qu'il y bien un objet sur la pile et que cet objet est un nom global, on tente de rappeler cet objet. S'il existe, alors on le purge.

PURGEDIR

```
ASSEMBLE
        con(5)  #02c67      * DOCOL
        con(5)  #0c3e1      * CK1
        con(5)  #0c7a5      * Dispatch1
        con(5)  #071e5      * SIX type GN
        con(5)  #02c67      * DOCOL
        con(5)  #0bffd      * rcl_gn&err
        con(5)  #02106      * DROP
        con(5)  #06a25      * purge_gn
        con(5)  #02f90      * SEMI
        con(5)  #02f90      * SEMI
```

Enfin une troisième commande de la HP48 absente de la HP28 : DEBUG.

Si l'on veut suivre l'exécution d'un programme pas à pas il faut éditer le programme pour y inclure HALT. Pour revenir à un exécution normale il faut rééditer ce programme pour enlever cette commande. Tout cela est perte de temps surtout en éditant un programme encombrant. Donc encore une fois il faut créer cette commande DEBUG.

Le principe du programme est le suivant :

On vérifie qu'il y a bien un objet sur la pile et que c'est un programme. Une liste est constituée avec la commande User Rpl HALT et le programme. Puis on évalue la liste avec la commande User Rpl EVAL.

Quelques adresses supplémentaires:

- EIGHT syst. binary correspondant au type programme.
- No_eval_nxt (rien d'officiel) n'évalue pas l'objet suivant ce qui permet, dans notre programme, de réaliser un liste avec xHALT et le programme à exécuter.
- + list\prog any ajoute dans une liste un programme et quelque chose ou une liste et quelque chose.
- xHALT et XEVAL sont les commandes User Rpl.

On remarquera que certaines mnémoniques n'en sont pas véritablement mais il fallait bien un moyen de les identifier.

DEBUG

```
ASSEMBLE
        con(5)  #02c67      * DOCOL
        con(5)  #0c3e1      * CK1
        con(5)  #0c7a5      * Dispatch1
        con(5)  #071f9      * EIGHT type prog
        con(5)  #02c67      * DOCOL
        con(5)  #0541f      * no_eval_nxt
        con(5)  #0e842      * xHALT
        con(5)  #020e5      * SWAP
        con(5)  #08fac      * + list/prog any
        con(5)  #0a418      * xEVAL
        con(5)  #02f90      * SEMI
        con(5)  #02f90      * SEMI
```

Dans l'article *HP-28S RETRO* je faisais allusion au programme SPEED qui double la vitesse d'une HP-28S. Ce programme est si connu et utile qu'il se trouve dans la plupart des machines. Toutefois pour ceux qui ne l'aurait pas voici une version plus courte que celles habituelles.

SPEED

```
                                    version habituelle
                                      (Paul Courbis)
    con(5) #02c96                     con(5) #02c96
    rel(5) fin                        rel(5) fin
    ad0ex                             ad1ex
    lchex  fff00                      d1=(5) #fff00
    d0=c                              lchex  f
    dat0=c xs                         dat1=c 1
    govlng #03801 * d0=a              d1=a
fin               * a=dat0 a          a=dat0 a
                  * d0=d0+ 5          d0=d0+ 5
                  * pc=(a)            pc=(a)
```

Si les lignes 6 à 8 paraissent différentes d'une version à l'autre, elle représentent au total la même longueur de code donc c'est juste pour ne pas faire pareil. Ma version est plus courte ensuite de 3 octets grâce à la routine #03801.

Vous trouverez les chaînes de codes à assembler dans le *coin des codes* pour tous les programmes ci-dessus.

Je dois ici remercier un ami : Mathieu Clabaut, un taupin qui se passionne pour sa HP28S (il rêve de trouver le moyen de réaliser des librairies comme pour la HP-48!), et qui a mis à ma disposition un ensemble d'informations sans lesquelles cette chronique HP-28S aurait été beaucoup plus longue et difficile à réaliser.

Ma reconnaissace va aussi à Jean-François Garnier (PPC 242) dont les deux décompilateurs MON48 et MON28 m'ont été d'un grand secours.

Enfin nous avons toujours une dette envers tous ceux qui ont contribué à découvir les secrets de la HP-28S mais dont les noms me sont inconnus.

Il reste un problème à élucider : combien, parmi les lecteurs de JPC, sont intéressés par cette rubrique. Faut-il continuer, aller plus loin ? Qui a une HP-28S munie de connecteurs ? Qui envisage de transformer (c'est à la portée de beaucoup) sa machine ou de la faire transformer ? Combien sont prêt à apporter leur contribution ?

Manifestez-vous en m'écrivant. Je rappelle mon adresse:

Guy Toublanc
21, rue Henri Becquerel
59120 LOOS-les-Lille

Si vous désirez des informations précisez vos coordonnées avec éventuellement votre numéro PPC et dans tous les cas ce qu'il faut matériellement pour la réponse.

Guy Toublanc (276)

HP48

LES GOODIES DISKS
ACTE II

Le volume 8 des Goodies Disks est maintenant disponible ! Encore une fois, on y trouve quantité de choses passionnantes, que je vais sans plus attendre vous présenter.

Directory DETLEF :

Cette section contient un système de développement très complet dû à Detlef Mueller. Il s'agit de la version 1.12 beta. Ceux d'entre vous qui étaient présents à la réunion des 30-31 janvier avaient pu goutter à la version 1.11 présente sur la disquette anniversaire du club anglais.

Ces outils tournent entièrement sur 48, et permet de développer sans l'aide d'un PC. On y trouve principalement :

- Librairie ←RPL→ : un compilateur et un décompilateur system RPL, un assembleur et un désassembleur Saturn.
- Librairie ←LIB→ : un créateur et un désassembleur de librairies, ainsi que divers utilitaires de création d'objets.
- Librairie DEBUG : un débogueur system RPL.

La particularité de ←RPL→, par rapport à d'autres déjà connus, est l'utilisation des mnémoniques HP des points d'entrées, aussi bien pour le compilateur et l'assembleur que pour leurs inverses (décompilateur et désassembleur). On écrit le source dans une chaine de caractère, avec possibilité de définir des symboles, d'utiliser des INCLUDE, bref de travailler de façon analogue au processus sur PC. Le rapidité de compilation est satisfaisante. Bien sûr le Saturn n'est pas un 386.

L'ensemble, d'une qualité exceptionnelle, fait environ 65 ko et nécessite une carte 128ko en port indépendant. La documentation très complète en fait un produit quasi-professionnel. Cette version beta (pouvant donc contenir encore quelques imperfections, en clair des bugs ...) est diffusée en tant que Giftware (cadeau). La version définitive sera distribuée sous forme de Shareware (diffusion libre, mais utilisation soumis au paiement d'un license).

Directory HACKER :

Vous trouverez dans cette rubrique (entre autres) :

- Une nouvelle version de la célèbre librairie HACKIT.LIB, avec une commande →DIR améliorée.

- Des informations sur un virus attaquant les 48 (eh oui ....), et sur les moyens de s'en prémunir.

- Une liste des ID des librairies actuellement en circulation (NDLR : Faisant déjà référence à des programmes devant se trouver dans la Goodies 9 !), où l'on se rend compte que malgré le nombre élevé d'ID disponibles, il n'est pas rare de trouver 2 ou 3 librairies de même numéro (environ 200 librairies commerciales ou du domaine public).

Directory POSTINGS :

Divers articles intéressants, en particulier sur la dernière version J de la rom de la 48. Contrairement à des rumeurs, tous les points d'entrées supportés sont présents. Mais attention aux nombreuses applications utilisant des points d'entrées non-supportés ! Par exemple la librairie GATEWAY (protection de la 48 par mot de passe) présentée dans les disquettes Goodies 3 et 7 semble crasher monumentalement les versions J. Les plus hardis d'entre vous pourront suivre le mode d'emploi du démontage de leur 48 préférée (avec perte de la garantie !).

Directory HP :

Cette rubrique contient divers programmes d'origine HP et de diffusion libre, bien que restant copyright HP. On y trouve les dernières versions de APDIR, CLK, EPSPRINT, INPRT, PCLPRINT, STPWATCH, USAG déjà connus (kit d'interface série HP48-PC), mais aussi CALCULUS, EZPLOT, HAMILTON, SUITE3D4 (tracé de courbes en 3 dimensions, avec les sources).

Directory TUNES :

Après les graphismes et les animations, voici les effets sonores, sinon musicaux. S'il y a des amateurs...

Directory GAMES :

Quelques jeux supplémentaires pour occuper vos (courts) temps de loisirs. Notez en particulier quelques superbes réalisations françaises (ARKANOID, BOULDER DASHP et TRONII).

Et bien sûr les habituelles rubriques MATH, UTILS, PROGRAMR, WORK.

D'une manière générale, si le niveau des programmes est devenu très élevé, on constate que l'utilisation de la programmation en system RPL ou assembleur, et la diffusion sous forme de librairies font que les sources sont de moins en moins fournis.

Le prochaine disquette ne devrait pas apparaitre avant l'automne. En attendant, tenez nous au courant de vos découvertes et de vos réalisations.

Happy Hunting.

Jean-François GARNIER (242)

LE MENU CST

Tous ce qui sera dit ci-dessous est valable pour le menu CST et les menus temporaires (TMENU).

Le menu CST est destiné par définition à l'utilisateur, cependant ses capacités sont limitées. Nous allons donc voir comment accéder pleinement aux possibilites de la HP48.

Le menu CST prend dans la variable CST une liste. Cette liste contient la plupart du temps des sous-listes. Ces sous-listes contiennent une chaine qui sera le label puis eventuellement un programme a exécuter en cas d'appui sur la touche correspondante.

Il est possible de remplacer la chaine par un GROB 21×8 qui sera affiché a la place du label.

Mais nous allons faire plus...

Nous allons remplacer la chaine ou le Grob par un programme. Ce programme sera executé et il laissera sur la pile une chaine ou un Grob.

On peut imaginer le CST suivant:

```
{ { « DEPTH →STR » « DEPTH 1 DISP » } }
```

qui afficherait dans le menu le nombre d'objets sur la pile.

Le resultat obtenu n'est pas celui attendu, car on voit dans le menu :

```
----------
| << DEP |
----------
```

Est-il donc impossible d'exécuter un programme qui creerait un label ??

Patience on va y arriver...

La HP48 exécute le programme d'un label seulement si il commence par le programme en ROM TakeOver (#40788h).

Voici donc un programme qui transforme un programme normal en programme executable dans un menu :

SMENU
#ED66h (125)

```
«
  [8] CTRL #40788h SYSRCL SWAP #54AFh SYSEVAL
  #3DEFh SYSEVAL #3FB3h SYSEVAL #5331h SYSEVAL
»
```

CTRL et SYSRCL sont deux programmes listés ci-dessous.

Si vous tapez donc :

```
« DEPTH →STR »
SMENU 1 →LIST →LIST TMENU
```

Ca marche mais le chiffre obtenu est superieur de 6 a la bonne taille. Voyons ce que contient la pile quand on exécute DEPTH :

```
« 1 6 FOR I I PICK I DISP 7 FREEZE »
SMENU 1 →LIST →LIST TMENU
```

Nous voyons s'afficher 6 *System binary* qui correspondent a l'abscisse de l'angle gauche des Grobs menu.

Si le programme se trouve en deuxieme position dans le menu, il n'y a plus que 5 objets sur la pile.

