JPC

JUILLET-AOUT 1988

NUMERO 56

Le numéro 40 F

A PROPOS DU CLUB

P. David	Editorial	*1*
	Courrier du coeur	*2*

HP28

P. Courbis	Choc en retour	*4*
H. Thévenon	Jeu intégral	*4*
P. Courbis	PEEK et POKE pour HP-28S	*5*
P. Heilbronn	Le facteur est constant (acte II)	*7*
S. Lalande	Matrice de fraction	*9*

HP41

J. Hulaas	Multiplications de matrices	*12*

HP71

X. Bille	Catalogue des mots (acte II)	*16*
J. Baudier	L'assembleur du HP-71 (acte VII)	*18*
F. Vadez	Le temps enchaîné	*21*
S. Vaudenay	La diagonale du Fou	*24*
J.C. Kursner	Directory	*24*
A. Boldireff	Paginez vos lettres et textes	*24*
M. Markov & E. Gengoux	Codes barres HP-41 sur HP-71	*26*
	Le coin des Lhex	*35*

EDITORIAL

Chers Membres,

Je suis bien triste de vous annoncer une mauvaise nouvelle : des rumeurs convergentes annoncent que le HP-71 ne sera plus fabriqué d'ici la fin de l'année. Hewlett-Packard ne l'a pas encore confirmé, mais cela ne saurait tarder. Je pense que nous aurons l'occasion d'en reparler plus longuement dans ces colonnes.

En revanche, deux nouvelles machines seraient sur le point de sortir : il s'agit des HP-22 et HP-32 qui viendraient compléter la gamme actuelle par le bas.

Nous avons déjà mentionné à plusieurs reprises la disponibilité des programmes HP-41 de la défunte *Bibliothèque de Genève*. Tous ces programmes encombrent actuellement la cave d'un de nos membres. Personne ne s'étant manifesté, nous avons décidé d'accorder un délai jusqu'au 31 octobre : si à cette date, vous n'avez trouvé aucune solution, cette bibliothèque de programmes sera détruite.

Enfin, une très bonne nouvelle pour terminer : vous avez reçu avec votre dernier *JPC* une carte d'adhérent. Celle-ci vous permet d'obtenir des réductions allant jusqu'à 25% chez certains distributeurs HP. Alors, profitez-en !

A bientôt,

Pierre David (37)

COURRIER DU COEUR

Patrick Tassignon
Rue des combattants, 15/6
B-4370 Waremme
Belgique

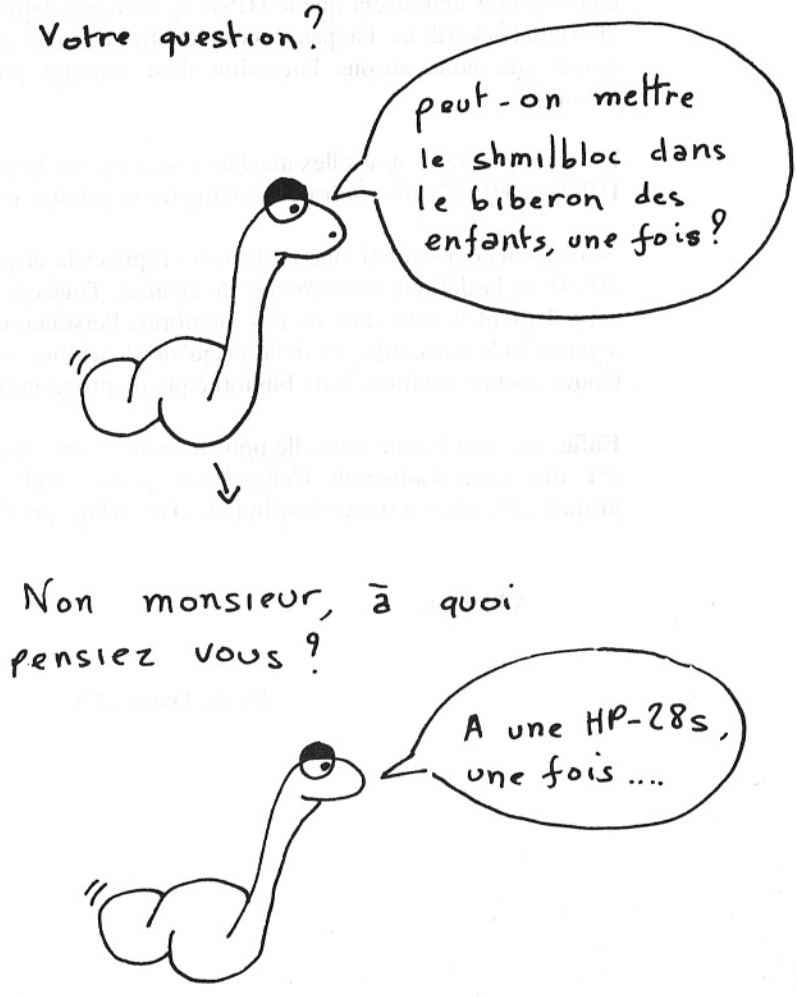
Vend :

Un HP-71 : 2500 FF, un module HP-IL : 800 FF, un module Forth/Assembleur : 800 FF, un module Math : 480 FF et deux modules 4 Ko Ram : 320 FF.

Philippe Ferreira Do Amaral
5 rue Daguerre
93110 Rosny sous Bois
Tél : (1) 45 28 33 32

Vend :

HP-28C sous garantie, état neuf : 1100 FF.



HP28

P. Courbis	Choc en retour	*4*
H. Thévenon	Jeu intégral	*4*
P. Courbis	PEEK et POKE pour HP-28S	*5*
P. Heilbronn	Le facteur est constant (acte II)	*7*
S. Lalande	Matrice de fraction	*9*

ERRATUM

Dans *JPC 55*, une vilaine erreur s'etait glissée dans la chaîne de codes assembleur... Voici la bonne chaîne :

```
76C20A63C069C20C31008F180501BF510CD21441B9F00C14234A
00008BE913410200DA8F8B0508DA69301B6110CAF01431562A26
A265D01B6900C158E13114334E4A208A2D0342020061BF84A174
14334500008A249EA81C1008F8B0501188FEA3505328F8B05086
A908DE39308F49A4085A65DF1361098F8B0501191341l0D8CD14
313117914B3103B6A31909EA903170B6A1580160171CD59D8F8B
0501111411421648O8C09F20
```

Paul Courbis (392)

JEU INTEGRAL

Bonjour à celui qui lira cet article... Devant le manque évident d'agressivité dont font preuve mes congénères (les utilisateurs de HP-28), je m'insurge et je tente de remédier à cette situation de manque... Le seul problème est qu'étant nouveau venu à PPC, je ne sais pas écrire d'articles, aussi je vais m'efforcer de faire de mon mieux pour vous faire part de ces quelques programmes élaborés pendant les cours de philo qui hantent les heures de ma classe de Terminale.

LE JEU

Voici tout d'abord un programme de jeu destiné à égayer ces cours de philo. Le programme principal GAME lance le jeu et gère les six autres sous-programmes. Le but du jeu est de traverser l'écran de gauche à droite en empruntant le sentier que dessine le programme. Il s'efface bien sûr au bout de cinq secondes : il faut donc le mémoriser avant de vous laisser aller dans vos errances...

On déplace le curseur, qui se trouve au début du jeu sur le sentier à gauche de l'écran, avec les touches grises fléchées.

Voici les sept programmes tels qu'ils sont édités par la HP-28C.

```
GAME
«
  'PPAR' PURGE                    mise en place de PPAR
  (0,0) PMIN
  (0,1.2) 0 RAND 2 *
  DUP *W 1.54 *H
  'SIN(X)+1=SIN(X)+1.4' STEQ          tracé du sentier
  CLLCD DRAW
  5 WAIT
  6.8 * DUP 136 / 'S' STO             calcul du step S
  'SIN(X)+1.2' STEQ
  CLLCD
  FOR X
    DO                                  début de boucle
      KEY
      UNTIL 0 ≠
    END                             avance et vérification
    2000 .05 BEEP
    STR→                                 de la position
    EVAL + DUP PIXEL                du curseur par rapport
    3000 .05 BEEP
    DUP                                        au sentier
    C→R SWAP
    'X' STO EQ
    EVAL - ABS .2
    IF >
      THEN STOP ABORT           appel à STOP si plantage
    END
    S
  STEP
  FIN                             appel à FIN si réussite
»


RIGHT
« S 0 R→C »                          déplacement du curseur

LEFT
« 0 S - 0 R→ »

DOWN
« (0,-.08) »

UP
« (0,.08) »

STOP
«
  "  out of the way"
  4 DISP 10 3 BEEP
  CLLCD CLEAR 7 1
  "Votre erreur vous est fatale... Paix a votre ame"
  1 DISP
  FOR X
    X 200 *
    .5 BEEP -1
  STEP
»
```

```
FIN
«
  1 11
  FOR X
    X 200 *
    "La femme du chef de la tribu adore les hommes
     blancs... au court-bouillon..."
    1 DISP .1 BEEP
  NEXT
»
```

Voici maintenant pour les nostalgiques du Basic, deux fonctions : INKEY pour la saisie d'un caractère et INPUT pour la saisie d'une chaîne validée par [ENTER].

Le fonctionnement de INKEY est simple, quant à INPUT c'est une boucle qui répète INKEY jusqu'à ce que l'on presse [ENTER].

```
INKEY
«
  DO KEY UNTIL 0 ≠
  END
»

INPUT
«
  ""
  WHILE
    DO
      KEY
      UNTIL 0 ≠
    END
    DUP "ENTER" ≠
    REPEAT +
  END
  DROP
»
```

LES INTEGRALES

Maintenant, pour les matheux qui cherchent à tracer les fonctions définies par une intégrale, voici INTEG. Si on a la fonction F(x) définie sur $[B_i,B_s]$ par $F(x) = \int f(t)\,dt$ (entre a et x) alors INTEG trace la fonction F.

```
INTEG
«
  CLLCD DRAX                    préparation de l'écran
  Bi Bs
  FOR x
    x
    'f(t)'                      fonction à intégrer
    t a x 3 →LIST
    .1
    ∫                           intégration
    DROP                        erreur de ∫
    R→ PIXEL                    on forme (x,F(x))
    .1
  STEP
»
```

Pour améliorer la précision, il suffit de changer le step et la précision demandée en effectuant ∫.

Merci d'avoir lu jusqu'au bout...

Hervé Thévenon (476)

PEEK ET POKE POUR HP-28S

Les programmes PEEK et POKE sont à entrer à l'aide du programme ASSEMBLEUR pour HP-28S. Pour cela entrer ASSEMBLEUR, si ce n'est déja fait, et taper les chiffres hexadecimaux de la colonne *codes* sous forme d'une chaîne, en une seule ligne, sans espace puis executer ASSEMBLEUR. Vous aurez alors dans le niveau 1 de la pile deux System Object et DROP2 (pour POKE) ou deux System Objects (pour PEEK).

Utilisation de ces deux nouvelles commandes :

POKE

Syntaxe :

```
2:#01251          adresse où poker
1:"ABF4324D3"     quartets à poker
```

Pas de sortie

Par exemple, `#FF840 "FFFFFFFF"` place F (15 en décimal) dans les cases mémoire #FF840 à #FF847 (noircit la première colonne de l'écran).

Les messages d'erreur possibles sont Too Few Arguments et Bad Argument Type.

PEEK

Syntaxe :

```
2:#011CA          adresse du peek
1:#10             longueur à lire
```

En sortie : chaîne contenant les codes hexadecimaux des quartets lus.

Par exemple, #0 #10 PEEK renvoie les 16 premiers quartets de la Rom : "8FDB400000CFE".

Les messages d'erreur possibles sont Too Few Arguments, Bad Argument Type et Too Few Memory.

Les programmes

POKE

```
Code     Lbl Instructions   Commentaires

76C20        CON(5) #02C67  début de structure
A63C0        CON(5) #0C36A  mise à jour Stacksize
69C20        CON(5) #02C96  programme LM
91100    DEB CON(5) FIN-DEB longueur du code
8F18050      GOSBVL #05081  sauvegarde D0 D1 B D
1BF510C      D0=(5) #C015F  No de l'instr. en cours
D2           C=0    A       fixé à zero (sans nom) :
144          DAT0=C A       nécessaire pour les msg
1B9F00C      D0=(5) #C00F9  d'erreur
142          A=DAT0 A       A=Stacksize
34E0000      LCHEX  0000E
8BE          ?A>=C  A       y a t'il assez d'objets ?
91           GOYES  OK1     oui ---> OK1
3410200      LCHEX  00201   erreur Too Few Arguments
DA       ERR A=C A
8F8B050      GOSBVL #050B8  récupération D0 D1 B D
8DA6930      GOVLNG #0396A  affichage erreur
1B6110C OK1  D0=(5) #C0116  sauvegarde éventuelle
AF0          A=0    W       dans LAST
15B9         A=DAT1 10
1562         C=DAT0 XS
A26          C=C+C  XS
A26          C=C+C  XS
5D0          GONC   L1
1B6900C      D0=(5) #C0096
158E         DAT0=A 15
34E4A20 L1   LCHEX  02A4E   C=prologue chaîne
D5           B=C    A
147          C=DAT1 A
137          CD1EX
143          A=DAT1 A       A=prologue obj. niveau 1
8A4          ?A#B   A       obj. niv. 1=chaine ?
F1           GOYES  L2      non --> erreur
135          D1=C
174          D1=D1+ 5
3407A20      LCHEX  02A70
D5           B=C    A       B=prologue entier
147          C=DAT1 A
137          CD1EX
143          A=DAT1 A       A=prologue obj. niv. 2
8A0          ?B=A   A       obj. niv. 2=chaîne?
D0           GOYES  OK2     oui--> on continue
3420200 L2   LCHEX  00202
6E8F         GOTO   ERR     --> Bad Argument Type
179      OK2 D1=D1+ 10      D1=^mantisse entier
143          A=DAT1 A       A=^du poke
8F8B050      GOSBVL #050B8  récupération D0 D1 B D
130          D0=A
143          A=DAT1 A
133          AD1EX
174          D1=D1+ 5       D1=^longueur chaîne
143          A=DAT1 A       A=longueur chaîne
174          D1=D1+ 5       D1=^mantisse chaîne
D8           B=A A
3450000      LCHEX  00005
E1       L4  B=B-C  A       Transfert
8A9          ?B=0   A
13           GOYES  L3
14B          A=DAT1 B
3103         LCHEX  30
B6A          A=A-C  B
3190         LCHEX  09
9EA          ?A<=C  B
90           GOYES  L5
3170         LCHEX  07
B6A          A=A-C  B
1580     L5  DAT0=A 1
160          D0=D0+ 1
171          D1=D1+ 2
3420000      LCHEX  00002
6DCF         GOTO   L4
8F8B050  L3  GOSBVL #050B8
142          A=DAT0 A       fin de routine
164          D0=D0+ 5
808C         PC=(A)
44670    FIN CON(5) #07644  DROP2
09F20        CON(5) #02F90  fin de structure
```

