JPC

NOVEMBRE 1987
NUMERO 49

Le numéro 40 F

A PROPOS DU CLUB

HP28

HP41

HP75

HP71

EDITORIAL

Chers co-membres,

L'Assemblée Générale se tiendra le samedi 16 janvier 1987 au Centre Jean Verdier, de 14 à 16h, juste avant la réunion mensuelle habituelle. A noter dans vos tablettes.

Nous vous enverrons prochainement une convocation. D'ici là, n'hésitez pas à nous prévenir si vous désirez faire partie du Bureau ou si vous souhaitez voir abordé un point particulier.

Nous parlerons de sujets importants pour la vie du Club. Aussi je compte sur votre présence.

A bientôt donc,

Pierre David (37)

PPC PARIS SE REUNIT UNE FOIS PAR MOIS

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

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

Si donc cela vous intéresse, n'hésitez plus un seul instant, venez nous rejoindre tous les premiers samedis de chaque mois (sauf en période de vacances scolaires) au :
Centre de Jeunesse et de Loisirs Jean Verdier
11 rue de Lancry
75010 Paris

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

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

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

Les dates des prochaines réunions sont :
Samedi 21 novembre 1987
Samedi 5 décembre 1987
Samedi 16 janvier 1988
Samedi 20 février 1988
Samedi 5 mars 1988
Samedi 16 avril 1988
Samedi 7 mai 1988
Samedi 4 juin 1988

Pierre David (37)

AH ! VOUS ECRIVEZ

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

Dans la mesure du possible, vous devez nous envoyer vos écrits sur support magnétique (carte, cassette ou disquette). Soyez sans crainte, nous vous retournerons vos biens après copie.

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

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

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

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

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

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

N'oubliez pas de mettre les accents. Utilisez le jeu de caractères Roman8. Les possesseurs de HP71 utiliseront les redéfinitions de touches ci-dessous, ainsi que le fichier CHARLEX listé dans le coin des Lhex en fin de journal.

Jean-Jacques Dhénin (177)

```
DEF KEY 'fW', CHR$(197);      (é)
DEF KEY 'fE', CHR$(193);      (ê)
DEF KEY 'fR', CHR$(201);      (è)
DEF KEY 'fY', CHR$(203);      (ù)
DEF KEY 'fU', CHR$(195);      (û)
DEF KEY 'fI', CHR$(209);      (î)
DEF KEY 'fO', CHR$(194);      (ô)
DEF KEY 'f/', CHR$(92);       (\)
DEF KEY 'fA', CHR$(192);      (â)
```

```
DEF KEY 'fS', CHR$(200);      (à)
DEF KEY 'fD', CHR$(205);      (ë)
DEF KEY 'fJ', CHR$(207);      (ü)
DEF KEY 'fK', CHR$(221);      (ï)
DEF KEY 'f*', CHR$(124);      (|)
DEF KEY 'fC', CHR$(181);      (ç)
```

COURRIER DES LECTEURS

Dominique Marcaillou
29, rue du Moulin
31320 Castanet
Tél : 64 81 74 92

Cher Pierre, cher Janick,

Je viens de me rendre compte d'une très grosse gaffe financière : avoir acheté le lecteur de cartes pour HP-71 en dehors du Club (1400F au lieu de 500F) : quelle erreur ! Je viens de m'apercevoir que malgré une attention particulière à la propreté des cartes, le message `ERR:R/W Error` revenait trop souvent. C'est pour cela qu'avant de lire la carte, l'utilisation d'un coton tige et d'alcool à 90° réduit considérablement ce facteur d'erreur.

Je tiens à féliciter les auteurs des Lex FIND et RWLEX qui sont carrément géniaux. A ce sujet, je propose deux définitions de touches qui permettent une utilisation plus facile et plus rapide de `FIND` :

```
DEF KEY "f ","IFA$#'' THENFINDA$":

DEF KEY "g ","DIMA$[96]@A$=DISP$@IFA$#''THEN
              DELAY0,0@EDITCAT$(0)@FINDA$":
```

- la première (touche `[f][SPC]`) recherche toutes les occurences de la variable A$ dans le fichier courant,

- la seconde recherche la première position de l'affichage dans le fichier en remettant le pointeur de la ligne courante à la première ligne du fichier. Il subsiste envore un problème : lorsque `FIND` recherche le message, il ne traite pas la première ligne du programme, même si le pointeur est au début et même si le message est contenu dans la première ligne.

J'aimerais bien savoir comment pratiquement Michel Martinet a pu pousser son HP-71 à 86 Ko. Il avait succintement relaté la méthode théorique dans JPC 30 de Décembre 85 / Janvier 86. Cela serait sympa si quelqu'un m'expliquait ce bricolage, car les 17 Ko sont déjà saturés. Car avec les facilités qu'arbore le HP-71 dans le stockage de fichiers par l'adjonction de modules et par la gestion de la mémoire, cela serait idiot de s'en passer.

Voila, c'est terminé, et que l'ensemble de mes lecteurs passent de bonnes vacances,

Amicalement,

Dominique Marcaillou (315)

COURRIER DU COEUR

PPC-Paris
B.P. 604
75028 Paris Cedex 01

Vend :
Lecteurs de cartes magnétiques HP82400A neufs pour HP-71 (dans leur boîte, avec documentation et 5 cartes magnétiques) : 500 F seulement.

1 lot de cartes magnétiques pour HP-41, HP-67, HP-97 et même HP-65 : 100 F seulement.

Jean-Jacques Moreau
ENST de Bretagne
B.P. 832
29285 Brest Cedex
Tél : 98 00 17 43

Vend :
HP-71 : 4000 F, lecteur de disquettes : 3000 F, module HP-IL : 1400 F, module Forth / Assembleur : 1400 F, module Maths : 800 F, excellent état.

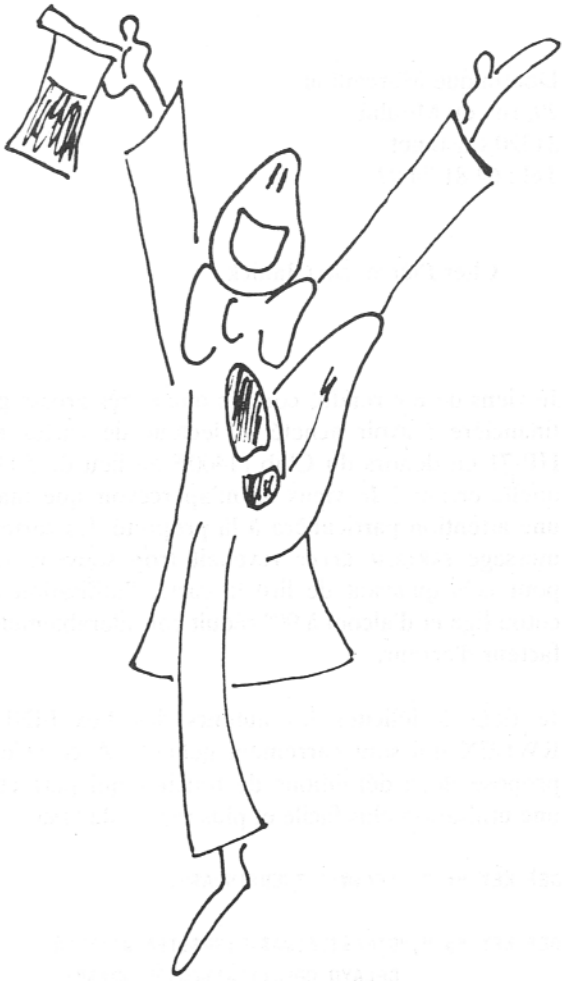
Laurent Eymard
22 rue Lechantre
02100 Saint Quentin
Tél : 23 62 05 92 (le week-end)

Vend :
Module HP-IL + littérature (manuels, Autour de la Boucle) + cable : 770 F.

Jacques Dupuis
26 rue de Chartres
91400 Orsay
Tél : (1) 69 28 82 58

Vend :
Lecteur de cassettes HP82161A + 4 cassettes : 3500 F, interface vidéo HP82163A : 1000 F, extension de ports (8 ports) pour HP-41 : 500 F, 2 modules mémoire HP82106A : 200 F.

SOS



Dr A. Bezzaoucha
8 rue Girardin
Parc de la Liberté
16000 Alger
Algérie

Cherche :

Des programmes d'analyses statistiques et épidémiologiques, y compris l'analyse multi-variée pour HP-41CV.

Roger Miollan
Infirmier-Hygiéniste
Centre Hospitalier
01012 Bourg-en-Bresse Cedex

Cherche :

Un programme de calcul de coefficient de corrélation pour un groupement de données à deux dimensions suivant la formule :

$$r = \frac{\Sigma n_{ij} x_i y_j - (\Sigma n_i x_i \times \Sigma n_j y_j)/n}{\sqrt{[\Sigma n_i x_i^2 - (\Sigma n_i x_i)^2/n] \times [\Sigma n_j y_j^2 - (\Sigma n_j y_j)^2/n]}}$$

HP28

E. Gengoux

MODE HORLOGE (ACTE II)

JPC 46 (Juillet - Août 1987) contenait un programme pour simuler une horloge sur le HP-28C.

La version présentée fonctionnait parfaitement pour la version 1BB du HP-28C.
Le numéro de version est obtenu en faisant `#10 SYSEVAL` en mode décimal.

Une nouvelle version de HP-28C va être mise prochainement en circulation, ce sera la version 1CC. Le programme diffusé dans JPC 47 devra donc contenir : `#1266 SYSEVAL` au lieu de `#123E SYSEVAL`.

Note de la rédaction : il passera encore beaucoup d'eau sous les ponts avant que ces nouvelles versions soient disponibles chez votre revendeur ! Pas la peine de l'embêter tous les jours en lui demandant : « alors, la nouvelle version est arrivée ? » Sachez également que HP ne remplacera pas les versions 1BB en 1CC. Dommage...

ENCORE DES COURBES !

Il est bien établi que « tracer des courbes sur la HP-28, mais c'est très simple ! » (voir l'article de Pierre David dans JPC 44). Nous irons un peu plus loin cette fois, avec par ordre d'entrée en scène :

- un petit perfectionnement permettant de modifier plus facilement une courbe déjà tracée,
- un programme de tracé de courbes données par leur équation polaire,
- et enfin, un autre programme de tracé de courbes données par leur équation implicite, c'est à dire données sous la forme $f(x,y)=0$.

Ainsi, votre trousse à outils de petit botaniste collectionneur des belles fleurs qui poussaient jadis dans tous les bons ouvrages de Math, avec des noms à faire rêver : Néphroïde de Machin-Chose, non, Docteur, ce n'est pas une maladie de l'appareil génito urinaire ! Sera-t-elle complète, et pourrez-vous, Taupin mon frère, ne plus « sortir de la route » dans un virage mal négocié au détour d'une branche que vous voyiez tout autrement qu'elle n'était...

UN PERFECTIONNEMENT UTILE EN PARAMETRIQUES

L'utilisation toute bête de la routine PARAM (ibid), en lui passant ses arguments par la pile opérationnelle, ne permet pas commodément de redéfinir la fenêtre de tracé (si ça dépasse, ou que c'est trop petit ou trop grand), ni de répéter le tracé avec des paramètres différents dans les équations. Songez aux courbes de Lissajous ($x=\sin(at)$, $y=\sin(bt)$, avec a et b entiers), bien connues des électroniciens...

Créons donc un programme appelant qui va définir le domaine de tracé et les axes, et passer à notre routine PARAM les deux équations. Il sera facile de modifier ce programme et de le relancer, sans toucher PARAM et sans avoir à se souvenir de ce qu'on avait mis dans la pile...

Ecrivons donc comme suit :

```
« RAD
  (-1.1,-1.1) PMIN
  (1.1,1.1) PMAX
  (0.0,0.0) AXES
  'SIN(8*T)' 'SIN(3*T)' 0.0 6.28
  PARAM
  PRLCD
»
```

Bien entendu, PARAM est toujours la même :

```
« → y m1 m2
  « CLLCD DRAX
    m2 m1 - 400 /
    m1 m2
    FOR t
      t 'T' STO
      x EVAL y EVAL R→C PIXEL
      DUP
    STEP
    DROP
  »
»
```

Notons au passage qu'il est plus naturel d'utiliser `[PMIN]` et `[PMAX]` que `[*W]` et `[*H]` pour définir ou modifier la fenêtre de tracé.

CLIMAT POLAIRE

Avec la même philosophie de programme appelant, définissons une routine qui prend dans la pile, d'une part l'expression $r=f(t)$, où t est l'angle et r le rayon, et d'autre part les limites de l'intervalle de variation de t. On aura la routine de tracé POLR suivante :

```
« → r t1 t2
  « CLLCD DRAX
    t2 t1 - 100 /
    t1 t2
    FOR t
      t 'T' STO
      r EVAL t R→C P→R PIXEL
      DUP
    STEP
    DROP
  »
»
```

Ecrivons à présent le programme appelant pour un Limaçon de Pascal d'équation $r = \cos t + 0.2$:

```
« RAD
  (-0.5,-1) PMIN
  (1.5,1) PMAX
  (0,0) AXES
  'COS(T)+0.2' 0.00 12.57
  POLR
  PRLCD
»
```

On voit que ça reste tout à fait simple et de bon goût...

IMPLICITE, MAIS TRES EXPLICITE

Les deux routines qui suivent (toujours le même système d'appel de l'une par l'autre) sont d'exécution nettement plus lente : en effet, on n'y fait plus varier un seul paramètre t par pas sur un intervalle, mais deux (x et y) pour balayer toute la fenêtre à la manière du « balayage lignes » de votre téléviseur, et on teste pour chaque point s'il vérifie ou non l'équation, ou plus exactement s'il est suffisamment rapproché de la courbe pour qu'on puisse allumer le pixel correspondant. L'exemple est un cercle de rayon unité, ce qui va permettre de comparer les temps de tracé pour chacun des trois systèmes (paramétriques, polaires, implicite).

Voici donc la routine graphique FXY :

```
« → x1 x2 y1 y2
  « CLLCD DRAX
    x2 x1 - 50 / x0 STO
    y2 y1 - 50 / y0 STO
    x1 x2
    FOR x
      y1 y2
      FOR y
        x 'X' STO
        y 'Y' STO
        f EVAL ABS 0.03
        IF ≤
          THEN
            x y R→C PIXEL
        END
        y0
      STEP
      x0
    STEP
  »
»
```

Passons à la routine appelante :

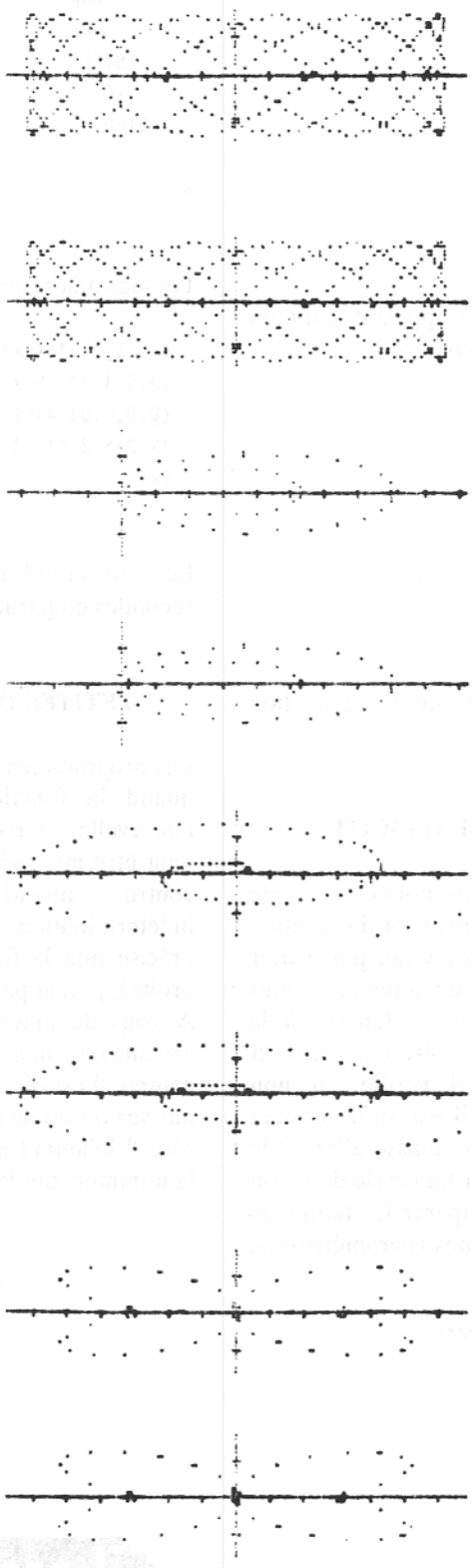
```
« (-3.3,-1.1) PMIN
  (3.3,1.1) PMAX
  (0.0,0.0) AXES
  'X^2+Y^2-1' -1.0 1.0 -1.0 1.0
  FXY
»
```

Le tout prend une dizaine de minutes, contre 30 secondes en paramétriques !

PETITES DIFFICULTES RESIDUELLES

Les programmes vus jusqu'ici marchent sans difficulté quand la fonction est définie en tout point de l'intervalle ; lorsqu'il y a division par zéro, l'erreur peut être masquée sans difficulté (flag 59 armé) ; par contre, quand se présente une pseudo indétermination de la forme zéro sur zéro (étant précisé que la fonction admet une limite gauche et droite), je n'ai pas encore trouvé de truc qui marche ! A vous de jouer ; prenez par exemple la fonction $y = \sin(x)/x$, une bonne et brave fonction à tracer comme il est dit dans le manuel HP ; eh bien, n'est-ce pas vexant de ne pas pouvoir la tracer au voisinage de zéro ? D'autant que la Casio FX-7000G, pour ne pas la nommer, sait le faire, elle !

Eric Gengoux (108)



HP41

M. Markov

GESTION DE DISQUES
(ACTE II)

Le lecteur de cassettes HP-82161 a été un réel bienfait pour les utilisateurs de HP-41. Il fournit une grande capacité de stockage et un accès relativement rapide, surtout si on le compare au lecteur de cartes ou au crayon optique. Il élimine également la ponction de courant que ces appareils exercent sur les batteries du calculateur.

Le nouveau disque HP-IL offre un meilleur temps d'accès, principalement parce que les opérations prennent peu ou pas de temps. Par exemple, le temps maximum d'accès aux données n'est que de 1,742 seconde sur le HP-9114A.

Les lecteurs de disques ont un inconvénient majeur : leur consommation électrique est beaucoup plus importante. Vous pouvez décharger les batteries du lecteur de disquettes en 40 minutes de lecture ou d'écriture continue. Le chargeur n'est pas d'un très grand secours : ses 3 watts ne sont que la moitié de la puissance des batteries. De plus, les pointes de consommation se situent très au delà de ces chiffres. Il faut environ 5 heures pour recharger les batteries à 80 % de leur capacité maximum.