Voici deux autres programmes pour créent des menus repertoires ou des menus inversés comme dans SOLVE :

ME.INV
#EDC8h (95.5)

```
« [2] CTRL #40788h SYSRCL SWAP #3A44Eh SYSRCL 3 →PRG »
```

ME.DIR
#0F73h (95.5)

```
« [2] CTRL #40788h SYSRCL SWAP #3A3ECh SYSRCL 3 →PRG »
```

Ils prennent une chaine dans la pile le programme créé par ME.DIR, qui devrait normalement être suivi d'un programme du type :

```
« { { .. } { .. } } TMENU »
```

avec éventuellement des messages affichés par des DISP comme dans les menus TIME et PLOT.
La derniere entree de ce menu temporaire pourra etre:

```
{ "EXIT" « .. » }
```

ou ".." correspond a un programme appelant le menu "père".

Le dernier programme que nous allons voir créera un menu "option" comme dans le menu MODE un carré blanc signalera qu'une condition est remplie. Le programme pris en argument devra retourner un réel (0 ou 1) puis se terminer par #2A76Bh SYSEVAL pour le transformer en flag systeme.

ME.OPT
#1643h (106)

```
«
  [8 2] CTRL #40788h SYSRCL ROT ROT
  #3ED0Ch SYSRCL 4 →PRG
»
```

Exemple d'utilisation:

```
"ALPHA"
« -60 FS? #2A76Bh SYSEVAL » ME.OPT
1 →LIST 1 →LIST TMENU
```

Le menu ainsi obtenu contient un carré blanc si le alpha se bloque au premier appui sur la touche [alpha].

J'espère que tous ces programmes vous seront utiles et qu'il vous permettront de realiser des programmes utilisants de manière intensive les menus temporaires.

Bonne programmation.

CTRL
#ADDCh (142.5)

```
«
  DUP SIZE EVAL → t ts
  «
   IF DEPTH ts <
   THEN
     #201h DOERR END 1 ts
     FOR I
       IF I PICK TYPE t I GET == NOT
       THEN
         #202h DOERR
       END
     NEXT
   »
»
```

On pourra remplacer == NOT par le signe différent.

SYSRCL
#DD21h (41)

```
« #5A03h SYSEVAL #C612h SYSEVAL »
```

→PRG
#325Dh (43.5)

```
« →LIST #3FB3h SYSEVAL #5ACCh SYSEVAL »
```

Pierre Silvestre de Sacy (572)

POLYEDRES

Voici deux programmes permettant de construire un polyèdre obtenu à partir d'un cube auquel on a fait subir une rotation de 180° suivant ses quatres diagonales (un cube a 8 sommets, donc 4 diagonales), le cube de base étant exclu des figures.

Les programmes MAT1 et MAT2 calculent les matrices de rotation. Le premier est une adaptation pour HP28 ou HP48 d'une partie d'un programme de de transformation de coordonnées contenu dans le module MATH de la HP-41.

MAT1

Il prend en entrée dans la pile 4 nombres: A, B, C et T. A, B, C sont les composantes d'un vecteur et T l'angle de la rotation ayant pour axe ce vecteur. La matrice de rotation a pour nom MR1.

```
«
  → A B C T
  «
    DEG A B C { 3 } →ARRY DUP ABS /
    ARRY→ DROP 1 T COS - T SIN T COS → A1
    B1 C1 D1 TS TC
    «
      A1 SQ A1 B1 * DUP A1 C1 * SWAP B1 SQ B1 C1 *
      DUP A1 C1 * SWAP C1 SQ { 3 3 } →ARRY D1 *
      TC TS C1 * NEG DUP B1 TS * SWAP NEG TC A1
      TS * NEG DUP B1 TS * NEG DUP B1 TS *
      NEG SWAP NEG TC { 3 3 } →ARRY + 'MR1' STO
    »
```

```
  »
»
```

MAT2

Il a comme arguments 2 nombres A et B qui sont les angles que font les axes des X et des Z avec l'axe des Y, qui est toujours parallèle au bord vertical de la feuille. En sortie on a la matrice de rotation MR2 ainsi que les réels KX, KY et KZ qui sont les coefficients de réduction suivant les axes des X, Y et Z.

```
«
  DEG DUP2 + NEG 360 + → A1 A2 A3
  «
    « COS SWAP SIN / NEG DUP ROT SIN * √ / »
    'COEF' STO A2 A3 A1 COEF 'KZ' STO A1 A2 A3 COEF
    'KY' STO A3 A1 A2 COEF 'KX' STO 'COEF' PURGE
    KZ SQ KX SQ KY SQ → B1 B2 B3
    «
      B2 B3 + 1 - B3 / √ B3 1 - B2 1 - * B3 /
      √ NEG KX ASIN COS 0 KY KY ASIN COS B3 B1 + 1
      - B3 / √ NEG B3 1 - B1 1 - * B3 / √ NEG
      KZ ASIN COS { 3 3 } →ARRY 'MR2' STO
    »
  »
»
```

Exemple

Une projection isométrique suivant la norme NF E 04-108 a une matrice de rotation obtenue par : 120 120 MAT2. Pour une projection dimétrique redressée on fait 105 105 MAT2.

Les coordonnées du cube de base sont introduites :

```
    [[ 1  1  1]
     [-1  1  1]
     [-1 -1  1]
     [ 1 -1  1]
     [ 1  1 -1]
     [-1  1 -1]
     [-1 -1 -1]
     [ 1 -1 -1]]
'CUBE' STO
```

Les sommets du polyèdre sont calculés par CUBE MR1 * MR2 *. Le report des points se fait à la main ou avec une table traçante.

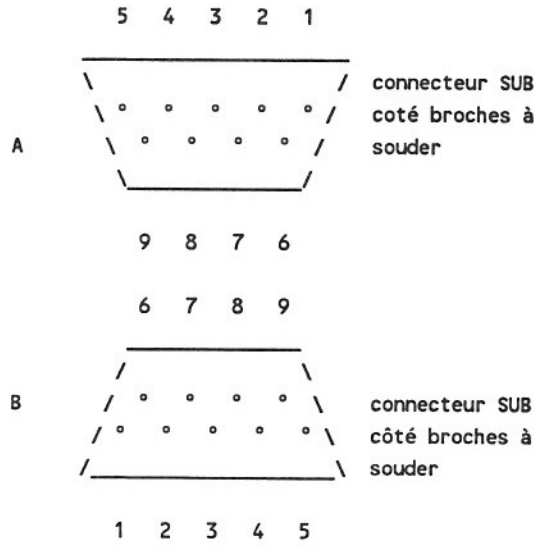
Alexandre Buchmann (386)

BRICOLAGE SIMPLE POUR TRANSFERT 48 <-> 48

Le transfert d'objets entre deux HP-48 se fait en général par voie infrarouge à la vitesse de 2400 bauds. Cette vitesse nous semble souvent un peu lente lorsque les objets ne sont pas de petite taille. Il existe des programmes qui effectuent ce transfert à des vitesses plus acceptables, mais il faut que les deux machines soient déjà pourvues de ces programmes et ceux-ci prennent de la place en mémoire, constituant encore un répertoire où une librairie de plus. Or il existe une autre solution que certains utilisent déjà depuis longtemps: la transmission par interface série à la vitesse de 9600 bauds. Pour cela il faut bricoler un cable et les connexions ne sont pas des plus faciles à réaliser. Ma solution implique que les deux HP-48 soient équipées d'un cable série HP. Il suffit de réaliser un connecteur amovible pour relier ces cables ce qui est à la portée de chacun (pour la circonstance vous pouvez emprunter un fer à souder à un ami).

Matériel :

- 2 connecteurs SUB-D RS232 mâles connexion par broches à souder à 9 contacts
- 1 capot plastisque de montage pour 2 connecteurs SUB-D à 9 contacts
- 20 cm de fil électrique fin
- un peu de fil d'étain à souder
- et évidemment 1 petit fer à souder électrique



Après acquisition de ce matériel dans un magasin de matériel électronique, relier les connecteurs SUB-D avec 3 longueurs de 4 cm environ de fil fin électrique soudé sur les broches et suivant les liaisons:

le 2 de A au 3 de B
le 3 de A au 2 de B
le 5 de A au 5 de B

Placer l'ensemble dans une moitié du capot et fermer avec la partie restante du capot. Les deux cables série HP pourront être assemblés le moment venu, votre IOPAR sera le même que pour une autre liaison avec un PC.

Bon bricolage et bons transferts.

Guy Toublanc (276)

RECHERCHE DE CHAINES DANS LA MEMOIRE

Bien que la ROM de nos chères machines soit assez bien connue de nos jours, il arrive parfois que l'on cherche des objets ou des parties d'objets dans la mémoire.

Le programme FINDMEM recherche une suite de caractères hexadécimaux (donnée sous forme d'une chaîne de caractères) dans la mémoire à partir d'une adresse spécifiée par un nombre binaire. Il retourne l'adresse de la première occurence de la suite s'il en trouve une, #0 sinon.

Voici le listing :

```
; FINDMEM ( 2:chaine 1:adresse -> idem )

RPL
::
    CK2&Dispatch FIFTYNINE
    ::
    OVER TOTEMPOB

    ASSEMBLE

        include addr-48.as

        nibhex  begin_code
        rel(5)  fincode
        gosbvl  save_reg
        a=dat1  a
        d0=a
        d0=d0+5
        c=0     w
        c=dat0  a
        csrb
        c=c-1   a
        c=c-1   a
        b=c     a
        ?c!=0   a
        goyes   ok1
        goto    fin
ok1     d0=d0+5
        ad0ex
        d0=a
        r0=a
        d1=d1+5
        c=dat1  a
        d1=c
        d1=d1+10
        cd1ex
        r1=c
        d1=a
        c=b     a
        d=c     a
loop1   a=dat0  b
        lchex   #30
        a=a-c   b
        lchex   #0a
        ?c>a    b
        goyes   digit
        lchex   #07
        a=a-c   b
digit   dat1=a  p
        d1=d1+1
        d0=d0+2
        d=d-1   a
        ?d!=0   a
        goyes   loop1
        a=r0
        d0=a
        c=dat0  p
        d=c     p
        a=r1
        d1=a
        a=dat1  a
        d1=a
loop2   c=dat1  p
        ?c=d    p
        goyes   test
increm  d1=d1+1
        gonc    loop2
found   cd1ex
        a=r1
        d1=a
        dat1=c  a
fin     gosbvl  load_reg
        d1=d1+5
        d=d+1   a
        a=dat0  a
        d0=d0+5
        pc=(a)
test    ad1ex
        d1=a
        r2=a
        c=b     a
        d=c     a
```

```
loop3   d=d-1   a
        ?d!=0   a
        goyes   cont
        a=r2
        d1=a
        goto    found
cont    d1=d1+1
        goc     found
        d0=d0+1
        c=dat1  p
        a=dat0  p
        ?a=c    p
        goyes   loop3
        a=r0
        d0=a
        c=dat0  p
        d=c     p
        a=r2
        d1=a
        goto    increm
fincode ENDCODE

    ;
;
```

Vous trouverez les codes à taper dans la rubrique *Le coin des codes*.

Laurent Grand (516)

RECOVER - ACTE II

N'ayant pas poussé mes tests a fond avant d'envoyer le programme RECOVER, je me suis aperçu après coup que le port 0 posait problème.

Ceci est du au fait que le port 0 n'est pas sur une carte.

Les ports 1 et 2, lorsque les cartes ne sont pas mergées, commencent a des adresses fixent qui sont soit #80000h soit #C0000h suivant le port et suivant le nombre de cartes mergées.