PEEK

```
Code     Lbl Instructions   Commentaires

76C20        CON(5) #02C67  début de structure
A63C0        CON(5) #0C36A  mise à jour Stacsize
69C20        CON(5) #02C96  programme LM
88100    DEB CON(5) FIN-DEB longueur code
8F18050      GOSBVL #05081  ]
1BF510C      D0=(5) #C015F  ]
D2           C=0    A       ]
144          DAT0=C A       ]
1B9F00C      D0=(5) #C00F9  ]
142          A=DAT0 A       ]          Voir les
34E0000      LCHEX  0000E   ]         commentaires
8BE          ?A>=C  A       ]             dans
91           GOYES  OK1     ]             PEEK
3410200      LCHEX  00201   ]
DA       ERR A=C    A       ]
8F8B050      GOSBVL #050B8  ]
8DA6930      GOVLNG #0396A  ]
1B6110C OK1  D0=(5) #C0116  ]
AF0          A=0    W       ]
15B9         A=DAT1 10      ]
1562         C=DAT0 XS      ]
A26          C=C+C  XS      ]
```

```
A26           C=C+C  XS      ]
5D0           GONC   L1      ]
1B6900C       D0=(5) #C0096  ]
158E          DAT0=A 15      ]
3407A20 L1    LCHEX  02A70   Vérification du type des
D5            B=C    A       arguments (deux entiers)
147           C=DAT1 A
137           CD1EX
143           A=DAT1 A
8A4           ?A#B   A
61            GOYES  L2
135           D1=C
174           D1=D1+ 5
147           C=DAT1 A
137           CD1EX
143           A=DAT1 A
8A0           ?B=A
D0            GOYES  OK2
3420200 L2    LCHEX  00202   Bad Argument Type
679F          GOTO   ERR
84A     OK2   ST=0    10      1er essai de réservation
179           D1=D1+ 10
143           A=DAT1 A       A=adresse du peek
100           R0=A
8F8B050       GOSBVL #050B8  récupération D0 D1 B D
147           C=DAT1 A
137           CD1EX
179           D1=D1+ 10
143           A=DAT1 A       A=longueur du peek
C4            A=A+A  A
34A0000       LCHEX  0000A
C2            C=A+C  A       C=place à réserver
109           R1=C
8F8B050       GOSBVL #050B8
119     L3    C=R1
8FEA350       GOSBVL #053AE  réservation place en RAM
532           GONC   OK3     se réussit --> OK3
8F8B050       GOSBVL #050B8  sinon...
86A           ?ST=0  10      premier essai ?
90            GOYES  RET     oui --> on essaye encore
8DE3930       GOVLNG #0393E  non --> Too Few Memory
8F49A40 RET   GOSBVL #04A94  garbage collector
85A           ST=1   10      second essai
65DF          GOTO   L3      on recommence
136     OK3   CDOEX
10A           R2=C           R2=^place réservée
134           D0=C
34E4A20       LCHEX  02A4E
144           DAT0=C A       écriture prologue chaîne
3450000       LCHEX  00005
164           D0=D0+ 5
111           A=R1
EA            A=A-C  A       A=longueur chaîne
140           DAT0=A A       écriture longueur
164           D0=D0+ 5       D0=^où écrire
D8            B=A    A
110           A=R0
131           D1=A           D1=^du peek
E1      L4    B=B-C  A
8A9           ?B=0   A
43            GOYES  L6
AE0           A=0 B
15B0          A=DAT1 1
3103          LCHEX  30
A6A           A=A+C  B
3193          LCHEX  39      Transfert
9EA           ?A<=C  B
90            GOYES  L5
3170          LCHEX  07
A6A           A=A+C  B
148     L5    DAT0=A B
161           D0=D0+ 2
170           D1+D1+ 1
3420000       LCHEX  00002
6ACF          GOTO   L4
8F8B050 L6    GOSBVL #050B8  récupération D0 D1 B D
174           D1=D1+ 5       on drope la longueur
E7            D=D+1  A
112           A=R2           on remplace l'adresse par
141           DAT1=A A       la chaîne
142           A=DAT0 A       fin de routine
164           D0=D0+ 5
808C          PC=(A)
09F20   FIN   CON(5) #02F90  fin de structure
```

Paul Courbis (392)

LE FACTEUR EST CONSTANT (ACTE II)

Une suite est donnée ici à l'article « Le facteur est constant » paru dans *JPC 54*.

L'un des défauts principaux du HP-28C est la taille restreinte de sa mémoire. Ce défaut ajouté à l'impossibilité de faire apparaître, aux libellés des menus, les signes ^, +, -, / et * avait conduit à utiliser les sigles les plus courts et les plus explicites possible. Au premier menu : X→ pour X^ - niveau 1(X) « à la puissance facteur constant » (à présent, au deuxième menu), A pour Addition, S pour Soustraction, D pour Division, M pour Multiplication. Au deuxième menu : R pour Reste, Q pour Quotient, →X pour ^X - facteur constant « à la puissance niveau 1 ».

Ces sigles sont relativement peu lisibles.

Le contenu des articles de Paul Courbis et de Sébastien Lalande (principalement dans *JPC 47* et *JPC 51*) permet à présent de pallier cet inconvénient et, grâce à l'assembleur, de faire apparaître aux menus les libellés suivants : X^, ^X, +, -, /, * et -HP, /HP. Pour ces deux derniers libellés (Soustraction HP, Division HP), voir les pages 5 et 10 de *JPC 54* et l'assertion de Wlodek Mier-Jedrzejowicz sur l'avantage de « soustraire d'une constante et de diviser une constante ».

Les fonctions - et / (du premier menu) sont placées respectivement au-dessus des fonctions -HP et /HP (du deuxième menu), les symboles - et / se trouvant au même emplacement de l'affichage (appuyer alternativement sur [USER] et sur [NEXT], pour le voir).

REMARQUES

1) Les libellés de menus créés par l'assembleur ne peuvent être purgés de la manière habituelle. Cependant, si par appuis successifs de MEM, on sature volontairement la mémoire, à un moment apparaît le message :

```
Out of memory
Purge
objet à purger
```

Lorsque le libellé de la « fonction à purger » se présente, appuyer sur YES : le libellé disparaît de l'affichage.

2) Sur l'ensemble des cinq menus, on ne peut obtenir, comme de coutume, un ordre différent des libellés par la commande ORDER.

3) On ne peut pas visiter le contenu des variables obtenues par l'assembleur.

Il serait certainement utile de supprimer ces inconvénients.

DESCRIPTION DES MENUS ET MODE D'EMPLOI

Le maniement du programme est plus aisé si l'on retrouve toujours les mêmes menus dans l'ordre indiqué ci-dessous (avec la petite modification apportée à l'article de *JPC 54* et que constitue l'interversion aux menus 1 et 2 des fonctions X^ et μ). Il est préférable, avant d'entrer les programmes, d'entrer les noms des variables (leur libellé ayant été écourté au maximum). Pour faire cela, stocker dans la pile sur vingt niveaux une valeur (1, par exemple) en tapant sur la ligne de commande, en mode alpha : `{ 20 } 1 CON ARRY→ DROP` puis [ENTER].

Activer le mode COMMAND. Invalider les modes LAST et UNDO. Activer le mode hexadécimal pour les opérations sur les entiers binaires (HEX). Entrer ensuite les noms des variables de droite à gauche et de bas en haut (du menu 5 au menu 1). Entrer le programme assembleur (merci encore à Sébastien Lalande) :

```
« "69A20" SWAP + "09F20" + → z « HEX "" 1 z SIZE
FOR X "#" z X DUP2 1 + DUP SUB ROT ROT DUP SUB +
+ STR→ B→R CHR + 2 STEP 'ASS■' STO ASS■ SIZE 2
* # 50000 SWAP - SYSEVAL LIST→ DROP 'ASS■' PURGE
» »
```

puis X [STO], W [STO], etc... E [STO], N [STO], '%Σ' [STO], etc... 'Σ' [STO].

Entrer dans la pile le programme →X (*JPC 54* page 10) :
« H ^ " ^" K ».
Se référant à Paul Courbis (*JPC 51*, page 15 au paragraphe « Global Name »), entrer dans la pile :
"21D2020E585" X [ENTER] [STO]
^X apparaît au menu.

Par la suite, pour entrer les programmes, utiliser au mieux la récupération des données à l'aide de [] [COMMAND].

Entrer de la même manière :
le programme Q
"21D2030F28405" X [ENTER] [STO]
le programme R
"21D2030D28405" X [ENTER] [STO]
le programme X→
"21D202085E5" X [ENTER] [STO]
?? [STO]
'I% [STO]
le programme M
"21D2030A20202" X [ENTER] [STO]
le programme D
"21D2030F20202" X [ENTER] [STO]
le programme S
"21D2030D20202" X [ENTER] [STO]
le programme A
"21D2030B20202" X [ENTER] [STO]
le programme ?
« → z « CLEAR z » » ? [STO]
μ [STO]

A présent que tous les libellés des menus ont été entrés, libérer de la mémoire en invalidant COMMAND et en stockant en X un nombre, 1 par exemple. La variable X est conservée au menu afin d'y pouvoir stocker éventuellement un objet sans voir les menus se décaler.

On pourra alors entrer les programmes dont on aura le plus besoin. Ils peuvent être tous présents en même temps dans la mémoire, mais, dans ce cas, le programme 'Σ' (portefeuille) ne pourra stocker que trois valeurs. Si l'on souhaitait en stocker un plus grand nombre, on pourrait, par exemple, vider de leur contenu les variables N et E et leurs routines de service Z, U, V et W.

Si toutes les routines sont entrées sur une HP-28C, version 1CC, MEM indique 212.

Attendons la venue du programme assembleur pour la HP-28S. On pourra alors, avec le programme 'Σ', stocker et étudier un portefeuille de plus de trois valeurs...

MENUS

```
Initialisations

          |___|  |_____________________|
1)    u   | ? |  |  +    -    /    *   | 'Foncteurs'
          |   |  |----|  |    |  |-----|  arithmé-
2)   I%   | ??|  | X^ |  -HP  /HP | ^X  |  tiques
          |___|  MarkUp
     |_____________________| |_________|  Taux nomi-
3)   |  Σ  | ■ |     C    %Σ| | N   E |  nal  <-->
     |_____|___|_____________| |_____|_|  taux ef-
   | P o r t e f e u i l l e |            fectif
     |_____|_____|_____|_____|

4)   G     H     I     J     K     L     Routines
                                          de
5)   Y     Z     U     V     W     X     service
```

EXPLICATION DES SYMBOLES

Par analogie avec le calculateur HP-41 : T est le contenu du niveau supérieur, X du niveau 1.

```
    Libellé    Opération      Sort réservé à X
    du menu    effectuée      ________________
    _______    _________

1er menu:
       +         "X+ "        lui Ajouter (Add)  T
       -         "X- "        lui Soustraire     T
       /         "X/ "        le  Diviser par    T
       *         "X* "        le  Multiplier par T

2ème menu:
       X^        "X^ "     l'élever à 'puissance T'
                              Sort réservé à T
                              ________________

       -HP       " -X"        Reste    de T moins X
       /HP       " /X"        Quotient de T par   X
       ^X        " ^X"     l'élever à 'puissance X'
```

EXEMPLE

A titre d'exercice, pour vérifier la bonne marche des fonctions arithmétiques, appuyer sur les touches indiquées, en examinant l'état de la pile après chaque appui de touche (en mode STD) :

```
Appuis de touches              Résultats en ligne 1
-----------------              --------------------
2 [?]
3 [+] [-] [/] [*]                        3
[NEXT]
[-HP][-HP]   [/HP][/HP]                  3
[^X]                                     8 (2^3)
3 [X^]                                   9 (3^2)
```

Philippe Heilbronn (233)

MATRICE DE FRACTION

Admettons qu'après un long calcul avec votre machine vous obteniez une matrice de réels dont les nombres ne vous disent rien du tout. Vous allez répondre : facile ! Je prends chaque réel et j'utilise le brillant programme →FRACTION qui prend un réel de la pile et renvoie au niveau 2 le numérateur et au niveau 1 le dénominateur.

Tout d'abord, voici le programme en question. Attention : il traite des approximations et risque donc de donner des résultats aberrants pour des valeurs inférieures à 10^{-4} ou supérieures à 10^{3}. Pour ces valeurs, ramenez-vous à un chiffre virgule les autres. Lancez le programme et arrangez la fraction ensuite, par exemple :

3333.333333
divisez par 1000
faites 3.333333333 →FRACTION
vous obtenez :

```
2:         10
1:          3
```

La fraction est 10/3*1000 soit 10000/3.