Si vous avez stocké de nombreux fichiers sur un support (disque ou cassette), vous vous apercevrez que l'accès aux fichiers situés vers la fin du catalogue est long. Les problèmes de consommation électrique deviennent prépondérants et un nouveau problème apparaît : l'usure du support dans la partie contenant le catalogue. Ce problème est particulièrement sensible pour les utilisateurs de HP-41 car leurs fichiers sont généralement petits. Ceci conduit, pour maximiser l'occupation du support, à stocker de nombreux fichiers et donc à avoir des catalogues très longs. Comme le HP-41 est plus lent que les autres contrôleurs, ceci ne fait qu'augmenter le temps de travail du lecteur et donc les problèmes de consommation électrique.

Les utilitaires DIRL, DIRLX, GDIRX et SRCHX ont été développés pour remédier à ces problèmes.

GDIRX (Get DIRectory by X) vous permet d'éviter le balayage du catalogue qui prend plus de temps et consomme plus de courant que la copie proprement dite du fichier en mémoire. Vous devez cependant fournir le numéro du fichier dans le registre X. Si vous n'entrez pas le bon numéro, ce n'est pas grave : les fonctions standard se chargeront d'effectuer la recherche dans le catalogue.

DIRL et DIRLX sont utilisés pour déterminer le numéro de fichier dont vous avez besoin pour utiliser GDIRX. DIRL est similaire à DIR dans la mesure où il commence au début du catalogue. Il numérote les fichiers, y compris les fichiers purgés s'il y en a, et en imprime le nom suivi de trois nombres : le type de fichier, la piste et le secteur de début. Cette liste doit être utilisée comme un complément de DIR. DIRLX fonctionne exactement comme DIRL. La principale différence est que vous spécifiez dans X un numéro de fichier. La liste commencera avec le premier fichier du secteur du catalogue contenant l'entrée que vous avez demandée.

SRCHX est une variante simple de GDIRX qui vous permet de faire une recherche dans le catalogue en commencant au fichier dont vous donnez le numéro dans X. Il vous permet d'utiliser la liste produite par un DIR normal et ignore les fichiers purgés. La recherche dans le catalogue est limitée à 3 secteurs de catalogue en commencant au secteur contenant le fichier donné en X. Ceci est fait pour limiter le temps pris par une recherche, en langage utilisateur, dans le catalogue.

MODE D'EMPLOI

Un module *Extended I/O* est nécessaire, ainsi qu'un module HP-IL et une mémoire de masse.

GDIRX

Si vous souhaitez éviter la recherche dans le catalogue pour, par exemple, le fichier 223, mettez ce nombre dans le registre X, et tapez XEQ "DIRX". Il vous suffit ensuite d'utiliser les fonctions ordinaires (GETAS, READP, READSUB, READK, READS, READA, SEEKR, READR, ZERO et WRTR) comme elles sont décrites dans le manuel. READRX et READR fonctionnent également très bien : rappelez-vous simplement que le support doit être préalablement positionné en utilisant l'ordre SEEKR.

GDIRX utilise la pile et le registre ALPHA. Il restaure le registre M, si bien qu'un nom de fichier (7 caractères au plus) peut être stocké en ALPHA avant d'utiliser GDIRX : c'est une option et non une obligation. Si, en utilisant GETAS, vous voulez spécifier un nom de fichier en mémoire étendue différent du nom sur le support magnétique, vous devez rentrer les noms de fichiers après avoir exécuté GDIRX.

SRCHX

SRCHX utilise également un numéro de fichier dans le registre X. Cependant, avec SRCHX vous devez avoir également le nom du fichier sur le support dans le registre ALPHA et rien d'autre. Le nom du fichier est

utilisé lors de la recherche dans le catalogue. Les noms de fichiers peuvent avoir jusqu'à 7 caractères. Le nom du fichier destination et la virgule de séparation ne peuvent être ajoutés qu'après avoir exécuté SRCHX.

GDIRX et SRCHX peuvent être utilisés comme sous-programmes. Vous devrez faire particulièrement attention en entrant les nom et numéro de fichiers : une erreur provoquera une longue recherche dans le catalogue pour aboutir finalement au message NOT FOUND. Voici quelques exemples d'appel :

```
223 XEQ "GDIRX" "FILENAME" nn SEEKR
"FILEAME" ...  125 XEQ "GDIRX" READP
"FILENAME" 198 XEQ "SRCHX" READS
```

DIRL

DIRL est utilisé exactement comme DIR. Toute imprimante ou interface vidéo HP-IL fonctionnera. Assurez vous que vous êtes bien en modes AUTOIO et ADRON. Evitez le mode trace.

DIRLX

DIRLX est utilisé exactement comme DIR. Vous devez fournir un numéro de fichier en X qui détermine le début du listage.

DIRL et DIRLX utilisent l'instruction FMT. Cette fonction facilite le formattage en colonnes quand on utilise une imprimante HP-82162A. D'autres imprimantes peuvent répondre à FMT d'une manière différente : ils accumulent deux octets de formattage. Avec l'interface vidéo 80 colonnes ou l'imprimante HP-82905 ceci fonctionne bien et économise des octets. Avec d'autres appareils vous pouvez avoir à remplacer l'instruction FMT par " " ACA ou 32 ACCHR ACCHR.

UN PEU DE THEORIE

L'interface HP-IL utilise le buffer numéro 1 du lecteur comme zone de stockage du catalogue. C'est à dire qu'il contient une copie du secteur de catalogue le plus récemment utilisé, à moins que le lecteur n'ait été remis à zéro. La plupart des fonctions d'accès au support, mais pas celles qui créent une nouvelle entrée dans le catalogue, examinent d'abord le contenu du buffer 1 avant de se lancer éventuellement dans un balayage complet du catalogue. Si l'entrée souhaitée se trouve dans le buffer, il n'y aura pas de recherche longue et coûteuse.

GDIRX et SRCHX accèdent au secteur du catalogue qui devrait contenir l'entrée de fichier dont vous avez spécifié le numéro en X. SRCHX s'assure que le nom de fichier que vous avez donné se trouve bien dans ce secteur. Quant à GDIRX, il copie ce secteur dans le buffer 1 de telle façon que les fonctions assembleur standard puissent l'utiliser.

En modifiant le contenu du secteur 1, vous pouvez faire croire aux fonctions du module HP-IL qu'un fichier sécurisé ne l'est pas, ou qu'un fichier WRTA est en fait un une entrée de catalogue pour récuperer des fichier après que le catalogue ait été endommagé.

ANALYSE DETAILLEE DES PROGRAMMES

Lignes synthétiques

Dans GDIRX :

```
09:  RCL M:  144, 117
10:  "D":  242, 68, 0.
25:  STO M:  145, 117.
```

Dans SRCHX :

```
09:  Ajoute 7 espace : n'est pas synthétique.
13:  "D":  242, 68, 0.
29:  X<>O:  206, 119.
```

Dans DIRL et DIRLX :

```
11:  242, 68, 0.
34:  " ":  241, 32 Non synthétique : un espace.
```

DIRX

Les lignes 2 à 5 calculent l'adresse du secteur de catalogue que vous voulez. Les lignes 6 à 8 sélectionnent l'unité de stockage comme appareil principal. La ligne 9 sauve le nom du fichier, s'il existe, dans ALPHA pour usage ultérieur. Les lignes 10 à 16 positionnent le lecteur sur l'entrée demandée et la ligne 19 copie le contenu de ce secteur dans le buffer 0 du lecteur. La boucle au LBL 09 est destinée à attendre que le lecteur se positionne : cette attente est indispensable avec un lecteur de cassettes. Les lignes 22 et 23 copient le buffer 0 dans le buffer 1 et enfin les lignes 24 et 25 restituent le nom de fichier dans ALPHA. Le CLST est facultatif.

SRCHX

Les lignes 2 à 8 sont identiques à celles de GDIRX. Les lignes 9 à 11 complètent le nom de fichier à 10 caractères par des blancs. Les lignes 13 à 24 font le même travail que les lignes 10 à 21 de GDIRX. A ce moment, puisque nous risquons d'avoir à lire plusieurs secteurs avant de trouver la bonne entrée, il est intéressant de rester en mode DDT 2. Ceci signifie que nous devons tester le nom de fichier avant de lire la totalité des informations de cette entrée, sinon nous pourrions nous retrouver en train de lire le secteur suivant.

La boucle au LBL 00 utilise le 16 (ligne 6) qui a été stocké dans LASTX par FINDAID comme compteur des fichiers trouvés dans les deux premiers secteurs à explorer. Si rien n'est trouvé dans ces deux secteurs, le lecteur est laissé positionné sur le secteur suivant. Les lignes de programme synthétiques (lignes 12, 29 et 39) permettent d'utiliser des noms de fichiers sur 7 caractères. Notez que chaque entrée est lue en deux morceaux : d'abord 21 caractères, commençant au nom du fichier puis les 11 restants. Vous pouvez travailler avec 14 et 18 caractères si vous changez la ligne 29 par X<>N.

Une fois la recherche terminée, le nom de fichier est recopié dans le registre ALPHA et le buffer 0 est copié dans le buffer 1 (lignes 37 à 41).

Que se passe-t-il si le nom de fichier ou le numéro introduits sont incorrects ? Le pire qu'il puisse arriver est que vous passiez par la recherche standard, en assembleur, du module. Dans ce cas, ou bien vous trouverez le fichier, ou bien vous obtiendrez le même message que si vous n'étiez pas passé par les utilitaires.

DIRLX et **DIRLX**

Le code utilisé dans ces deux sous-programmes est très semblable à celui de SRCHX. La même procédure d'accès à un secteur du catalogue se retrouve aux lignes 2 à 24. La boucle au LBL 00 est toujours la boucle de lecture du catalogue. La principale différence est que vous devez avoir un pointeur de fichier. Celui-ci est créé par les lignes 6 à 8 et 27 à 30. La boucle du LBL 00 accumule les données contenues dans chaque entrée de manière à faciliter la production d'une liste :
- lignes 32 à 35 : numéro du fichier et espace,
- lignes 36 à 39 : nom de fichier,
- lignes 40 à 42 : les deux octets du type de fichier,
- lignes 43 et 53 : ajoutent deux espaces ou marques de format,
- lignes 44 à 48 : test de la fin du catalogue et fin si terminé,
- lignes 49 à 55 : secteur de début du fichier. Cette information n'est pas utilisée par GDIRX ou SRCHX, elle est utile pour des applications spécialisées.
- lignes 56 à 61 : impression des données que vous avez collectées, déplacement jusqu'à l'entrée suivante et retour au LBL 00 pour son traitement.

REMARQUES GENERALES

Ces programmes ont été conçus principalement pour fonctionner avec un lecteur de disquettes HP-9114. Cependant, il s'est avéré qu'ils étaient utiles avec la cassette. Le programme DIRLX donne aux utilisateurs de HP-41 les possibilités du programme DIRALL qui utilise un HP-75 et une imprimante 80 colonnes pour vous dire tout sur vos fichiers HP-41, HP-75 ou HP-71. Par ailleurs, avec un listing produit par DIR ou DIRL, vous pouvez recréer sans trop de difficultés vos entrées de fichier.

Michael Markov (301)

Programmes DIRL et DIRLX :

```
01*LBL "DIRL"
0
03*LBL "DIRLX"
8 / INT ENTER^ ENTER^ 2 + "D" XTOAR 16
FINDAID SELECT 4 DEVL SQRT OUTAN
20*LBL 09
SF 25 DEVT FC?C 25 GTO 09 CF 29 FIX 0 R^ 8
* SIGN
31*LBL 00
LASTX ACX " " ACA 10 INAN ATOX ACA INXB
INXB ACX FMT 255 X=Y? CLRDEV X=Y? RTN INXB
INXB INXB ACX FMT INXB ACX PRBUF 16 INAN
ISG L "" GTO 00 END
```

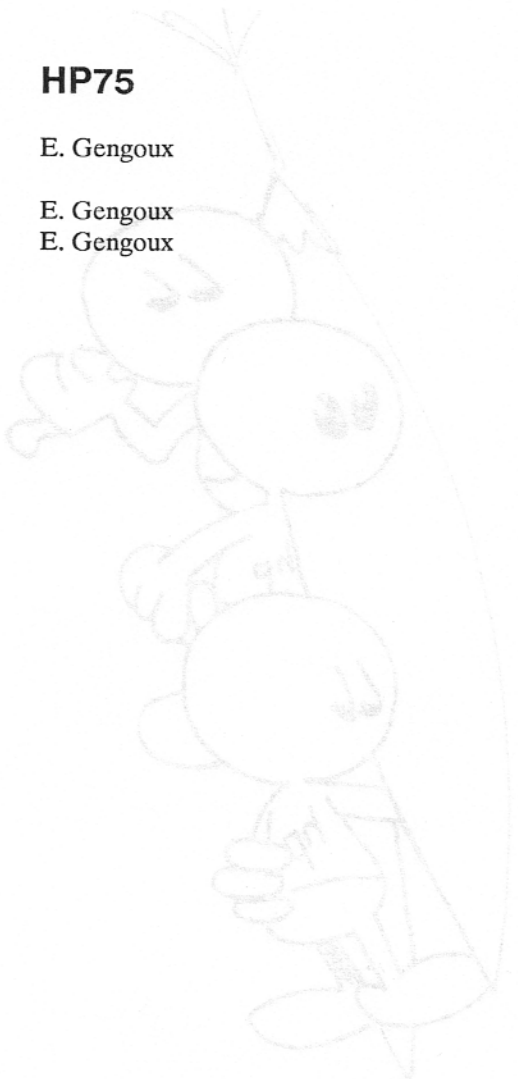
Programme GDIRX :

```
01*LBL "GDIRX"
8 / 2 + 16 FINDAID SELECT RCL M "D"
RCL Z XTOA 4 DEVL SQRT OUTAN
17*LBL 09
SF 25 DEVT FC?C 25 GTO 09 5 DEVT R^ STO M
CLST END
```

Programme SRCHX :

```
01*LBL "SRCHX"
8 / 2 + 16 FINDAID SELECT "⊢      " 7
AROT RCL M "D" R^ XTOAR 4 DEVL 2 OUTAN
20*LBL 09
SF 25 DEVT FC?C 25 GTO 09 R^
26*LBL 00
21 INAN X<> 0 X=Y? GTO 01 11 INAN X<> Z
DSE L GTO 00
37*LBL 01
CLA STO M 5 DEVT END
```

HP75



L'AUTRE COIN DES LHEX

Tout d'abord, voici le truc pour créer, sans risque d'erreurs, un fichier Lex directement dans votre 75. C'est presque aussi facile qu'avec le 71, grâce au programme LEXPOKE joint. Il faut seulement disposer de la ROM I/O et du Lex PEEKPOKE (celui diffusé par la Bibliothèque Utilisateurs de Corvallis, mais nous pouvons vous le copier - joindre une carte magnétique).

Saisissez votre fichier sous forme de texte, en numérotant les lignes de un en un (AUTO 1,1) et en respectant la disposition indiquée (numéro de ligne, espace, chiffres hexadécimaux, espace, checksum). Une fois saisi, renommez le fichier, si possible en utilisant le nom du Lex suivi de la lettre T (par exemple : TOTOLEXT), car le programme supprimera ce T après avoir converti le texte en Lex.

Le programme vérifie les saisies et, s'il constate une erreur, indique le numéro de la ligne et vous replace sous l'éditeur de texte (tiens, comme chez Borland !). Vous n'avez plus qu'à faire un FETCH xx, corriger la ligne, et taper CONT [RTN] pour poursuivre le travail... Commode, non ?

Faisons donc RUN"LEXPOKE". Le premier menu demande le nom du fichier texte, puis indique "Vérif. CHKSMs en cours". S'il découvre des anomalies ou si vous n'avez pas mis du tout de checksums, il demande "Présence CHKSMs O/N". Bien entendu, vous pouvez *shunter* ce contrôle. Ceci est utile, par exemple, pour rentrer un Lex ancien qui n'utilise pas ce système : c'est le cas de ceux publiés il y a longtemps dans *PPC-CJ* ou *PPCT*. C'est assez risqué quand-même ! Si donc vous répondez O, le programme s'arrêtera à chaque erreur rencontrée, comme indiqué ci-dessus. Il fera de même à chaque caractère non-hexadécimal (lettre O à la place du chiffre 0, ou lettre l à la place du chiffre 1...).

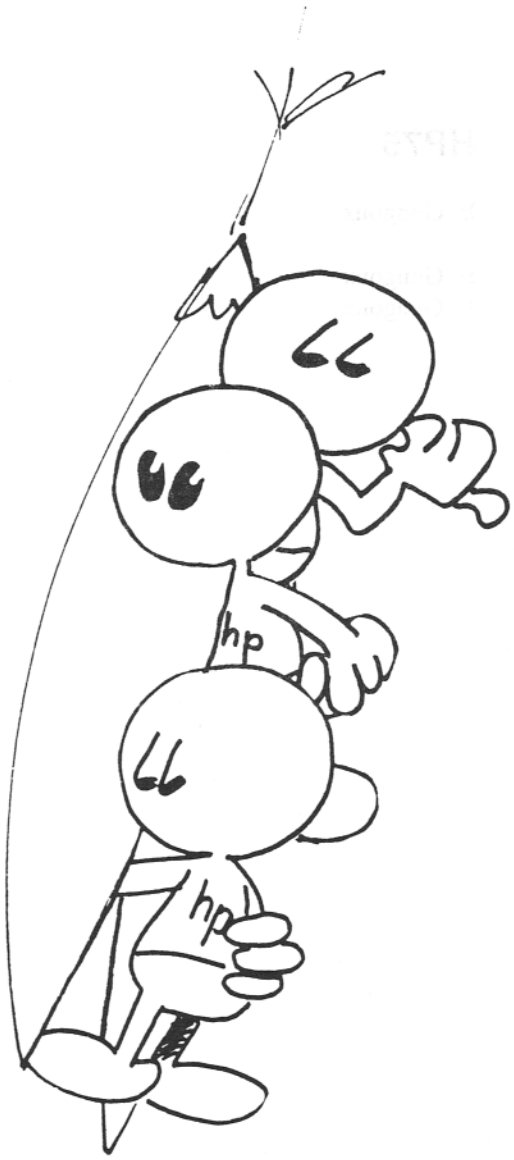
Une fois le fichier dûment vérifié, le programme affiche le message "POKE en cours", puis "Terminé". Faites alors un CAT ALL pour vérifier (dernière précaution!) que le CAT du Lex a pour taille le nombre d'octets indiqué en tête moins 18 (ce sont ceux de la directory interne). Si oui, vous pouvez passer à la dernière formalité, qui va activer votre LEX : il s'agit de positionner ses *access bits* et la ROM I/O intervient de nouveau ici, ou plutôt l'un de ses trésors cachés. Tapez :

```
SET ACCESS "<nom du LEX>","RxelPCnT"
```

en respectant bien la position des lettres et les majuscules sinon, plantage spectaculaire garanti ! C'est tout !