Ces ports finissent lorsque on trouve cinq quartets a 0. Mais dans le programme RECOVER la recherche continuait jusqu'a la fin de la carte car si on a perdu des librairies celles-ci seront au delà de la fin du port. Du fait de cette recherche sur toute la carte, il arrive que l'on retrouve des librairies que l'on a purgées il y a un certain temps.

Le port0 quant a lui commence normalement juste apres le répertoire HOME et finit normalement aux adresses suivantes:

- #7FFFBh si il n'y a pas de carte mergée.
- #8FFFBh si une carte de 32 ko est mergée (peut importe dans quel port).
- #BFFFBh si une carte de 128 ko est mergée (peut importe dans quel port).
- #9FFFBh si deux cartes de 32 ko sont mergées.
- #CFFFBh si une carte de 32 ko et une carte de 128 ko sont mergées.
- #FFFFBh si deux cartes de 128 ko sont mergées.

Lorsque l'un des objets du port 0 est "abimé" la 48 reconnait le port 0 depuis la fin du repertoire HOME jusqu'a l'objet abimé. Tous les autres objets existent toujours, mais ils ne sont plus dans le port 0. Ils sont inaccessibles.

Ces objets inaccessible ne seront jamais detruits par la 48 car elle ne les connait pas. Même un garbadge collector ne recupère pas la memoire qu'ils occupent.

En remplacant dans le programme RECOVER :

```
PORT0
        D0=(4) #0597
        D1=(2) #4D
        GOTO   SUITE2
```

par:

```
PORT0
        LCHEX   80000
        D1=(2)  #2C
        A=DAT1  1
        ?ABIT=0 1
        GOYES   NOMERGE1
        D1=D1+  1
        A=DAT1  A * TAILLE CARTE1
        A=-A-1  A
        C=C+A   A
        D1=D1-  1
  NOMERGE1
        D1=D1+  11
        A=DAT1  1
        ?ABIT=0 1
        GOYES   NOMERGE2
        D1=D1+  1
        A=DAT1  A * TAILLE CARTE2
        A=-A-1  A
        C=C+A   A
        D1=D1-  1
  NOMERGE2
        B=C     A * @ FIN
        D0=(4)  #0597 * @ DEBUT
```

```
A=DATO  A
DO=A
GOTO RECHERCHE
```

ces objets peuvent être récuperés.

Cependant si on les restocke dans le port, le repertoire HOME est alors deplacé et ils sont stockés au debut du port 0. Mais ils existent toujours après la fin du port.

Le seul moyen de reconstituer le port 0 de manière propre est ON-A-F puis YES. La 48 reconstitue alors le port 0 avec toutes les Librairies qu'elle trouve en RAM.

Le programme RECOVER modifié permet toutefois de faire une sauvegarde de tout le port 0 au cas où le ON-A-F n'arriverait pas a le reconstituer.

D'autre part, une instruction à été oubliée dans le listing paru le mois dernier. En effet, il faut remplacer :

```
SUITE
      DO=DO+    1
      C=DATO    A
```

par :

```
SUITE
      DO=DO+    1
      C=DATO    A
      C=-C-1    A
```

Sinon le programmme ne marche pas, car la taille est stocke en memoire en complementation a 2.

Pierre Silvestre de Sacy (572)

FAUSSE RESOLUTION

Comme le dit l'un de mes amis, une fausse erreur s'est glissée dans mon article *Résolution de Polynômes* de *JPC 79*. Je prie les victimes de cette inattention de m'excuser.

L'erreur se trouve dans le programme BAIRSTOW, page 7, colonne 2, ligne 18. Elle se produit pour la dernière racine des polynômes de degré impair. Cette racine doit être divisée par le coefficient de plus grand degré. Pour la supprimer, il suffit de remplacer la ligne.

```
SWAP DROP
```

par

```
SWAP /
```

Laurent Grand (516)

HP95

UNE TACHE DANS LES PIXELS

Comme d'habitude dans mes articles (si si, vérifiez !), quelques erreurs se sont glissées dans la présentation des fonctions graphique du HP95, parue le mois dernier.

Cette fois ci, je me suis fait trahir d'une part par une petite faute d'inattention suivant une opération facilitée par la puissance des éditeurs de textes actuels, et d'autre part par une mauvaise relecture de mes notes après mes essais préliminaires d'une fonction.

La première faute concerne la fonction 04h (sélection d'une fenêtre graphique, page 19). Un lecteur attentif s'aperçevra que je cite pour les registres SI et DI le coin inférieur *gauche* alors qu'il s'agit du coin inférieur *droit*. En effet, comment le programme pourrait connaitre la largeur de la fenêtre, si je ne donne que les coordonnées du côté gauche ?

Si la première faute ne concernait que l'article, la seconde a le désavantege de se retrouver aussi dans le listing présenté après l'article. En effet, les éléments des coordonnées du coin inférieur droit de la fonction 0dh (get image) sont inversées pour les registres SI et BP.

Il faudra donc, dans la fonction `get_image()`, remplacer les lignes :

```
    asm  mov  si,y1    par   asm  mov  si,x1
et
    asm  mov  bp,x1    par   asm  mov  bp,y1
```

Jacques Belin (123)

Note finale : Après relecture de cet article, je me suis aperçu que je venais de reproduire la première erreur sur le correctif de la deuxième erreur ! Je suis vraiment incorrigible...

EDITEUR DE MASQUES

Le mois dernier, vous avez pu remarquer une page contenant des motifs de remplissage graphiques, à la suite de mon article sur le graphisme sur le HP95. La création de certains ne m'aurait pas été possible sans l'aide du programme que je vais vous présenter ce mois-ci.

Il s'agit en fait d'un éditeur graphique permettant de créer et visualiser les motifs de remplissage et leur donner un nom le cas échéant. Bien sûr, ces motifs peuvent être sauvegardés dans un fichier.

Mode d'emploi du programme

Au lancement du programme, vous pouvez spécifier, si vous le désirez, le nom d'un fichier. Par défaut, ce sera `C:\_DAT\MASKS.MSK`.

Vous obtiendrez l'écran suivant :

Le nom du fichier courant est situé en haut et à droite de l'ecran. La première fenêtre montre le résultat de l'exécution de la fonction `set_fill_mask` (voir précédent numéro) pour le motif défini dans la deuxième fenêtre, qui permet l'édition proprement dit.

Les codes hexadécimaux présents dans cette fenêtre sont ceux que vous devrez spécifier pour la fonction précédement citée.

Sous les deux fenêtres, une troisième permet de donner un nom au motif que vous avez créé.

Touches définies :

```
Espace ou DEL : Inversion d'un pixel.
Flèches       : Déplacement du curseur graphique.
Page Up/Down  : Motif précédent/suivant.
Début/Fin     : Premier/dernier motif.
F1            : Affiche l'écran d'aide.
F2            : Sauve le fichier courant.
Shift F2      : Sauve sous un nouveau nom.
F3            : Charge un nouveau fichier.
F4            : Edite le nom du motif.
F5            : Inverse tous les bits du motif.
F6            : Insère un motif.
F7            : Efface le motif courant.
F10           : Quitte le programme.
ESC           : Annulation des changements sur le
                motif courant.
```

Fonctionnement du programme

J'ai l'habitude, lorsque j'expérimente un nouveau domaine ou de nouveaux outils, de créer un "vrai" programme utilisant les nouvelles fonctions, plutôt que de faire une simple démo sans utilité. Ceci me permet donc de me placer dans un cas réel d'utilisation, et de mieux connaitre les limites de ces fonctions.

Ce programme ne faillit pas à la règle, puisqu'il utilise quasiment toutes les fonctions graphiques du HP95, à l'exception notoire de celles permettant d'afficher un point ou une ligne !

Il ne devrait pas être trop difficile à analyser, je n'expliquerais ici que quelques points spécifiques.

Architecture générale du programme

Elle est très simple, puisqu'il s'agit avant tout d'une boucle comportant dans l'ordre : les affichages, la fonction d'attente de touche et les différents traitements associés à ces touches. La sortie de cette boucle, et donc du programme, se fait par l'activation de la variable `fin` (dans le traitement de la touche F10, pour ce programme). Les touches peuvent effectuer des actions simples (positionnement du curseur, par exemple) ou appeler des sous-programmes gérant des fenêtres, qui peuvent avoir exactement la même structure que le programme principal (voir le traitement de Shift F2 avec le couple `get_in_box/ginput`).

Si vous désirez créer des programmes de ce type, vous êtes bien entendu tout à fait autorisés de récupérer le squelette de ce programme et de n'apporter que les modifications propres à votre application !

Notez aussi que j'ai pour cela séparé les fonctions spécifiques à ce programme et les fonctions généralistes (affichage de fenêtre, entrée de chaine), qui peuvent être réutilisées dans n'importe quel autre programme sans modification.

Affichage des fenêtres

Le programme permet une gestion simple de l'affichage de fenêtres. En effet, certaines touches nécessitent un petit dialogue avec l'utilisateur.

Pratiquement, le principe est de sauvegarder dans un buffer la zone devant être recouverte par le fenêtre ainsi que les paramêtres graphiques courants, puis positionner l'origine graphique sur le coin supérieur/droit de la fenêtre (pour simplifier les calculs de coordonnées) puis d'afficher le contenu de la fenêtre et effectuer les différentes actions que nous désirons effectuer. En fin de traitement, l'effacement est effectué en affichant le contenu du buffer à la position initiale. Enfin, il est toujours souhaitable de replacer les modes vidéo à ce qu'il étaient avant l'appel de la fonction.

Par souci de simplicité les fonctions d'affichage de fenêtres de ce programme ne gèrent que des fenêtres de deux lignes de textes, placées automatiquement au centre de l'écran. Cependant, la largeur est automatiquement pris en compte en fonction de la longueur du texte, qui est lui aussi centré.

Vous remarquerez que la plupart des affichages sont précédés d'un appel d'une fonction visant à effacer ce qu'il y avait originellement à l'ecran. Il est dommage que le mode `FORCE` ne soit pas totalement appliqué (comme pour la fonction `put_image()`) car cela augmente souvent inutilement la taille du programme.

Notez aussi que ces sous-programmes ayant une structure commune, il aurait été intéressant de les simplifier en remplacant les parties communes situées au début et à la fin par des fonctions plus génériques que nous aurions appelé `open_window()` et `close_window()` par exemple. Ceci aurait grandement diminué la taille du programme, mais je n'ai pas eu le temps de le faire.

Optimisation de la vitesse d'affichage

Je n'ai pas cherché à optimiser le programme, que ce soit en vitesse ou en taille. Cependant, la partie du programme chargée d'afficher le motif dans la fenêtre et les codes hexadécimaux étant relativement lente, j'ai un peu modifié le code afin de diminuer le temps d'affichage.

Pour représenter un pixel dans la fenêtre d'edition, je l'ai agrandi dans un carré de 8×8 pixels. La méthode qui vient immédiatement à l'esprit est d'utiliser la fonction `draw_rectangle()` avec le paramètre `SOLID`. Cependant, avant d'utiliser cette fonction, nous devons appeler les fonctions `move_pen()` et `set_color()` pour déplacer le curseur et selectionner la couleur du tracé. Ce qui fait 3 appels pour un seul tracé. Pour l'effectuer un seul appel, il vaut donc mieux utiliser la fonction `put_image()` qui est à priori plus lente que `draw_rectangle()` en mode `SOLID`, mais plus rapide si nous nous en servons bien. La sélection de la couleur du pixel est alors assurée par le paramètre `NORMAL/INVERSE`.