Voici le programme :

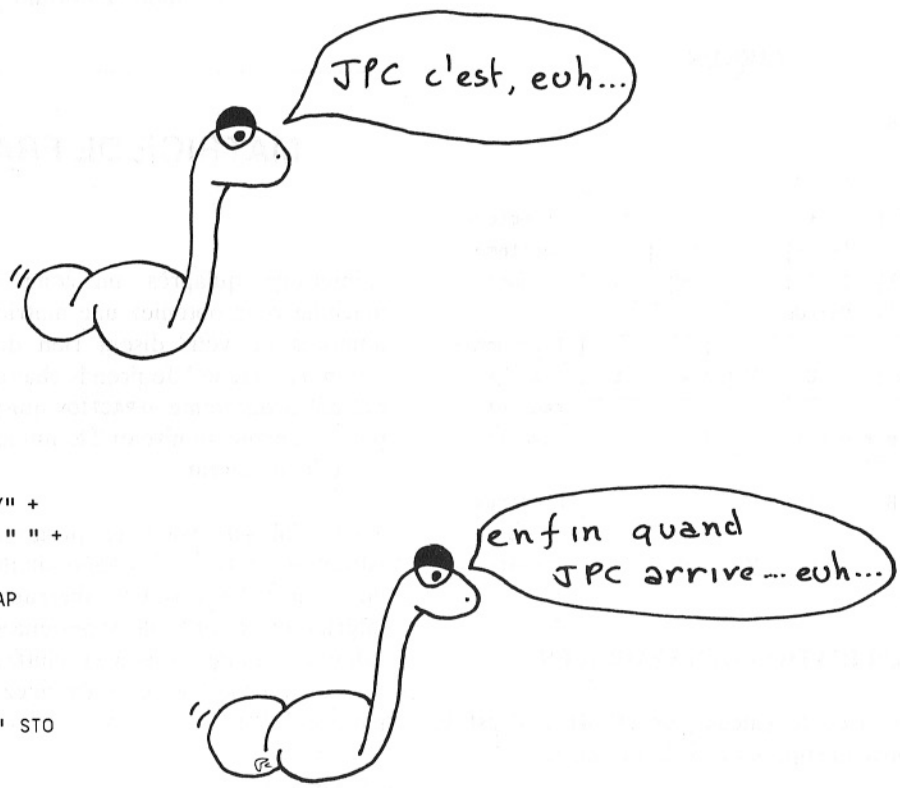
```
→FRACTION
«
  1E11 * 1E11 → n m
  « n ABS m
    DO
      SWAP OVER MOD
```

```
    UNTIL
      DUP 1000 / IP NOT
    END
    DROP
    n OVER / IP
    m ROT / IP
  »
»
```

Quant à notre matrice, si elle est 4×4 ce peut être long. Voici donc un programme qui le fait pour vous. Le résultat est affiché et stocké dans C. Placez une matrice de réels dans le niveau 1 de la pile et voici le programme pour la transformer :

```
M→FR :

«
  →STR "" SWAP
  DO
    DUP " " POS
    DUP2
    1 SWAP SUB
    3 ROLLD
    1 + 999 SUB
    SWAP
    IFERR
      STR→
    THEN
    ELSE
      →FRACTION
      SWAP →STR "/" +
      SWAP →STR + " " +
    END
    ROT SWAP + SWAP
  UNTIL
    DUP SIZE 2 ==
  END
  + DUP 1 DISP 'C' STO
»
```



Sébastien Lalande (442)



J.P.C. ce sont les vers qui en parlent le mieux

HP41

J. Hulaas

Multiplications de matrices *12*

MULTIPLICATIONS DE MATRICES

Le petit chef-d'oeuvre que j'ai l'honneur et l'avantage de vous présenter est le fruit d'une expérience inégalée (la mienne), de méditations approfondies ainsi que d'années de recherche intensive. Le logiciel en question représente en fait la quintessence de tous les travaux jamais effectués dans cette merveilleuse - mais ô combien ardue ! - discipline qu'est l'algèbre matricielle (Rien de tel qu'une petite boutade pour détendre l'atmosphère !).

Bien. Il est temps de passer aux choses sérieuses: ma création, soit dit en passant, est un véritable petit prodige de vélocité et de compacité. Le tout est matérialisé par la bagatelle de nonante-cinq pas ou instructions (quatre-vingt-quinze pour nos amis Français, que je profite de saluer au passage. Quelle satisfaction que d'écrire dans un journal a portée mondiale et universelle !). Vous conviendrez que c'est négligeable si l'on prend en considération la complexité de la tache effectuée : ce programme interactif sait en effet multiplier des matrices de n'importe quelle taille pourvu qu'elles soient mutipliables et qu'il y ait assez de mémoire à disposition pour les stocker. C'est quasiment de l'intelligence artificielle ! Pour plus d'efficacité, la matrice-résultat n'est pas conservée, mais des modifications mineures permettront d'y remédier pour ceux qui se sentiraient lésés.

Autre caractéristique : la robustesse et la transparence de l'interface utilisateur. Il vous apparaîtra que ce logiciel est totalement implantable si vous suivez à la lettre le mode d'emploi dont vous pourrez d'ailleurs rapidement vous passer avec un peu d'exercice.

Mode d'emploi

Admettons que vous désiriez effectuer le produit suivant :

```
                |2  0  0  0|
|1  2  3|  *    |0  2  0  0|
|4  5  6|       |0  0  2  0|
```

1- Avant de lancer le programme, il faut spécifier la taille des deux matrices. Quoi de plus approprié que la pile pour entrer quatre données numériques ? Vous faites donc [2] [ENTER] (la première matrice comporte deux lignes), puis [3] [ENTER] (nombre de colonne de cette matrice), ensuite [ENTER] (la deuxième matrice a trois lignes), et, enfin [4] [XEQ "MM"] (on lance le programme avec en X le nombre de colonnes de la deuxième matrice). Si ces données sont exactes, et qu'un sournois DATA ERROR vient s'afficher, c'est que les deux matrices ne sont pas mutipliables.

N.B. Ces valeurs doivent être des nombres entiers. Les dimensions réelles, négatives et complexes seront implémentées dans une version ultérieure du produit.

2- Le programme calcule la taille mémoire nécessaire. Si besoin est, il demande d'effectuer un SIZE.

3- A l'apparition du prompt OK, vous pouvez commencer à entrer la première matrice, ligne par ligne, donc [1] [R/S] [2] [R/S] [3] [R/S] ... jusqu'à l'affichage du message FIN. Commence alors la saisie de la deuxième matrice. Comme le programme ne modifie jamais la pile, vous pouvez l'utiliser pour effectuer toutes sortes de calculs, ou simplement pour stocker des valeurs qui apparaissent souvent. Saisie de la deuxième matrice: [2] [R/S] [0] 4*[R/S] [X<>Y] [R/S] [X<>Y] [R/S] ...

4- Affichage des composantes successives de la matrice résultat:

```
|2  4  6  0|
|8 10 12  0|
```

Appuyez sur [R/S] pour faire apparaître la composante suivante, jusqu'à affichage du message FIN (si vous faites [R/S] une fois de trop). Ici également, vous pouvez utiliser la pile pour effectuer des calculs à votre guise ; cela n'affectera en rien l'exécution du programme.

Mon exposé touche à sa fin ; je pense que le jury n'aura pas été insensible à la sincérité de mon plaidoyer et je ne puis donc que lui demander de se montrer indulgent envers les autres candidats du concours.

COMMENTAIRES

Ligne 1 : C'est bien choisi comme nom je trouve. C'est très parlant.

Ligne 2 : Permet d'utiliser la même routine de saisie pour les deux matrices.

Ligne 3 : Permet des VIEW AVIEW sans interrompre l'exécution.

Ligne 4-6 : Stockage du nombre de colonnes de la deuxième matrice.

Lignes 8-13 : Peut-on manipuler les deux matrices ? Si ce n'est pas le cas, affichage de DATA ERROR en deux octets par grâce à un subterfuge que je vous laisse apprécier. RTN en cas de SF 25.

Lignes 14-27 : Calcul du nombre de composantes des matrices et de la mémoire nécessaire a leur stockage.

Lignes 28-40 : Routine aimablement empruntée à HP (module MATH I); teste si le nombre de registre à disposition est suffisant. Sinon, on demande à l'utilisateur de reconfigurer la mémoire. Les instructions SIZE? et PSIZE permettent d'automatiser cette procédure, pour ceux qui possèdent une 41CX ou un module X-Functions.

Lignes 41-44 : Le prompt "OK" dans le registre X indique que la machine est prête, que l'utilisateur ait ou non effectué un SIZE. Le message FIN reste dans le registre ALPHA jusqu'à le fin du programme; des AVIEW indiqueront successivement le fin de la saisie de la première et de la seconde matrice, et l'affichage des résultats.

Lignes 44-55 : Saisie des matrices à multiplier: Cela peut sembler beaucoup, mais dans ce cas, c'est tout de même plus économique que de calculer les paramètres d'une boucle ISG simple. La pile n'est pas touchée, donc entièrement a la disposition de l'utilisateur pour effectuer des calculs pendant la saisie (ce qui n'est pas le cas des programmes du module MATH). LBL 05: instruction bidon necessaire après ISG.

Lignes 56-61 : Initialisation avant la multiplication. SF 21 pour imprimer les résultats. Le registre 04 doit être modifié plusieurs fois, dons on lui assigne une valeur interprétée rapidement et qui prend peu de place (CLX). De nouveau, on utilise un registre de plus que le minimum, mais c'est préférable au calcul des valeurs de départ, d'arrivée et d'incrémentation d'une boucle ISG simple.

Lignes 62-95 : Multiplication. Les lignes 70 à 81 implémentent le produit scalaire des deux vecteurs pointés par R03 et R05. Ainsi que vous pouvez le constater, le programme utilise les registres 00 à 08 comme pointeurs ou pour le stockage d'informations diverses. Les matrices sont placées séquentiellement ligne par ligne à partir de R09. Si vous désirez modifier le programme pour le rendre compatible avec le module MATH I, il y a donc suffisamment de marge, puisque ce dernier réquisitionne les 15 premiers registres pour son utilisation (cf mode d'emploi du module pages 56-58). N'oubliez pas que ce module ne traite que les matrices carrées.

```
01*LBL "MM"
SF 05  CF 21  STO 02  STO 04  STO 08  RDN  X=Y?
GTO 00  CLX  1/X  RTN

13*LBL 00
STO 01  STO 03  STO* 04  RCL Z  STO 00  STO* 03
RCL 04  8  STO 05  RCL 03  STO 06  +  STO 07  +
SF 25  ST IND X  FS?C 25  GTO 01  1  +  "ZE "
ARCL X  ASTO X  "SET SI"  ARCL X  AVIEW

40*LBL 01
"OK"  ASTO X  "FIN"

44*LBL 02
ISG 05

46*LBL 05
STOP  STO IND 05  DSE 06  GTO 02  AVIEW  RCL 04
STO 06  FS?C 05  GTO 02  SF 21  CLX  STO 04  9.9
STO 05  RCL 01

62*LBL 03
STO 06  ISG 04

65*LBL 05
RCL 04  RCL 07  +  STO 03  CLST  RCL 02

72*LBL 04
RCL IND 05  RCL IND 05  *  ST+ Z  ISG 05  RDN
ST+ 03  DSE 06  GTO 04  VIEW Y  RCL 01  STO- 05
DSE 08  GTO 03  RCL 02  X<> 08  STO 04  X<>Y
ST+ 05  DSE 00  GTO 03  AVIEW  END
```

J. Hulaas
PPC Lausanne





FORTH

X. Bille

Catalogue des mots (acte II) *16*

ASSEMBLEUR

J. Baudier
F. Vadez

L'assembleur du HP-71 (acte VII) *18*
Le temps enchaîné *21*

BASIC

S. Vaudenay
J.C. Kursner
A. Boldireff
M. Markov & E. Gengoux

La diagonale du Fou *24*
Directory *24*
Paginez vos lettres et textes *24*
Codes barres HP-41 sur HP-71 *26*

S. Vaudenay
J.C. Kursner
A. Boldireff
M. Markov & E. Gengoux

Programme "DIAG" *28*
Programme "DIRDOS" *28*
Programme "PAGI" *30*
Programme "BAR" *32*

LE COIN DES LHEX

35

CATALOGUE DES MOTS
(ACTE II)

Au mois de juin 1987 paraissait une série de programmes qui pouvaient décrypter le dictionnaire, c'est à dire en donner le contenu en séparant mots programmes et variables des mots vocabulaires. Cependant, il comportait un bug que je développerai plus loin. Alors voici la version 2.1, corrigée et améliorée car elle fournit l'indication de mot immédiat. D'autre part, j'ai découvert que les procédures pouvaient s'appliquer en lieu et place du mot VLIST (voir manuel). Je vous invite donc à reconsidérer l'ensemble.

LE BUG ET LE FLYTOX

Flytox est le nom d'un insecticide produit par l'Ethiopian Drug & Food Corporation ou quelqu'un dans le même genre. Avec l'ancienne version de CATALOG, vous avez peut-être remarqué que le programme affichait correctement les entrées du dictionnaire à deux lettres près. En effet, la première et la dernière lettre étaient remplacées par des caractères différents. Ce défaut disparaissait avec l'ordre Basic : CHARSET "". En fait le problème réside dans la structure d'un mot Forth.

Un mot est codé selon la trilogie LFA, NFA et CFA. Rappelons que le LFA (Link Field Address) est un pointeur qui relie le mot à celui qui le précède dans le vocabulaire d'attache (voir l'article de juin 87). Le contenu du LFA est le NFA du mot précédent. Le CFA (Code Field Address) contient l'adresse du mot qui assure l'exécution. L'illustration du propos est donnée par la structure : ... CREATE ... DOES> ... ; un mot qui aurait été défini par cette structure a un CFA dont le contenu pointe après le mot DOES> . Mais revenons au générateur de bug, je veux parler du NFA.

Le NFA (Name Field Adress) contient le nom du mot mais aussi quelques paramètres. Le premier octet du NFA est divisé comme suit :

```
1 | pr | sm | lon |
```

- *sm* signifie *smudge bit*. C'est un bit qui indique au système que le mot est bien défini. Il est d'ailleurs le point de départ de manipulations d'intérêt.
- *pr* signifie *precedence bit*, c'est lui qui donne le caractère immédiat d'un mot. Un mot est dit immédiat lorsque son action n'est pas modifiée par le mode compilation. Exemple :

```
: HOP 4 EMIT ; IMMEDIATE
```

L'action de IMMEDIATE est de rendre immédiat la dernière entrée dans le dictionnaire. Si on fait HOP, on a un α à l'écran.

```
: TOTO ." bonjour" HOP ;
```

Tiens, bizarre, le α apparaît. HOP s'est exécuté pendant la définition. De plus TOTO ne produit aucun α pendant son exécution. HOP est un mot immédiat.

- *lon* est la longueur du mot, codée sur 5 bits. Indépendamment de cela il existe une variable Forth qui définit la longueur maximale des mots : WIDTH.

Dans le NFA, les octets suivants contiennent le nom. Seulement le premier et le dernier caractère sont codés au delà de 127 en ASCII. Ce codage sert au système pour sa gestion des mots, en particulier le mot TRAVERSE s'en sert.

LE CATALOGUE SYSTEME

Les mots du système sont regroupés dans des tables situées en E0005. A cette adresse se trouvent 13 pointeurs dont le contenu est équivalent à celui de CONTEXT pour le dictionnaire FORTH. Donc partant des derniers NFA, il suffit de remonter la chaîne. C'est assez simple non ? Les mots sont classés par taille égale, un à douze caractères.