Ah, si ! Pour faire bonne mesure, nous avons inclus le programme LEXDUMP qui a servi à faire les exemples. Comme cela, si vous « commettez » un jour un Lex, vous pourrez à votre tour le diffuser !

Eric Gengoux (108) et HPCC



FORTH

ASSEMBLEUR

BASIC

LE COIN DES LHEX

TOURS DE FORTH

Le module *Translator Pac* a un grave défaut : si vous créez des mots Forth dans le vocabulaire HP41V, il ne vous est pas possible d'utiliser tout le vocabulaire standard. En effet, beaucoup de mots seront interprétés comme des mots HP41 et ce n'est pas forcément ce que vous souhaitez...

Par exemple, essayez de créer le mot suivant dans ce vocabulaire

```
: BOUCLE 1000 0 DO I . LOOP ;
```

Les bornes de la boucle seront déposées sur la pile flottante et non sur la *Data Stack* ; de même, le mot I mettra sur la pile flottante le contenu de la variable Basic I.

Il faut donc trouver un moyen de contourner ce problème et ce moyen, le voici :

- Structurez vos écritures afin de découpler les mots HP41 des mots Forth dans la mesure du possible.

- Ecrivez vos définitions Forth dans ce vocabulaire.

- Déposez le CFA des mots que vous venez de définir sur la pile, puis entrez dans le vocabulaire qui vous intéresse; ce peut être le vocabulaire HP41V par exemple, ou bien votre propre vocabulaire.

- Compilez ces adresses (CFA) dans des LITERAL.

- Récrivez les mots précédents en déposant le CFA correspondant sur la pile et en l'exécutant; le tour est joué. Vous pouvez même prévoir un comportement différent selon le vocabulaire dans lequel vous opérez.

Bien entendu, ce principe n'est applicable que pour les mots que vous pouvez définir complètement dans le vocabulaire FORTH.

A titre d'exemple, voici un mot pour le vocabulaire HP41V qui extrait du registre ALPHA une sous-chaîne représentée par un double pointeur dans le registre X de la pile flottante. Dans l'environnement Forth, ce mot dépose simplement sur la pile l'adresse et le nombre de caractères de la sous-chaîne (format TYPE). Dans l'environnement HP41V, la sous-chaîne remplace la chaîne initiale dans le registre ALPHA.

```
( Mots définis simultanément dans 2 vocabulaires  )

HEX

: DecodeX ( décodage du registre X )
  2FBDF 2FBDA DO I N@ LOOP
  2FBD0 4N@
  CASE
    0 OF 5 ROLL DROP 0 ENDOF
    1 OF ENDOF -1 CHIRP ABORT" FTH ERR:Data Error"
  ENDCASE
  A * +
  2SWAP 4 ROLL
  A * + A * + ;

( Mot A[ défini dans l'environnement FORTH        )

: A[ 2FC84 2FC82 C@ DecodeX SUB$ ;

' A[ ( dépose le CFA sur la pile )

HP41V DEFINITIONS

( Dans le vocabulaire HP41V, on récupère le CFA   )
( sous la forme d'un litéral; remarquez que cette )
( valeur se trouve déjà sur la pile; il faut donc )
( la dupliquer en suspendant la compilation sans  )
( oublier de la détruire par la suite. En effet,  )
( il doit y avoir le même nombre d'objets sur la  )
( pile avant et après la compilation d'un mot     )
( compilant lui-même un litéral.                  )

: addr(A[) [ DUP ] LITERAL ; DROP

( Mot A[ défini dans l'environnement HP41V        )

: A[ addr(A[) EXECUTE

( on recopie la chaîne dans le registre ALPHA     )

DUP 2FC82 C! 2FC84 OVER S! ;
DECIMAL
FORTH FORTH DEFINITIONS
```

Une autre application de ce principe peut être le comportement après erreur d'un logiciel opérant dans un environnement particulier.

Si vous ne prévoyez rien, chaque erreur vous ramène dans l'environnement FORTH, ce qui peut être génant ou pour le moins peu pratique.

Commencez par écrire votre vocabulaire : par exemple, le logiciel de gestion d'une pile complexe (JPC 40 du même auteur). Revenez dans l'environnement FORTH et créez deux mots : l'un constitue votre routine d'erreur et se termine a priori par QUIT (cela permet de reprendre en douceur le contrôle du calculateur sans réinitialiser l'environnement) ; l'autre mot sert à initialiser l'environnement de votre logiciel et place le CFA du mot précédent dans la variable ONERR.

On a donc, par exemple :

```
: ERREUR ." Erreur" QUIT ;
: VAZY Vocabulaire DEFINITIONS ['] ERREUR ONERR ! ;
```

Supposons maintenant que vous vouliez que votre logiciel reprenne la main après l'affichage du message d'erreur tout en permettant à l'utilisateur de voir l'intitulé plus précis de cette erreur.

Avant de revenir à l'environnement FORTH pour construire le mot d'initialisation, déposez le CFA du ou des mots qui permettront à votre logiciel de reprendre la main :

```
' REVIENZY
```

Puis, dans l'environnement FORTH :

```
: ON-Y-VA [ DUP ] LITERAL ; DROP
: ERREUR ." Erreur" KEY DROP ON-Y-VA EXECUTE ;
```

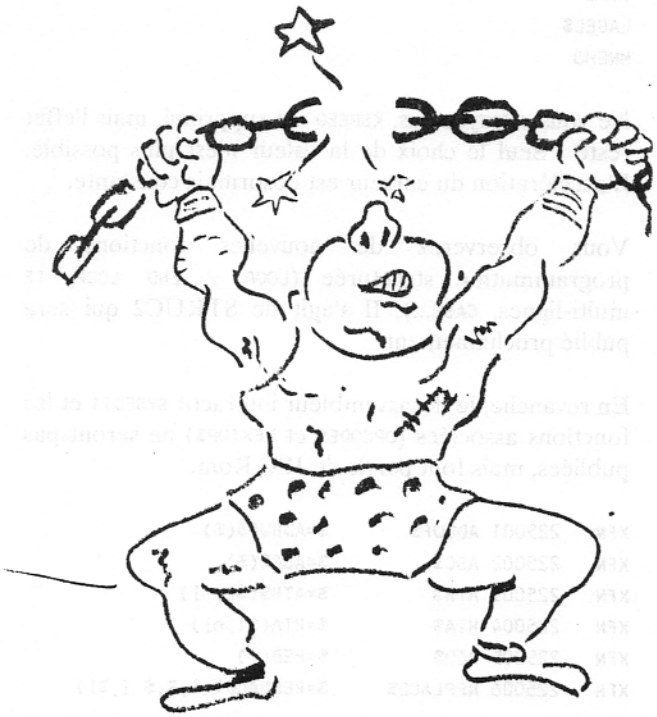
Au KEY, appuyez sur [g][ERRM] pour connaître l'intitulé de l'erreur ; l'appui sur toute autre touche rend la main à la fonction REVIENZY de votre logiciel, et vous restez dans cet environnement.

Enfin, n'oubliez pas de remettre ONERR à 0 lorsque vous quittez l'environnement de votre logiciel.

Il y a sûrement d'autres applications possibles de ce principe. Ce qui prouve bien que Forth a plus d'un tour dans son sac et recèle des merveilles qui me font oublier les SIGouillages rébarbatifs.

Pour terminer, je lance un appel pour qu'on écrive un mot Forth (et uniquement en Forth) qui permettra de stocker des codes machine dans le PFA d'un autre mot sans passer par un assembleur bien entendu. Autrement dit, faire pour les mots Forth ce qui existe déjà pour les Lex grâce au programme MLLEX. Alors, à vos méninges !

Alain Goubault de Brugière (308)



RECAPITULATION DES TOKENS

Comme chaque année maintenant, nous publions la liste complète des tokens attribués dans JPC Rom (ID #E1).

Certains changements sont intervenus au cours du processus d'homologation des tokens chez HP, ainsi que lors de la phase de remaniement de certaines fonctions.

Certains mots-clefs ont été renommés, ainsi :

FILE?	devient	FILESIZE
HMS+	devient	HMSADD
HMS-	devient	HMSSUB
CR	devient	PCR
FF	devient	PFF
LF	devient	PLF
PL	devient	PAGELEN
FPRM	devient	FPRIM
NPRM	devient	NPRIM
BLIST	devient	DBLIST

DISABLE et ENABLE vont être modifiés et rebaptisés vraisemblablement LEX ON/OFF.

D'autre part, certains mots-clefs ont été supprimés :

INV$
KSPEED
TYPE
LABEL$
MNEMO

Ne vous effrayez pas, KSPEED est supprimé, mais l'effet reste ! Seul le choix de la valeur n'est plus possible. L'accélération du curseur est désormais constante.

Vous observerez de nouvelles fonctions de programmation structurée (LOOP / END LOOP, IF multi-lignes, CASE...). Il s'agit de STRUC2 qui sera publié prochainement.

En revanche, le désassembleur interactif SYSEDIT et les fonctions associées (OPCODE$ et NEXTOP$) ne seront pas publiées, mais font partie de JPC Rom.

```
XFN   225001 ADBUF$       $=ADBUF$($)
XFN   225002 ASC$         $=ASC$($)
XFN   225003 ATH$         $=ATH$($[,n])
XFN   225004 HTA$         $=HTA($[,n])
XFN   225005 RED$         $=RED($)
XFN   225006 REPLACE$     $=REPLACE$($,$,$ [,$])
XFN   225007 FILESIZE     n=FILESIZE($)

XWORD 225008 ATTN         ATTN ON|OFF

XWORD 225009 DISABLE      DISABLE $
XWORD 225010 ENABLE       ENABLE $

XWORD 225011 FKEY         FKEY $

XFN   225012 CONTRAST     n=CONTRAST()
XWORD 225013 INVERSE      INVERSE
XFN   225014 INV$         $=INV$($)
XFN   225015 PAINT        n=PAINT(n,n,n)

XFN   225016 ENDUP$       $=ENDUP$()
XWORD 225017 ENDUP        ENDUP $
XFN   225018 STARTUP$     $=STARTUP$()
XWORD 225019 EXECUTE      EXECUTE $

XFN   225020 ARR          n=ARR(n,n)
XFN   225021 COMB         n=COMB(n,n)

XFN   225022 HMSADD       n=HMSADD(n,n)
XFN   225023 HMSSUB       n=HMSSUB(n,n)
XFN   225024 HMS          n=HMS(n)
XFN   225025 HR           n=HR(n)

XWORD 225026 EDIT         EDIT $ [TO $]

XWORD 225027 STACK        STACK n

XWORD 225029 MARGIN       MARGIN n

XFN   225030 NEXTOP$      NEXTOP$($)
XFN   225031 OPCODE$      OPCODE$($)
XWORD 225032 SYSEDIT      SYSEDIT $

XFN   225033 MENU         n=MENU(n [,n])

XFN   225034 CENTER$      $=CENTER$($,n)
XFN   225035 CESURE       n=CESURE($,n)
XFN   225036 FORMAT$      $=FORMAT$($,n)
XFN   225037 REDUCE$      $=REDUCE$($)
XFN   225038 SPACE$       $=SPACE$(n)

XWORD 225039 BELL         BELL
XWORD 225040 BOLD         BOLD ON|OFF
XWORD 225041 PCR          CR
XFN   225042 ESC$         $=ESC$([$])
XWORD 225043 PFF          FF
XWORD 225044 PLF          LF [n]
XWORD 225045 MODE         MODE n
XWORD 225046 PERF         PERF ON|OFF
XWORD 225047 PAGELEN      PAGELEN [n [,n]]
XWORD 225048 UNDERLINE    UNDERLINE ON|OFF
XWORD 225049 WRAP         WRAP ON|OFF
```

```
XFN    225050 DATESTR$     $=DATESTR$($|n)
XFN    225051 DATEADD      n=DATEADD($|n,n)
XFN    225052 DDAYS        n=DDAYS($|n,$|n)
XFN    225053 DMY          DMY
XFN    225054 DOW$         $=DOW$($|n)
XFN    225055 DOW          n=DOW($|n)
XFN    225056 MDY          MDY

XFN    225057 MAXD         MAXD(n|$)
XFN    225058 MAXM         MAXM(n|$)
XFN    225059 MEMD         MEMD(n|$)
XFN    225060 MEMM         MEMM(n|$)

XWORD  225061 EXIT         EXIT var

XFN    225062 NLOOP        n=NLOOP([n])
XFN    225063 PPOLL        n=PPOLL([n])
XWORD  225064 SLEEP        SLEEP
XFN    225065 SRQ          n=SRQ([n])


XWORD  225066 END          END WHILE/LOOP/SELECT/IF
XWORD  225067 WHILE        WHILE n
XWORD  225068 REPEAT       REPEAT
XWORD  225069 UNTIL        UNTIL n
XWORD  225070 LEAVE        LEAVE

XWORD  225071 SWAP         SWAP var,var

XFN    225072 ENTRY$       $=ENTRY$($ [,n])
XFN    225073 TOKEN        n=TOKEN($ [,n])

XWORD  225075 FIND         FIND $

XFN    225076 MAP$         MAP$($,$,$)
XWORD  225077 MAP          MAP $|#n,$,$[,n[,n]]

XWORD  225078 GLINE        GLINE n,n,n,n,n
XWORD  225079 GPSET        GPSET n

XFN    225080 RPLC$        $=RPLC$($,$,$)

XWORD  225081 SHRINK       SHRINK $

XFN    225082 FPRIM        n=FPRIM(n [,n])
XFN    225083 NPRIM        n=NPRIM(n,n)
XFN    225084 PGCD         n=PGCD(n,n...n)
XFN    225085 PHI          n=PHI(n)
XFN    225086 PPCM         n=PPCM(n,n...n)
XFN    225087 PRIM         n=PRIM(n [,n])

XFN    225088 FRAC$        $=FRAC$(n [,n])

XFN    225089 POSI         n=POSI($,n [,n])

XWORD  225090 BLIST        BLIST [$ [,n [,n]]]
XWORD  225091 PBLIST       PBLIST [$ [,n [,n]]]
XWORD  225092 RENUMREM     RENUMREM [n[,n[,n[,n]]]]
XWORD  225093 FINPUT       FINPUT $,$[,$],var

XFN    225094 RREC$        $=RREC$($|n,$|n)
XWORD  225095 WREC         WREC $,n|$,$|n

XWORD  225096 LOOP         LOOP
XWORD  225097 SELECT       SELECT n|$
XWORD  225098 CASE         CASE n|$ [TO n|$] / ELSE
XWORD  225099 IF           IF n THEN
XWORD  225100 ELSE         ELSE
```

CHRONOMETRE

Le HP-71 est une machine dotée d'une horloge interne, d'où un bon nombre d'applications.

Le Lex que je vous propose utilise ces capacités : il s'agit d'une fonction chronomètre.

Un chronomètre est tout simplement un compteur que l'on peut arrêter. La qualité d'un chronomètre est fonction de sa précision (plus petite mesure du temps mesurable). Ici, l'objectif atteint est le centième de seconde.

Dès lors, des problèmes se posent. Il faut que l'affichage et le décodage du clavier soient suffisamment rapides. De ce fait, des routines comme KEYSCN ou DSPCHA ne peuvent être utilisées : elles sont beaucoup trop lentes. Il a fallu donc improviser.

Pour le clavier, on n'avait pas besoin du décodage (l'ordinateur n'a pas besoin de savoir quelle touche a été pressée). Aussi, l'instruction C=IN suffit. Pour l'affichage, on écrit directement les caractères ASCII dans le buffer d'affichage, puis on modifie les drapeaux de la routine d'affichage (écran non correct, enlever le curseur, etc). Le programme fait ensuite appel à la routine BLDDSP et le tour est joué.

Le problème suivant était la sortie du résultat. Faut-il sortir un nombre en secondes, une chaîne au format *hh:mm:ss.cc* ou encore un nombre au format *hh.mmsscc* ?

J'ai choisi les deux premières solutions, deux fonctions ont donc été créées : CHRONO et CHRONO$.

CHRONO renvoie le nombre en secondes, alors que CHRONO$ renvoie une chaîne représentant le temps écoulé sous le format *hh:mm:ss.cc*.