Notez aussi qu'on aurait eu de bien meilleurs résultats en effectuant des écritures directes en mémoire d'ecran, sans vraiment compliquer le programme. Cependant, cela aurait sûrement rendu le programme incompatible avec les successeurs du HP95. J'ai donc choisi la sécurité par rapport à la rapidité.

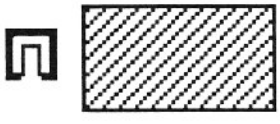
Pour afficher le code Hexadécimal, nous devons toujours utiliser la fonction `write_text()`. Cependant, ce n'est pas la plus gourmande en temps de calcul. En effet, nous devons auparavant convertir le code en hexadécimal et le transférer dans une chaine de caractères, qui sera utilisé par la fonction graphique. La méthode "normale" est d'utiliser la fonction `sprintf()` du C. Cependant, cette fonction appelle une routine très complexe d'analyse des opérandes, donc très lente. Il faut donc remplacer l'appel à cette fonction par du code moins généraliste, mais beaucoup plus rapide (voir listing du programme).

Notons finalement qu'ecrire cette partie en assembleur n'aurait quasiment pas ammélioré les performances, puisque la plus grande partie du temps est passé à l'intérieur des fonctions internes qui sont (enfin espérons le) déjà optimisées.

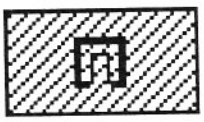
Affichage du logo

La fonction `put_image()` permet d'afficher une image au dessus d'un fond. Cependant, si cette image contient des zones blanches, le fond apparaitra au travers, car cette fonction ne faisant qu'une opération au niveau du bit n'a aucun moyen de savoir si une zone blanche est à l'intérieur ou à l'exterieur d'une image !

Par exemple, supposons que nous voulons afficher un motif quelconque sur un fond hachuré :

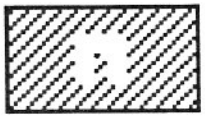


Si nous exécutons la fonction `put_image` avec les paramêtre `NORMAL` et `OR`, nous obtiendrons ceci :

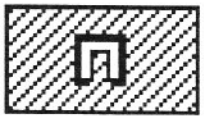


Si nous voulons obtenir le bon résultat, il faut faire une deuxième version de l'image, avec tous les pixels intérieurs activés, qui nous servira à créer un cache :

Nous afficherons ensuite ce cache avec les paramètres `INVERSE` et `AND`, ce qui nous donnera le résultat suivant :



Il ne nous restera plus qu'à afficher l'image originale de façon classique (avec les paramètres `NORMAL` et `OR`) pour obtenir le résultat désiré :



Pour l'affichage de mon logo, j'ai appliqué ce principe dans le but de créer une bordure blanche autour du logo (en créant un masque plus grand d'un pixel), afin qu'il apparaisse bien détaché du fond.

Structure du fichier

Le fichier est constitué d'un en-tête puis des enregistrements décrivant les motifs.

L'en-tête contient une valeur prédéfinie (`MAGIC`) permettant au programme de savoir si il s'agit vraiment d'un fichier de motifs.

Les enregistrements sont constitués des huits octets décrivant le motif, suivis de son nom.

Pour finir, je ne ferais qu'un petit avertissement quant à l'utilisation du programme : il y a de grandes chances que vous n'ayez aucune difficulté à créer des motifs intéressants, jusqu'au moment où vous devrez leur donner un nom explicite (qui ne soit pas "passe partout")... Si vous passez cette épreuve, n'oubliez pas de nous transmettre vos oeuvres !