US ET COUTUMES

```
CATALOG ( -- )
```

Liste les mots du dictionnaire en commençant par les derniers définis. CATALOG ne remonte pas au delà du mot définissant le vocabulaire de contexte. Les touches flèches ([↑] et [↓]) permettent de se déplacer dans la chaîne. A tout instant, il est possible de sortir du programme en appuyant sur la touche [ON].
Codification :
i : indique la classe immédiate du mot,
* : indique que le mot est protégé par FENCE,
w : signifie simple mot (word),
vo. : indique que le mot représente un vocabulaire,
ct. : si le vocabulaire est celui de contexte,
cu. : si le vocabulaire est celui courant.

```
SYSCAT ( n -- )
```

permet de faire le catalogue des mots système dans l'une des 13 tables (0 < n < 12). SYSCAT s'utilise comme CATALOG.

```
MEM ( -- n )
```

ce mot donne la place disponible.

```
HEX

: PR-WORD ( nfa -- )
  DUP COUNT 1F AND
  ( affichage lettre par lettre du nom )
  2* OVER + SWAP DO
    I C@ 7F AND EMIT
  2 +LOOP
  SPACE
  ( test des mots protégés )
  DUP 5- FENCE @ <
  IF ." *" ELSE SPACE THEN
  ( test des mots immédiats )
  SPACE DUP C@ 40 AND
  IF ." i" ELSE SPACE THEN
  SPACE
  1 TRAVERSE 2+ DUP @ E7160 =
  ( décryptage d'un mot vocabulaire )
  IF ." vo: " 9 + DUP
    ( le tester par rapport au voc. courant )
    CONTEXT @ =
    IF ." ct. " ELSE 4 SPACES THEN
    CURRENT @ =
    IF ." cu." ELSE 3 SPACES THEN
  ELSE DROP ." w."
  THEN ;

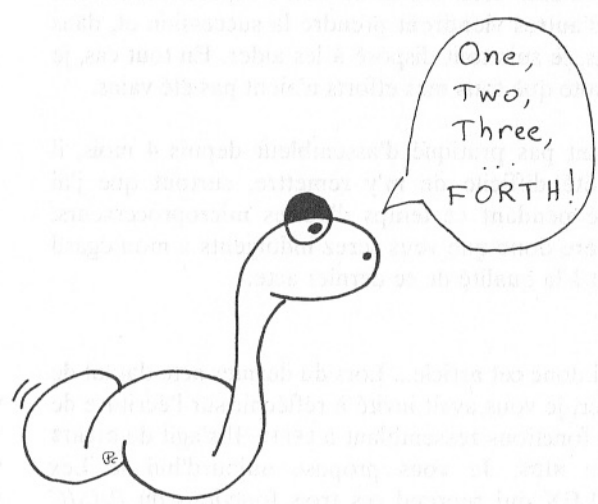

: CATALOG  ( -- )
  ( empêcher l'interruption ON )
  F 2F441 N!
  ( déplacer la pile )
  S0 >R SP@ SP0 !
  ( on remonte la chaîne à partir du dernier NFA )
  CONTEXT @ @
  BEGIN
    DUP PR-WORD
    KEY CASE
     E OF ( [ON] )
       SP! R> SP0 ! 0 2F441 !
     -1 ENDOF CR
     12 OF  ( [^] )
       CONTEXT @ OVER U<
      IF DUP 5- @ DUP 0=
       IF DROP THEN THEN
     0 ENDOF
     13 OF ( [v] )
      DEPTH 1 > IF DROP THEN
     0 ENDOF
     0 SWAP
    ENDCASE
  UNTIL ;


: MEM ( -- n )
  ( MEM donne le nombre de quartets disponibles )
  SP@ 1C1 - HERE - ;

: SYSCAT ( n -- )
  ( destiné à lister le système )
  ( on teste que 0 < n < 12 )
  DUP 0< NOT ABS *
  DUP B - DUP 0> * +
  ( on se positionne dans la table )
  5 * E0005 +
  ( on sauvegarde la variable CONTEXT )
  CONTEXT DUP @ ROT ROT !
  CATALOG CONTEXT ! ;

DECIMAL
```

Xavier Bille (203)



L'ASSEMBLEUR DU 71
(ACTE VII)

Chers amis, bonjour. Avant de commencer la septième partie de notre initiation à l'assembleur du HP-71, je voudrais préciser certaines choses.

Comme vous le savez depuis le début du mois de mars, je n'ai plus de HP-71 à ma disposition. Il m'était donc difficile de continuer cette série d'articles. Grâce à Jean-Jacques Dhénin qui m'a gentiment prêté un HP-71, j'ai pu finir cet article commencé en octobre dernier. Je tiens donc à remercier Jean-Jacques.

Nous sommes en mai au moment où je rédige cet article. Ce dernier achève la série des actes consacrés à l'écriture des fonctions. Il y a maintenant environ un an que cette série à débuté. A l'origine, mon intention était d'aborder, après l'étude des fonctions, celle des ordres ainsi que divers aspects du HP-71 et du module HP-IL.

N'ayant plus de HP-71 et pour l'instant pas l'intention de m'en procurer un autre, il m'est donc difficile de continuer. De plus, je dois quitter la France pour quelques mois.

Les choses n'iront donc pas plus loin. Vous l'aurez compris, cet article est le dernier de cette série jusqu'à nouvel ordre.

Depuis le début, j'avais espéré que d'autres viennent me soutenir dans ma démarche. J'espère maintenant que d'autres viendront prendre la succession et, dans ce cas, je suis tout disposé à les aider. En tout cas, je souhaite que tous mes efforts n'aient pas été vains.

N'ayant pas pratiqué d'assembleur depuis 4 mois, il m'a été difficile de m'y remettre, surtout que j'ai utilisé pendant ce temps d'autres microprocesseurs. J'espère donc que vous serez indulgents à mon égard quant à la qualité de ce dernier acte.

Voici donc cet article... Lors du dernier acte datant de février, je vous avait invité à réfléchir sur l'écriture de deux fonctions ressemblant à LEFT$. Il s'agit de RIGHT$ et de MID$. Je vous propose aujourd'hui le Lex MICLEX qui reprend ces trois fonctions du *BASIC Microsoft*.

J'ai modifié un peu le code de LEFT$ mais je pense que le dernier article et les commentaires suffiront pour comprendre son fonctionnement.

Je rappelle que RIGHT$(A$,n) renvoie les *n* caractères les plus à droite de A$. Comme le ferait A$[LEN(A$)-n]. La fonction MID$ prend 3 paramètres dont deux numériques. Ainsi MID$(A$,p,n) renvoie les *n* premiers caractères de A$[p]. De même on pourrait remplacer MID$(A$,p,n) par A$[p][1,n].

Pour compliquer les choses et pour rendre MID$ complètement compatible avec son homologue de *Microsoft,* le troisième paramètre est optionnel. Ainsi, lorsque *n* est absent, MID$(A$,p) renvoie la même chose que LEFT$(A$,p) ou encore que A$[p].

Comme d'habitude, plutôt qu'un long discours, je vous livre le Lex. Et quelques explications supplémentaires le suivent.

```
       LEX    'MICLEX'
       ID     #5C
       MSG    0
       POLL   0

       TITLE  Fonctions Microsoft
        *     J.B. janvier 1988 pour Acte VII
        *          avril   1988

POP1N  EQU    #0BD1C   dépile un nombre de la MS
FLTDH  EQU    #1B223   conversion en hexadécimal
POP1S  EQU    #0BD38   "dépile un chaîne"
ADHEAD EQU    #181B7   ajout de l'en-tête
ARGERR EQU    #0BF19   renvoie ERR:Invalid Arg

       ENTRY  LEFTe
       CHAR   #F       fonction
       ENTRY  MIDe
       CHAR   #F       fonction
       ENTRY  RIGHTe
       CHAR   #F       fonction

       KEY    'LEFT$'
       TOKEN  15
       KEY    'MID$'
       TOKEN  16
       KEY    'RIGHT$'
       TOKEN  17

       ENDTXT

       STITLE LEFT$
************************************************
************************************************
**                                            **
**  LEFT$ :                                   **
**                                            **
**    syntaxe : LEFT$(<chaîne>,<nombre>)      **
**                                            **
**    but     : renvoie les nombre caractères **
**              les plus à gauche de chaîne   **
```

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.

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)
```

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 17 septembre 1988
Samedi 1 octobre 1988
Samedi 5 novembre 1988
Samedi 3 décembre 1988
Samedi 7 janvier 1989
Samedi 4 février 1989
Samedi 4 mars 1989
Samedi 6 mai 1989
Samedi 3 juin 1989

Pierre David (37)

NOUS EN AVONS

La coopérative du Club vous propose :

- de **lecteurs de cartes** magnétiques pour HP-71, neufs, dans leur boîte d'origine, avec 5 cartes magnétiques, pour 500 F (port compris),
- des **anciens numéros** de JPC, au prix de 40 F + 7,40 F de frais d'affranchissement,
- d'une **année complète** de numéros de JPC (février à janvier) pour 300 F (offre spéciale) port compris,
- des **I.D.S.** du module Forth / Assembleur (listing interne commenté par HP) pour 250 F (port compris),
- des **VASM** pour HP-41 (listings des Roms internes commenté par HP) pour 300 F (port compris),
- de **manuels de service** du HP-41 au prix de 75 F (port compris),
- de **manuels de service** du HP-75 au prix de 75 F (port compris).

En outre, le module **JPC Rom** pour HP-71 est disponible. Vous nous adressez votre Eprom CMT (de préférence 64 Ko), et nous la programmons suivant une des options ci-dessous :
- JPC Rom + Manuel, pour 600 F,
- JPC Rom + Manuel + vos propres programmes, pour 800 F.

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

VOUS EN VOULEZ

Nom :
Prénom :
No de membre :
Adresse :

Commande :

	Qté	Prix Unitaire	Prix Total
lecteur de cartes pour HP-71	x	500 FF	
anciens numéros de JPC	x	47,40 FF	
année complète de JPC	x	300 FF	
I.D.S. du module Forth	x	250 FF	
V.A.S.M.	x	300 FF	
Manuel de service pour HP-41	x	75 FF	
Manuel de service pour HP-75	x	75 FF	
JPC Rom + Manuel	x	600 FF	
JPC Rom + Manuel + vos propres programmes	x	800 FF	
		Total	FF

Préciser éventuellement les
numéros de JPC commandés :

PPC PARIS

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

BULLETIN D'ADHESION

Nom |_|_|_|

```
**                                                **
************************************************
************************************************

       NIBHEX 8422     Syntaxe : LEFT$(A$,n)
LEFTe  GOSUB  getN&S

* ici : R0 = n
*       D1 ^ fin de A$
*       A[A] = LEN(A$)
* on fait R1 = D1 + A[A]

       CD1EX
       C=C+A  A         A ^ début de A$
       R1=C             R1 aussi

       GOSUB  getmin    renvoie MIN(R0,A[A])

* ici : A[A] = MIN(n,LEN(A$))
* on calcule alors R1 - A[A]

       C=R1
       C=C-A  A
       D1=C

* ici : D1 ^ fin de chaîne à renvoyer
*       R1 ^ début de A$

adhead ST=0   0         pour ADHEAD
       GOVLNG ADHEAD

       STITLE MID$
************************************************
************************************************
**                                            **
**  MID$ :                                    **
**                                            **
**    syntaxe : MID$(<chaîne>,<pos>[,<nombre>]) **
**                                            **
**    but     : renvoie au plus nombre        **
**              caractères de chaîne à partir **
**              du p-ième compris             **
**                                            **
************************************************
************************************************

       NIBHEX 88423    Syntaxe : MID$(A$,p[,n])
MIDe   CPEX   15        P = nombre de paramètres

* on utilise ST[1] pour coder le nombre de
* paramètres de la façon suivante :
* codage : ST[1] = 0 indique 3 paramètres
*          ST[1] = 1 indique 2 paramètres

       ST=1   1         on suppose qu'on a 2 param.
       ?P=    2         est-ce vrai ?
       GOYES  para2
       ST=0   1         sinon ST[1]=0 pour 3 param.

para3  GOSBVL POP1N     on récupère n
       GOC    argerr    vérifie si n est complexe
       D1=D1+ 16        pointe l'objet suivant : p
       GOSBVL FLTDH     A[W] --> A[A] en hexa
       GONC   argerr    si out of range
       A=A+A  A         multiplication par 2
       R2=A             sauvegarde

para2  GOSUB   getN&S

* ici : R0 = p
*       R1 ^ début de A$ (A[A] aussi)
*       A[A] = LEN(A$)
* on fait R1 = D1+LEN(A$)-p+2
* en vérifiant au passage que p n'est pas nul

       CD1EX
       B=C    A         sauvegarde de D1
       D1=C             on remet D1 à sa place
       C=C+A  A         C = D1+LEN(A$)
       A=R0             A[A] = p
       ?A=0   A         si p = 0
       GOYES  argerr    alors erreur

       C=C-A  A         C = D1+LEN(A$)-p
       C=C+1  A         C = D1+LEN(A$)-p+1
       C=C+1  A         C = D1+LEN(A$)-p+2

* ici : C[A]=D1+LEN(A$)-p+2
* on vérifie que C n'est pas remonté au dessus
* de D1 en utilisant gtmin2

       A=B    A         A ^ fin de A$
       GOSUB  gtmin2
       R1=C             R1=min(D1,D1+LEN(A$)-p+2)

* R1 est donc bien placé en début de la chaîne
* à renvoyer et D1 en fin. On appelle alors ADHEAD
* si MID$ à été appelé avec deux paramètres.

       ?ST=1  1         2 paramètres ?
       GOYES  adhead

* sinon on ajuste D1 en fonction de p
* ici : C ^ toujours le début de la chaîne
* on fait C = C-p

       A=R2             on récupère p
       C=C-A  A         C = C-R2

* on vérifie également que l'on remonte pas au
* dessus de la fin de la chaîne. On utilise encore
* gtmin2 et on met dans D1 le maximum de D1 et C.