Lionel Guillou (326)

```
        LEX     'CHRONOLX'
        ID      #5E       ID inutilisé
        MSG     0
        POLL    0
FLREG   EQU     #2F6F8    Adr drapeaux utilisateurs
BSPCL?  EQU     #020B6    Effacer l'écran
CMPT    EQU     #125B2    Heure actuelle en 1/512s
*                         depuis le 1 jan 0000
AFF1    EQU     #2F494    Adr dans le buffer du
*                         début de 00:00:00.00
DSP     EQU     #2F478    Drapeaux d'affichage
KEYPTR  EQU     #2F443    Pointeur buffer de touches
SLEEP   EQU     #006C2    Met la machine en attente
SETALR  EQU     #12917    Régler une alarme
DELAYT  EQU     #2F948    Où est stockée la valeur
*                         de delay
ALMSRV  EQU     #1257D    Actualisation des alarmes
ATNFLG  EQU     #2F442    Compteur de touches
CKSREQ  EQU     #00721    Controleur des timers
RCD0D1  EQU     #12A83    Rappel D0&D1 situés en R4
STD0D1  EQU     #12A5C    Sauvegarde D0&D1 dans R4
BLDDSP  EQU     #01898    Construit l'affichage
*                         d'après le buffer
FLOAT   EQU     #1B322    Conv. déc ent./12digFloat
STRHDR  EQU     #0F09A
TRUNCC  EQU     #12B4A    Conv 15dig/12dig
EXPR    EQU     #0F23C    Pour sortir de chrono$
DEBAFF  EQU     #2F480    Add. debut buffer d'aff.
CLRFRC  EQU     #0C6F4    Efface la partie frac.
DV2-12  EQU     #0C4A8    Division de 2 nb
FNRTN1  EQU     #0F216    Pour sortir de chrono
HXDCW   EQU     #0ECB4    Convs. hex/déc
IDIV    EQU     #0EC7B    Divi. de 2 nb entiers
        ENTRY   T1
        CHAR    #F
        ENTRY   T2
        CHAR    #F
        KEY     'CHRONO$'
        TOKEN   100
        KEY     'CHRONO'
        TOKEN   101
        ENDTXT
* Routine prise dans TIME
TIME    GOSBVL  CMPT      Lit l'heure en 1/512s
        A=C     W
        C=0     W
        P=      4
        LCHEX   2A3       C=24 Heures en 1/512s
        GOSBVL  IDIV      Divise l'heure depuis 0
        GOSBVL  HXDCW     par 24H & conv en déc.
        A=C     W
        GOSBVL  FLOAT     Transf. le nb en virgule
*                          flottante(12digit)
        C=0     W
        P=      12
        LCHEX   512       C=5.12
        GOSBVL  DV2-12    Divise l'heure par 5.12
        GOSBVL  CLRFRC    Supprime la partie frac.
        GOSBVL  TRUNCC    Convs. 15dig/12dig
        P=C     0         Exposant du nb
        CSR     M         Origine n=.ccccccc.......
        CSR     M
        CSR     M
        CSR     M
        CSR     M
        CSR     W
        CSR     W         Maint. n=........ccccccc.
        P=P-1             Retranche à p 7 ou
        P=P-1             ajoute 9 à p
        P=P-1             p représente l'exposant:E
        P=P-1
        P=P-1
        P=P-1
        P=P-1
LOU     CSR     W         Décale C pour éliminer
        P=P+1             les 0 si E<5
        GONC    LOU
        RTN               C=Heure en 1/100s(entier)
        NIBHEX  00
T2      GOSBVL  STD0D1    Sauvegarde de D0&D1
        P=      1
        C=P     2         Mise à 1 du
        D0=(5)  FLREG     drapeau 60
        DAT0=C  XS
        GOTO    OK
        NIBHEX  00
T1      GOSBVL  STD0D1
        C=0     XS
        D0=(5)  FLREG     Mise à 0 du
        DAT0=C  XS        drapeau 60
OK      SETHEX
        ST=1    8         raz du curseur
        ST=1    5         Effacer l'écran
        GOSUB   STOSTA    Sauvegarde des drapeaux
        P=      0
        GOSBVL  BSPCL?    Routine pour effacer
        D0=(5)  DEBAFF
        P=      0
        LCASC   'onorhC '  Pour afficher
        DAT0=C  14
        D0=D0+  14        la présentation
        LCASC   '0:00 : '
        DAT0=C  14
        D0=D0+  14
        LCASC   '00.00:0' Stocke la chaîne dans le
        DAT0=C  14        buffer d'affichage
        D0=D0+  14
        ST=1    6         Enlève le curseur
```

```
        ST=0    1       Ecran non correct donc à
        GOSUB   STOSTA  refaire&construction
        GOSBVL  BLDDSP
        C=0     A
ATEN    C=IN            Touche pressée ?
        ?C=0    A
        GOYES   ATEN
ATT     C=IN            Attend que la touche
        ?C#0    A       soit relachée
        GOYES   ATT
        GOSUB   TIME
        R2=C            Valeur initiale du temps
BOF     GOSUB   TIME    Heure actuelle
        A=C     W
        C=R2            Rappel heure initiale
        A=A-C   W       Temps ecoulé
        R1=A
        D1=(5)  AFF1    Affichage temps
        ASR     W       Eliminer les centièmes
        ASR     W
        C=A     W
        GOSUB   DV60    A=Min,B=Secondes
        D=C     W
        GOSUB   DV60    A=Heures,B=C=Min,D=Sec
        R3=C
        GOSUB   ASCII   Copie heures ds le buffer
        LCASC   ':'
        DAT1=C  B       Copie ':' ds le buffer
        D1=D1+  2
        A=R3
        GOSUB   ASCII   Copie min ds le buffer
        LCASC   ':'
        DAT1=C  B
        D1=D1+  2
        C=D     A
        A=C     A
        GOSUB   ASCII   Copie les sec ds le buffer
        LCASC   '.'
        DAT1=C  B
        D1=D1+  2
        A=R1
        GOSUB   ASCII   Copie les centièmes
        SETHEX
        ST=1    6       Enlève le curseur
        ST=0    1       Ecran non correct
        GOSUB   STOSTA
        GOSBVL  BLDDSP  Afficher
        C=0     A
        C=IN
        ?C#0    A       Touche pressée?
        GOYES   BOUSO   Sort de la boucle
        GOTO    BOF
BOUSO   SETHEX
        A=0     A
        ST=1    5       Efface l'ecran
        ST=1    8
        GOSUB   STOSTA
        P=      0
        GOSBVL  BSPCL?  Effacer
        D0=(5)  KEYPTR
        A=0     XS      Vide le buffer de touches
        DAT0=A  XS
        SETDEC
        A=R1            temps
        R2=A
        D1=(5)  DELAYT
        A=0     A
        A=DAT1  B       Valeur du delay*32
        ASL     A       A=A*16(1/512s)
        ?A=0    A
        GOYES   TEST
        LCHEX   4       Règle l'alarme 5
        GOSBVL  SETALR  avec la valeur dans A
WAIT10  GOSBVL  ALMSRV  Actualisation des alarmes
        ?ST=1   4
        GOYES   TEST    Si le temps est écoulé
        D1=(5)  ATNFLG  Touche pressée?
        C=DAT1  P
        ?C#0    P
        GOYES   TEST
        C=0     A
TESA    C=IN            Touche relachée?
        ?C#0    A
        GOYES   TESA
        GOSBVL  SLEEP   Machine en attente
        GONC    TEST    Sort si une touche a été
*                       pressée lors de l'attente
        GOSBVL  CKSREQ  Controleur de timer
        GOTO    WAIT10
INTER   GOTO    NOMB
* BUT : teste le drapeau 60 pour voir sous quelle
* forme il faut envoyer le résultat
TEST    D0=(5)  FLREG
        C=DAT0  XS
        ?C#0    XS
        GOYES   INTER   Fin fonction chrono
* BUT : construire la chaîne représentant le
* résultat(format=hh:mm:ss.cc) sur la math stack
        GOSBVL  RCD0D1
        C=0     A
        P=      0       Assez de mémoire?
        LCHEX   16      Si oui,crée l'entête de la
        GOSBVL  STRHDR  chaine sur la Math Stack
        SETDEC
        A=R2
        GOSUB   TRA     Ecrit les centièmes sur MS
        A=R2
        ASR     W       Elimine les
        ASR     W       centièmes
        LCASC   '.'
        DAT1=C  B
        D1=D1+  2
        C=A     W
        GOSUB   DV60    A=Minutes,B=Secondes
        D=C     W
        GOSUB   DV60    A=Heures,B=C=Minutes,D=Sec
```

```
        CDEX   A
        R3=A
        A=C    A
        GOSUB  TRA      Ecrit les secondes sur MS
        LCASC  ':'
        DAT1=C B        Ecrit ':' sur MS
        D1=D1+ 2
        CDEX   A
        A=C    B
        GOSUB  TRA      Ecrit les minutes sur MS
        LCASC  ':'
        DAT1=C B
        D1=D1+ 2
        A=R3
        GOSUB  TRA      Ecrit les secondes sur MS
        D1=D1- 16
        D1=D1- 6        Replace D1 au debut
        D1=D1- 16       de la math stack (MS)
        GOVLNG EXPR     Fin de la fonction chrono$
* BUT : envoyer le temps en secondes sur la
* Math Stack
NOMB    A=R2
        GOSBVL FLOAT    Convs. déc/12dig
        C=A    W
        C=C-1  X        Divise le temps par 100
        C=C-1  X        pour faire app. les sec.
        GOSBVL RCD0D1
        GOVLNG FNRTN1
* BUT : enregistrer les drapeaux d'affichage
* puis lancer la routine d'affichage
STOSTA  D0=(5) DSP
        C=ST            Sauvegarde des
        DAT0=C X        status bits
        RTN
* BUT : Divise A par C
DV60    P=     1
        C=0    W        C=60
        LCHEX  6
        GOVLNG IDIV     Divise
* BUT : envoyer à l'espace memoire pointé par D1
* la chaine ASCII représentant les 2 premiers
* chiffres de A(B)
* A la sortie, la chaine est en A
ASCII   P=     0
        LCHEX  30       Chr$(48)="0"
        B=0    W
        B=A    P        Premier chiffre
        B=B+C  B        Convertit en ASCII
        BSL    A        Décale les 2 premiers
        BSL    A        quartets de la
* représentation ASCII du premier chiffre de A
        ASR    A        Elimine le 1er chif. de A
        B=A    P        de même pour
        B=B+C  B        le deuxième
        A=B    A        chiffre
        DAT1=A 4        Ecrit la chaine
        D1=D1+ 4        en mémoire
        RTN
TRA     P=     0        I
        B=0    W        I inversion
        B=A    P        I des
        BSL    B        I 2
        BSL    X        I premiers
        P=     2        I caractères
        A=B    P        I
        ASR    X        I
        GOTO   ASCII

        END
```

OUI OU NON ?
IL FAUT CHOISIR !

Voici ma première proposition d'article pour JPC. Il s'agit d'un petit Lex contenant la fonction YESNO.

Il arrive en effet souvent qu'il faille, au cours d'un programme, répondre à une question par *oui* ou par *non*. Jusqu'à présent, cela se faisait à l'aide de diverses routines Basic, quelquefois en appelant la fonction KEYWAIT$. Tant que cela n'apparaissait qu'une fois dans le Basic, pas de problème. Mais bien vite, il fallut faire un sous-programme faisant cela, sous-programme pouvant être appelé d'un autre sous-programme, ce qui prend du temps.

Je décidais alors de remplacer ce sous-programme par la fonction YESNO.

UTILISATION

Afficher votre question (avec DISP) sans terminer la ligne d'affichage (terminer l'ordre DISP avec ";") et exécuter la fonction YESNO. Cette fonction rajoute à la fin de votre question le message suivant : " ? Y/N". Le point d'interrogation est là pour préciser qu'il s'agit d'une question et les lettres Y/N pour préciser qu'il faut répondre uniquement par la lettre [Y] ou par la lettre [N]. Toute autre touche est sans effet, y compris la touche [ATTN]. Une fois votre choix effectué les lettres Y/N sont remplacées soit par Yes, soit par No selon votre réponse, ceci afin que le HP-71 vous indique qu'il a accepté votre réponse. Tant que l'affichage n'est pas modifié, une trace de votre réponse subsistera ainsi.

Voila pour ce premier Lex. La prochaine fois, j'espère pouvoir vous proposer une routine permettant d'otenir facilement des caractères soulignés à l'affichage.

Jean-Yves Naour (313)

```
       LEX    'YESNOLEX'
       ID     #5C
       MSG    0          Pas de messages
       POLL   0          Pas de pollhandler
*
ATNFLG EQU    #2F442     Drapeau ATTN
BF2DSP EQU    #01C0E     Affichage d'1 buffer
BLDDSP EQU    #01898     Construction de l'affichage
CKSREQ EQU    #00721     Vérifie une requète
CRLFOF EQU    #02296     Termine l'affichage
FNRTN2 EQU    #0F219     Retour à BASIC
POPBUF EQU    #010EE     Prend une touche du buffer
SLEEP  EQU    #006C2     Mise en sommeil léger
kcN    EQU    #00022     Code de la touche 'N'
kcY    EQU    #00006     Code de la touche 'Y'
*
       ENTRY  yesno
       CHAR   #F         C'est une fonction
       KEY    'YESNO'
       TOKEN  223        à vous de mettre le bon
       ENDTXT

       NIBHEX 00         Aucun paramètre
yesno  CD0EX              Sauvegarde de D0
       R0=C                 dans R0
       CD1EX              Sauvegarde de D1
       R1=C                 dans R1
       GOSUB  pop1        Pour récupérer l'adresse de
       *                    la chaîne à afficher dans
       *                    la pile de retour
       NIBHEX B1          chr$(27) }
       NIBASC '>'                  } curseur ON
       NIBASC ' ? Y/N'    message à afficher
       NIBHEX B1          chr$(27) \
       NIBASC 'D'                  | curseur
       NIBHEX B1                   | 3 caractères
       NIBASC 'D'                  | en arrière
       NIBHEX B1                   |
       NIBASC 'D'                  /
       NIBHEX B1          chr$(27) }
       NIBASC '<'                  } curseur off
       NIBHEX FF          Marqueur de fin de chaîne
       *
pop1   C=RSTK             Adresse de chaîne dans C(A)
       GOSUB  dsp         Affichage de la chaîne
*
* Routine principale : mise en sommeil et attente
*   d'une touche ou d'une alarme ou autre (HPIL)
*
key?   GOSBVL SLEEP       Met le 71 en sommeil léger
       GOC    srq?        Si aucune touche dans
       *                    le buffer de touche (le
       *                    71 a été réveillé par
       *                    autre chose qu'1 touche)
       GOSBVL POPBUF      Code de touche dans B(A)
       C=0    A           C(A) = 00000
       LC(2)  kcY         C(A) = 00006
       ?C=B   A           Touche pressée = 'Y' ?
       GOYES  yes           Oui : traiter touche 'Y'
       LC(2)  kcN         C(A) = 00022
       ?C=B   A           Touche pressée = 'N'
       GOYES  no            Oui : traiter touche 'N'
*
* Il faut maintenant éviter tout effet indésirable
* de la touche ATTN : Purger tout le tampon
* de touche et attendre une nouvelle touche.
*
       D1=(5) ATNFLG      Adresse du tampon de touche
       C=0    W
       DAT1=C W
       D1=D1+ 16
       DAT1=C W
       GONC   key?        B.E.T.

srq?   GOSBVL CKSREQ      Chercher qui réveilla Titan
       GOTO   key?          et retourner dormir
*
* Traitement de la touche 'Y'
*  - renvoyer 1 à BASIC
*  - afficher 'Yes' à la place de 'Y/N'
*
yes    C=0    W           C(W) = 0000000000000000
       C=C+1  S           C(W) = 1000000000000000
       CSRC               C(W) = 0100000000000000
       R2=C               Résultat dans R2
       GOSUB  pop2        Pour avoir l'adresse
       *                    du compte rendu
       *                    à afficher
       NIBASC 'Yes'       Chaine à afficher
       NIBHEX FF          Marqueur de fin de chaîne
*
* Traitement de la touche 'N' :
*   - renvoyer 0 à BASIC
*   - remplacer 'Y/N' à l'affichage par 'No'
*
no     C=0    W           C(W) = 0000000000000000
       R2=C               Résultat dans R2
       GOSUB  pop2        Pour avoir l'adresse
       *                    du compte rendu
       *                    à afficher
       NIBASC 'No '       Chaine à afficher
       NIBHEX FF          Marqueur de fin de chaîne
* Affichage du compte-rendu et renvoi
*   de la réponse à BASIC
* L'adresse de la chaîne à afficher est dans
*  la pile de retour. Il ne reste plus
*   qu'à l'afficher.
```

```
pop2    C=RSTK          C(A) = adresse de la chaîne
        GOSUB  dsp      Pour affichage
        GOSBVL CRLFOF    Termine ligne d'affichage
        C=R2            Récupérer résult. dans C(W)
        A=R1            Restaurer D1
        D1=A
        A=R0            Restaurer D0
        GOVLNG FNRTN2   Envoi du résultat
        *                 et retour à BASIC
*
* Routine d'affichage d'une chaîne pointée par C(A)
dsp     D1=C            Adresse de chaîne dans D1
        GOSBVL BF2DSP   Envoie la chaîne au display
        GOSBVL BLDDSP   Construction de l'affichage
        RTNCC           Retour

        END
```

ENCORE DES DATES !

Vous vous rappelez certainement deux articles majeurs parus dans JPC. Il s'agissait de DATELEX, de Laurent Istria dans JPC 28 (octobre 1985). Quelques mois après, François Le Grand nous faisait part de ses «améliorations» dans JPC 31 (avril 1986).

Las, Laurent nous avait présenté quelques fonctions de calcul sur les dates, mais sans respecter quelques points essentiels :

- Il existe dans le HP-71 un format de dates reconnu par le système. Il s'agit de *"aaaa/mm/jj"*, ou *"aa/mm/jj"* en format alphanumérique, ou *aajjj* en format numérique. L'introduction d'un nouveau format, même s'il se référait à celui de la HP-41, ne faisait qu'accroître la confusion.

- Quand un Lex doit renvoyer des messages, les concepteurs du HP-71 encouragent à utiliser les tables de messages.

Par ailleurs, et c'est bien normal, quelques bogues subsistaient sur les vérifications de dates. Ainsi, des dates comme 117.041987, ou 117.001987 étaient considérées comme valides. Personne n'est parfait, même Laurent !

Notre excellent camarade François Le Grand nous a présenté une nouvelle version, qui ajoutait la fonction DATE+, et ôtait du source les commentaires...

Hélas, trois fois hélas ! Les noms de fonctions comportant des symboles tels que "+" sont refusés par Hewlett-Packard lors de l'homologation. Nous nous vîmes donc refuser cette orthographe par les Laboratoires HP de Corvallis.

La situation était grave. Cette anarchie ne pouvait plus régner. Munis de nos IDS, nous avons donc entrepris un ravalement général. Et voici le résultat aujourd'hui :

UTILISATION

Les dates peuvent être introduites de deux manières différentes :

- format HP-71 (*"aa/mm/jj"*, ou *"aaaa/mm/jj"*), par exemple "1789/07/14".

- format HP-41 (*jj.mmaaaa* ou *mm.jjaaaa*), par exemple 14.071789.

Les dates doivent être comprises entre le 15 octobre 1582 et le 31 décembre 9999.

Les fonctions disponibles sont :

DMY change le format numérique en *jj.mmaaaa* (format européen).

MDY change le format numérique en *mm.jjaaaa* pour nos camarades américains.

DOW (Day Of Week) renvoie la numéro du jour correspondant à la date donnée en paramètre, ou à la date courante si elle n'est pas spécifié. 0 correspond à dimanche, 1 à lundi... et 6 à samedi.

Exemples :

```
DOW("1789/07/14") -> 2  (mardi)
DOW(14.071789) -> 2 si on est en mode DMY
DOW(14.071789) -> "Invalid Arg" si mode MDY
```

DOW$ est la même fonction que précédemment, sauf que le résultat est le nom du jour et non son numéro. Ce nom est donné en anglais, et est traduisible par un Lex de traduction. La correspondance entre les noms anglais et les noms français est :

Sunday	Dimanche
Monday	Mardi
Tuesday	Mardi
Wednesday	Mercredi
Thursday	Jeudi
Friday	Vendredi
Saturday	Samedi

Exemples :

```
DOW$("1789/07/14") -> "Tuesday" (mardi)
DOW$(18.041987) -> "Saturday" (samedi)
```

DDAYS renvoie le nombre de jours entre deux dates. DDAYS(*d1*,*d2*) renvoie donc *d1-d2*.

Exemples :

```
DDAYS(18.041987,"1789/07/14") -> 72230
DDAYS(31.121999,DATE$)*24*3600 -> ?
```

faites ce dernier exemple en comparant avec l'horloge du Centre Georges Pompidou.

DATEADD (autrefois DATE+) renvoie la date correspondant à la date donnée plus (ou moins) le nombre de jours spécifié.

Exemples :

```
DATEADD("1789/07/14",1) -> 15.071789
DATEADD(18.041987,-1) -> 17.041987
```

DATESTR$ convertit une date au format numérique HP-41 (*jj.mmaaaa* ou *mm.jjaaaa*) en format alphanumérique HP-71 (*"aaaa/mm/jj"*). Notez que l'année renvoyée est obligatoirement sur quatre chiffres, alors que DATE$ la renvoie sur deux.

Exemples :

```
DATESTR$(14.071789) -> "1789/07/14"
DATESTR$(DATE$) -> l'année sur 4 chiffres
DATESTR$("87/04/18") -> "1987/04/18"
```

Vous pouvez déterminer par programme dans quel mode vous vous trouvez en testant l'état du flag -53 qui nous a été alloué par HP.