Jacques Belin (123)

```
/*****************************************************************************/
/*****************************************************************************/
/**                          MASKEDIT.C                                     **/
/**              Editeur de masques de remplissages pour HP95               **/
/**                          Version 1.0                                    **/
/**                                                                         **/
/**   Création : 20/02/93                          Auteur : Jacques Belin   **/
/**   Dernière modification : Changement du nom du fichier par défaut       **/
/**                                                                         **/
/**  Note : Développé et testé uniquement sur Turbo C++ 1.01                **/
/**         TASM nécessaire (pour l'assemblage en ligne de grafx95.c)       **/
/*****************************************************************************/
/*****************************************************************************/
#include <stdio.h>
#include <dos.h>
#include <io.h>
#include <conio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <bios.h>
#include <dir.h>
#include <ctype.h>
#include <sys\stat.h>
#include "grafx95.c"  /* fonctions graphiques. Voir JPC 84+correction JPC 85 */



/*--------- Codes de quelques touches tels que GetKey() les renvoie ---------*/
#define BS      8                               /* Code de touche Back-Space */
#define CR      13                                 /* Code de touche Return */
#define ESC     27                                 /* Code de touche Escape */
#define F1      315                                   /* Code de touche F1  */
#define F2      316                                   /* Code de touche F2  */
#define F3      317                                   /* Code de touche F3  */
#define F4      318                                   /* Code de touche F4  */
#define F5      319                                   /* Code de touche F5  */
#define F6      320                                   /* Code de touche F6  */
#define F7      321                                   /* Code de touche F7  */
#define F8      322                                   /* Code de touche F8  */
#define F9      323                                   /* Code de touche F9  */
#define F10     324                                  /* Code de touche F10 */
#define HOME    327                                /* Code de touche Home */
#define UP      328                               /* Codes de curseur haut */
#define PgUp    329                              /* Code de touche page up */
#define LEFT    331                             /* Codes de curseur gauche */
#define RIGHT   333                             /* Codes de curseur droite */
#define END     335                                /* Code de touche Fin */
#define DOWN    336                              /* Codes de curseur bas */
#define PgDn    337                            /* Code de touche page down */
#define DEL     339                          /* code de touche d'effacement */
#define SF2     341                              /* Code de touche Shift+F2 */

#define YES     'Y'                /* codes de retour pour la fonction yesno() */
#define NO      'N'

#define X_FILE   120                           /* position du nom du fichier  */
#define Y_FILE   1

#define X_ZOOM   134                       /* position de la fenêtre d'édition */
#define Y_ZOOM   24

#define X_PATERN 20                          /* position de la fenêtre gauche */
#define Y_PATERN 14
#define W_PATERN 84                                /* largeur de la fenêtre */
#define H_PATERN 84                                /* hauteur de la fenêtre */

#define LEN_NAME 20                              /* longueur du nom du masque */
#define X_NAME   49                               /* position du nom du motif */
#define Y_NAME   102
#define W_NAME   ((LEN_NAME+2)*6)                    /* largeur de la fenêtre */
#define H_NAME   12                                  /* hauteur de la fenêtre */

#define X_LOGO   205                                  /* position du logo J*B */
#define Y_LOGO   100

#define MAX_NB_FILLS  500          /* nombre maximum de masques dans fichier */

#define LEN_FILENAME  30                    /* longueur maxi du nom du fichier */
#define DEF_FILENAME  "c:\\_dat\\masks.msk"            /* fichier par défaut */
#define MAGIC         0x930401L          /* code de reconnaissance du fichier */

typedef struct                /* description d'un enregistrement d'un fichier */
        {
         char mask[8];                                              /* motif */
         char name[LEN_NAME+1];                                      /* nom */
        } FILL_MASK;


DESCRIPT_IMAGE(32, 17, IMAGE_32_17)                 /* définition de structure */
                                                       /* Description du Logo */
IMAGE_32_17 logo_jb0 = {                          /* masque de détourage du logo */
                        1, 1, 32, 17,
                        {
                        0x07, 0xf8, 0x3f, 0xe0,
                        0x07, 0xf8, 0x3f, 0xe0,
                        0x07, 0xf8, 0x3f, 0xf0,
                        0x07, 0xfb, 0xbf, 0xf0,
                        0x01, 0xe3, 0x9e, 0xf0,
                        0x01, 0xef, 0xff, 0xf0,
                        0x01, 0xef, 0xff, 0xf0,
                        0xff, 0xff, 0xff, 0xff,
                        0xff, 0xff, 0xff, 0xff,
                        0xff, 0xff, 0xff, 0xff,
                        0x01, 0xef, 0xff, 0xf0,
                        0x3d, 0xef, 0xff, 0xf0,
                        0x3d, 0xe3, 0x9e, 0xf0,
                        0x3f, 0xe3, 0xbf, 0xf0,
                        0x3f, 0xe0, 0x3f, 0xf0,
                        0x3f, 0xe0, 0x3f, 0xf0,
                        0x1f, 0xc0, 0x3f, 0xe0,
                        }
                       };

IMAGE_32_17 logo_jb1 = {                                  /* logo proprement dit */
                        1, 1, 32, 17,
                        {
                        0x00, 0x00, 0x00, 0x00,
                        0x03, 0xf0, 0x1f, 0xc0,
                        0x03, 0xf0, 0x1f, 0xe0,
                        0x00, 0xc0, 0x0c, 0x60,
                        0x00, 0xc1, 0x0c, 0x60,
                        0x00, 0xc1, 0x0c, 0x60,
                        0x00, 0xc5, 0x4c, 0xc0,
                        0x00, 0x03, 0x80, 0x00,
                        0x7f, 0xff, 0xff, 0xfe,
                        0x00, 0x03, 0x80, 0x00,
                        0x00, 0xc5, 0x4c, 0xc0,
                        0x00, 0xc1, 0x0c, 0x60,
                        0x18, 0xc1, 0x0c, 0x60,
                        0x18, 0xc0, 0x0c, 0x60,
                        0x1f, 0xc0, 0x1f, 0xe0,
```

```
                        0x0f, 0x80, 0x1f, 0xc0,
                        0x00, 0x00, 0x00, 0x00,
                        },
                       };

                                  /* Description du Pixel pour fenêtre zoom */
DESCRIPT_IMAGE(8, 8, IMAGE_8_8)

IMAGE_8_8 big_pixel  = {
                        1, 1, 8, 8,                                  /*  HEADER  */
                        {
                         0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
                        }
                       };


                                                        /* VARIABLES GLOBALES */
int  updated = 0;              /* flag indiquant que le fichier a été modifié */
int  num_mask=0;                                    /* numéro du motif courant */
int  nb_masks=0;                          /* nombres de motifs dans le fichier */
FILL_MASK curr_mask;             /* variable de travail pour le motif courant */
FILL_MASK far *masks;                   /* pointeur sur le tableau des motifs */
unsigned  int seg_m;                  /* segment de départ du tableau "masks" */



                                  /* prototypes des fonctions déclarées plus bas */
void disp_help(void);
void disp_filename(char *filename);
void disp_cursor(int x, int y);
void update(void);
int  save(char *file);
int  load(char *file);

int  detect_hp95(void);
int  GetKey(void);
int  msg_box2(char *message1, char *message2, int color, char *exit_codes);
void wait_box(char *message, int color);
int  yesno_box(char *message, int color);
int  ginput(char *string, int len, int x1, int y1, int color);
int  get_in_box(char *titre, char *string, int len_string, int color);


/*****************************************************************************/
/*****************************************************************************/
/*                          Module Principal                                 */
/*****************************************************************************/
/*****************************************************************************/
void main(int argc, char *argv[])
   {
    int  fin=0;                                    /* flag de fin du programme */
    int  key, key2;                                        /* codes de touche */
    int  ret;                                    /* code de retour de fonction */
    int  errflag=0;                                          /* flag d'erreur */
    int  regen;             /* flag indiquant que l'on veut regénérer le dessin */
    int  curs_x=0;                  /* position courante du curseur graphique */
    int  curs_y=0;
    char hex_code[3];                 /* chaine contenant le code hexadécimal */
    char filename[LEN_FILENAME+1];                     /* nom complet du fichier */
    char new_name[LEN_FILENAME+1];      /* tampon pour nouveau nom de fichier */
    int  i, j;
    char c, c2;


    if(!detect_hp95())                    /* si l'ordinateur n'est pas un HP95 */
      {
       fprintf(stderr, "This program runs only on an HP95LX !\n");
       exit(-1);
      }

    if(argc > 2)                          /* si 2 parametres (ou plus) -> erreur */
      errflag=1;

    if(errflag == 1)                                                /* si erreur */
      {
       fprintf(stderr, "usage : maskedit  [file]\n");
       exit(-1);
      }

                                  /* allocation du buffer contenant les motifs */
    if(allocmem(((sizeof(FILL_MASK)*(MAX_NB_FILLS+1))+15)/16, &seg_m) != -1)
      {
       fprintf(stderr, "Error : not enough memory to run\n");
       exit(-1);
      }
    masks = MK_FP(seg_m, 0);


    init_graph();                            /* initialisation du mode graphique */

    set_fill_mask("\x49\x92\x24\x49\x92\x24\x49\x92");  /* affichage du fond */
    draw_rectangle(239, 127, PATERN);

    set_color(CBLACK);   /* affichage des lignes hautes et basses de l'écran */
    move_pen(0, 0);
    draw_rectangle(239, 8, SOLID);
    move_pen(0, 118);
    draw_rectangle(239, 127, SOLID);
    set_color(CWHITE);
    write_text(2, 1, "FILL MASK EDITOR", 0);
    write_text(2, 119, "Hlp Sav Lod Nam Inv Ins Del         End", 0);

                                                          /* affichage du logo */
    put_image((IMAGE *) &logo_jb0, X_LOGO, Y_LOGO, INVERSE, AND); /*aff mask */
    put_image((IMAGE *) &logo_jb1, X_LOGO, Y_LOGO, NORMAL, OR);   /*aff logo */

    move_pen(X_PATERN-1, Y_PATERN-1);   /* affichage bordure fenêtre hachure */
    set_color(CWHITE);
    draw_rectangle(X_PATERN+W_PATERN, Y_PATERN+H_PATERN, SOLID);
    set_color(CBLACK);
    draw_rectangle(X_PATERN+W_PATERN, Y_PATERN+H_PATERN, OUTLINE);

    move_pen(X_ZOOM-10, Y_ZOOM-11);           /* affichage bordure zone zoom */
    set_color(CWHITE);
    draw_rectangle(X_ZOOM+86, Y_ZOOM+74, SOLID);
    set_color(CBLACK);
    draw_rectangle(X_ZOOM+86, Y_ZOOM+74, OUTLINE);
    move_pen(X_ZOOM-2, Y_ZOOM-2);          /* affichage bordure fenêtre zoom */
    set_color(CWHITE);
    draw_rectangle(X_ZOOM+63+2, Y_ZOOM+63+2, SOLID);
    set_color(CBLACK);
    draw_rectangle(X_ZOOM+63+2, Y_ZOOM+63+2, OUTLINE);

    set_color(CWHITE);                                 /* Affichage fenêtre nom */
    move_pen(X_NAME, Y_NAME);
    draw_rectangle(X_NAME+W_NAME-1, Y_NAME+H_NAME-1, SOLID);
    set_color(CBLACK);
    draw_rectangle(X_NAME+W_NAME-1, Y_NAME+H_NAME-1, OUTLINE);

    if(argc == 2)                                /* si un paramètre à été donné */
       strcpy(filename, argv[1]);                          /* copie dans filename */
    else
       strcpy(filename, DEF_FILENAME);           /* sinon, fichier par défaut */
```

```
strcpy(new_name, DEF_FILENAME);
load(filename);                                  /* chargement du fichier */
disp_filename(filename);                      /* affichage du nom du fichier */

disp_cursor(curs_x, curs_y);               /* affichage du curseur graphique */

hex_code[2] = '\0';    /* initialisation du dernier octet d'une chaine C */
regen = 1;                      /* autorisation de régénération de l'écran */
do
  {
   if(regen)                    /* si autorisation régénération de l'écran */
     {                                    /* effacement fenêtre hachurage */
      move_pen(X_PATERN, Y_PATERN);
      set_color(CWHITE);
      draw_rectangle(X_PATERN+W_PATERN-1, Y_PATERN+H_PATERN-1, SOLID);
      set_color(CBLACK);
      set_fill_mask(curr_mask.mask);             /* sélection nouveau motif */
                                          /* affichage nouveau hachurage */
      draw_rectangle(X_PATERN+W_PATERN-1, Y_PATERN+H_PATERN-1, PATERN);

      move_pen(X_NAME+1, Y_NAME+1);              /* effacement nom hachurage */
      set_color(CWHITE);
      draw_rectangle(X_NAME+W_NAME-2, Y_NAME+H_NAME-2, SOLID);
      set_color(CBLACK);                              /* affichage nouveau nom */
      write_text(X_NAME+3, Y_NAME+2, curr_mask.name, 0);

      move_pen(X_ZOOM+73, Y_ZOOM);              /* effacement des codes Hexa */
      set_color(CWHITE);
      draw_rectangle(X_ZOOM+85, Y_ZOOM+63, SOLID);
      set_color(CBLACK);
                                    /* affichage  du masque de hachurage */
      for(j=0; j<8; j++)                    /* pour chaque octet du masque */
        {
         c=curr_mask.mask[j];                                 /* lecture octet */
                                          /* affichage de l'octet en Hexa */
           /*  (plus rapide que : sprintf(hex_code, "%02X", c & 0xff);) */
         c2 = c & 0xf;
         hex_code[1] = c2 > 9 ? c2-10+'A' : c2+'0';
         c2 = (c>>4) & 0xf;
         hex_code[0] = c2 > 9 ? c2-10+'A' : c2+'0';
         write_text(X_ZOOM+73, Y_ZOOM+(8*j), hex_code, 0);

         for(i=7; i>=0; i--)           /* pour chaque bit (à partir du LSB) */
           {
            put_image((IMAGE *) &big_pixel, X_ZOOM+(i*8),    /*aff pixel */
                                 Y_ZOOM+(j*8), c & 1, FORCE);
            c >>= 1;                                          /* bit suivant */
           }
        }
     }

   regen = 0;
   key=GetKey();                          /* acquisition du code de la touche */
   switch(key)                                /* suivant le code de la touche */
     {
      case PgUp  : update();
                   if(num_mask > 0)  /*positionnement sur motif précédent*/
                     {
                      num_mask--;
                      memcpy(&curr_mask, &masks[num_mask],
                                                      sizeof(FILL_MASK));
                      regen = 1;
                     }
                   break;

      case CR    :
      case PgDn  : update();                                  /* motif suivant */
                   if(num_mask < nb_masks)
                     {
                      num_mask++;
                      memcpy(&curr_mask, &masks[num_mask],
                                                      sizeof(FILL_MASK));
                      regen = 1;
                     }
                   break;

      case HOME  : update();           /* positionnement sur premier motif */
                   num_mask = 0;
                   memcpy(&curr_mask, &masks[num_mask],
                                                      sizeof(FILL_MASK));
                   regen = 1;
                   break;

      case END   : update();             /* positionnement sur fin fichier */
                   num_mask = nb_masks;
                   memcpy(&curr_mask, &masks[num_mask],
                                                      sizeof(FILL_MASK));
                   regen = 1;
                   break;

      case LEFT  : curs_x--;         /* déplacement pointeur vers la gauche */
                   if(curs_x < 0)
                      curs_x = 7;
                   disp_cursor(curs_x, curs_y);
                   break;

      case RIGHT : curs_x++;         /* déplacement pointeur vers la droite */
                   if(curs_x == 8)
                      curs_x = 0;
                   disp_cursor(curs_x, curs_y);
                   break;

      case UP    : curs_y--;           /* déplacement pointeur vers le haut */
                   if(curs_y < 0)
                      curs_y = 7;
                   disp_cursor(curs_x, curs_y);
                   break;

      case DOWN  : curs_y++;            /* déplacement pointeur vers le bas */
                   if(curs_y == 8)
                      curs_y = 0;
                   disp_cursor(curs_x, curs_y);
                   break;

      case ' '   :
      case DEL   : c=curr_mask.mask[curs_y];  /* inversion du bit pointé */
                   i = 7-curs_x;
                   c =  c ^ (1 << i);
                   curr_mask.mask[curs_y]=c;
                   regen = 1;
                   break;