       A=B    A         A ^ fin de chaîne
       GOSUB  gtmin2
       D1=C             C ^ fin de la chaîne à
        *               renvoyer
```

```
       GOTO   adhead
argerr GOVLNG ARGERR

       STITLE RIGHT$
************************************************
************************************************
**                                            **
**  RIGHT$ :                                  **
**                                            **
**    syntaxe : RIGHT$(<chaîne>,<nombre>)     **
**                                            **
**    but     : renvoie les nombre caractères **
**              les plus à droite de chaîne   **
**                                            **
************************************************
************************************************

       NIBHEX 8422     Syntaxe : RIGHT$(A$,n)
RIGHTe GOSUB  getN&S

* ici : R0 = n
*       D1 ^ fin de A$
*       A[A] = LEN(A$)
* D1 est donc déjà placé pour ADHEAD
* on fait R1 = D1 + MIN(n,LEN(A$))

       GOSUB  getmin

* ici : A[A] = MIN(n,LEN(A$))
* on calcule alors D1 + A[A]

       CD1EX
       A=A+C  A
       R1=A            R1=MIN(n,LEN(A$))+D1
       CD1EX

* ici : D1 ^ fin de A$
*       R1 ^ début de la chaîne à renvoyer

       GOTO   adhead

       STITLE Divers
************************************************
************************************************
** Nom : getN&S
**
** Catégorie : Math Stack Utilities
**
** But :
**      dépile un nombre (n) et une chaîne (A$)
**      de la Math Stack
**
** Entrée :
**      D1   = haut de la Math Stack
**
** Sortie :
**      D1   = fin de la chaîne
**      A[A] = LEN(A$)
**      R0   = 2.n
**
** Appels : POP1N, FLTDH, POP1S, argerr
**
** Utilise : A,B,C,D1,R0,XM
**
** Histoire :
**
**    Date      Programmeur      Modification
**  --------    -----------   ------------------
**  12/01/88    J.B.          Conception et codage
**
************************************************
************************************************

getN&S GOSBVL POP1N    A[W] = n
       GOC    argerr   si complexe : erreur
       D1=D1+  16      pointe A$
       GOSBVL FLTDH    A[W] --> A[A] en hexa
       GONC   argerr   si out of range ou NaN
       A=A+A  A        multiplication par 2
       R0=A            sauvegarde

       GOSBVL POP1S    A[A] = LEN(A$),D1 ^ chaîne

       RTN

************************************************
************************************************
** Nom : getmin
**       gtmin2
**
** Catégorie :
**
** But :
**      getmin : renvoie A[A] = min(R0,A[A])
**      gtmin2 : renvoie A[A] = min(C[A],A[A])
**
** Entrée :
**      getmin : données dans R0 et A[A]
**      gtmin2 : données dans C[A] et A[A]
**
** Sortie :
**      A[A] = plus petite des deux données
**      C[A] = plus grande des deux données
**
** Appels : aucun
**
** Utilise : getmin R0,C
**           gtmin2 A,C
**
** Histoire :
**
**    Date      Programmeur      Modification
**  --------    -----------   ------------------
**  12/01/88    J.B.          Conception et codage
**  29/04/88    J.B.          Commentaires
```

```
**
************************************************
************************************************

getmin C=R0             on récupère n
gtmin2 ?A<C   A         LEN(A$) < n ?
       RTNYES           si oui on laisse

       ACEX   A         si non on échange
       RTN

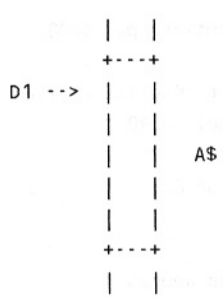
       END
```

Comme je l'ai déjà dit, le code de LEFT$ a été légèrement modifié, mais rien de bien compliqué. La sous-routine getN&S sert à dépiler une valeur numérique de la pile, à la convertir et à la sauvegarder avant d'appeler POP1S qui se charge de la chaîne à traiter.

Le code de RIGHT$ est identique à celui de LEFT$ pour le début. Les choses varient pour MID$:

En effet, dans ce cas, il faut gérer la présence ou non d'un troisième paramètre. Je rappelle que, lors de l'entrée dans un Lex, C[S] contient le nombre de paramètres. On utilise alors le registre P comme intermédiaire et on dépile la valeur de n si besoin est à partir de l'étiquette para3 et on la sauvegarde dans R2.

Ensuite, et pour les trois fonctions, une fois getN&S appelé on se retrouve avec la pile dans cette configuration :



Et A[A] contient la longueur de A$. Il faut donc positionner correctement D1 et R1 avant d'appeler ADHEAD qui terminera notre code. Il faut placer R1 au début de la chaîne à renvoyer et D1 en fin.

Dans le cas de LEFT$, il faut placer R1 en début de A$ (R1 = D1+A[A]) et D1 en R1-n. Pour RIGHT$, il faut mettre R1 en D1+LEN(A$)-n et on ne touche pas à D1.

Enfin, dans le cas de MID$, on place R1 comme pour RIGHT$ et D1 en fonction de valeur du troisième paramètre.

Dans tous les cas, lors du calcul d'un nouveau pointeur, il faut s'assurer que l'on pointe toujours sur des caractères de A$ et qu'en particulier on ne remonte pas au dessus de la valeur de D1 du schéma. En effet LEFT$('PPC',45) ne doit renvoyer que trois caractères et pas plus.

Cette vérification est l'objet de la routine getmin que l'on utilise dans les trois fonctions.

Comme pour LOCK$, on quitte donc le Lex en renvoyant le résultat par ADHEAD.

Les routines HP utilisées dans ce Lex ont déjà été vues lors d'articles précédents. Vous ne devriez donc avoir aucune difficulté à comprendre le fonctionnement. Je vous renvoie à la bibliographie des actes précédents pour plus d'informations.

Voici donc qui termine cette série d'articles consacrée à l'assembleur de notre chère machine. Je tiens à dire que quel qu'ait été l'impact de ces articles auprès des membres du Club, cette expérience aura été pour moi enrichissante.

Si je n'ai plus de HP-71, j'ai toujours ma HP-41. Je reste donc inscrit au Club et en contact avec mes amis membres avec qui j'apprend beaucoup, même si maintenant je vais prendre mes distances avec PPC Paris pendant quelques temps.

A tous merci... et au revoir,

Jacques Baudier (192)

LE TEMPS ENCHAINE

Je profite d'un petit moment de tranquillité pour prendre le temps de vous écrire. Je suis, en général, trop pris par mes études ou mon club de natation pour le faire, mais, en ce moment, mes études me laissent un peu de répit.

Je tiens à vous remercier de m'avoir si vite envoyé l'assembleur Areuh. Il est fantastique, et j'en ferais la publicité. Grâce à lui, j'ai repris un vieux Lex que j'avais écrit en pokant dans un fichier texte. Ce Lex implémente une fonction que j'ai connue sur la HP-41 (que de souvenirs, sniff...), mais que je n'ai jamais vue sur le HP-71.

Il s'agit de la fonction HMS$ qui transforme un réel représentant un nombre de secondes (comme TIME par exemple) en chaîne au format :

HH:MM'SS"CC

En cas de donnée erronnée, la fonction rend la chaîne :

" -??-"

J'ai placé des blancs au début de cette chaîne, car dans mes applications, j'utilise rarement les heures et j'écris souvent HMS$(X)[4] pour les supprimer.

Je vous envoie ce petit Lex, et espère qu'il sera le premier d'une longue série. J'ai en préparation la suite logique de cette fonction : un chronomètre dans le plus pur style HP-41, mais pour l'instant sans stockage des temps intermédiaires.

Assembleurement vôtre,

Frédéric Vadez (79)

```
        LEX     'VADLEX'
        ID      #5D

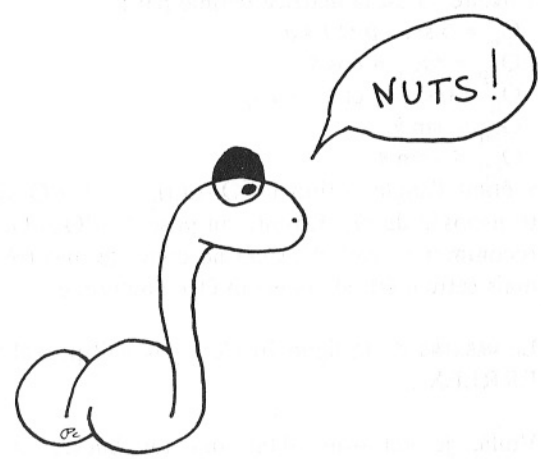
        MSG     0
        POLL    0
EXPR    EQU     #0F23C
STRHDR  EQU     #0F09A

        ENTRY   xHMS
        CHAR    #F
        KEY     'HMS$'
        TOKEN   1
        ENDTXT

        NIBHEX 811
xHMS    A=DAT1 W        Lecture opérande dans A
        ST=0   0        Initialisation du marqueur
          *             de nombre négatif
        GOSUB  FORMAT   Vérifie format de l'opérande
        GOSUB  TRANSF   Effectue la transformation
          *             proprement dite
SORTIE  GOVLNG EXPR     Termine la fonction
*
* Si opérande hors norme alors rendre "-??-"
*
PASBON  SETHEX
        P=     0        Préparation d'un en-tête
        C=0    W        de chaîne alpha
        LC(2)  #10      de la longueur
        D1=D1+ 16       voulue pour placer
        GOSBVL STRHDR   le message réponse:
        LCASC  '-??-'
        DAT1=C W        Ecrit message dans la pile
        C=RSTK          on saute un RTN
        D1=D1- 16       on pointe le haut de pile
        GOTO   SORTIE   et on sort
* Formattage de l'opérande et éjection des données
* invalides: INF, NaN et hors normes donnent "-??-"
**********************************************
FORMAT  P=     2        vérification
        LC(1)  #F       NaN et INF
        ?A=C   P        d'après l'octet #2
        GOYES  PASBON   éjecte les NaN et INF
        SETDEC
        A=A+1  X        Multiplication par 100
        A=A+1  X        pour avoir des centièmes de
        P=     0        seconde
        ?A=0   S        Détection des positifs
        GOYES  POSITI   Si oui on saute à la suite,
          *             sinon...
        ST=1   0        Caractérise un opérande
          *             négatif
        A=0    S        annule le signe pour la
          *             routine DIV
POSITI  LC(3)  #006     éjection des temps > 10^4
        ?C<A   X
        GOYES  PASBON
        LC(2)  #13      Annulation des chiffres
          *             après la virgule
        C=C-A  X        C(X)= nbre de 0 à placer
        P=C    0
        A=0    WP       annulation de ces chiffres
          *             en trop
BCLE    ASR    W        puis décalage du nombre
        C=C-1  X        qui ressortira cadré à droite
        GONC   BCLE
        RTN             terminé pour le formattage
*
* transformation proprement dite
*
TRANSF  P=     4        calcul des heures
        C=0    W        par
        LC(2)  #36      division entière par 3600
        GOSUB  DIV
        BCEX   A        DIV rend le résultat en B(A)
        R0=C            et on le met en R0
        C=0    W
        P=     3        division par 60
        LC(1)  #6
        GOSUB  DIV
        C=R0            lecture des heures
        CSL    W        décalage d'un
        CSL    W        octet
        BCEX   B        placer les minutes après
        CSL    W        puis nouveau décalage pour
        CSL    W        préparer place des secondes
        CSL    W        et des
        CSL    W        centièmes
        P=     3
        ACEX   WP       Ce que l'on fait ici
        R0=C            Mise à l'abri de HHMMSSCC
        SETHEX          dans R0
        P=     0
        C=0    W
```

```
       LC(2)  #16      Longueur de l'en-tête de la
         *             chaîne à sortir = 22
       ?ST=0  0        Nombre positif?
       GOYES  SUITE0
       LC(2)  #18      Non lg de chaîne = 24 nibs
SUITE0 D1=D1+ 16       On écrase l'opérande
       GOSBVL STRHDR   Création de l'en-tête
       GOSUB  NCHR     Conversions ASCII des
         *             centièmes de seconde
       LCASC  '"'      Marque les secondes
       DAT1=C B        Ajout à la chaîne en
         *             construction
       D1=D1+ 2        Pointage sur la suite pour...
       GOSUB  NCHR     conversion ASCII des secondes
       LC(2)  #27      le caract. ' marque les min.
       DAT1=C B        Ajout à la chaîne
       D1=D1+ 2        pointage sur la suite pour...
       GOSUB  NCHR     Conversion ASCII des minutes
       LCASC  ':'      Suivies du marqueur heures
       DAT1=C B        Ajouté à la chaîne
       D1=D1+ 2        On pointe la suite pour...
       GOSUB  NCHR     Conversion ASCII des heures
       ?ST=0  0        Nombre positif?
       GOYES  SUITE1
       LCASC  '-'      Non, on rajoute un '-' devant
       DAT1=C B        la chaîne
SUITE1 D1=D1- 6        puis on
       D1=D1- 16       repointe le sommet
       D1=D1- 16       de la pile
       RTN             et c'est terminé
*
* Sous-routine de division entière par soustractions
* successives.
* nombre formatté en A, diviseur formatté en C
* résultat en B(B)
*
DIV    B=0    W
DIV0   ?A<C   W
       RTNYES
       A=A-C  W
       B=B+1  B
       GOTO   DIV0
*
* Prend un nombre de 2 chiffres en R0, décale R0, et
* range les codes ASCII des chiffres à l'adresse D1
*
NCHR   A=0    A
       C=R0
       A=C    B        000XY en A(A)
       CSR    W        décalage pour éliminer
       CSR    W        ce qu'on vient de lire
       R0=C
       ASL    A        000XY devient 00XY0
       ASR    B        puis 00X0Y
       LC(4)  #3030
       A=A+C  A        ?3X3Y (on se fiche de A[4])
       DAT1=A 4        ce qu'on écrit dans la chaîne
       D1=D1+ 4        avant de pointer la suite
       RTN             et de retourner