Flag = 1 : format DMY
Flag = 0 : format MDY (par défaut)

Nous vous laissons à vos calculs de dates...

Pierre David (37)
Janick Taillandier (246)

```
        LEX    'DATELEX'
* Premiere version :
*   Laurent Istria
*   Parue dans JPC 28, Octobre 1985
*   Creation du Lex
*   Mots-clefs DDAYS, DMY, DOW$, DOW, et MDY
* Deuxieme version :
*   Francois Legrand
*   Parue dans JPC 35, Avril 1986
*   Ajout de DATE+
*   Retrait des commentaires
*   Non correction des defauts
* Troisieme version :
*   Janick Taillandier & Pierre David
*   16 au 18 Avril 1987
*   Reconception complete du Lex
*   Suppression du conflit de DATE+ par renommage
*     en DATEADD
*   Verification de la date corrigee (117.041987)
*   Changement de la signification du flag flDATE
*     (1 = DMY, 0 = MDY)
*   Ecriture des commentaires
*   Creation de la table de messages pour DOW$
*   Ajout du type de parametre alphanumerique
*     pour les dates ("aaaa/mm/jj" ou "aa/mm/jj")
*   Ajout de la fonction de conversion DATESTR$
*   DOW et DOW$ peuvent ne pas avoir de parametre
*   Amelioration de la verification des dates,
*     et correction de jj.991582, jj.00aaaa, et
*     mm.00aaaa)

=id     EQU    #E1
        CON(2) =id
        CON(2) 50
        CON(2) 56

        CON(5) 0
        NIBHEX F
        REL(4) 1+TxTbSt
        REL(4) MSGTBL
        CON(5) 0

* Main Table, triée par tokens
        CON(3) (TxEn02)-(TxTbSt)
        REL(5) =DATESe
        CON(1) #F
        CON(3) (TxEn01)-(TxTbSt)
        REL(5) =DATEAe
        CON(1) #F
        CON(3) (TxEn03)-(TxTbSt)
        REL(5) =DDAYSe
        CON(1) #F
        CON(3) (TxEn04)-(TxTbSt)
        REL(5) =DMYe
        CON(1) #D
        CON(3) (TxEn05)-(TxTbSt)
        REL(5) =DOW$e
        CON(1) #F
        CON(3) (TxEn06)-(TxTbSt)
        REL(5) =DOWe
        CON(1) #F
        CON(3) (TxEn07)-(TxTbSt)
        REL(5) =MDYe
        CON(1) #D
```

```
* Text Table, triée dans l'ordre alphabétique
 TxTbSt
 TxEn01 CON(1) 13
        NIBASC 'DATEADD'
        CON(2) 51
 TxEn02 CON(1) 15
        NIBASC 'DATESTR$'
        CON(2) 50
 TxEn03 CON(1) 9
        NIBASC 'DDAYS'
        CON(2) 52
 TxEn04 CON(1) 5
        NIBASC 'DMY'
        CON(2) 53
 TxEn05 CON(1) 7
        NIBASC 'DOW$'
        CON(2) 54
 TxEn06 CON(1) 5
        NIBASC 'DOW'
        CON(2) 55
 TxEn07 CON(1) 5
        NIBASC 'MDY'
        CON(2) 56
        NIBHEX 1FF

 flDATE EQU    -53      Flag systeme pour la date
 sDMY   EQU    0        Flag temporaire
 sDOW$  EQU    1        1 si DOW$, 0 si DOW

=ADHEAD EQU    #181B7
=ARGERR EQU    #0BF19
=CMPT   EQU    #125B2
=D0=AVS EQU    #09B2C
=D=AVMS EQU    #1A460
=DAYYMD EQU    #13335
=DRANGE EQU    #1B076
=ERRM$f EQU    #09806
=FLOAT  EQU    #1B322
=FNRTN1 EQU    #0F216
=FPOLL  EQU    #1250A
=FUNCD1 EQU    #2F8C0
=HXDCW  EQU    #0ECB4
=IDIV   EQU    #0EC7B
=NXTSTM EQU    #08A48
=OUTELA EQU    #05303
=POP1R  EQU    #0E8FD
=POP1S  EQU    #0BD38
=R3=D10 EQU    #03526
=RNDAHX EQU    #136CB
=SFLAG? EQU    #1364C
=SFLAGC EQU    #13601
=SFLAGS EQU    #135FA
=STKCHR EQU    #18504
=TBMSG$ EQU    #099AB
=TODT   EQU    #13229
=YMDDAY EQU    #13304

=pTRANS EQU    #000EF
```

```
        STITLE TABLE DE MESSAGES
* Le premier message de la table est un message
* sans aucune signification. Il est la uniquement
* pour satisfaire la contrainte du premier message
* dans les tables. D'ou son nom.
 MBASE  EQU    8
=eSUN   EQU    (MBASE)+0     Sunday
=eMON   EQU    (MBASE)+1     Monday
=eTUE   EQU    (MBASE)+2     Tueday
=eWED   EQU    (MBASE)+3     Wednesday
=eTHU   EQU    (MBASE)+4     Thursday
=eFRI   EQU    (MBASE)+5     Friday
=eSAT   EQU    (MBASE)+6     Saturday
 bidon  EQU    (MBASE)+7
 BB16   EQU    16            day

 MSGTBL CON(2) (MBASE)+0     Lowest message #
        CON(2) (MBASE)+7     Highest message #
*
        CON(2) 16
        CON(2)  bidon   Message # 15
        CON(1) 4
        NIBASC '    '
        CON(1) 12
* Sunday
        CON(2) 15
        CON(2) =eSUN    Message # 8
        CON(1) 2
        NIBASC 'Sun'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* Monday
        CON(2) 15
        CON(2) =eMON    Message # 9
        CON(1) 2
        NIBASC 'Mon'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* Tuesday
        CON(2) 17
        CON(2) =eTUE    Message # 10
        CON(1) 3
        NIBASC 'Tues'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* Wednesday
        CON(2) 21
        CON(2) =eWED    Message # 11
        CON(1) 5
        NIBASC 'Wednes'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* Thursday
        CON(2) 19
```

```
        CON(2) =eTHU    Message # 12
        CON(1) 4
        NIBASC 'Thurs'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* Friday
        CON(2) 15
        CON(2) =eFRI    Message # 13
        CON(1) 2
        NIBASC 'Fri'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* Saturday
        CON(2) 19
        CON(2) =eSAT    Message # 14
        CON(1) 4
        NIBASC 'Satur'
        CON(1) 13
        CON(2) BB16
        CON(1) 12
* day
        CON(2) 12
        CON(2) BB16     Message # 16
        CON(1) 2
        NIBASC 'day'
        CON(1) 12
        NIBHEX FF       Table terminator

        STITLE UTILITAIRES
************************************************
* getdat
*
* But: obtenir une date sous un format utilisable
*   a partir d'un objet sur la Math Stack
* Entree:
*  - D1 = ^ M.S.
* Sortie:
*  - A, B et C = numero du jour depuis le jour 0
*  - D1 reactualise
*  - ST(sDMY) = 1 si mode DMY, 0 si mode MDY
* Abime: A-D, R0, R1, FUNCD1, ST(0), ST(8)
* Niveaux: 3
* Appelle: POP1R, POP1S, conv2, chk/, verdat
* Algorithme:
*
*   DECODAGE :
*
*   si type numerique
*     alors
*       x := IP(arg) ;
*       y := IP(FP(arg)*100) ;
*       A := IP(FP(arg*100)*10000) ;
*       si DMY
*         alors
*           D := x ;
*           B := y ;
*         sinon
*           D := y ;
*           B := x ;
*       fin si ;
*     sinon      (type alphanumerique)
*       p := 3 ;
*       si arg$(p)="/"
*         alors
*           x := arg$(1,2) ;
*           si x < 60
*             alors A := 2000 + x ;
*             sinon A := 1900 + x ;
*           fin si ;
*         sinon
*           p := 5 ;
*           A := arg$(1,4) ;
*       fin si ;
*       si arg$(p) # "/" alors erreur ; fin si ;
*       B := arg$(p+1,p+2) ;
*       p := p+3 ;
*       si arg$(p) # "/" alors erreur ; fin si ;
*       D := arg$(p+1,p+2) ;
*       si il reste des caracteres alors erreur ;
*   fin si ;
*
*   VERIFICATION :
*
* Modifications:
*   Ajout du parametre alphanumerique
*   Essayez 117.041987 avec l'ancienne version !
*   Separation du decodage et de la verification
*   Ajout des commentaires
* Historique:
*   85/10/  : L.I.
*   87/04/16: J.T. & P.D. reconception & recodage
************************************************
* Lecture de flDATE pour avoir le mode DMY ou MDY
 getdat LC(2)  flDATE   C(B) = flag number
        GOSBVL =SFLAG?
* En sortie de SFLAG?
*   Cy = flag teste (1 si DMY, 0 si MDY)
*   HEX mode
*   P=0
        ST=1    sDMY    Mode DMY par defaut
        GOC     getd10  DMY, on ne change rien
        ST=0    sDMY    pour les ricains !
* Test du type
 getd10 A=DAT1 S        Signature de l'element
        A=A+1  S        Chaine = F ==> Cy := 1
        GOC    getstr
 getnum GOSBVL =POP1R
        D1=D1+ 16       On passe le reel
* En sortie de POP1R :
*   A = 12 digits form
*   DEC mode
        P=     0        Apres POP1R, P = ?
        SETHEX
* Test du signe
```

```
        ?A#0   S        signe different de "+"
        GOYES  Ivarg
* Test de l'exposant du nombre lu. Il doit valoir
* 0 ou 1. Le registre A a donc la forme suivante :
* (exemple dans le cas DMY)
*
* A(W) = 0jmmaaaa.....000
* A(W) = 0jjmmaaaa....001
        C=0    X
        ?A=C   X
        GOYES  getn20
        C=C+1  X
        ?A#C   X
        GOYES  Ivarg
        ASL    W
* On a dans A :
* A(W) = jjmmaaaa........  ou encore
* A(W) = mmjjaaaa........  si MDY
 getn20 ASLC
        ASLC
        C=A    B          C(B) := jj (si DMY)
* A(W) = mmaaaa........jj  si DMY
* A(W) = jjaaaa........mm  si MDY
        ASLC
        ASLC
* Si DMY
*   A(B) = mm
*   C(B) = jj
* Si MDY
*   A(B) = jj
*   C(B) = mm
* dans tous les cas, A(W) = aaaa........yyxx
*
* Si MDY alors ACEX B
        ?ST=1  sDMY
        GOYES  getn30
        ACEX   B
* quelque soit le mode, on a maintenant :
* A(B) = mm
* C(B) = jj
 getn30 D=C    B        D(B) := jj
        B=A    B        B(B) := mm
* Reste a isoler la date dans A(A) :
        A=0    A
        ASLC
        ASLC
        ASLC
        ASLC           A(A) := 0aaaa
* Nous avons donc maintenant :
*   A(A) = 0aaaa
*   B(B) = mm
*   D(B) = jj
        GOTO   verdat
 Ivarg  GOTO   ivarg
 getstr GOSBVL =POP1S
        CD1EX
        C=C+A  A        C(A) := ^ item suivant
        D1=(5) =FUNCD1  Sauvegarde de D1
        DAT1=C A
        D1=C            D1 := ^ debut de la chaine
        P=     5
        A=0    P
        P=     0
        ASRB            A(A) := longueur en octets
        C=0    A
        ST=1   8        8 caracteres pour la date
        LC(1)  8
        ?A=C   A
        GOYES  gets10
        ST=0   8        10 caracteres pour la date
        LC(1)  10
        ?A#C   A
        GOYES  Ivarg
 gets10 GOSUB  conv2    convertit 2 caracteres
* C(B) = l'annee (ou le siecle)
        ?ST=1  8
        GOYES  gets20
* L'annee est sur 4 chiffres, il faut lire les
* deux derniers.
        A=0    W
        A=C    B
        ASL    A
        ASL    A
        R0=A            R0 := 0000000000000aa00
        GOSUB  conv2
        A=R0
        A=C    B        A(W) := 0000000000000aaaa
        GOTO   gets50
* L'annee est sur 2 chiffres.
*   si <60 alors 20aa
*          sinon 19aa
 gets20 A=0    W
        A=C    B
        C=0    A
        LCHEX  60
        ?A<C   B
        GOYES  gets30
        LCHEX  19
        GONC   gets40
 gets30 LCHEX  20
 gets40 CSL    A
        CSL    A
        A=A+C  A        A(W) := 000000000000aaaa
 gets50 R0=A
        GOSUB  chk/
        GOSUB  conv2    C(B) := mm
        R1=C
        GOSUB  chk/
        GOSUB  conv2
        D=C    B        D(B) := jj
* restauration de D1
        D1=(5) =FUNCD1
        C=DAT1 A
        D1=C
* restauration du mois et de l'annee
        C=R1            C(B) := mm
```

```
       B=C    B        B(B) := mm
       A=R0            A(A) := 0aaaa
* Attention. Le code continue !!!

************************************************
* verdat
*
* But: verifier la validite d'une date
* Entree:
*  - A(A) = 0aaaa
*  - B(B) = mm
*  - D(B) = jj
* Sortie:
*  - A, B et C = numero du jour depuis le jour 0
*  - HEX
*  - P=0
* Abime: A-D
* Niveaux: 2
* Appelle: YMDDAY
* Algorithme:
*   erreur si j=0 ;
*   erreur si m=0 ;
*   erreur si a<1582 ;
*   erreur si m>12 ;
*   si a>1582
*     alors
*       si m#2
*         alors
*           jmax := dernier jour du mois ;
*           erreur si j>jmax ;
*           (date valide)
*         sinon
*           erreur si j>29 ;
*           si j=29
*             alors
*               erreur si a non divisible par 4 ;
*               si a divisible par 100
*                 alors
*                   erreur si non divis. par 400;
*               fin si ;
*           fin si ;
*       fin si ;
*     sinon (annee = 1582)
*       erreur si m<10 ;
*       erreur si m=10 et j<15 ;
*       jmax := dernier jour du mois ;
*       erreur si j>jmax ;
*       (date valide)
*   fin si ;
*
* Modifications:
*   separation logique du reste du sous programme
*   eclaircissement de l'algorithme
*   tests corrects pour 01.991582 (DMY)
*   tests corrects pour j = 0 ou m = 0
* Historique:
*   87/04/18: J.T. & P.D. conception & codage
************************************************
* erreur si mois = 0 ;
 verdat ?B=0   B
        GOYES  erreur
* erreur si jour = 0 ;
        ?D=0   B
        GOYES  erreur
* erreur si mois > 12 ;
        LCHEX  12
        ?B>C   B
        GOYES  erreur
* erreur si annee < 1582 ;
        LCHEX  01582
        ?A<C   A
        GOYES  erreur
* si annee > 1582
        ?A=C   A
        GOYES  verd50   annee = 1582
*   alors
*     si mois # 2
*       alors verification normale
        LCHEX  02       Fevrier
        ?C#B   B
        GOYES  verd70   mois normal
*       sinon (mois = fevrier)
*         erreur si jour > 29 ;
        LCHEX  29
        ?D>C   B
        GOYES  erreur
*         si jour # 29
*           alors ok
        ?D#C   B
        GOYES  verd99   Ok, jour dans [1..28]
*           sinon
*             erreur si a non divisible par 4 ;
        C=A    A
        SB=0            Attention !
        CSRB
        CSRB
        ?SB=0
        GOYES  verd20
 erreur GOTO   ivarg
*             erreur si a non divisible par 400 ;
 verd20 ?A#0   B
        GOYES  verd99   Ok, non divisible par 100

        C=A    A
        CSR    A
        CSR    A        C(B) := siecle
        SB=0            inutile ?
        CSRB
        CSRB
        ?SB=0
        GOYES  verd99   Ok, divisible par 400
        GONC   erreur   B.E.T.
*         fin si ;
* (annee 1582)
*     erreur si mois < octobre
 verd50 LCHEX 10        Octobre
```

```
        ?B<C  B
        GOYES erreur
*     erreur si mois = 10 et jour < 15 ;
        ?B#C  B
        GOYES verd70    mois normal
        LCHEX 15
        ?D<C  B
        GOYES erreur
* Attention ! le code continue !
* (mois normal)
* jmax := dernier jour du mois
 verd70 C=B    B
        R0=C            Sauvegarde du mois
        LCHEX  07
        ?B<=C  B
        GOYES  verd80
        SETDEC
        B=B+1  B
        SETHEX
 verd80 LCHEX  01
        B=B&C  B        B := bit de poids faible
        LCHEX  30
        C=C+B  B        C(B) := jmax
* erreur si jour > jmax ;
        ?D>C   B
        GOYES  erreur
        C=R0            sauvegarde du mois
        B=C    B
* Ok, c'est bon
 verd99 C=A    A
        A=0    W
        A=C    A
        C=B    B
        B=0    W
        B=C    B
        C=D    B
        D=0    W
        D=C    B
        GOVLNG =YMDDAY
************************************************
* conv2
*
* But: convertir deux caracteres en deux chiffres
*   BCD.
* Entree:
*  - D1 = ^ M.S.
* Sortie:
*  - C(B) = valeur lue et convertie
*  - D1 actualise
* Abime: A, B(B), C
* Niveaux: 2
* Appelle: conv1, DRANGE
* Historique:
*   87/04/17: J.T. & P.D. conception & codage
************************************************
 conv2  GOSUB  conv1    poids fort
        B=A    B
        GOSUB  conv1
        C=B    B
        CSL    B
        C=A+C  B
        RTN
 conv1  D1=D1- 2
        A=DAT1 B
        GOSBVL =DRANGE
        GOC    ivarg    byte not in range
        LCASC  '0'
        A=A-C  B        A(B) = 0d  (d = 1..9)
        RTN
************************************************
* chk/
*
* But: verifier que le caractere courant est bien
*   un slash.
* Entree:
*  - D1 = ^ M.S.
* Sortie:
*  - si le caractere etait bien un "/", D1 est
*    reactualise
*  - sinon erreur
* Abime: A(B), C(B)
* Niveaux: 0
* Historique:
*   87/04/17: J.T. & P.D. conception & codage
************************************************
 chk/   D1=D1- 2
        A=DAT1 B
        LCASC  '/'
        ?A=C   B
        RTNYES
 ivarg  GOVLNG =ARGERR
************************************************
* send2
*
* But: fonction inverse de conv2 : envoie 2
*   chiffres BCD sur la M.S. en ASCII
* Entree:
*   - D1 = ^ M.S.
*   - D(A) = AVMEMS
*   - A(B) = les deux chiffres en BCD
* Sortie:
*   - D1 reactualise
* Abime: C(B)
* Niveaux: 2
* Appelle: STKCHR
* Historique:
*   87/04/17: J.T. & P.D. conception & codage
************************************************
 send2  ASRC
        GOSUB  send1
        ASLC
 send1  LCASC  '0'
        C=A    P
 stkchr GOVLNG =STKCHR

        STITLE LES ORDRES BASIC
```

```
************************************************
* DATESTR$
*
* But: renvoyer la date alphanumerique au format
*   HP71 ("aaaa/mm/jj") a partir de la date au
*   format numerique jj.mmaaaa ou mm.jjaaaaa
* Note: La date renvoyee par DATE$ est de la forme
*   "aa/mm/jj". La date renvoyee par DATESTR$ est
*   de la forme "aaaa/mm/jj". DATESTR$(DATE$)
*   convertit donc une date "aa" en date "aaaa".
* Syntaxe: DATESTR$ ( <date> )
* Detail: fonction nécessaire pour passer du
*   format "utilisateur" au format "machine".
* Historique:
*   87/04/17: P.D. & J.T. conception & codage
************************************************
        CON(1) 8+4      alpha ou num   1er  param.
        NIBHEX 11       2 parametres obligatoires
=DATESe GOSUB  getdat
* A, B, C = date au format interne
        GOSBVL =DAYYMD
        SETHEX
        P=     0
* A(3-0) = aaaa
* B(B) = mm
* D(B) = jj
        R0=A            R0 := annee
        C=B    B
        R2=C            R2 := mois
        C=D    B
        R3=C            R3 := jour
        GOSBVL =D=AVMS  ne modifie pas A(W)
        R1=C            R1 := ^ bottom of M.S.
        ASR    A
        ASR    A        A(B) := siecle
        GOSUB  send2
        A=R0            A(B) := annee
        GOSUB  send2
        LCASC  '/'
        GOSUB  stkchr
        A=R2            A(B) := mois
        GOSUB  send2
        LCASC  '/'
        GOSUB  stkchr
        A=R3            A(B) := jour
        GOSUB  send2
        ST=0   0        No return desired
        GOVLNG =ADHEAD
************************************************
* DATEADD
*
* But: renvoyer la date correspondant a : date + n
* Syntaxe: DATEADD ( <date> , <n> )
* Modifications:
*   Ajout du parametre de type alpha
*   Ajout des commentaires
*   Clarification du code
*   Extension des dates jusqu'au 31/12/9999
* Historique:
*   86/03/  : F.D.
*   87/04/17: P.D. & J.T. reconception & recodage
************************************************
        CON(1) 8        num            2eme param.
        CON(1) 8+4      alpha ou num   1er  param.
        NIBHEX 22       2 parametres obligatoires
=DATEAe GOSBVL =RNDAHX
* Ce n'est pourtant pas si dur d'utiliser les
* registres dans leur totalite.
        C=0    W
        C=A    A
        GOC    DTAD10
* Parametre negatif
        C=-C   A
        C=-C   W        parametre negatif sur 16 q.
 DTAD10 D1=D1+ 16
        R3=C            R3 := n
        GOSUB  getdat
        C=R3
        C=A+C  W        C(W) := date + n
* Attention ! le code continue
************************************************
* rtndat
*
* But: convertir une date en format interne (nb de
*   jours depuis le 1er janvier 0) en reel au
*   format jj.mmaaaa (ou mm.jjaaaa), et retourner
*   a Basic.
* Entree:
*  - C(W) = date au format interne
*  - ST(sDMY) indique le format (DMY ou MDY)
* Sortie: par FNRTN1
* Appelle: DAYYMD
* Historique:
*   87/04/17: P.D. & J.T. conception & codage
************************************************
 rtndat GOSBVL =DAYYMD
* A = aaaa
* B = mm
* D = jj
        ?ST=1  sDMY
        GOYES  DTAD20
        C=B    B        C := mm
        DCEX   B        D := mm ; C := jj
        B=C    B        B := jj
 DTAD20
* Le nombre que l'on devra retourner doit etre de
* la forme :
*   0ddbbaaaa0000001  ou
*   0dbbaaaa00000000  si dd<10
* (dd = D(B), bb = B(B), aaaa = A(3-0))
        C=A    A
        A=0    W
        P=     4
        A=C    WP
        P=     0
* A(W) = 000000000000aaaa
```

```
        ASRC
        ASRC
        ASRC
        ASRC
* A(W) = aaaa000000000000
        A=B     B
        ASRC
        ASRC
* A(W) = bbaaaa0000000000
        LCHEX   10
        DCEX    B       D := 10 ; C(B) := dd
        A=C     B       A(W) := bbaaaa00000000dd
        ?C<D    B       dd < 10
        GOYES   rtnd10  un seul shift, Cy := 1
* deux shifts, Cy := 0
        ASRC
 rtnd10 ASRC
        ASR     W
* A(W) = 0dbbaaaa00000000
* A(W) = 0ddbbaaaa0000000
        GOC     rtnd20
        A=A+1   X
 rtnd20 C=A     W
        GOTO    fnrtn1
************************************************
* DDDAYS
*
* But: renvoyer date1-date2
* Syntaxe: DDAYS ( <date1> , <date2> )
* Modifications:
*   Ajout des parametres de type alpha
*   Ajout des commentaires
* Historique:
*   85/10/  : L.I.
*   87/04/17: P.D. & J.T. reconception & recodage
************************************************
        CON(1) 8+4      alpha ou num
        CON(1) 8+4      alpha ou num
        NIBHEX 22       2 parametres obligatoires
=DDAYSe GOSUB   getdat
        R3=A            R3 := date2
        GOSUB   getdat
        C=R3
* A(W) = date1, C(W) = date2
        D=0     S       signe := "+"
        ?A>=C   W       date1 >= date2
        GOYES   DDAY10  oui : ok
* signe "-"
        ACEX    W       non : on echange
        SETDEC
        D=-D-1  S       et signe := "-"
        SETHEX
 DDAY10 C=A-C   W       C(W) := date1 - date2
        GOSBVL  =HXDCW  full word hex-dec conv.
        GOSBVL  =FLOAT  A(W) = resultat, mode DEC
        C=A     W
        C=D     S       C(S) := signe
        GOTO    fnrtn1
************************************************
* DOW$
*
* But: renvoyer le nom du jour
* Syntaxe: DOW$ ( [ <date> ] )
* Modifications:
*   Ajout du parametre de type alpha
*   Parametre optionnel = date d'aujourd'hui
*   Ajout des commentaires
*   Nom des jours en messages
* Historique:
*   85/10/  : L.I.
*   87/04/17: P.D. & J.T. reconception & recodage
************************************************
        CON(1) 8+4      alpha ou num
        NIBHEX 01       1 parametre optionnel
=DOW$e  ST=1    sDOW$
        GOTO    DOW00
************************************************
* DOW
*
* But: renvoyer le numero du jour
* Syntaxe: DOW ( [ <date> ] )
* Modifications:
*   Ajout du parametre de type alpha
*   Parametre optionnel = date d'aujourd'hui
*   Ajout des commentaires
* Historique:
*   85/10/  : L.I.
*   87/04/17: P.D. & J.T. reconception & recodage
************************************************
        CON(1) 8+4      alpha ou num
        NIBHEX 01       1 parametre optionnel
=DOWe   ST=0    sDOW$
* Algorithme :
* si nb parametre = 1
*   alors decoder la date
*   sinon obtenir la date d'ajourd'hui
* fin si ;
* jour := (date - 1) mod 7
 DOW00  ?C#0    S
        GOYES   DOW10
* Sauvegarde temporaire de D0 et D1
        CD1EX
        RSTK=C
        CD0EX
        RSTK=C
        CSTEX
        R3=C
        GOSBVL =CMPT    C = R1 := current time
* Restauration de D0 et D1 apres le monstre CMPT
        C=R3
        CSTEX
        C=RSTK
        D0=C
        C=RSTK
        D1=C
* Et on reprend le cours de nos investigations...
```

```
        C=R1
        CSR   W
        CSR   W
        CSRB          C / 512 (in seconds)
        GOSBVL =TODT  A = day number
        P=    0
        GOTO  DOW20
 DOW10  GOSUB getdat
 DOW20  A=A-1  W
        C=0    W
        LC(1)  7
        GOSBVL =IDIV  C := a-1 mod 7
        P=    0
* C(0) = numero du jour (0:dimanche ... 6:samedi)
        ?ST=1  sDOW$
        GOYES  DOW30
* Sortie numerique
        CSRC
        CSRC          C(14) := a-1 mod 7
 fnrtn1 GOVLNG =FNRTN1
* Sortie alphanumerique
 DOW30  A=C    A      A(A) := numero du jour
        LC(5)  (=id)~(=eSUN)
        C=C+A  A      C(A) := numero du message
        R0=C
* Pompe dans les IDS I, page 17-60, d'apres MSG$
        GOSBVL =R3=D10  Sauver D1 et D0
        GOSBVL =FPOLL
        CON(2) =pTRANS
        GOSBVL =D0=AVS  D0 := (AVMEMS)
        C=R0
        GOSBVL =TBMSG$
        GOVLNG =ERRM$f  Et c'est supporte !!!
* Fin du pompage...
************************************************
* DMY
*
* But: passer en mode jj.mmaaaa
* Syntaxe: DMY
* Modifications:
*   Utilisation de routines supportees
* Historique:
*   85/10/  : L.I.
*   87/04/17: P.D. & J.T. reconception & recodage
************************************************
        REL(5) =DMYd
        REL(5) =DMYp
=DMYe   LC(2)  =flDATE
        GOSBVL =SFLAGS  Set system flag
        GONC   nxtstm   B.E.T.
************************************************
* MDY
*
* But: passer en mode mm.jjaaaa
* Syntaxe: MDY
* Modifications:
*   Utilisation de routines supportees
* Historique:
*   85/10/  : L.I.
*   87/04/17: P.D. & J.T. reconception & recodage
************************************************
        REL(5) =MDYd
        REL(5) =MDYp
=MDYe   LC(2)  =flDATE
        GOSBVL =SFLAGC  Clear system flag
 nxtstm GOVLNG =NXTSTM
=DMYp
=MDYp   RTNCC
=DMYd
=MDYd   GOVLNG =OUTELA
        END
```