                                          /* retour au motif original */
      case ESC   : memcpy(&curr_mask, &masks[num_mask],
                                                      sizeof(FILL_MASK));
                   regen = 1;
                   break;

      case F1    : disp_help();             /* Affichage de l'écran d'aide */
                   break;

      case F2    : update();                                    /* sauvegarde */
```

```
                if(updated)
                  save(filename);
                break;

        case SF2  : strcpy(new_name, filename);     /* sauvegarde sous... */
                key2 = get_in_box("Save Under...", new_name,
                                  LEN_FILENAME, CWHITE);
                if(key2 != ESC)           /* si touche sortie != ESC */
                  {
                   update();
                   strcpy(filename, new_name);
                   save(filename);     7* sauvegarde sous nouveau nom */
                   disp_filename(filename);
                   regen = 1;
                  }
                break;

                                          /* chargement d'un fichier */
        case F3   : strcpy(new_name, filename);
                key2 = get_in_box("Load File...", new_name,
                                  LEN_FILENAME, CWHITE);
                if(key2 != ESC)         /* si touche de sortie != ESC */
                  {
                   update();
                   if(updated)          /* si fichier courant modifié */
                     {
                      if(yesno_box("DO YOU WANT TO SAVE ?", CWHITE)
                           == YES)
                         save(filename);
                     }
                   if(load(new_name) == 0)      /* chargement fichier */
                     {
                      strcpy(filename, new_name);  /*aff. nom fichier */
                      disp_filename(filename);
                     }
                   regen = 1;
                  }
                break;

                                       /* modification du nom du motif */
        case F4   : ginput(curr_mask.name, LEN_NAME, X_NAME+3,
                                          Y_NAME+2, CBLACK);
                regen = 1;
                break;

        case F5   : for(i=0; i<8; i++)              /* inversion du motif */
                    curr_mask.mask[i] = (char) ~curr_mask.mask[i];
                regen = 1;
                break;

                                         /* insertion d'un motif */
        case F6   : if(nb_masks == MAX_NB_FILLS) /* si nbre maxi de masks */
                  {                                      /* ...atteint */
                   wait_box("Maximum number of masks reached !",
                                                         CBLACK);
                  }
                else                                    /* sinon */
                  {
                   if(num_mask != nb_masks)   /* insertion d'un motif */
                     {
                      update();
                                 /* deplacement des motifs suivants */
                      for(i=nb_masks; i>=num_mask; i--)
                        memcpy(&masks[i+1], &masks[i],
                                            sizeof(FILL_MASK));
                      nb_masks++;           /* incr. nombre de motifs */
                      memset(&curr_mask, 0, sizeof(FILL_MASK));
                      memset(&masks[num_mask], 0, sizeof(FILL_MASK));
                      updated = 1;
                      regen = 1;
                     }
                  }
                break;

                                                /* effacement d'un motif */
        case F7   : if((nb_masks != 0) && (num_mask != nb_masks))
                     {                          /* deplacement motifs suivants */
                      for(i=num_mask; i<=nb_masks; i++)
                        memcpy(&masks[i], &masks[i+1], sizeof(FILL_MASK));
                      nb_masks--;     /* decrémentation nombre de motifs */
                      memcpy(&curr_mask, &masks[num_mask],
                                                 sizeof(FILL_MASK));
                      updated = 1;
                      regen = 1;
                     }
                break;

                                                /* fin de programme */
        case F10  : if(yesno_box("DO YOU REALLY WANT TO QUIT ?", CBLACK)
                          == YES)
                     {
                      update();
                      if(updated)                   /* si fichier modifié */
                        {
                         if(yesno_box("DO YOU WANT TO SAVE ?", CWHITE)
                               == YES)
                            save(filename);
                        }
                      fin=1;                          /* fin du programme */
                     }
                break;

        default   : regen = 0;                          /* autres touches */
                    break;
       }
     }
    while(!fin);        /* boucle tant que la touche F10 n'a pas été pressée */

    exit_graph();                                   /* retour en mode texte */

    freemem(seg_m);          /* lib. mémoire utilisée par le tableau "masks" */

    printf("MASKEDIT Version 1.00\n");
   }


/*****************************************************************************/
/* DISP_HELP : Affiche l'écran d'aide                                        */
/* Entrées : Aucune                                                          */
/* valeur retournée : Aucune                                                 */
/*****************************************************************************/
void disp_help(void)
   {
    GRAPHINFO95 grinfo;              /* sauvegardes des paramètres initiaux */
    unsigned seg;                       /* segment du buffer de sauvegarde */
    IMAGE *buff_box;                       /* pointeur sur le buffer image */
    int len_mess1;                               /* longueur des messages */
    int len_mess2;

    get_graph_info(&grinfo);          /* sauvegarde des paramètres courants */

          /* annulation des paramêtres peut-être modifiés par le programme */
```

```
    set_logorigin(0, 0);
    set_clip_region(0, 0, 239, 127);
    set_line_type(SOLID_LINE);
    set_rule(FORCE);

    allocmem((LEN_IMAGE(240, 128)+15)/16 , &seg);      /* creation du buffer */
    buff_box = (IMAGE *) MK_FP(seg, 0);

    get_image(buff_box, 0, 9, 240, 109);              /* sauvegarde ds buffer */

    set_logorigin(0, 9);          /*prendre coin de la fenêtre comme origine */
    move_pen(0, 0);
    set_color(CWHITE);
    draw_rectangle(239, 108, SOLID);         /* on affiche la fenêtre (vide) */
    set_color(CBLACK);
    draw_rectangle(239, 108, OUTLINE);             /* on affiche une bordure */

                                          /* affichage de l'ecran d'aide */
    write_text(3,   3, "Page Up/Down : Previous/Next Fill Mask.", 0);
    write_text(3,  11, "Home/End     : First/Last Fill Mask.",    0);
    write_text(3,  19, "Space or Del : Inverse a pixel",          0);
    write_text(3,  27, "ESC          : Cancel the changes",       0);
    write_text(3,  35, "F1           : Help about the keys.",     0);
    write_text(3,  43, "F2           : Save the current file.",   0);
    write_text(3,  51, "Shift F2     : Save under a new name.",   0);
    write_text(3,  59, "F3           : Load a file.",             0);
    write_text(3,  67, "F4           : Edit the Fill Mask Name.", 0);
    write_text(3,  75, "F5           : Inverse the patern.",      0);
    write_text(3,  83, "F6           : Insert a new Fill Mask.",  0);
    write_text(3,  91, "F7           : Delete the current Mask.", 0);
    write_text(3,  99, "F10          : Quit the program",         0);

    GetKey();

                                                /* effacement de la fenêtre */
    set_logorigin(0, 0);
    put_image(buff_box, 0, 9, NORMAL, FORCE);       /* réaffichage précédent */
    freemem(seg);          /* libération de la mémoire utilisée par le buffer */

                                        /* restauration valeurs originales */
    set_logorigin(grinfo.col_orig, grinfo.row_orig);
    move_pen(grinfo.pen_col, grinfo.pen_row);
    set_clip_region(grinfo.left_clip, grinfo.top_clip,
                    grinfo.right_clip, grinfo.bottom_clip);
    set_line_type(grinfo.line_type);
    set_color(grinfo.color);
    set_rule(grinfo.rep_rule);
   }


/*****************************************************************************/
/* DISP_FILENAME : Affichage du nom du fichier                               */
/* Entrée  : filename = nom du fichier (avec path, eventuellement)           */
/* Sortie  : Aucune                                                          */
/* NOTE    : par souci de sécurité et de simplicité, resort toujours         */
/*           avec couleur courante CBLACK                                    */
/*****************************************************************************/
void disp_filename(char *filename)
   {
    char string[20];                        /* chaine transmise à write_text() */
    char drive[MAXDRIVE];                 /* variables utilisées par fnsplit() */
    char dir[MAXDIR];
    char name[MAXFILE];                        /* nom et extension du fichier */
    char ext[MAXEXT];

    strupr(filename);
    fnsplit(filename, drive, dir, name, ext);
    move_pen(X_FILE, Y_FILE);
    set_color(CBLACK);
    draw_rectangle(X_FILE+(6*19), Y_FILE+6, SOLID);
    set_color(CWHITE);
    sprintf(string, "File : %s%s", name, ext);
    write_text(X_FILE, Y_FILE, string, NORMAL);
    set_color(CBLACK);
   }


/*****************************************************************************/
/* DISP_CURSOR : affichage du pointeur de la fenêtre zoom                    */
/* Entrée  : x, y = coordonnées du pointeur (x, y = [0..7])                  */
/* Sortie  : Aucune                                                          */
/*****************************************************************************/
void disp_cursor(int x, int y)
   {
    int x1, y1;              /* coordonnées écran du bit dans la fenêtre zoom */

    x1 = X_ZOOM + 3 + (8*x);              /* conversion en coordonnées ecran */
    y1 = Y_ZOOM + 3 + (8*y);

    set_color(CWHITE);
    move_pen(X_ZOOM-8, Y_ZOOM);                  /* effacement curseur gauche */
    draw_rectangle(X_ZOOM-4, Y_ZOOM+63, SOLID);
    move_pen(X_ZOOM+63+4, Y_ZOOM);                /* effacement curseur droit */
    draw_rectangle(X_ZOOM+63+4+4, Y_ZOOM+63, SOLID);
    move_pen(X_ZOOM, Y_ZOOM-9);                    /* effacement curseur haut */
    draw_rectangle(X_ZOOM+63, Y_ZOOM-4, SOLID);
    move_pen(X_ZOOM, Y_ZOOM+63+4);                  /* effacement curseur bas */
    draw_rectangle(X_ZOOM+63, Y_ZOOM+63+4+5, SOLID);

    set_color(CBLACK);
    move_pen(X_ZOOM-8, y1);                              /* tracé curseur gauche */
    draw_rectangle(X_ZOOM-4, y1+1, SOLID);
    move_pen(X_ZOOM+63+4, y1);                            /* tracé curseur droit */
    draw_rectangle(X_ZOOM+63+4+4, y1+1, SOLID);
    move_pen(x1, Y_ZOOM-9);                               /* tracé curseur haut */
    draw_rectangle(x1+1, Y_ZOOM-4, SOLID);
    move_pen(x1, Y_ZOOM+63+4);                             /* tracé curseur bas */
    draw_rectangle(x1+1, Y_ZOOM+63+4+5, SOLID);
   }


/*****************************************************************************/
/* UPDATE  : mise à jour du tableau masks                                    */
/* Entrée  : Aucune                                                          */
/* Sortie  : Aucune                                                          */
/*****************************************************************************/
void update(void)
   {
    if(num_mask >= MAX_NB_FILLS)     /* si masque édité est le dernier et ...*/
      {          /* ...que son numéro est plus grand que le maxi admissible */
       wait_box("Maximum number of masks reached !", CBLACK);
      }
    else                                                        /* cas normal */
      {
       if(memcmp(&curr_mask, &masks[num_mask],           /* si motif modifié */
                         sizeof(FILL_MASK)) != 0)
      {                                 /* sauvegarde du motif courant dans buffer */
          memcpy(&masks[num_mask], &curr_mask, sizeof(FILL_MASK));
          updated = 1;
          if(num_mask == nb_masks)           /* si création du dernier masque */
             nb_masks++;                    /* incrémentation nombre de masques */
        }
      }
```

```
    }


/*****************************************************************************/
/* LOAD    : Lecture du fichier                                              */
/* Entrées : file = nom du fichier                                           */
/* Sortie  : 0 si tout s'est bien passé                                      */
/*****************************************************************************/
int load(char *filename)
   {
    int  fd;                                    /* handle du fichier ouvert */
    int errflag = 0;                                        /* flag d'erreur */
    long magic;                          /* code de reconnaissance du fichier */

    fd = open(filename, O_RDONLY+O_BINARY);          /* ouverture du fichier */
    if(fd == -1)                                /* si le fichier n'existe pas */
      {
       memset(masks, 0, sizeof(FILL_MASK)*(MAX_NB_FILLS+1));/*init buffer à 0*/
       memcpy(&curr_mask, &masks[0], sizeof(FILL_MASK));
       nb_masks = 0;
       num_mask = 0;
       updated = 0;
      }
    else                                            /* sinon, on lit le fichier */
      {
       read(fd, &magic, sizeof(long));          /* lecture en-tête du fichier */
       if(magic != MAGIC)  /* si en-tête différent du code de reconnaissance */
         {
          errflag = 1;                          /* ce n'est pas le bon fichier */
         }
       else
         {
          memset(masks, 0, sizeof(FILL_MASK)*(MAX_NB_FILLS+1)); /*eff. buffer*/
          nb_masks = 0;
                                                          /* lecture du fichier */
          while(read(fd, &masks[nb_masks], sizeof(FILL_MASK))
                                                            == sizeof(FILL_MASK))
             nb_masks++;
          memcpy(&curr_mask, &masks[0], sizeof(FILL_MASK));
          num_mask = 0;
          updated = 0;
         }
       if(errflag)                                    /* si erreur de lecture */
         {
          wait_box("Error : Invalid Masks File", CBLACK);
         }
      }
    close(fd);                                          /* fermeture du fichier */
    return(errflag);
   }


/*****************************************************************************/
/* SAVE    : Sauvegarde du fichier                                           */
/* Entrées : file = nom du fichier                                           */
/* Sortie  : 0 si tout s'est bien passé                                      */
/*****************************************************************************/
int save(char *file)
   {
    int fd;                                          /* handler du fichier */
    long magic=MAGIC;                    /* code de reconnaissance du fichier */
    int errflag = 0;
    int i;

    fd = _creat(file, FA_ARCH);                       /* création du fichier */
    if(fd != -1)
      {                                  /* ecriture du code de reconnaissance */
       if(write(fd, &magic, sizeof(long)) != sizeof(long))
          errflag = 1;
       if(!errflag)                                          /* si pas d'erreur */
         {
          for(i = 0; (i<nb_masks) && !errflag; i++)  /* pour tous les motifs */
            {
             if(write(fd, &masks[i], sizeof(FILL_MASK)) != sizeof(FILL_MASK))
                errflag=1;
            }
         }
       if(errflag)                      /* si il y a eu une erreur d'écriture */
          wait_box("Error : cannot write file", CBLACK);
       else
          updated = 0;
      }
     close(fd);                                          /* fermeture du fichier */
     return(errflag);
    }



/*****************************************************************************/
/*****************************************************************************/
/**            FONCTIONS REUTILISABLES dans un autre programme              **/
/*****************************************************************************/
/*****************************************************************************/


/*****************************************************************************/
/**  DETECT_HP95 : detection d'exécution du programme sur HP95               **/
/**  Entrées  : aucune                                                       **/
/**  sortie   : 1 si HP-95 trouvé                                            **/
/*****************************************************************************/
int detect_hp95(void)
   {
    union REGS regs;
    int retour=0;

    regs.x.ax= 0x4dd4;                        /* fonction BIOS spécifique HP95 */
    int86(0x15, &regs, &regs);
                                    /* valeurs de retour définis pour le HP95 */
    if((regs.h.bh == 'H') && (regs.h.bl == 'P') && (regs.h.ch == 0x01))
       retour=1;

    return(retour);
   }


/*****************************************************************************/