END
```



LA DIAGONALE DU FOU

Il m'est arrivé une chose étrange : j'ai vu la lumière du jour. Je me suis aperçu que j'avais un module Maths sur mon HP-71, et que celui-ci pouvait travailler sur des matrices. Donc, après une « pale » (les initiés savent ce que c'est) d'Ulm 79, j'ai entrepris de faire un programme appliquant bêtement les résultats du problème.

Pour ceux que cela intéresse, je signale que cette méthode est connue sous le nom de *Méthode de Jacobi*.

Ce programme consiste à diagonaliser une matrice symétrique réelle (comme chacun sait, toutes ces matrices sont diagonalisables). En prime, le programme fournit une base orthonormale de vecteurs propres.

L'exécution effectue un MAT INPUT, et donne le résultat avec MAT DISP. Je rappelle l'algorithme à ceux qui ont oublié le problème de 79 : si A est symétrique réelle, on pose p et q tels que :

$$|A_{pq}| = \max \{ |A_{ij}| ; i \neq j \} \text{ et } p \neq q$$

Ensuite, O est la matrice définie par :
$O_{ii} = 1$ si $i \neq p$ et $i \neq q$
$O_{pp} = O_{qq} = \cos \theta$
$O_{ij} = 0$ si $i \neq j$ et $i, j \neq p, q$
$O_{pq} = \sin \theta$
$O_{qp} = - \sin \theta$
θ étant l'angle vérifiant $({}^tO.A.O)_{pq} = 0$ (^tO signifie transposée de O). Ensuite on pose $A' = {}^tO.A.O$, et on recommence avec A'. Cela ne converge pas très vite, mais cette méthode pourrait être améliorée.

Le WARNING de la ligne 70 n'est pas indispensable (Cf ERRLEX).

Voilà, je retourne dans ma taupinière, et à la prochaine.

Serge Vaudenay (124)

DIRECTORY

Voici un petit programme de débutant écrit pour la rubrique du même nom. Je suis particulièrement heureux d'y utiliser JPC Rom.

Les fonctions CAT et CAT$ ne permettent un catalogue que par *nom de fichier*. J'ai éprouvé le besoin de cataloguer par nom abrégé, nom partiel, type ou année. Ce que permet l'utilitaire ci-joint. Il est simple d'utilisation :

If faut donner le nom du périphérique, puis le critère *nom*, *type* ou *année* :

- *nom* : les premiers caractères suffisent (par exemple, ME listera MENU, MESURES, etc.). Un * comme premier caractère listera un groupe (par exemple, *UTIL listera TEXTUTIL, LEXUTIL2...).

- *type* : choisir dans le menu. Si le critère est exclusif et LEX est choisi, les LEX seront listés sinon, les D-LEX le seront simultanément.

- *année* : introduire le millésime (8) ou l'année (88).

Les résultats sortent sur les appareils 2 et 3 (Thinkjet et moniteur) de ma boucle HP-IL.

Remarques et suggestions sont les bienvenues.

Jean-Claude Kursner (444)

PAGINEZ VOS LETTRES ET TEXTES

Vous possédez le module *JPC Rom* ? Félicitations ! Pour les indécis, les quelques possibilités que j'ai utilisées dans le programme PAGITEXT, peuvent peut-être contribuer à votre décision d'acquérir ce merveilleux outil.
J'en serais personnellement très heureux.

Les réfractaires à KEYWAIT$ peuvent remplacer cet ordre par KEY$.

PAGINATION DES TEXTES

Vos mémoires, rapports ou lettres peuvent utiliser les possibilités offertes par EDTEXT pour centrer, justifier, marger, paginer, etc... C'est très long !

Par exemple, comparez le temps de formattage d'une de vos proses par EDTEXT et par FORMAT. Sans commentaire n'est-ce pas ?

Comme beaucoup d'entre-vous, j'utilise donc FORMAT pour mes lettres. De plus un truc utile, qui se trouve dans *JPC Rom*, pour décaler la marge de nos textes, au préalable justifiés par FORMAT, avec 67 caractères, par exemple, est d'employer (sous EDTEXT) la tabulation assembleur de *JPC Rom*, dont on utilise le réglage à la colonne 9 pour chaque ligne :

- le mode tabulation est actif (appui de la touche [CALC] sous EDTEXT),
- [I/R] : le curseur devient curseur d'insertion,
- [SPC] : l'on insère 8 blancs en début de ligne automatiquement,
- [ENDLINE] : l'on valide la ligne puis passage à la ligne suivante, etc...

Cela peut paraître long, mais ça l'est beaucoup moins que d'attendre les instructions de formattage du traitement de textes, surtout si vous avez plusieurs copies à imprimer.

Vos lettres seront ainsi présentées avec une marge de 8 blancs + 67 caractères de votre prose + 5 blancs, soit 80 caractères par ligne, mais cela peut être 8 blancs + 72 caractères, ou toute autre présentation de votre choix. Personnellement, je sépare mes textes de l'en-tête par 6 lignes vides, de même que mon nom de la fin du texte.

GENERALITES

Au début du programme, il y a une petite attente due au chargement des variables du menu de départ. Vous lirez donc pendant quelques secondes, sur votre fenêtre d'affichage :
Pagination des textes

Menu pour l'imprimante

J'ai utilisé l'ordre FINPUT plutôt que MENU, car, dans le message, sont introduites les flèches vers le haut et vers le bas (caractères 162 et 163), pour indiquer le sens de déroulement du menu, en appuyant sur les touches habituelles de déroulement des lignes. Avec MENU, je n'y suis pas parvenu. Jean-Jacques Dhénin nous indiquera sans doute le moyen de le faire.

La première ligne du menu explique qu'il faut se déplacer vers le bas pour faire un choix et appuyer sur [RUN] pour valider. Si, par accident, vous appuyez à ce moment pour valider, comme il s'agit d'un message, il ne se passera rien. Donc appuyez sur la touche 51.

La seconde ligne vous demande si vous voulez imprimer 66 lignes (soit 6 lignes au pouce), sur une feuille de 12 pouces.

La troisième propose le dernier choix, 88 lignes imprimées sur une feuille de 12 pouces. Pour les autres formats de papier (11 pouces) ou tailles logiques, il importe de modifier le programme.

Les flèches vers le haut, vous rappellent que vous pouvez remonter à la ligne précédente (voire dans d'autres applications avec de longs menus, monter, descendre ligne par ligne, revenir au début ou à la fin comme avec MENU), tant que vous n'avez pas validé votre choix avec [RUN].

VOS LETTRES

Logo

Vous avez vraisemblablement un logo ou tête de lettre, à vos noms et adresses. Dans la négative je présente deux exemples sous forme de sous-programmes qui peuvent être appelés par le programme : LOGOPPC et LOGOEXMPL.
Attention : si votre propre logo n'a que trois lignes, pensez à conserver le FLAG(0), ou bien sûr, le supprimer si quatre lignes vous sont utiles.

Date

S'il ne s'agit pas d'une lettre votre document n'a peut-être pas besoin d'être daté. Répondez comme il convient au test O/N.
Par contre, une lettre est, naturellement, datée. Cependant, même si votre hiérarchie peut apprécier que vous emportiez du courrier à faire au cours de vos week-end, il est préférable de dater vos missives du prochain jour ouvrable. C'est pourquoi un test O/N vous est proposé pour dater le jour courant. Dans la négative vous entrez par FINPUT la date choisie.
Nota : la barre oblique doit être insérée après l'entrée du jour et du mois. En effet il est possible avec FINPUT de protéger des champs. Dans le cas qui nous préoccupe, les barres obliques seront imprimées donc elles seront à inclure dans la variable alphanumérique. Vous pouvez faire la modification si tel est votre bon plaisir.

Références

Il est possible d'inclure les deux lignes habituelles :
- V/Ref :
- N/Ref :
Un test O/N vous le permet si vous en avez l'utilité. L'entrée s'effectuera de suite après l'entrée des coordonnées de vos destinataires.

Destinataire(s)

Un même texte, tel qu'une circulaire, peut être adressé à plusieurs destinataires, il suffit d'en indiquer le nombre.

Le programme prend en compte quatre lignes au maximum par destinataire. Si trois sont suffisantes, deux possibilités s'offrent à vous :

- sauter "Lieu-dit éventuellement",
- y entrer le code postal et la ville puis sauter la question suivante.

La comptabilité du nombre de lignes utilisées s'effectuera automatiquement lors de l' impression.
Rappel : la validation des entrées multilignes par FINPUT, s'effectue par [ENDLINE] sauf pour la dernière par [RUN]. Si vous l'avez oublié, la dernière entrée réapparaît à l'écran avec son message, il suffit d'appuyer sur la touche [RUN]. Avant de le faire, vous pouvez vérifier vos entrées en remontant les lignes, que vous pouvez inspecter en long, en large, j'allais dire en travers, puis valider le tout en sortant de FINPUT par [RUN].

Texte

Entrez le nom de votre fichier Text. Puis le nombre de copies à réaliser.

Pagination

Avez-vous remarqué que dans certains cas votre lettre est trop longue pour contenir dans une page, mais que seule la signature, ou, au mieux, une ligne de la formule de politesse + quelques lignes vides + votre nom figurent dans la seconde (ou dernière) page. C'est moche !

Pour éviter ces désagréments, le programme évalue ce qu'il reste comme lignes pour la dernière page. S'il s'avère que s'est insuffisant, quelques lignes sont supprimées de la première page et reportées quantitativement dans la dernière.

La pagination n'est effective que si votre lettre comporte plus d'une page. Vous trouverez cette pagination au bas de chaque page, après un saut de quelques lignes (4 en principe sauf cas exprimé précédemment et la dernière page rarement complète), sous la forme : np/tp avec :
- np = numéro de la page
- tp = total des pages

Si vous avez plus de 99 pages, une petite modification du programme s'imposera à la ligne 890.

MEMOIRES, RAPPORTS, etc...

Vous avez les mêmes possibilités que pour vos lettres, mais en répondant NON à toutes les questions, vos proses seront imprimées telles quelles, après un saut de quelques lignes.
Nota important : a la question :
"Nb. de destinataires: ?"
répondez par le chiffre zéro si vous ne désirez pas imprimer de pavé(s) "DESTINATAIRES".

Ouf ! Je termine cette longue mais indispensable explication en remerciant Pierre David. En moins de temps qu'il me faut pour vous le dire, il m'a sorti du mauvais pas que j'ai rencontré avec les boucles et EXECUTE. Voyez le programme ! Enfin, le même Pierre m'a répété à maintes reprises que je devais faire des organigrammes pour programmer. Je n'arrive pas à m'y contraindre. Cela donne à mes "oeuvres", que je n'ose qualifier de "chef-", une impression b...que ! Mais comme l'a déjà dit Galilée avant moi, elle tourne...

Alexandre Boldireff (254)

IMPRIMEZ VOS CODES BARRES HP-41 AVEC LE 71 !

Le programme que nous vous proposons ici a été initialement écrit pour HP-75 et pouvait piloter au choix une imprimante ThinkJet ou Epson (voir *CHHU Chronicle* V3N1 de janvier 86). Bien sûr, j'en connais qui vont dire : « ça y est, on va encore nous parler du 75 ! ».

Eh bien, non ! L'idée naturelle était de le faire migrer sur HP-71, de tester au passage la possibilité d'impression sur LaserJet (machine a priori compatible ThinkJet et utilisée pour faire votre journal préféré), et d'adapter le module Epson à la 82905, plus répandue en France (avec une excellente définition des codes-barre obtenus avec un ruban encreur neuf). Malheureusement, au début, il est apparu que les fonctions hexadécimales du HP-75 et celles du HP-71 réagissaient différemment, voire qu'il n'y avait pas de fonction appropriée sur le HP-71. Une première version fût faite, utilisant plusieurs Lex inclus dans le projet, maintenant défunt, de module CHHU aux USA.

Enfin vint la version actuelle de JPC Rom, dont les fonctions avaient été réécrites pour pouvoir, entre autres choses, « désinverser les quartets » (les initiés comprendront...). Tout ceci devait permettre de convertir une chaîne d'octets en son image bit et d'y insérer les caractères de contrôle voulus, sans acrobatie notable. Le présent programme utilise donc JPC Rom, seulement ce module...

J'ai failli oublier l'essentiel : le logiciel lit directement sur la cassette ou la disquette, sans aucun risque puisqu'on n'y écrit rien, les octets du fichier programme. Il fait appel à des menus. Tout est paramétrable : hauteur des barres, nombre de rangées par page, nombre d'octets par rangée, longueur de page, indication ou non de la traduction hexadécimale au-dessus de la rangée, nombre de points pour les barres 0 et 1, l'intervalle, etc... C'est énormément plus rapide que tous les programmes d'impression de codes-barre par une HP-41 parus à ce jour. Ceux-ci nécessitaient souvent les deux modules HPIL-Devel et Plotter et demandaient parfois plusieurs heures pour un programme un peu long - il ne faut plus maintenant que quelques minutes !

On ne risque plus de vider les accumulateurs de la cassette ou de la HP-41... Finis le folklore des batteries de blocs-secteur pour éviter la panne sèche, la colle et les ciseaux, l'interminable bande de papier thermique...

Pour démarrer, c'est tout simple : mettez sur la boucle le HP-71, l'imprimante (ThinkJet, 82905 ou LaserJet) et la cassette ou disquette, faites `RUN BAR713`, et laissez-vous guider par les menus... Les programmes HP-41 sont lus par défaut sur la première unité de mémoire de masse sur la première boucle ; cependant, rien ne vous interdit d'utiliser deux boucles sur le même HP-71, et de spécifier l'identificateur complet du fichier, par exemple : `PROG41:3:2` ou `PROG41:TAPE(2):2`... Tous les types d'instructions sont traités (sauf, peut-être, quelques synthétiques vraiment, mais alors là vraiment très tordues !). Espions, sachez qu'un programme privé donnera des codes-barre eux-mêmes privés.

Le principe est extrêmement simple : on commence par calculer l'emplacement du fichier sur le support, puis on en lit les octets dans une variable-chaîne P$. Après quoi, on analyse la chaîne pour isoler chacune des lignes (c'est à dire instructions 41). Ceci n'est pas absolument indispensable, mais permet d'écrire le numéro des lignes au-dessus des rangées, et simplifie le problème des instructions à cheval sur deux rangées de code. On convertit au fur et à mesure les octets en binaire, sans oublier de calculer les numéros de rangée, les limites de la rangée de code et le checksum correspondant.

L'impression se fait à partir de la chaîne binaire qui est la même pour toutes les imprimantes, au moyen de l'un des deux modules spécialisés par type d'imprimante. Les séquences d'échappement, et surtout le principe de fonctionnement graphique, différent entre ThinkJet et Laserjet d'une part, et 82905 de l'autre.

Il est possible d'imprimer une seule rangée ce qui est très utile si votre cartouche d'encre se met à « baver » ou s'épuise en cours de travail, ou de relancer une nouvelle impression en modifiant les paramètres : longueur de page, hauteur, etc, sans avoir à relire le fichier sur le support. Le maximum de simplicité a donc été recherché pour l'interface utilisateur.

Si vous avez des questions, vous pouvez nous écrire (et ce n'est pas la peine de le faire en anglais pour aucun de nous deux !) :

Michael Markov
P.O.Box 17
Lockwood, NY 14859
USA

ou

Eric Gengoux
8 rue de Furstenberg
75006 Paris

Alors, Happy barcoding ! Et soyez gentils : ça fait bien longtemps qu'on ne voit plus beaucoup de programmes pour 41 dans JPC... Maintenant, vous n'avez plus d'excuse !

Michael Markov (301)
Eric Gengoux (108)



Programme "DIAG" (diagonalisation de matrices, nécessite module Math et ERRLEX)