CHOC EN RETOUR
DERIVEE SYMBOLIQUE
(ACTE I TER)

Le programme DIFF paru dans JPC 44 contenait malheureusement d'autres fautes que celles signalées par Dominique Marcaillou dans JPC 47 .

Au total, il y a 3 formules de dérivées fausses.

ligne 910: f(x) = sqr(x)
lire `F1$='2*SQR('&U$&')'` au lieu de :
`F1$='.5*SQR('&U$&')'`

ligne 1010: f(x) = acosh(x)
lire `F1$='SQR('&FNK$(U$,5)&'^2-1)'` au lieu de :
`F1$='SQR(1-'&FNK$(U$,5)&'^2)'`

ligne 1040: f(x) = acoth(x)
lire `F1$='1-'&FNK$(U$,5)&'^2'` au lieu de :
`F1$='1+'&FNK$(U$,5)&'^2'`

La fonction `ATAN` renvoie l'erreur `PARENTHESIS OMITTED`. Pour y remédier, il faut corriger la ligne 940 avec : `F1$='(1+'&FNK$(U$,5)&'^2)'` (oubli de la première parenthèse).

Voilà pour les erreurs bénignes.

Le programme reste cependant bogué car il accepte des expressions du type `S(x)`, `SI(x)`, `FACT(x)`... qu'il dérive sous la forme: `-1/`...
De plus la dérivée de $u(x)^n$ est fausse. Qui sait où se trouve l'erreur (le programme est très compliqué à suivre) ? Quelqu'un ne pourrait-il pas en donner un organigramme ? On pourrait alors passer à l'acte II...

Thierry Besançon (292)

CATALOGUE DES
MEMOIRES DE MASSE

Je vous avais proposé, il y a quelques temps, un programme pour imprimer le catalogue des mémoires de masse.

CCAT était assez lent, disons-le, pour les fichiers de masse. Les séquences HP-IL y faisaient défaut.

Michel Martinet ayant créé les fontions pour lire directement les pistes, sans utiliser les `SEND`, j'ai donc repris le tout en y incluant d'autres fonctions, et surtout celles de structures (super) qui vous permettront de rajouter tous les codes vous intéressant et n'étant pas traités ici, et cela sans avoir à tout remanier.

Ainsi, les fichiers BIN et SDATA étant absents, à vous de travailler s'ils vous sont utiles. De même pour les fichiers pour 41, 75 ou futurs fichiers.

Venons en au programme :

On a le choix entre 3 fichiers (FMAIN, FPORT, FTAPE)
- FPORT vous en demande le numéro
- FTAPE sélection du support entre 2 (au cas où)
- FMAIN (pas d'option prévue)

Si le fichier voulu existe déjà, Q > un nouveau ?

- Si différent de "N" on construit le fichier "N", ou fin de construction fichier, choix du mode `Clavier Imprimante Video Echange Quitter`. *Echange* remplace FTAPE du support avec celui/71 et retour question choix de mode.

Et on s'arrête proprement en appuyant sur `[Q]`.

Voilà, il vous faudra alors environ 2 minutes et 30 secondes pour 250 fichiers d'une disquette double face.

On peut regretter de ne pas disposer de fonction de tri alphanumérique qui permettrait d'étendre les options sur la taille, le type. Mais on peut parier que cette petite merveille verra le jour.

Pour fonctionner, ce programme fait appel à des `SUB`, qui sont numérotes de la ligne 6010 à 6300.

Maintenant, une question perfide : il y a t-il un (ou *des*) volontaires pour m'expliquer comment transférer données et fichiers avec l'interface HPIL/RS232 entre 71 et un autre ordinateur ? Tout listing sera évidemment le bienvenu...

A vous de decoder maintenant

Gérard Kossmann (244)

Programme "LEXPOKE" (rentrer un Lhex sur le HP-75, necessite ROM I/O et le lex PEEKPOKE)

```
  1 DIM A$[256] ! LEXPOKE adapté de HPCC/Datafile V3N6P5
  2 INPUT 'Fichier Txt source: '; F$ @ IF F$='' THEN CAT ALL @ GOTO 2
  3 DISP 'Verif. CHKSUMs en cours"
  4 ASSIGN # 1 TO F$ @ A$=FNA$(1) @ L1=LASTLN?(F$)
  5 L=HTD(HEX$(REV$(A$[3,4]))) @ T$=A$[6,6]
  6 A1$='rxelpcnt'
  7 FOR I=1 TO 7 @ IF FLAG?(A$[5,5],8-I) THEN A1$[I,I]=UPRC$(A1$[I,I])
  8 NEXT I
  9 C2=0 @ Y=1 @ IF NOT POS('TALWB',T$) THEN DO ERROR 68
 10 FOR I=1 TO L1-1 @ A$=FNA$(I) @ C1=0
 11 FOR B=1 TO LEN(A$)-1 @ B1=NUM(A$[B]) @ C1=MOD(C1+B1,255) @ C2=MOD(C2+B1,65535) @ NEXT B
 12 IF C1=NUM(A$[B]) THEN NEXT I ELSE 23
 13 IF C2>32767 THEN C2=C2-65536
 14 IF C2#HTD(HEX$(FNA$(I))) THEN 25
 15 F1$=F$[1,LEN(F$)-1] @ DISP 'Creation LEX : '&F1$
 16 D=HTD(CREATE$(F1$,T$,A1$,L))
 17 DISP 'POKE en cours' @ R=18
 18 FOR I=1 TO L1-Y @ A$=FNA$(I)
 19 Q=LEN(A$)-Y @ IF R=0 THEN 21
 20 IF R>Q THEN R=R-Q @ NEXT I ELSE A$=A$[R+1] @ R=0 @ GOTO 19
 21 APOKE DTH$(D),A$[1,Q] @ D=D+Q @ NEXT I
 22 BEEP @ DISP 'Termine' @ END
 23 INPUT 'Presence Chksms ?','O'; A1$ @ Y=POS(UPRC$(A1$),'O')
 24 IF NOT Y THEN 15
 25 EDIT F$ @ DISP 'Ligne';I;': ERR verif.' @ BEEP @ STOP @ GOTO 10

================================================================
 26 DEF FNA$(X)
 27 READ # 1,X ; A$ @ FNA$=ASC$(MAP$(' ','',A$)) @ END DEF
```