/* GETKEY  : lecture d'un caractère au clavier                               */
/* Entrée  : Aucune                                                          */
/* Sortie  : Code de touche reçue          < 256 : touche normale            */
/*                                        >= 256 : touche étendue            */
/*****************************************************************************/
int  GetKey(void)
   {
    union REGS Register;  /* Variable de registres pour appel d'interruption */

    do
      {
       Register.h.ah = 2;        /* Numéro de fonction pour Lire état clavier */
       int86(0x16, &Register, &Register);    /* Appeler int. clavier du BIOS */
      }
    while(!bioskey(1));                 /* Répéter jusqu'à ce que touche prête */
    Register.h.ah = 0;                      /* N° de fonction pour Lire touche */
```

```
    int86(0x16, &Register, &Register);  /* Appeler interruption clavier BIOS */
    return((Register.h.al) ? Register.h.al : Register.h.ah | 256);
   }


/*****************************************************************************/
/* MSG_BOX2: Affichage d'une fenêtre de message de deux lignes, avec attente */
/*           de pression de touche.                                          */
/* Entrées : message1 = chaine contenant texte 1ere ligne (35 caract. maxi)  */
/*           message2 = chaine contenant texte 2eme ligne (35 caract. maxi)  */
/*           color : couleur du fond de la fenêtre                           */
/*           exit_codes = chaine contenant les caractères autorisant la      */
/*                        sortie (chaine vide si attente de touche simple).  */
/*                        Touches de fonctions non autorisées                */
/* valeur retournée : code de touche ayant provoqué la sortie                */
/*****************************************************************************/
int msg_box2(char *message1, char *message2, int color, char *exit_codes)
   {
    int color_text;                                  /* couleur du texte */
    int code;                                        /* code de la touche */
    GRAPHINFO95 grinfo;              /* sauvegardes des paramètres initiaux */
    int x_box;                              /* Coordonnées de la fenêtre */
    int y_box=64-16;
    int w_box;                          /* largeur de la fenêtre en pixels */
    int h_box=32;                       /* hauteur de la fenêtre en pixels */
    unsigned seg;                       /* segment du buffer de sauvegarde */
    IMAGE *buff_box;                       /* pointeur sur le buffer image */
    int len_mess1;                               /* longueur des messages */
    int len_mess2;

    color_text = (color == CWHITE) ? CBLACK : CWHITE;

    get_graph_info(&grinfo);           /* sauvegarde des paramètres courants */

          /* annulation des paramêtres peut-être modifiés par le programme */
    set_logorigin(0, 0);
    set_clip_region(0, 0, 239, 127);
    set_line_type(SOLID_LINE);
    set_rule(FORCE);

    len_mess1 = strlen(message1);
    len_mess2 = strlen(message2);
    w_box = (max(len_mess1, len_mess2) + 4) * 6;   /* calcul largeur fenêtre */
    x_box = 120 - (w_box/2);                       /* centrage de la fenêtre */

    allocmem((LEN_IMAGE(w_box, h_box)+15)/16 , &seg);  /* creation du buffer */
    buff_box = (IMAGE *) MK_FP(seg, 0);

    get_image(buff_box, x_box, y_box, w_box, h_box); /* sauvegarde ds buffer */

    set_logorigin(x_box, y_box);  /*prendre coin de la fenêtre comme origine */
    set_color(color);
    move_pen(0, 0);
    draw_rectangle(w_box-1, h_box-1, SOLID);   /*on affiche la fenêtre (vide)*/
    set_color(color_text);
    draw_rectangle(w_box-1, h_box-1, OUTLINE);     /* on affiche une bordure */
    set_line_type(DOTTED_LINE);
    move_pen(3, 3);
    draw_rectangle(w_box-4, h_box-4, OUTLINE);  /* un cadre, pour faire joli */
    move_pen(0, 0);
    write_text((w_box-(len_mess1*6)) / 2,  8, message1, 0); /*affichage texte*/
    write_text((w_box-(len_mess2*6)) / 2, 17, message2, 0);

    if(strlen(exit_codes) == 0)          /* si aucun code de sortie spécifié */
      {                          /* on sortira après n'importe quelle touche */
       GetKey();                              /* attente de pression de touche */
       code = 0;
      }
    else           /* sinon, on ne sortira qu'après une pression sur une ... */
      {                                  /* ... touche définie dans exit_codes */
       do
         {
          code=GetKey();                        /* attente de pression de touche */
          if(islower(code))              /* conversion des caractères minuscules */
            code=_toupper(code);
         }
       while(strrchr(exit_codes, code) == NULL); /* tant que code de sortie..*/
      }                                                      /* ..non activé */

                                                    /* effacement de la fenêtre */
    set_logorigin(0, 0);
    put_image(buff_box, x_box, y_box, NORMAL, FORCE); /*réaffichage précédent*/
    freemem(seg);          /* libération de la mémoire utilisée par le buffer */

                                              /* restauration valeurs originales */
    set_logorigin(grinfo.col_orig, grinfo.row_orig);
    move_pen(grinfo.pen_col, grinfo.pen_row);
    set_clip_region(grinfo.left_clip, grinfo.top_clip,
                    grinfo.right_clip, grinfo.bottom_clip);
    set_line_type(grinfo.line_type);
    set_color(grinfo.color);
    set_rule(grinfo.rep_rule);

    return(code);
   }


/*****************************************************************************/
/* WAIT_BOX: Affichage d'une fenêtre de message de deux lignes, avec attente */
/*           de pression de touche.                                          */
/* Entrées : message = chaine contenant texte 1ere ligne (35 caract. maxi)   */
/*           color : couleur du fond de la fenêtre                           */
/* valeur retournée : aucune                                                 */
/*****************************************************************************/
void wait_box(char *message, int color)
   {
    msg_box2(message, "Press any key", color, "");
   }


/*****************************************************************************/
/* YESNO_BOX : Affichage d'une fenêtre de message, avec attente de pression  */
/*           de touche Y ou N.                                               */
/* Entrées : message = chaine contenant le texte du message (35 caract. maxi)*/
/*           color : couleur du fond de la fenêtre                           */
/* valeur retournée : 'Y' ou 'N'                                             */
/*****************************************************************************/
int yesno_box(char *message, int color)
   {
    return(msg_box2(message, "Press Y or N", color, "YN"));
   }


/*****************************************************************************/
/**  GINPUT  : entrée d'une chaine de caractères en mode graphique          **/
/**  Entrées  : string = chaine recevant les données (peut-être déjà init.) **/
/**             len = longueur maximum de la chaine                         **/
/**             x, y = Position de la chaine (Coin sup/gauche 1er caractère)**/
/**             color = couleur du fond de la fenêtre                       **/
/**  sortie   : code de la touche ayant provoqué la sortie                  **/
/*****************************************************************************/
                                                    /* Description du Curseur */
```

```
DESCRIPT_IMAGE(7, 8, IMAGE_7_8)
                                        /* Curseur en mode Overwrite */
IMAGE_7_8 curs_overw = {
                         1, 1, 7, 8,                          /*  HEADER  */
                         {
                          0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
                         }
                        };
                                        /* Curseur en mode Insertion */
IMAGE_7_8 curs_ins   = {
                         1, 1, 7, 8,                          /*  HEADER  */
                         {
                          0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff
                         }
                        };

int  Insert;                /* flag indiquant que l'on est en mode insertion */

int ginput(char *string, int len, int x, int y, int color)
   {
    char str_init[80];                      /* stockage de la chaine intiale */
    int curpos2=0;                                  /* position du curseur */
    int color_text;                                   /* couleur du texte */
    GRAPHINFO95 grinfo;             /* sauvegarde des paramètres graphiques */
    union REGS Register;  /* Variable de registres pour appel d'interruption */
    int  Ins;                                       /* flag de mode insertion */
    int j, key, fin;

    strcpy(str_init, string);               /* sauvegarde chaine originale */
    color_text = (color == CWHITE) ? CBLACK : CWHITE;

    get_graph_info(&grinfo);           /* sauvegarde des paramètres courants */

    set_rule(FORCE);

    move_pen(x-2, y-1);       /* curseur en haut/droite de la zone d'edition */

    *((unsigned far *) MK_FP(0x40, 0x17)) &= 63;    /* force INS et CAPS à 0 */
    Insert = 0;

    fin=0;
    do
      {
       set_color(color);                    /* effacement du texte précédent */
       draw_rectangle(x+(6*(len+1))+1, y+8, SOLID);
       set_color(color_text);                  /* affichage du nouveau texte */
       write_text(x, y, string, 0);

       Insert = 2;                          /* on force l'affichage du curseur */
       do
         {
          Register.h.ah = 2;    /* Numéro de fonction pour Lire état clavier */
          int86(0x16, &Register, &Register);  /*Appeler int. clavier du BIOS */
          Ins = (Register.h.al & 128) ? 1 : 0;     /* lecture mode insertion */
          if(Ins != Insert)                     /* si mode insertion modifié */
            {
             Insert = Ins;
             if(Insert)                                /* affichage du curseur */
                put_image((IMAGE *) &curs_ins, x+(6*curpos2), y, NORMAL, XOR);
             else
                put_image((IMAGE *) &curs_overw, x+(6*curpos2), y,
                                                                 NORMAL, XOR);
            }
         }
       while(!bioskey(1));            /* Répéter jusqu'à ce que touche prête */
       Register.h.ah = 0;               /* N° de fonction pour Lire touche */
int86(0x16, &Register, &Register); /*Appeler interruption clavier BIOS*/
key = (Register.h.al) ? Register.h.al : Register.h.ah | 256;

if((key>31) && (key<255))                    /* si caractere editable */
  {
   if(curpos2 != len)                 /* si curpos diff. de long. max */
     {
      if(Insert)                             /* si mode Insertion activé */
        {
                                    /* si long. chaine diff. long. max */
         if(strlen(string) != len)
           {
            for(j=len; j>=curpos2; j--)              /* inserer caract*/
               string[j]=string[j-1];
            string[curpos2]=key;
            curpos2++;                                /* deplacer curseur */
           }
        }
      else                                                    /* sinon */
        {
         if(curpos2 == strlen(string))           /* si dernier caract */
            string[curpos2+1]='\0';             /* terminer avec '\0' */
         string[curpos2]=key;
         curpos2++;                                /* deplacer curseur */
        }
     }
  }
else                                        /* si caractere non éditable */
  {
   switch(key)                          /* suivant la valeur de la touche */
     {
      case BS    : if(curpos2 != 0)  /* si on peut effacer un caract. */
                     {                          /* decalage vers la gauche */
                      for(j=curpos2; j<=len; j++)
                         string[j-1]=string[j];
                      curpos2--;           /* repositionnement curseur */
                     }
                   break;

      case DEL   : if(curpos2 != strlen(string))
                     {                          /* decalage vers la gauche */
                      for(j=curpos2; j<=len; j++)
                         string[j]=string[j+1];
                     }
                   break;

      case LEFT  : if(curpos2 != 0)   /* deplacement curseur a gauche */
                      curpos2--;
                   break;
                                    /* deplacement du curseur à droite */
      case RIGHT : if(curpos2 < strlen(string))
                      curpos2++;
                   break;
                                                /* curseur en fin de chaine */
      case END   : curpos2 = strlen(string);
                   break;

      case HOME  : curpos2 = 0;        /* deplacement curseur a gauche */
                   break;

                                                          /* si touche ESC */
      case  ESC  : if(strcmp(string, str_init) == 0)
                      fin = 1;         /* si chaine originale, on sort */
                   else                                       /* sinon... */
                     {                /* ...on affiche chaine originale */
                      strcpy(string, str_init);
```

```
                        if(curpos2 > strlen(string))
                           curpos2 = strlen(string);
                        }
                      break;

          case  CR   : fin=1;                /* si touche Entrée pressée */
                       break;

          default    :  break;                          /* autres touches */
                     }
        }
      }
    while(!fin);           /* boucle tant que CR ou ESC n'a pas été pressée */

    *((unsigned far *) MK_FP(0x40, 0x17)) &= 63;    /* force INS et CAPS à 0 */

    move_pen(grinfo.pen_col, grinfo.pen_row);
    set_rule(grinfo.rep_rule);


    return(key);
   }


/*****************************************************************************/
/* GET_IN_BOX : Entrée d'un chaine de caractères dans une fenêtre            */
/* Entrées : titre = titre de la fenêtre                                     */
/*           string = chaine à ecrire                                        */
/*           len_string = longueur maximale de la chaine                     */
/*           color : couleur du fond de la fenêtre                           */
/* valeur retournée : code de touche ayant provoqué la sortie                */
/*****************************************************************************/
int get_in_box(char *titre, char *string, int len_string, int color)
   {
    int color_text;                                  /* couleur du texte */
    int code;                                       /* code de la touche */
    GRAPHINFO95 grinfo;              /* sauvegardes des paramètres initiaux */
    int x_box;                              /* Coordonnées de la fenêtre */
    int y_box=64-16;
    int w_box;                          /* largeur de la fenêtre en pixels */
    int h_box=32;                       /* hauteur de la fenêtre en pixels */
    unsigned seg;                       /* segment du buffer de sauvegarde */
    IMAGE *buff_box;                       /* pointeur sur le buffer image */
    int len_titre;                               /* longueur des messages */

    color_text = (color == CWHITE) ? CBLACK : CWHITE;

    get_graph_info(&grinfo);           /* sauvegarde des paramètres courants */

           /* annulation des paramêtres peut-être modifiés par le programme */
    set_logorigin(0, 0);
    set_clip_region(0, 0, 239, 127);
    set_line_type(SOLID_LINE);
    set_rule(FORCE);

    len_titre = strlen(titre);
    w_box = (max(len_titre, len_string) + 4) * 6;  /* calcul largeur fenêtre */
    x_box = 120 - (w_box/2);                       /* centrage de la fenêtre */

    allocmem((LEN_IMAGE(w_box, h_box)+15)/16 , &seg);  /* creation du buffer */
    buff_box = (IMAGE *) MK_FP(seg, 0);

    get_image(buff_box, x_box, y_box, w_box, h_box); /* sauvegarde ds buffer */

    set_logorigin(x_box, y_box);  /*prendre coin de la fenêtre comme origine */
    set_color(color);
    move_pen(0, 0);
    draw_rectangle(w_box-1, h_box-1, SOLID);   /*on affiche la fenêtre (vide)*/
    set_color(color_text);
    draw_rectangle(w_box-1, h_box-1, OUTLINE);     /* on affiche une bordure */
    set_line_type(DOTTED_LINE);
    move_pen(3, 3);
    draw_rectangle(w_box-4, h_box-4, OUTLINE);  /* un cadre, pour faire joli */
    move_pen(0, 0);
    write_text((w_box-(len_titre*6)) / 2,  5, titre, 0);  /* affichage titre */

    code = ginput(string, len_string, ((w_box-(len_string*6)) / 2)-3,
                                                          16, color_text);

                                                  /* effacement de la fenêtre */
    set_logorigin(0, 0);
    put_image(buff_box, x_box, y_box, NORMAL, FORCE); /*réaffichage précédent*/
    freemem(seg);          /* libération de la mémoire utilisée par le buffer */

                                              /* restauration valeurs originales */
    set_logorigin(grinfo.col_orig, grinfo.row_orig);
    move_pen(grinfo.pen_col, grinfo.pen_row);
    set_clip_region(grinfo.left_clip, grinfo.top_clip,
                    grinfo.right_clip, grinfo.bottom_clip);
    set_line_type(grinfo.line_type);
    set_color(grinfo.color);
    set_rule(grinfo.rep_rule);

    return(code);
}
```