```
   - Diagonalisation des matrices carrées symétriques.
     D'après une pale d'Ulm 79.
  10 INPUT 'Ordre de A ? ';N
  20 OPTION BASE 1
  30 DIM A(N,N),D(N,N),O(N,N)
  40 MAT INPUT A
  50 MAT D=TRN(A) @ MAT O=D-A
  60 IF ABS(FNORM(O))>.000000001 THEN DISP 'Matrice non symétrique' @ END
  70 BEEP OFF @ WARNING 240010 @ BEEP ON
  80 CALL DIAG(A(,),N,D(,),O(,))
  90 DELAY INF
 100 DISP 'On a: O.D.tO=A'
 110 DISP 'D'; @ MAT DISP D;
 120 DISP 'O'; @ MAT DISP O;
 130 DELAY .4
-------------------------------------------------------------------
 140 SUB DIAG(A(,),N,B(,),R(,))
 150     DIM O(N,N)
 160     REAL T,D
 170     D=ANGLE(0,1)
 180     INTEGER I,J,P,Q
 190     MAT R=IDN(N,N)
 200     P=1 @ Q=2
 210     FOR I=1 TO N @ FOR J=1 TO N
 220             IF I#J AND ABS(B(I,J))>ABS(B(P,Q)) THEN P=I @ Q=J
 230         NEXT J @ NEXT I
 240     T=1/2*(MOD(D+ANGLE((B(Q,Q)-B(P,P))/2,B(P,Q)),D*2)-D)
 250     MAT O=IDN(N,N)
 260     O(P,P)=COS(T) @ O(Q,Q)=COS(T)
 270     O(P,Q)=SIN(T) @ O(Q,P)=-SIN(T)
 280     MAT R=R*O
 290     MAT B=TRN(R)
 300     MAT O=B*A
 310     MAT B=O*R
 320     DISP '.';
 330     IF ABS(T)>.000000001 THEN 200
```

****

Programme "DIRDOS" (catalogue évolué, nécessite JPC Rom)

```
  1 ON ERROR GOTO 'ERR'
  2 GOSUB 'INIT'
  3 ATTN OFF @ M=1

  4 DATA Catalogue selon NOM,Catalogue selon TYPE,Catalogue selon ANNEE

  5 RESTORE 4 @ M=MENU(3,M) @ ON M GOTO 6,9,12
  6 ATTN ON @ T$="NOM" @ FINPUT N$,"Fichier(s): ",A @ D1=1
  7 IF N$[1,1]="*" THEN SFLAG 1 @ F=8 @ N$=N$[2,LEN(N$)] ELSE F=LEN(N$)
```

```
 8 GOSUB 'IMP' @ GOTO 'RECH'
 9 ATTN ON @ T$="TYPE" @ GOSUB 'TS' @ IF P$=":TAPE" THEN D1=14 ELSE D1=12
10 F=D1+4 @ IF FLAG(1) THEN N$=RED$(N$)
11 GOSUB 'IMP' @ GOTO 'RECH'
12 ATTN ON @ T$="ANNEE" @ FINPUT N$,"Année: ",A @ IF P$=":TAPE" THEN D1=32 ELSE D1=30
13 F=D1+LEN(N$)-1 @ N$=N$[1,2] @ GOSUB 'IMP'

14 'RECH': X=1
15 WHILE CAT$(X,P$)#""
16     IF FLAG(1) THEN
17         T=POS(CAT$(X,P$)[D1,F],N$)
18     ELSE
19         T=N$=CAT$(X,P$)[D1,F]
20     END IF
21     IF T#0 THEN
22         A$=CAT$(X,P$) @ OUTPUT :D ;A$
23         O=O+VAL(A$[D2,F2]) @ Y=Y+1 @ X=X+1
24     ELSE
25         X=X+1
26     END IF
27 END WHILE
28 OUTPUT :D USING "A2/";" "
29 OUTPUT :D ;Y;" fichier(s) trouvé(s) représentant ";O;" octets utilisés."
30 END

31 'TS': ATTN OFF @ M=1
32 T1$="BASICLEX  TEXT DATA SDATAD-LEXADRS BIN  KEY  "

33 DATA BASIC,LEX,TEXT,DATA,SDATA,D-LEX,ADRS,BIN,KEY

34 RESTORE 33 @ M=MENU(9,M) @ N$=T1$[(M-1)*5+1,M*5]
35 FINPUT R$,"Type exclusif (O/N): ",A
36 IF NOT A THEN END
37 IF R$#"O" AND R$#"N" THEN 35
38 IF R$="N" THEN SFLAG 1
39 RETURN

40 'IMP': Z$="........" @ IF T$="TYPE" AND FLAG(1) THEN N1$=".."&N$&".." ELSE N1$=N$
41 IF T$="NOM" THEN N1$=N$&Z$[1,8-LEN(N$)] ELSE IF T$="ANNEE" THEN N1$="19"&N$&Z$[1,2-LEN(N$)]
42 IF FLAG(1) THEN N1$=".."&N$&".."
43 OUTPUT :D ;SPACE$(20);"Catalogue de ";P$
44 OUTPUT :D ;SPACE$(20);"Selon le critère ";T$;" = ";N1$
45 OUTPUT :D USING "A2/";" "
46 RETURN

47 'ERR': E=ERRN
48 SELECT E
49     CASE =64,=225032,=58
50         BEEP @ DISP "Faux périphérique!"
51     CASE =57
52         BEEP @ DISP "N'existe pas!"
53     CASE ELSE
54         STOP
55 END SELECT
56 END

57 'INIT':
58 DIM N$[8],P$[8],R$[1],T$[5],T1$[45],Z$[8],N1$[9],A$[43]
```

```
 59 INTEGER M,X,D,O,Y,F,D1,T @ CFLAG 1
 60 FINPUT R$,"Sortie: Ecr/Imp?  ",A
 61 IF NOT A THEN END
 62 IF R$#"E" AND R$#"I" THEN 60
 63 IF R$="E" THEN D=3 ELSE D=2
 64 INPUT "Catalogue ","MAIN";P$ @ P$=":"&P$
 65 IF P$=":TAPE" THEN D2=20 @ F2=24 ELSE D2=18 @ F2=22
 66 RETURN

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

Programme "PAGI" (Impression de texte, nécessite JPC Rom)

  - (c) A.BOLDIREFF - PPC(254) @ rev: 04-13/04/88
 20 DESTROY ALL
 30 B$=CHR$(163) @ H$=CHR$(162) @ E$=" PAGELEN " @ F$=CHR$(161) @ DISP 'Pagination des textes'

 40 DATA "Choix:"&B$&h$&" "&f$&"Valider:#RUN"
 50 DATA B$&E$&"72,66 "&B$&' 12"',H$&e$&"96,88 "&H$&' 12"',p,p,p

 60 OPTION BASE 1 @ DIM LO$(3)[1],M$(3)[22],P$(3)[1],T$[8],L$[8]
 70 RESTORE 40 @ READ M$,P$
 80 FINPUT LO$,M$,P$,A
 90 ON A GOTO 80,100,110
100 P=66 @ PRINT ESC$('&l6D'); @ PAGELEN @ GOTO 130
110 P=88 @ PRINT ESC$('&l8D'); @ PAGELEN 96

120 'P': IF P=88 THEN PERF ON
130 DISP 'Logo ? - O/N'
140 ON POS('ON',KEYWAIT$)+1 GOTO 140,160,150
150 DISP 'Document daté ? o/n' @ ON POS('ON',KEYWAIT$)+1 GOTO 150,180,190
160 SFLAG 2
170 FINPUT L$,'Choix logo : ',A @ IF L$='' THEN 170
180 DISP 'Date de ce jour ?(o/n)' @ ON POS('ON',KEYWAIT$)+1 GOTO 180,210,200
190 SFLAG 4 @ GOTO 220
200 DIM D$[8] @ FINPUT D$,'Date : jj/mm/an','7p8up',A @ IF NOT A THEN 200 ELSE 220
210 IF NOT FLAG(2) THEN D$='EURODATE' ELSE D$=""
220 DISP 'Référence ? (o/n)' @ ON POS('ON',KEYWAIT$)+1 GOTO 220,230,240
230 SFLAG 1
240 DIM Q$[1] @ FINPUT Q$,'Nb.Destinataires: 1','18pup',A @ N=VAL(Q$)
250 IF NOT N THEN N=1 @ SFLAG 5 @ GOTO 360

260 DATA Prénom Nom:,Adresse:,Lieu-dit(évent.):,Ville & Code:,v/ref:,n/ref:
270 DATA 11p30up,8p30up,17p30up,13p30up,6p30up,6p30up

280 DIM N$(N),A$(N),A1$(N),V$(N),M$(4),P$(4),I$(4)
290 IF FLAG(1) THEN DIM R$(N),S$(N),M$(6),P$(6),I$(6)
300 RESTORE 260 @ READ M$ @ RESTORE 270 @ READ P$
310 FOR I=1 TO N
320     FINPUT I$,M$,P$,A
330     N$(I)=I$(1) @ A$(I)=I$(2) @ A1$(I)=I$(3) @ V$(I)=I$(4)
340     IF NOT FLAG(1) THEN 350 ELSE R$(I)=I$(5) @ S$(I)=I$(6)
350 NEXT I
360 FINPUT T$,'TEXTE ? ',A @ IF T$='' THEN 360
370 FINPUT Q$,'Nb Copies à éditer: 1','20pup',A @ M=VAL(Q$)
```

```
380 IMAGE 5/,4(45x,32a,/),/
390 IMAGE 5/,4(45x,32a,/),5/
400 IMAGE 5/,3(45x,32a,/),/
410 IMAGE 5/,3(45x,32a,/),5/
420 I=1 @ J=1 @ BOLD ON
430 IF I>N THEN 940
440 IF J>M THEN CFLAG 3 @ J=1 @ GOTO 930
450 IF FLAG(5) AND FLAG(4) THEN 610 ELSE IF FLAG(5) AND NOT FLAG(4) THEN GOTO 470
460 IF FLAG(4) THEN 530
470 IF NOT FLAG(2) THEN
480     PRINT TAB(62);'le '; @ ON ERROR GOTO 510 @ CALL D$
490     IF FLAG(5) THEN 610 ELSE IF FLAG(2) THEN 520 ELSE 530
500 ELSE @ GOTO 490 @ END IF
510 OFF ERROR @ PRINT D$ @ IF FLAG(5) THEN 610
520 CALL L$(D$) @ IF D$='EURODATE' THEN D$=''
530 IF A1$(I)="" EXOR V$(I)="" THEN SFLAG 3
540 IF V$(I)='' THEN V$(I)=A1$(I)
550 IF FLAG(1) AND FLAG(3) THEN 560 ELSE IF FLAG(1) THEN 570 ELSE IF FLAG(3) THEN 580 ELSE 590
560 PRINT USING 400;N$(I),A$(I),V$(I) @ GOTO 600
570 PRINT USING 380;N$(I),A$(I),A1$(I),V$(I) @ GOTO 600
580 PRINT USING 410;N$(I),A$(I),V$(I) @ GOTO 610
590 PRINT USING 390;N$(I),A$(I),A1$(I),V$(I) @ GOTO 610
600 PRINT 'V/Réf : '&R$(I) @ PRINT 'N/Réf : '&S$(I) @ PLF 2
610 INTEGER T,T0,T1
620 IF FLAG(2) THEN T0=19 ELSE T0=16
630 IF FLAG(3) AND FLAG(0) THEN T0=T0-2 ELSE IF FLAG(0) EXOR FLAG(3) THEN T0=T0-1
640 IF FLAG(5) THEN T0=1
650 IF FLAG(4) THEN T0=T0-1
660 T1=P-T0-5 @ T=FILESZR(T$)
670 SELECT T
680     CASE <=T1
690         GOTO 900
700     CASE ELSE
710         INTEGER K,R,S @ R=RMD(T-T1,P-5)
720         IF R<=9 THEN S=12 @ R=R+8 @ T1=T1-8 ELSE S=4
730         K=CEIL((T-T1)/(P-5))+1
740         L=1 @ EXECUTE FNZ$(T$,1,T1,"F")

750         'F': PLF S
760         PRINT USING 890;L,47,K
770         L1=T1+1 @ L2=T1+P-5 @ H=2
780         IF H=K THEN 840
790         EXECUTE FNZ$(T$,L1,L2,"S")

800         'S': PLF 4
810         PRINT USING 890;H,47,K
820         L1=L2+1 @ L2=L1+P-6 @ H=H+1
830         GOTO 780
840         EXECUTE FNZ$(T$,L1,T,"E")

850         'E': PLF P-1-R
860         PRINT USING 890;K,47,K
870 END SELECT @ GOTO 910

880 DEF FNZ$(T$,L1,L2,G$)="PLIST "&T$&','&STR$(L1)&','&STR$(L2)&'@ run,"'&G$&'""

890 IMAGE 35x,2z,x,b,x,2z
900 PLIST T$
```

```
 910 IF NOT K THEN PFF
 920 J=J+1 @ GOTO 440
 930 I=I+1 @ GOTO 430
 940 CFLAG 0,1,2,3,4,5 @ BOLD OFF @ IF P=88 THEN PRINT ESC$('&l6d'); @ PAGELEN
 950 END

------------------------------------------------------------

 960 SUB LOGOPPC(D$)
 970     IF D$='' THEN SFLAG 0 @ D$='EURODATE'

 980     DATA PPC PARIS CHAPTER,B.P.604,75028 - PARIS Cedex01

 990     DIM A$(3) @ READ A$
1000     BOLD ON @ MODE 3
1010     FOR I=1 TO 2
1020         PRINT CENTER$(A$(I),21)
1030     NEXT I
1040     PRINT A$(3); @ MODE 0 @ PRINT TAB(58);'le ';
1050     IF FLAG(0,1) THEN CALL D$ ELSE PRINT D$
   -     L'on conserve le flag(0) pour mémoriser que le logo n'a que 3 lignes
1070 END SUB

------------------------------------------------------------

1080 SUB EURODATE
1090     DIM A$ @ A$=DATE$ @ PRINT A$[7,8]&A$[3,6]&A$[1,2] @ END SUB

------------------------------------------------------------

1100 SUB LOGOEXMP(D$)
1110     IF D$='' THEN SFLAG 0 @ D$='eurodate'
1120     BOLD ON @ MODE 3 @ PRINT CENTER$('PPC Paris Chapter',22.5) @ MODE 2
1130     PRINT 'Centre de Jeunesse et de Loisirs Jean Verdier' @ MODE 3
1140     PRINT CENTER$('11, Rue de Lancry',22.5)
1150     PRINT CENTER$('75009 - P A R I S',22.5); @ MODE 0 @ PRINT TAB(58);'le ';
1160     IF FLAG(0,0) THEN CALL D$ ELSE PRINT D$
   -     Logo de 4 lignes: Flag(0) effacé
1180 END SUB



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

Programme "BAR" (codes-barre HP-41 sur HP-71, nécessite JPC Rom et STRINGLX)