****

Programme "LEXDUMP" (imprimer un Lhex sur le HP-75, necessite ROM I/O et le lex PEEKPOKE)

```
  1 DIM A$[256],D$[60] ! LXDUMP75 (adapté de DATAFILE V3N6P4)
  2 W=12 @ R$,D$=" " @ FOR I=1 TO W+W/2-4 @ D$=D$&"-" @ NEXT I
  3 INPUT "Nom LEX ? ";F$ @ F$=LEFT$(F$,8) @ I=-31414
  4 B$=APEEK$(DTH$(I),18) @ IF B$[1,2]='' THEN DISP F$&' non trouve'; @ GOTO 3
  5 IF B$[11]#F$ THEN I=I+18 @ GOTO 4
  6 A=HTD(HEX$(REV$(B$[1,2]))) @ A$=B$&APEEK$(DTH$(A),18) @ A=A+18
  7 I,K,C1,C2=0 @ L1=1 @ L=HTD(HEX$(REV$(A$[3,4])))+18
  8 R=HTD(HEX$(REV$(A$[19,20]))) @ IF A$[6,6]='L' THEN R$='  ROM ID: '&STR$(R)
 10 PRINT F$;' '&B$[6,6];L;' Octets '&R$ @ PRINT
 11 PRINT 'Ligne'&D$&' Data'&D$&' Check' @ PRINT
 12 K=1 @ FOR J=1 TO L @ I=I+1 @ IF I=37 THEN I=1 @ A$=APEEK$(DTH$(A),36) @ A=A+36
 13 IF MOD(J,W)=1 THEN PRINT USING '3d,a' ; L1;' '; @ L1=L1+1 @ C1=0
 14 B1=NUM(A$[I]) @ PRINT FNH$(B1);
 15 C1=MOD(C1+B1,255) @ C2=MOD(C2+B1,65535) @ IF NOT MOD(J,W) THEN PRINT ' ';FNH$(C1)
 16 NEXT J @ IF NOT MOD(L,W) THEN 18
 17 FOR I=1 TO W-MOD(L,W) @ PRINT '   '; @ NEXT I @ PRINT ' ';FNH$(C1)
 18 IF C2>32767 THEN C2=C2-65536
 19 PRINT USING '3d,2x,k' ; L1,DTH$(C2)
 20 PRINT @ END
```

```
================================================================
  21 DEF FNH$(X) = ' '&HEX$(CHR$(X))


*****************************************************************************************************

Programme "CAT" (imprimez vos catalogues)


   - ------------------------------------------------------------------
     CAT * sans REM(!), ni CHR$ env=4450oct
     ------------------------------------------------------------------
     Les Lex utilisés:
     STRUCT1/SHRINK/FORMALEX/PRINTLEX/KEYWAIT/EDLEX/MMLEX/RWLEX/DESAL/
     Tous les codes ou caractères sont indiqués, pour l'édition du journal
     pour la lisibilité, et sauf quand c'est dû à un calcul, vous avez tout
     intérêt à les supprimer (la chasse est ouverte) FIND est prévu pour !!
     Vous trouverez inclus des affectations de touches pour mémoire, afin de
     vous faciliter les "manip", surtout "F(" permettant de reprendre le
     contenu d'une ligne,pour l'accoler à une autre, etc.
     Deux details qui ont leur importance :
     1)N'oubliez SURTOUT PAS apres un WHILE d'ecrire le END WHILE ou alors
     vous aurez,assez souvent,droit un MEMORY LOST (tres chatouilleuse la 71)
     2)Si vous voulez visualiser debut et fin de WHILE utiliser REM et non "!"
     ou vous vous resignerez a la longue a faire INIT 3
     ------------------------------------------------------------------

============================================================
 510 SUB CAT @ BEEP 30,.2
 520 DIM K$[5],E1$[6],N$[12],C$[32],T$[80],M$[50],E$[256],P$[34]
 530 E1$=ESC$&"E"&ESC$&"%"&CHR$(1)&CHR$(2)
 540 INTEGER B,C,E,F,I,L,N,Q,T,I1,M1,M2,V1 @ REAL P,S
   - SELECTION DU CATALOGUE
1010 WHILE 1 @ REM ----------------------------------------------------1
1020 LC OFF @ CALL PERI(M1,M2,I1,V1) @ M$="[" @ CFLAG 0 @ N=0 !          1
1030 DISP E1$;"main1 port2 tape3 ?["; @ K$=KEYWAIT$ @ DISP K$ !          1
1040 IF NOT POS("23",K$) THEN N$=":MAIN" !                              1
1050 IF K$="2" THEN N$=":PORT(" @ INPUT "port{N} ?[","5";P @ N$=N$&STR$(P)&")"
1060 WHILE K$="3" @ REM ---------------------------------------------a 1
1070 IF M1#-1 THEN N$=":TAPE" @ N=1 ELSE CALL ERMAS @ END !            a 1
1080 IF M2=-1 THEN LEAVE @ REM                                         a 1
1090 REPEAT @ REM ---------------------------------------------------b a 1
1100 BEEP @ DISP "2 Supports ";CAT$(1,N$)[1,10];"choix O/N ?["; !    b a 1
1110 K$=KEYWAIT$ @ DISP K$ !                                         b a 1
1120 UNTIL POS("ON",K$) @ REM ---------------------------------------b a 1
1130 IF K$="N" THEN N=2 @ N$=N$&"(2)" @ LEAVE @ REM                    a 1
1140 END WHILE @ REM ------------------------------------------------a 1

============================================================
   - LE CATALOGUE EXISTE DEJA , FAUT-IL EN CREER UN NOUVEAU ??
2010 M$=M$&N$&"]" @ CALL TOP(T$) @ T$=M$&T$ @ DISP E1$;T$ !             1
2020 CALL POSFCH("F"&N$[2,5],K$) @ IF K$="N" THEN K$="E" @ LEAVE @ REM  1
2030 DISP N$;" autre? O/N["; @ K$=KEYWAIT$ @ DISP K$ !                  1
2040 IF POS("+O",K$) THEN PURGE "F"&N$[2,5] @ K$="E" ELSE K$="L" !      1
2050 LEAVE @ END WHILE @ REM ----------------------------------------1
   - --- ECRITURE DE L'INTITULE DU FICHIER ---
2070 WHILE K$="E" @ REM ---------------------------------------------2
```

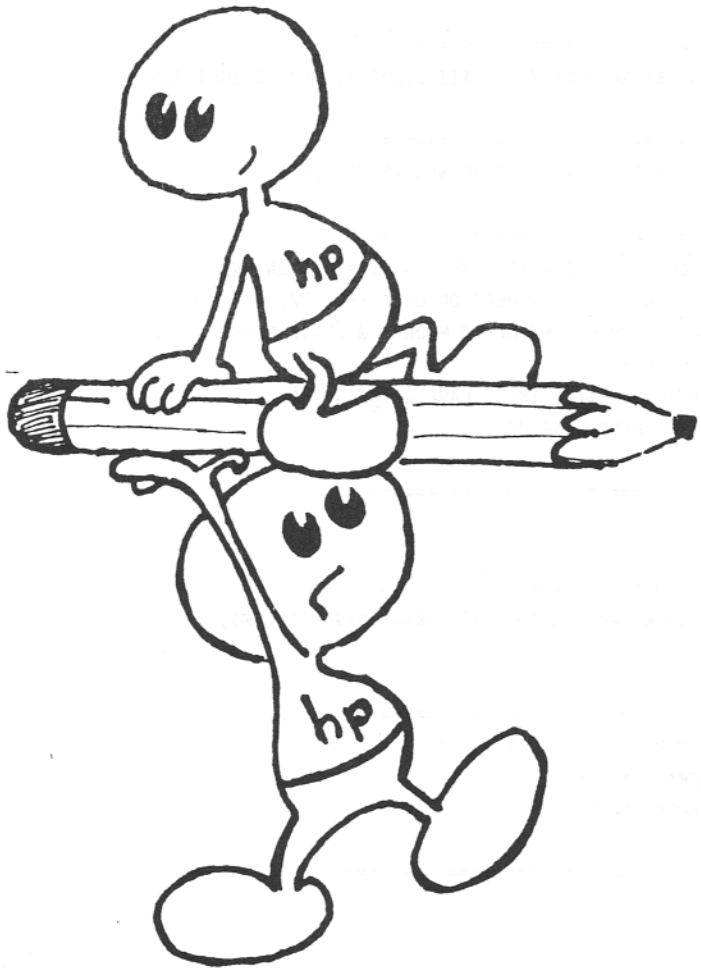
```
2080 CREATE TEXT "F"&N$[2,5] @ ASSIGN #1 TO "F"&N$[2,5] !                    2
2090 CALL TCAT(N,P,N$,E$) @ T$=E$&T$ !                                      2
2100 IF N THEN T$=T$&"{"&RREC$(0,N$)[3,8]&"}  " !                           2
2110 T$=CENTER$(T$,79) @ T$=(T$&SPACE$(80))[1,79] !                         2
2120 DISP E1$;T$ @ PRINT #1;T$ !                                            2
2130 IF N$[2,5]="MAIN" THEN INSERT #1,0;T$ !                                2
   - CATALOGUE :MAIN OU :PORT                                               2
3010 WHILE N=0 @ REM ------------------------------------------------a 2
3020 T$="" @ T$[1]=STR$(I+1) @ T$[4]=" " @ J=0 !                          a 2
3030 REPEAT @ J=J+1 @ I=I+1 @ REM ----------------------------------b a 2
3040 M$=CAT$(I,N$) @ IF LEN(M$)=0 THEN SFLAG 0 @ J=4 !                  b a 2
3050 WHILE LEN(M$)#0 @ REM -----------------------------------------c b a 2
3060 T$=T$&(RED$(M$[1,8])&".........")[1,10]&M$[10,10] !               c b a 2
3070 IF M$[14,14]="N" THEN T$=T$&"b" ELSE T$=T$&M$[12,12] !            c b a 2
3080 T$=T$&(RED$(M$[18,22])&SPACE$(5))[1,6]&" " @ LEAVE @ REM          c b a 2
3090 END WHILE @ REM -----------------------------------------------c b a 2

================================================================
3100 UNTIL J=4 @ REM -----------------------------------------------b a 2
3110 IF LEN(T$)>10 THEN T$[80]=" " @ T$=T$[1,79] !                        a 2
3120 WHILE N$[2,5]="MAIN" @ REM ------------------------------------d   a 2
3130 IF M$="" THEN REPLACE #1,CEIL(I/4);T$ @ LEAVE @ REM             d   a 2
3140 INSERT #1,CEIL(I/4);T$ @ LEAVE @ REM                            d   a 2
3150 END WHILE @ REM -----------------------------------------------d   a 2

================================================================
3160 IF N$[2,5]#"MAIN" THEN PRINT #1;T$ !                                  a 2
3170 IF FLAG(0) THEN CFLAG 0 @ LEAVE @ REM                                 a 2
3180 DISP ESC$&"%"&CHR$(5)&CHR$(5);P;"/";I !                               a 2
3190 END WHILE @ REM ------------------------------------------------a 2

================================================================
   - CAT :TAPE OU :TAPE(2) (ON DECODE TOUT,MAIS PLUS RAPIDE)
4010 WHILE N#0 @ REM ------------------------------------------------e 2
4020 P$="" @ RESTORE !                                                     e 2
4030 FOR I=1 TO 34 @ READ E @ P$=P$&CHR$(E) @ NEXT I !                     e 2
   - PURGE(2)+TEXT(4)+LEX(8)+BASIC(8)+KEY(4)+FORTH(4)+DATA(4)=34 !         e 2

================================================================
4050 DATA 0,0,0,1,224,209,226,8,226,9,226,10,226,11
4060 DATA 226,20,226,21,226,22,226,23
4070 DATA 226,12,226,13,226,24,226,25,224,240,224,241

================================================================
4080 M$=" TST LSLPLEL BSBPBEB KSK FSF DSD" @ E=1 @ I=0 !                   e 2
4090 CREATE TEXT PROV @ ASSIGN #2 TO PROV @ PRINT #2;"DEBUT" !             e 2
4100 REPEAT @ E=E+1 @ E$=RREC$(E,N$) @ PRINT #2;E$ @ REM --------f       e 2
4110 UNTIL POS(E$,CHR$(255)&CHR$(255)&CHR$(255)) @ REM ----------f       e 2
4120 WHILE 1 @ REM -------------------------------------------------g e 2
4130 C=0 @ DELETE #2,0 @ READ #2,0;E$ !                                  g e 2
4140 REPEAT @ F=0 @ T$="" @ T$[1]=STR$(I+1) @ T$[4]=" " @ REM -----h g e 2
4150 REPEAT @ F=F+1 @ I=I+1 @ C=C+1 @ REM ----------------------i h g e 2
4160 C$=E$[1,32] @ E$=E$[33] !                                         i h g e 2
4170 IF C$[1,1]=CHR$(255) THEN SFLAG 0 @ PURGE PROV @ LEAVE @ REM  h g e 2
4180 WHILE C$[1,5]="FTAPE" @ REM ---------------------------k   i h g e 2
4190 S=256*CEIL(((CEIL(P/4)+1)*78+2)/256) !                    k   i h g e 2
4200 C$[19,19]=CHR$(S DIV 2^16) !                              k   i h g e 2
4210 S=S-(S DIV 2^16) @ C$[20,20]=CHR$(S DIV 256) !            k   i h g e 2
```

```
4220 LEAVE @ END WHILE @ REM ------------------------------k   i h g e 2
4230 T$=T$&(RED$(C$[1,10])&".........")[1,10]&C$[10,10] !       i h g e 2
   - Si d'autres codes,veillez a gerer la bonne position
4250 WHILE 1 @ REM ----------------------------------------l   i h g e 2
4260 WHILE 1 @ REM ----------------------------------------m l   i h g e 2
4270 B=POS(P$,C$[11,12]) !                                 m l   i h g e 2
4280 IF B=0 OR B=1 THEN S=B+1 @ LEAVE @ REM                m l   i h g e 2
4290 T$=T$&M$[B-2,B-1] !                                   m l   i h g e 2
4300 IF B<7 THEN S=3 @ LEAVE @ REM                         m l   i h g e 2
4310 IF B<23 THEN S=4 @ LEAVE @ REM                        m l   i h g e 2
4320 IF B<31 THEN S=5 @ LEAVE @ REM                        m l   i h g e 2
4330 S=6 @ LEAVE @ REM                                     m l   i h g e 2
4340 END WHILE @ REM ---------------------------------------m l   i h g e 2

================================================================
4350 ON S GOTO 'VOIR','NUL','TXT','LEBA','KEFO','DAT' !      l   i h g e 2

================================================================
4360 'VOIR': T$=T$&"-a voir-" @ LEAVE @ REM                   l   i h g e 2

================================================================
4370 'NUL': T$=T$&"-purgee-" @ I=I-1 @ LEAVE @ REM            l   i h g e 2

================================================================
4380 'TXT': S=NUM(C$[19])*2^16+NUM(C$[20])*256 @ GOTO 4440 ! l   i h g e 2

================================================================
4390 'LEBA': IF B>14 THEN S=6 ELSE S=0 !                      l   i h g e 2
4400 S=CEIL(HTD(REV$(ATH$(C$[29,30]))))/2-S) @ GOTO 4440 !    l   i h g e 2

================================================================
4410 'KEFO': S=CEIL(NUM(C$[29])/2)+NUM(C$[30])*128 @ GOTO 4440 ! i h g e 2

================================================================
4420 'DAT': S=(NUM(C$[30])*256+NUM(C$[29]))*NUM(C$[31]) !     l   i h g e 2
4430 GOTO 4440 !                                              l   i h g e 2
4440 IF S<10^5 THEN T$=T$&(STR$(S)&".....")[1,6] !            l   i h g e 2
4450 IF S>=10^5 THEN T$=T$&(STR$(IP(S/1000))&"Ko-")[1,6] !    l   i h g e 2
4460 LEAVE @ END WHILE @ REM ------------------------------l   i h g e 2
4470 UNTIL F=4 @ IF FLAG(0) THEN LEAVE @ REM ----------------i h g e 2
4480 PRINT #1;T$[1,79] @ DISP ESC$&"%"&CHR$(5)&CHR$(5);P;"/";I !   h g e 2
4490 UNTIL C=8 @ IF FLAG(0) THEN LEAVE @ REM ----------------h g e 2
4500 END WHILE @ IF LEN(T$)>10 THEN PRINT #1;T$ @ REM --------g e 2

================================================================
4510 LEAVE @ END WHILE @ REM ----------------------------------------e 2
4520 ASSIGN #1 TO * @ K$="L" @ IF N THEN COPY "F"&N$[2,5] TO N$ !       2
4530 CFLAG 0 @ LEAVE @ END WHILE @ REM ---------------------------------2
   - OPERATIONS SUR CATALOGUE
5010 WHILE K$="L" @ REM ---------------------------------------------3
5020 REPEAT @ DISP "Cla Ech Imp Vid {Qt}" @ K$=KEYWAIT$ @ REM ---------a 3
5030 WHILE K$="E" AND N$[2,5]="TAPE" @ REM -------------------------b a 3
5040 CALL POSFCH("FTAPE",K$) !                                     b a 3
5050 IF K$#"N" THEN PURGE "FTAPE" !                                b a 3
5060 COPY :TAPE TO "FTAPE" @ SHRINK "FTAPE" @ K$="E" @ LEAVE @ REM   b a 3
5070 END WHILE @ REM ------------------------------------------------b a 3

================================================================
```

```
5080 UNTIL K$#"E" @ REM ---------------------------------------------a 3
5090 WHILE POS("IV",K$) OR NOT POS("QC",K$) @ REM -------------------c 3
5100 IF I1#-1 THEN PRINTER IS :I1 ELSE IF V1#-1 THEN PRINTER IS :V1 !  c 3
5110 IF I1=-1 AND V1=-1 THEN K$="C" @ LEAVE @ REM                      c 3
5120 DISP E1$ @ PLIST "F"&N$[2,5] @ LEAVE @ REM                        c 3
5130 END WHILE @ REM ------------------------------------------------c 3

================================================================
5140 WHILE K$="C" @ REM ---------------------------------------------d 3
5150 ASSIGN #1 TO "F"&N$[2,5] @ T=FILESZR("F"&N$[2,5]) @ READ #1;T$ !  d 3
5160 FOR L=1 TO T-1 @ READ #1;T$ @ C=VAL(T$) @ T$=T$[5] @ B=0 !        d 3
5170 FOR B=1 TO 4 @ DISP STR$(C+B-1)&" ="&T$[1,18] @ T$=T$[19] !       d 3
5180 K$=KEYWAIT$ !                                                     d 3
5190 IF K$="." THEN K$="Q" @ ASSIGN #1 TO * @ LEAVE @ REM              d 3
5200 NEXT B !                                                          d 3
5210 NEXT L @ K$="Q" @ LEAVE @ REM                                     d 3
5220 END WHILE @ REM ------------------------------------------------d 3

================================================================
5230 WHILE K$="Q" @ DISP "" @ LEAVE @ END WHILE @ REM                   3
5240 LEAVE @ END WHILE @ BEEP 30,.2 @ REM ---------------------------3
5250 END SUB

================================================================
   - Les SUB (sinon les reprendre en lieu et place,pour gagner des octets)

================================================================
6010 SUB LIST(N8$) @ DIM T$[80] @ CALL PCAT(I,N8$) @ CALL TOP(T$)
6020 T$=T$&">"&CAT$(I)[1,22] @ CALL IMP(N8$,T$)

================================================================
6030 SUB TOP(T$) @ D$=DATE$ @ T$=T$&"GK "&D$[7,8]&D$[3,6]&D$[1,2]&" "&TIME$[1,5]

================================================================
6040 SUB PCAT(I,N8$) @ REPEAT @ I=I+1 @ UNTIL N8$=RED$(CAT$(I)[1,8])

================================================================
6050 SUB IMP(N8$,T$) @ CALL PERI(E,R,I1,V1) @ IF I1=-1 AND V1=-1 THEN END
6060 WHILE I1#-1 @ PRINTER IS :%35 @ WRAP ON @ PERF ON @ PAGELEN 72,60 @ MODE 1
6070 UNDERLINE ON @ PRINT T$ @ PLF @ UNDERLINE OFF @ MODE 0 @ LEAVE @ END WHILE
6080 WHILE I1=-1 @ PRINTER IS :DISPLAY
6090 DISP ESC$&"E"&ESC$&"%"&CHR$(15)&CHR$(2);T$ @ LEAVE @ END WHILE
6100 PLIST N8$ @ IF I1#-1 THEN 'LF' ELSE DISP MEM

================================================================
6110 SUB ERMAS @ DISP MSG$(64) @ BEEP

================================================================
6120 SUB POSFCH(N$,K$) @ K$="N" @ ON ERROR GOTO 6130 @ K$=ADDR$(RED$(N$))
6130 OFF ERROR

================================================================
6140 SUB PERI(E,R,I1,V1) @ E=-1 @ R=-1 @ I1=-1 @ V1=-1
6150 E=DEVADDR("%16") @ R=DEVADDR("%16(2)")
6160 I1=DEVADDR("%35") @ V1=DEVADDR("%48")

================================================================
6170 SUB TCAT(E,P,N$,E$)
```

```
6180 WHILE N$[1,5]#":TAPE" @ IF N$=":MAIN" THEN S=MEM ELSE S=MEM(VAL(N$[6]))
6190 LEAVE @ END WHILE
6200 WHILE E=0 @ Y=8 @ P=0
6210 REPEAT @ P=P+Y
6220 IF LEN(CAT$(P,N$)[1,8]) THEN Y=Y*2 ELSE P=P-Y @ IF Y>1 THEN Y=Y/2 ELSE Y=0
6230 UNTIL Y=0 @ N=0 @ FOR Y=1 TO P @ N=N+VAL(CAT$(Y,N$)[18,22]) @ NEXT Y
6240 E$[1]=STR$(P) @ E$[4]="="&STR$(N) @ E$[12]="+ env "&STR$(S)
6250 LEAVE @ END WHILE
6260 WHILE E#0
6270 P=MAXD(N$)-MEMD(N$) @ E$[1]=STR$(P) @ E$[4]="="&STR$(MAXM(N$)-MEMM(N$))
6280 E$[11]="+ "&STR$(MEMD(N$)) @ E$[18]="="&STR$(MEMM(N$))
6290 LEAVE @ END WHILE
6300 E$=E$&" oct" @ END SUB
```

```
Deux types de fichiers d'affectations: 1)KBAS  2)KDEP
1) KBAS
DEF KEY "F=",ESC$&"Q"&"FIND'"&ESC$&"D";
A$=ESC$&"D"&ESC$&"P" @ A$=A$&A$ @ A$=A$&A$
A$=A$&ESC$&"Q"&"DEF KEY'F-','&"'" @ DEF KEY "F(",A$;
DEF KEY "F.",'N8$=RED$(CAT$(0)[1,8])@CALL LIST(N8$)@DESTROY N8$';
-----------------------------------------------------------------
2) KDEP Gestion curseur avec PAC-SCREEN( pratique sous EDTEXT )
DEF KEY "F7",ESC$&"%"&CHR$(1)&CHR$(2);
DEF KEY "F8",ESC$&"A";
DEF KEY "F9",ESC$&"S";
DEF KEY "F1",ESC$&"T";
DEF KEY "F2",ESC$&"B";
DEF KEY "F3",ESC$&"%"&CHR$(71)&CHR$(22);
-----------------------------------------------------------------
```



LE COIN DES LHEX

Comme de coutume, cette rubrique contient la liste des codes hexadécimaux des fichiers Lex parus ce mois-ci.