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 HP28

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.

Tapez les deux programmes ASSCOD.28 et INPUT.28 avec la plus grande attention car leur mauvais fonctionnement peut entraîner un désordre fatal pour les objets contenus dans votre machine. La commande SPEED peut être incluse si vous possédez ce programme. La commande ASC→ peut remplacer les lignes avec @@ jusqu'à *EVAL*.

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                     @@
»
```

INPUT.28

```
« 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.
6- si tout s'est bien déroulé vous pouvez purger tmpcod qui contient la chaîne de codes.

Programmes HP48

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)
»
```

Le mode d'emploi est le même que pour la HP28.

```
ASCX                    (HP28)
# 26C6h           85.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C069 C20D   E31
001: 8000 8F18 0501 4313   987
002: 0131 34E4 A201 438A   68F
003: 2918 0823 2020 8F8B   3F8
004: 0508 DA69 3017 4143   006
005: 818F 855F 0808 2430   C48
006: 2004 8D81 9F0D 8169   A37
007: 1743 1931 5B19 EA80   805
008: 8186 8615 8017 1160   356
009: CD54 E8D5 E521 AAEC   3C4
00A: 3D3F 4009 F20         1E6


BYTES28S                (HP28)
# F0FCh          189.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C0A4 0201   C47
001: F7E3 3ACD 376C 20DF   BFD
002: FB05 E020 F2C0 1D3F   A9F
003: 405E 0206 9C20 9300   686
004: 0147 174E 7137 174D   3CC
005: 014B C481 8F08 1351   06F
006: 4713 7179 1411 358D   D1F
007: CE52 109F 2076 C20F   B01
008: 2C01 D3F4 009F 205E   8FD
009: 0206 9C20 FC00 0143   544
00A: 174E 7137 0613 58F1   21F
00B: 8050 1B00 FFF3 0F15   072
00C: C0D6 1340 68F4 9E20   DED
00D: 0713 5132 EE10 8D7C   C81
00E: F071 3414 2130 169D   8E3
00F: 230F 10AD 2061 5B07   5EC
010: 2508 1AF1 20EF 6D62   486
011: 6C40 D5BF C224 C40D   362
012: 5BFC A07F 6782 0170   032
013: CF56 C128 142C A120   CFF
014: 1408 F8B0 50D2 208D   AB7
015: 3257 1D50 EFC0 EF2F   B36
016: E0EF 5012 2E13 ED21   8FC
017: 128D 1109 F20         6F6


XASC                    (HP28)
# 338Ch           91.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C069 C203   D21
001: A000 8F18 0501 4713   8B8
002: 4068 F49E 2007 132E   5E6
003: A103 C481 8F09 1027   267
004: 860D 68FA B9E1 1841   073
005: 1281 8F84 1401 6413   C1A
006: 2764 0130 1431 3111   699
007: BD7C F319 3AEA 15B0   55C
008: 9EA8 0818 6061 5811   12D
009: 7016 1CF5 1E7D 0011   E3F
00A: 8145 8DCE 5218 D8B0   CE4
00B: 5009 F20              2C7


RCLDIR28                (HP28)
# 34F9h          12.5 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C073 7E16   D91
001: 0120 09F2 0           6E9


PURGEDIR                (HP28)
# E95Ch            25 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C05   DE8
001: E170 76C2 0DFF B060   C76
002: 1205 2A60 09F2 009F   9F4
003: 20                    A86


DEBUG28S                (HP28)
# 2176h            30 octets

     0123 4567 89AB CDEF   sm

000: 76C2 01E3 C05A 7C09   E28
001: F170 76C2 0F14 5024   A67
002: 8E05 E020 CAF8 0814   80A
003: A009 F200 9F20        89C


SPEED                   (HP28)
# A389h            17 octets

     0123 4567 89AB CDEF   sm

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


FINDMEM                 (HP48)
# ED36h          159.5 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0FDE 818A B46D   FD4
001: 9D20 2C23 0756 60CC   D8D
002: D202 1100 8FB9 7601   A4D
003: 4313 0164 AF21 4681   6DD
004: ECEC ED58 AE60 6780   53C
005: 1641 3213 0100 1741   FD8
006: 4713 5179 1371 0913   B4E
007: 1D9D 714A 3103 B6A3   8E5
008: 1A09 E290 3170 B6A1   61F
009: 5101 7016 1CF8 AF7D   5D7
00A: 1101 3015 60A8 7111   175
00B: 1311 4313 1157 0903   C94
00C: A217 053F 1371 1113   805
00D: 1145 8F2D 7601 74E7   5C7
00E: 1421 6480 8C13 3131   16F
00F: 102D 9D7C F8AF C011   0A3
010: 2131 63CF 1704 CB16   E7A
011: 0157 0152 0902 BD11   AF3
012: 0130 1560 A871 1213   675
013: 1609 FB21 30B2 130    F46


RECOVER                 (HP48)
# A824h           254 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0ECE 819F F30D   FDA
001: 9D20 AEC8 1CCD 204D   F47
002: 1008 F146 608F B976   D6F
003: 0D33 4004 0713 4135   894
004: 8A83 2CC8 A8C6 CC8A   92E
005: 817D B108 8F73 5608   667
006: D48D 0034 0000 81DC   372
007: 215B 0808 61F0 1701   F6E
008: 43FC C21C 017A 15B0   D14
009: 8086 1F01 7014 3FCC   B7F
00A: 21C0 D51A 7950 1421   790
00B: 306B 3019 C277 116B   4C0
00C: 0019 737B 0116 0146   064
00D: FE16 4142 130C AD86   E5D
00E: E001 4213 0143 D831   A40
00F: 2014 A962 8116 0132   59A
010: 8B46 06D4 F132 65EF   472
011: 1823 404B 2014 28A2   0CB
012: 5134 26B2 08A2 9016   D2E
013: 36CB F164 1428 18F8   AEA
014: 3136 1088 FE79 50D2   8EB
015: 15E3 8A2C 0110 1306   4A1
016: E8FE 71B9 7507 1F47   27D
017: 5071 4214 7EE8 18FA   191
018: 44E3 818F 8413 1140   D74
019: 1108 18F8 4141 1FB0   ADB
01A: 1003 1F81 4D32 FFFA   B02
01B: 3E5C F310 814D 6C9F   A3E
01C: D230 1DA8 F2D7 608D   923
01D: 3205 0152 0808 63B0   52D
01E: 8087 1400 1D23 0A66   1B9
01F: DFB2 130B 2130        1BB
```

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

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

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

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

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

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

Imprimé par Paris Copie, 4 rue Linné, 75005 Paris