  - BAR713 codes à barre HP-41 sur HP-71/Thkjet, LaserJet ou HP82905
    (c)Michael Markov, adaptation Eric Gengoux V3.0 16/04/1987.
    LEX: JPCLEX
 40 DESTROY ALL @ DIM A$[32],F$[32]
 50 H5=0
 60 INPUT "Octets/Ligne? ","13";R6 @ R6=MAX(MIN(13,R6-3),1)
 70 INPUT "Lignes/Pg,Hauteur? ","24,26";H6,H7
 80 IF H5 THEN 620 ELSE H=0 @ H8=0 @ H9=0 @ LC OFF
 90 PWIDTH INF @ DIM O$[1290]
100 INPUT "Thinkjet/Hp82905 ? ","T";Q$ @ H8="T"=UPRC$(Q$)
110 IF NOT H8 THEN O1$=HTA$('FFFFFFFFFFFF')&HTA$('00000000') @ O0$=O1$[5] ! Réglage 82905B
```

```
120 INPUT "Fichier ? ";F$
130 P=POS(F$,":") @ IF NOT P THEN F$=REDUCE$(F$)&":TAPE:1" @ GOTO 130
140 D$=UPRC$(F$[P]) @ D=DEVADDR(D$)
150 L=POS(F$,":",P+1) @ IF L THEN L=VAL(F$[L+1]) ELSE L=1
160 F$=F$[1,P-1] @ F$[11]=CHR$(224)&CHR$(128)
170 SEND L; UNT UNL MTA LISTEN D DDL 4 DATA 0,2 UNL ! Recherche DIR et adresse fichier
180 WAIT 5 ! Arrêt progr. pendant les recherches sur cassette
190 SEND L; UNL UNT TALK D DDT 2
200 ENTER :D:L USING "#,32A";A$
210 IF NUM(A$)=255 THEN DISP "Fichier non trouvé" @ BEEP @ END
220 IF POS(A$,F$)#1 THEN 200
230 SEND L; UNT MTA LISTEN D DDL 4 DATA A$[15,16] UNL
240 L0=NUM(A$[29])*256+NUM(A$[30]) @ DIM P$[L0]
250 P9=BINAND(NUM(A$[31]),1) ! Bit PRIVATE armé si P=1
260 STD @ M$="#,"&STR$(L0)&"A"
270 WAIT 5 ! Lecture fichier HP-41 dans variable P$
280 SEND L; TALK D DDT 2 @ ENTER :D:L USING M$;P$
290 SEND L; UNT UNL LISTEN D SDC UNL
  - Analyse par lignes (selon type) du programme HP-41 ("parse")
310 P$[L0]=CHR$(47) @ GOTO 620
320 B=NUM(P$[R0]) @ B1=NUM(P$[R0+1]) @ B2=NUM(P$[R0+2])
330 X=1+(B>143)+(B>207)
340 IF B>239 THEN X=B-239 ! Instr. alpha
350 IF B>28 AND B<32 THEN X=MAX(2,B1-238) ! XEQ/GTO/W globaux
360 IF B>191 AND B<206 AND B2>240 THEN X=B2-237 ! Labels globaux
370 IF B>191 AND B<206 AND B2<241 THEN X=3 ! .END.
380 IF B>15 AND B<29 THEN GOSUB 400 ! Instr. numériques
390 L2=L2+(B#0) @ R0=R0+X @ RETURN
400 N=R0+1
410 B9=NUM(P$[N,N]) @ N=N+1
420 IF B9>15 AND B9<29 THEN X=X+1 @ GOTO 410 ELSE RETURN
430 IF H8 THEN PRINT CHR$(14)&ESC$('&l8D'); ELSE PRINT ESC$('0')&ESC$('E'); ! Bits rangée code
440 PRINT F$[1,10];L0;'Octets'
450 H5=1 @ L2=1 @ L1=1 @ R0=1 @ M1=0 @ M2=0 @ C=0 @ T=16*(1+P9)
460 IF H8 THEN PRINT ESC$('*r1280S'); ! Graph. hte. densité
470 FOR I=1 TO L0 STEP R6 @ E=MIN(L0,I+R6-1) @ B$=P$[I,E]
480     M1=MIN(R6,R0-I)
490     IF R0<E+1 THEN GOSUB 320 @ GOTO 490
500     M2=(X+E-R0+1)*(E+1#R0) @ R=CEIL(I/R6)
510     B$=CHR$(T+MOD(R-1,16))&CHR$(M1*16+M2)&B$
520     FOR K=1 TO LEN(B$) @ C=C+NUM(B$[K,K]) ! Checksum
530         C=MOD(C+INT(C/256),256) @ NEXT K
540     B$=CHR$(C)&B$
550     IF R9 AND R<>R9 THEN 600
560     IF H9 THEN H$=ATH$(B$,1) ELSE H$=""
570     PRINT USING '"ROW ",3D,"   Lignes",4D,"-",4D,3X,33A';R;L1;L2-1;H$
580     GOSUB 680 @ IF R=R9 THEN 620
590     IF NOT MOD(R,H6) THEN PRINT CHR$(12) ! Nbre. ROWs/page
600     L1=(R0=E+1)+L2-1
610 NEXT I
620 R9=0 @ DISP "Fin/Tout/Ligne/Hex ";
630 IF H9 THEN DISP "on" ELSE DISP "off"
640 BEEP @ ON POS("FTLHM",UPRC$(KEYWAIT$))+1 GOTO 640,670,430,660,650,60
650 H9=NOT H9 @ GOTO 620
660 INPUT "Ligne #? ";R9 @ GOTO 450
670 DISP @ PWIDTH 80 @ END
680 IF NOT H8 THEN 800 ELSE O$='7070' ! Routine THINKJET
690 FOR A=1 TO LEN(B$)
```

```
700     FOR A0=7 TO 0 STEP -1
710         IF SBIT(B$,A,A0) THEN O$=O$&'7F0' ELSE O$=O$&'70'
720     NEXT A0 @ NEXT A
730 O$=O$&'7F07' @ IF MOD(LEN(O$),2) THEN O$=O$&'0'
740 O$=HTA$(O$,1)
750 ENDLINE '' @ PRINT CHR$(27)&'*rA';
760 FOR H=1 TO H7 ! Hauteur barres
770     PRINT ESC$('*b'&STR$(LEN(O$))&'W')&O$;
780 NEXT H @ PRINT ESC$('*b1W')&CHR$(0)&ESC$('*b1W')&CHR$(0)
790 ENDLINE @ PRINT CHR$(27)&'*rB'; @ RETURN
800 O$=O0$&O0$ @ FOR A=1 TO LEN(B$) ! Routine HP82905
810     FOR A0=7 TO 0 STEP -1
820         IF SBIT(B$,A,A0) THEN O$=O$&O1$ ELSE O$=O$&O0$
830     NEXT A0 @ NEXT A @ O$=O$&O1$&O0$ @ C2=LEN(O$)
840 PRINT ESC$('&l9D') ! Lignes jointives (esp. 9 points)
850 PRINT ESC$('&k2S') ! Graphique mode condensé
860 FOR H=1 TO H7 DIV 8 @ PRINT CHR$(27)&'*b';C2;'G';O$ @ NEXT H
870 PRINT ESC$('&k0S') @ PRINT ! Titre ROW mode normal
880 RETURN
```



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 trouverez donc le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !"), le Lex de Frédéric Vadez ainsi que le Lex ERRLEX de Serge Vaudenay pour le programme "DIAG".

```
CHARLEX

VADLEX    HMS$        XFN     93001

ERRLEX    EEND        XWORD   92001
          ERROR       XWORD   92002
          WARNING     XWORD   92003
```

```
 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          624 octets

      0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E000000000000 68D
002: 5E4001EFF0000000 9FD
003: FE00000000800001F D57
004: F31BF961400032BF 0EA
005: 38F14A11DB10AD23 484
006: 07D532BFB8FD7911 837
007: 11AD754D7A101743 BBA
008: 11014D1CB15D0000 F25
009: 71450375FF864834 2A2
00A: 5655581008355654 5F9
00B: 5810070507701724 93F
00C: 7700775070077517 C92
00D: 2077040708364545 FE0
00E: 4A30000A49724000 333
00F: 0808094A2C180814 69C
010: A464242008355455 9F6
011: 581000054C714000 D3C
012: 0C3142404C700832 098
013: 41414A70002078A0 3F0
014: 2F30000000000000 71B
015: 0000000000000000 A2B
016: 0000000000000000 D3B
017: 0000000000000000 04B
018: 0000000000000000 35B
019: 0000000000000000 66B
01A: 0000000000000000 97B
01B: 0000000000000000 C8B
01C: 0000000000000000 F9B
01D: 0000000000000000 2AB
01E: 0000000000000000 5BB
01F: 0000000000000000 8CB
020: 0000000000000000 BDB
021: 000000000000080C F06
022: 1A28080008080A2C 270
023: 180008040E340800 5B9
024: 08001E3018000000 8F3
025: 0000000000000000 C03
026: 0000000000000000 F13
027: 0000000000000000 223
028: 0201000000010200 539
029: 0000000201020000 84E
02A: 0001000100000002 B62
02B: 0102010000000000 E76
02C: 0000000000000000 186
02D: 045E755142400101 4D2
02E: 0101010000000000 7E5
02F: 0000000000000000 AF5
030: 0000070507000000 E18
031: 000000000083444C4 156
032: 44400D7901112D70 4B6
033: 050D750509700000 800
034: 0D70000000384540 B43
035: 4020014E322E3140 E97
036: 084E794142400000 1E7
037: 00000000002E4559 525
038: 3200000000000000 83A
039: 0000000000000026 B52
03A: 5556587008365556 EB1
03B: 5810083645464830 202
03C: 0832414248700024 543
03D: 5655587008345655 8A0
03E: 5810083446454830 BEF
03F: 0C3042414C700024 F44
040: 5556587008355654 2A1
041: 5810083546444830 5F0
042: 0C3142404C700025 946
043: 5455587008355455 CA0
044: 5810083544454830 FEE
045: 0C3140414C700875 350
046: 14141870000A4972 6A1
047: 40000E3159454E30 A01
048: 0C7A0F7949400024 D79
049: 5554587000084A71 0D5
04A: 40000C523A262D10 436
04B: 0424587458400875 78D
04C: 1415187000094A70 ADD
04D: 4000083544454830 E21
04E: 0C3140414C300C74 189
04F: 5655545000054C71 4E0
050: 40000            5D9

VADLEX           205 octets

      0123456789ABCDEF sm

000: 651444C454850202 359
001: 802E000000000000 688
002: E9100D5101000000 9D2
003: F710000000000000 D00
004: 071000F784D43542 067
005: 101FF81115378407 3D2
006: F307D708DC32F004 76B
007: 20AF2310117F8FA9 B01
008: 0F03FD2F3F3D2020 EA0
009: 202021557071CF68 207
00A: CF2230F9024C05B3 599
00B: 4B342094880850AC 916
00C: 0326009B65A3131B C82
00D: 3280D0A90BF4A3E5 027
00E: 9F0124AF23163749 3A7
00F: 0DD108AF22330673 729
010: 80118BF2BF2AEDBF B01
011: 2BF2BF2BF223A9E1 EC6
012: 080420AF23161860 226
013: 60318117F8FA90F0 5B5
014: 7350312214D17175 90A
015: 40317214D1717730 C5E
016: 31A314D171792086 FC8
017: 09031D214D1C51CF 357
018: 1CF01AF19F200B7A 705
019: B6564FFD0118AEAB ACF
01A: F6BF6108F0BE4330 E79
01B: 303CA159317301F  1B6

ERRLEX           198 octets

      0123456789ABCDEF sm

000: 542525C454850202 358
001: 802E000000000000 687
002: 19100C5103000000 9BE
003: F920000000000000 CEF
004: 019000CB0035000D 04A
005: 81016000D75454E4 3B0
006: 44109542525F4252 70C
007: 0D751425E494E474 A92
008: 301FF8DE6A208D39 E44
009: 4509FFFFDEFFF7CB 255
00A: 0228DA93902EFFF6 60A
00B: DFFF75A02A8F5C39 9E2
00C: 08D84A804CFFF8BF DBC
00D: FF77801BBB8F2144 16E
00E: 8FFD0908F81EF043 528
00F: 28F4CEF039025454 8BA
010: E44415C91698FCBE C75
011: F08FA87914428FFF 037
012: 980100119A064229 38C
013: 0EA18FB51905318F 729
014: 92D708F4E02184D1 ABA
015: BBB8F2146694F8F6 E78
016: 81F08FDF8E0AC08F 24E
017: 322B18FFACE0AF2A 61F
018: BEBF4BF4BF4AF78F A26
019: 2D2B1AFEAFFAFE8F E34
01A: 2D2B1AFBF2F2AE60 204
01B: 1                236
```

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

La maquette de ce numéro a été préparée et réalisée par Pierre David et Laurent Istria grâce à un système comprenant un HP71B, un lecteur de disquettes HP9114A, deux HP9807A, deux HP9154 et une imprimante LaserJet.

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

Directeur de la publication : Pierre David
Numéro ISSN : 0762 - 381X

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

Imprimé par Copy-Express, 42 86 91 94.

ENGLISH SUMMARY

JPC 56 - JULY / AUGUST 1988

We are sad to report convergent rumors about the production stop of the HP-71 before the end of this year. We will probably discuss this point in a near future. Hewlett-Packard will also announce two new low-end machines, HP-22 and HP-32 similar to HP-27.

This issue of *JPC* begins as usual with HP-28 articles. Paul Courbis make a few corrections about his previous article. Next, Hervé Thévenon presents us a game and a program to plot the primitive of a function. On page 5, you will find PEEK and POKE functions for the HP-28S by Paul Courbis. On page 7, Philippe Heilbronn gives us a second part for his previous article. Last, Sébastien Lalande describes a program to compute fractional numbers and an application to matrices.

One of our Swiss friends (from PPC Lausanne), Jean Hulaas, feeds the HP-41 columns with a program to multiply compatible matrices. The program is well documented.

The HP-71 columns begins on page 16 with an article by our friend Xavier Bille about Forth vocabularies listing.

On page 18, Jacques Baudier presents his last article about HP-71 assembly language programming. This series is discontinued because Jacques's machine was lost a few month ago. That's too bad... The next assembly language article is from Frédéric Vadez describing a function HMS$ to convert a real to a string in *hms* format.

The HP-71 Basic columns includes four articles. The first one appears on page 24. It was written by Serge Vaudenay to compute the eigen values of a matrix using Jacobi method. The following article is by Jean-Claude Kursner (a Swiss friend from PPC Paris) and provides an extended directory facility. It is interesting to study for beginners. The penultimate article is from Alexandre Boldireff. Alexandre is using this program daily to print his own mail and reports. The last article is from Eric Gengoux and Michael Markov and provides most desired program to print HP-41 barcodes on a ThinkJet using an HP-71.

Until next time,

Happy Programming and JPC reading !