Rappelons ce qu'est un fichier Lex : c'est un programme pour le HP-71, en assembleur, qui apporte de nouvelles fonctions. Celles-ci sont utilisables directement, ou dans des programmes Basic.

Pour bénéficier de ces nouvelles fonctions, vous n'avez pas besoin de programmer vous-même en assembleur, ni de posséder un module Forth/Assembleur.

Il suffit de recopier le petit programme basic "MAKELEX" ci-dessous, de le lancer et de recopier les codes du fichier Lex désiré. Quand vous avez fini, les nouvelles fonctions sont accessibles, après avoir éteint et rallumé votre HP-71.

Si l'erreur "Erreur de somme" apparaît, vérifiez la ligne que vous avez introduite.

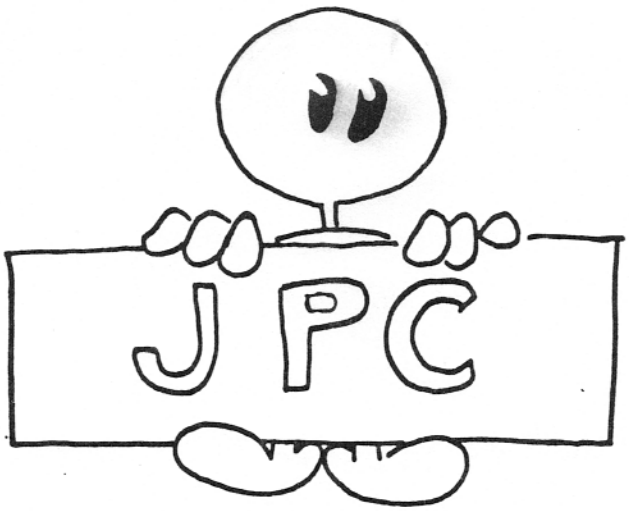
Vous trouverz donc le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !"), ainsi que les Lex de ce mois-ci.

```
CHARLEX

CHRONOLX  CHRONO$    XFN    94100
          CHRONO     XFN    94101

YESNOLEX  YESNO      XFN    92223

DATELEX   DATESTR$   XFN    225050      DATEADD    XFN    225051
          DDAYS      XFN    225052      DMY        XWORD  225053
          DOW$       XFN    225054      DOW        XFN    225055
          MDY        XWORD  225056
```

```
 10 CALL MLEX @ SUB MLEX @ SFLAG -1 @ PURGE AH @ INPUT "Nb. d'octets: ";N @ LC OFF
 20 CREATE DATA AH,1,N-4 @ A=HTD(ADDR$("AH")) @ B=A @ GOSUB 130
 30 Q=1 @ X=0 @ INPUT "000: ",P$;A$ @ C$=A$ @ S=0 @ GOSUB 90
 40 Q=2 @ X=1 @ GOSUB 80 @ A$=A$&C$ @ A=A+37 @ N=N*2+37 @ Q=3 @ SFLAG 5 @ FOR X=2 TO N DIV 16-1
 50 GOSUB 80 @ C$=C$[5*FLAG(5)+1] @ POKE DTH$(A),C$ @ A=A+16-5*FLAG(5,0) @ NEXT X @ Q=4
 60 DISP DTH$(X)[3]; @ INPUT ": ",P$[1,MOD(N,16)];C$ @ GOSUB 90
 70 POKE DTH$(A),C$ @ POKE DTH$(B),A$ @ CFLAG -1 @ END
 80 DISP DTH$(X)[3]; @ INPUT ": ",P$;C$
 90 DISP DTH$(X)[3]; @ INPUT " sm ","---";D$
100 M=S @ FOR Z=1 TO LEN(C$) @ M=NUM(C$[Z])+M+1 @ NEXT Z
110 IF D$=DTH$(MOD(M,4096))[3] THEN GOSUB 130 @ S=M @ RETURN
120 DISP "Erreur de somme" @ BEEP @ P$=C$ @ POP @ ON Q GOTO 30,40,50,60
130 P$="----------------" @ RETURN
```

```
CHARLEX  ID#E1  624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E004131810178 6AF
002: 5E4001E000000000 9F3
003: FE00000000800001F D4D
004: F31BF961400032BF 0E0
005: 38F14A11DB10AD23 47A
006: 07D532BFB8FD7911 82D
007: 11AD754D7A101743 BB0
008: 11014D1CB15D0000 F1B
009: 71450375FF864834 298
00A: 5655581008355654 5EF
00B: 5810002455565870 93C
00C: 0026555658700836 C8E
00D: 555581008364545 FE4
00E: 4A30000A49724000 337
00F: 0808094A2C180814 6A0
010: A466242008355455 9FA
011: 581000054C714000 D40
012: 0C3142404C700832 09C
013: 41414A70002078A0 3F4
014: 2F30000000000000 71F
015: 0000000000000000 A2F
016: 0000000000000000 D3F
017: 0000000000000000 04F
018: 0000000000000000 35F
019: 0000000000000000 66F
01A: 0000000000000000 97F
01B: 0000000000000000 C8F
01C: 0000000000000000 F9F
01D: 0000000000000000 2AF
01E: 0000000000000000 5BF
01F: 0000000000000000 8CF
020: 0000000000000000 BDF
021: 000000000000080C F0A
022: 1A28080008080A2C 274
023: 180008040E340800 5BD
024: 08001E3018000000 8F7
025: 0000000000000000 C07
026: 0000000000000000 F17
027: 0000000000000000 227
028: 0201000000010200 53D
029: 0000000201020000 852
02A: 0001000100000002 B66
02B: 0102010000000000 E7A
02C: 0000000000000000 18A
02D: 045E755142400101 4D6
02E: 0101010000000000 7E9
02F: 0000000000000000 AF9
030: 0000070507000000 E1C
031: 00000000083444C4 15A
032: 44400D7901112D70 4BA
033: 050D750509700000 804
034: 0D70000000384540 B47
035: 4020014E322E3140 E9B
036: 084E794142400000 1EB
037: 000000000002E4559 529
038: 3200000000000000 83E
039: 0000000000000026 B56
03A: 5556587008365556 EB5
03B: 5810083645464830 206
03C: 0832414248700024 547
03D: 5655587008345655 8A4
03E: 5810083446454830 BF3
03F: 0C3042414C700024 F48
040: 5556587008355654 2A5
041: 5810083546444830 5F4
042: 0C3142404C700025 94A
043: 5455587008355455 CA4
044: 5810083544454830 FF2
045: 0C3140414C700875 354
046: 14141870000A4972 6A5
047: 40000E3159454E30 A05
048: 0C7A0F7949400024 D7D
049: 5554587000084A71 0D9
04A: 40000C523A262D10 43A
04B: 0424587458400875 791
04C: 1415187000094A70 AE1
04D: 4000083544454830 E25
04E: 0C3140414C300C74 18D
04F: 5655545000054C71 4E4
050: 40000            5DD

CHRONOLX ID#5E  469 octets

     0123456789ABCDEF sm

000: 348425F4E4F4C485 39B
001: 802E005131810178 6ED
002: FA300E5465600000 A56
003: F020000000000000 D7E
004: 0EC000F1107A000F 0FC
005: D348425F4E4F4424 491
006: 6B348425F4E4F456 82B
007: 1FF8F2B521AFAAF2 C03
008: 24323A28FB7CE08F FB1
009: 4BCE0AFA8F223B1A 380
00A: F22C322158F8A4C0 718
00B: 8F4F6C08FA4B2180 AC9
00C: D0BD6BD6BD6BD6BD EC3
00D: 6BF6BF60D0D0D0D0 285
00E: D0D0DBF60C5AF010 63F
00F: 08FC5A212180C21B 9CC
010: 8F6F215426A10008 D44
011: FC5A21AA21B8F6F2 109
012: 1542048588557142 45E
013: 208F6B0201B084F2 7DF
014: 203D02348627F6E6 B5F
015: F615CD16D3D02A30 F00
016: 20303A30315CD16D 277
017: 3D03A30303E20303 5D8
018: 15CD16D85684170E 96C
019: 18F89810D28038AA D00
01A: AF8038AEAF75CE10 0CC
01B: A7EBEAFA11AB7A10 4A6
01C: 11F494F2BF4BF4AF 876
01D: 67BA1AF774A110B7 C1C
01E: CA131A314D171113 F90
01F: 7B9131A314D171DB 323
020: DA798131E214D171 6AE
021: 1117871048568417 A03
022: E418F89810D28038 D8E
023: AE606C7F04D08558 132
024: 587B21208F6B0201 4A6
025: B344F2AA01502051 81B
026: 111021F849F2D014 B8D
027: BF08A8B43048F719 F32
028: 218FD7521874531F 2B8
029: 244F2157090E52D2 632
02A: 8038AEAF8F2C6005 9E0
02B: 118F1270068CF629 D62
02C: 01B8F6F2156292EE 104
02D: E8F38A21D2203161 489
02E: 8FA90F00511275C0 80F
02F: 112BF4BF431E214D BAF
030: 171AF67670AF77F6 F5A
031: 0DF103DA799031A3 2EE
032: 14D171DFAEA76803 699
033: 1A314D1711137570 9F8
034: 1CF1C51CF8DC32F0 DBF
035: 1128F223B1AF6A3E 161
036: A3E8F38A218D612F 512
037: 01B874F209154301 877
038: 21AF23068DB7CE02 C1B
039: 03103AF1A88A61F1 FAA
03A: F1F4A88A61D41593 34E
03B: 1730120AF1A88BE1 6DD
03C: BB122A84BB463CF  A5C

YESNOLEX ID#5C  137 octets

     0123456789ABCDEF sm

000: 955435E4F4C45485 38F
001: 802E006131810178 6E2
002: 61100C5FDFD00000 A66
003: F710000000000000 D94
004: 081000F9955435E4 104
005: F4FD1FF001361081 499
006: 371097220B1E302F 80B
007: 30295F2E4B144B14 B91
008: 4B144B1C3FF07729 F2E
009: 08F2C6004538FEE0 2CB
00A: 10D231608A1E2312 633
00B: 28A1D31F244F2AF2 9D8
00C: 155717F155755C8F D65
00D: 127006BBFAF2B468 10C
00E: 1610A7A10955637F 487
00F: FAF210A7800E4F60 82C
010: 2FF077A108F69220 BBB
011: 11A1111311108D91 F0E
```

```
012: 2F01358FE0C108F8 2AC
013: 981003F          42E

DATELEX  ID#E1  731 octets

     0123456789ABCDEF sm

000: 44144554C4548502 35E
001: 802E007131810178 6B2
002: AB5001E238300000 A10
003: FD4002A000000011 D63
004: 0A8300F0004E300F 0D7
005: 420E5400F130B450 440
006: 0DA30A8400F540B8 7CE
007: 400FE4084500DD44 B52
008: 14455414444433F4 EAE
009: 4144554354525422 1F9
00A: 3944441495354354 550
00B: 4D49553744F47542 8CD
00C: 63544F475735D444 C48
00D: 95831FF80F001F04 FD7
00E: 0202020202CF0802 324
00F: 3557E6D01CF0902D 6C0
010: 4F6E6D01C11A0345 A52
011: 575637D01C51B057 DCF
012: 55646E65637D01C3 154
013: 1C044586572737D0 4C6
014: 1CF0D02642796D01 84D
015: C31E043516475727 BBC
016: D01CC0012461697C F3E
017: FF315E8FC4631850 2E4
018: 4508401534B444B5 64B
019: 8FDF8E017F200494 9F1
01A: C44AB2932D0B3693 D8A
01B: 643BF0810810AE68 115
01C: 1081087050AEEAE7 4AB
01D: AE8D081081081081 821
01E: 067B063C18F83DB0 BBC
01F: 137C21F0C8F21451 F41
020: 3525A802081CD285 2BA
021: 83088A2D084830A8 63C
022: A63C7341878D1AF0 9DB
023: AEAF0F01007D2111 D6F
024: 0AEA6520AF0AEAD2 137
025: 31069E2903191560 493
026: 3102F2F2CA100792 810
027: 176F01097E117BE0 B9A
028: AE71F0C8F2147135 F36
029: 119AE51109698496 2B1
02A: B3431219E1A33428 625
02B: 5108B2E28A294312 99C
02C: 09657531929E3719 D0D
02D: 6797D682281E81E8 0A4
02E: 326064C096C06D6F 434
02F: 6F682281E81E832C 7CE
030: 452E31019E59D965 B57
031: B031519E7BCAE910 EFD
032: 831709ED9005B650 27D
033: 431100E613103A69 5D9
034: 9E31A118AE5D6AF0 992
035: DAAE9AF1AE5AEBAF DA8
036: 3AE78D403317210A 12A
037: E87B00AE9BE2A620 4E1
038: 11C114B8F670B14A 871
039: 13103B6A011C114B BDF
03A: 31F2962008D91FB0 F6A
03B: 81473008103103A8 2BA
03C: 68D40581C1173AD8 64A
03D: F533310420100AE9 9B5
03E: 10AAEB10B8F064A1 D5C
03E: 10AAEB10B8F064A1 D5C
03F: 109F4F475BF1107E 0FC
040: AF31F277BF1127F9 4AF
041: F31F278AF113709F 852
042: 8408D7B1818C228F BEA
043: BC631AF2D6470FAB FB0
044: FA17F10B762D11BA 360
045: 728F53331870B0AE 6ED
046: 9AEFAE5D6AF024A9 ACE
047: A20814814814814A E36
048: E48148143101AEFA 1CB
049: EA9E350814814BF4 56D
04A: 450B34AF666B0CC2 912
04B: 27DBC10376BC11BA CBF
04C: C39FEC0AFE05BCF0 0A8
04D: 4B7E8F4BCE08F223 46B
04E: B1AF6ACB6770C018 821
04F: 516900C0184194E0 B89
050: 413706136060B10B EE3
051: 8F2B52111B0B0713 25E
052: 407135119BF6BF68 5F1
053: 1E8F922312067007 95C
054: F2CA7CAF23078FB7 D2C
055: CE020871F0816816 0AA
056: 8D612F0DA34801E0 43F
057: C21088F625308FA0 7CA
058: 521FE8FC2B901188 B6D
059: FBA9908D60890630 F04
05A: 00F2000315E8FAF5 294
05B: 31571E1000710003 5D6
05C: 15E8F106318D84A8 96B
05D: 0038D30350F      BC6
```



PPC PARIS CHAPTER

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

BULLETIN D'ADHESION

Nom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
Prénom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| Date de naissance |_|_|/|_|_|/|_|_|
Adresse |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Code Postal |_|_|_|_|_| Ville |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
Téléphone Domicile |_|_|_|_|_|_|_|_|_|_|_| Bureau |_|_|_|_|_|_|_|_|_|_|_|

Profession ____________________
Intérêts ____________________
____

Matériel HP en votre possession ____________________
____
____

Autre matériel informatique ____________________
____
____

Comment avez-vous connu PPC Paris ? ____________________
____

Que recherchez-vous au sein de PPC Paris ? ____________________
____
____

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

Date |_|_|/|_|_|/19|_|_|
Signature, précédée de la mention "Lu et Approuvé" ____________________

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

Paiement effectué le |_|_|/|_|_|/19|_|_| à l'ordre de PPC Paris Chapter,
par [ ] chèque bancaire, [ ] chèque postal ou [ ] mandat lettre

Eventuellement: je m'abonne à compter du |0|1|/|_|_|/19|_|_|

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

ENGLISH SUMMARY

JPC 49 - NOVEMBER 1987

Even if you, reader out of France, are less concerned with it, we inform you that PPC Paris will have a important meeting in January 1987. Subjects will be new PPC Paris board election, new statutes and other important things about our Club.

On page 3, you have a letter from a young member giving a few hints about using FIND.

Two articles in the HP-28 columns. First, we inform you that if you are using an HP-28C version 1CC, you can still use the clock program given in JPC 46 by changing #123E SYSEVAL into #1266 SYSEVAL. Next, Eric Gengoux gives us three little programs to plot on the HP-28C.

Continuing the series about HP-IL discs and HP-41, Michael Markov provides us with programs which are intended to decrease the power consumption and increase medium life.

We will soon have Lex files for the HP-75. However, it is difficult to load these files, because there is no assembler on this machine. So, Eric Gengoux gives us two programs similar to those already existing for the HP-71. They will allow you to easily load those files.

The Forth definitions from Alain Goubault de Brugière's article are intended to help you work with the Translator Pac. It explains the problem of conflicting names between the HP-41 and Forth vocabularies, and how he solves them. Next, he explains how to recover smoothly from an error in Forth environment.

We have a large HP-71 assembly language column this month. First, we start by a short summary of JPC Rom keywords and the various changes which have occurred this year (new keywords and renaming old ones, a few have been suppressed). This gives you an idea of the numerous possibilities of JPC Rom.

Two new assembly language programmers presents us their first productions. First, Lionel Guillou has written a stop watch for the HP-71. The two keywords return the time value, either as a string or as a numeric. The first key press starts the stop watch, the second stops it. Jean Yves Naour gives us a function asking a question and asking for a Yes/No answer.

The last Lex file this month is a new, very much improved version of DATELEX. All the bugs existing in previous versions have been corrected (we hope so !). It is a completely new version. This is the one used in JPC Rom.

The DIFF program published in JPC 44 was really bugged. Thierry Besançon gives on page 36 three modifications but the program is still not good : Thierry says that $u(x)^n$ has not been corrected.

Until next month,

Happy Programming and JPC reading !