NUMERO 50

JPC

DECEMBRE 1987
JANIVIER 1988
NUMERO 50

Le numéro 40 F

A PROPOS DU CLUB

P. David	Editorial	*1*
	PPC Paris se réunit	*2*
	Ah ! Vous écrivez !	*2*
	Des nouvelles de JPC Rom	*3*
	Courrier du coeur	*3*
	S.O.S.	*4*

HP28

P. Courbis	Choc en retour	*8*
P. Courbis	Lex sur HP-28C	*8*
L. Istria	Réduisez vos matrices	*9*

HP41

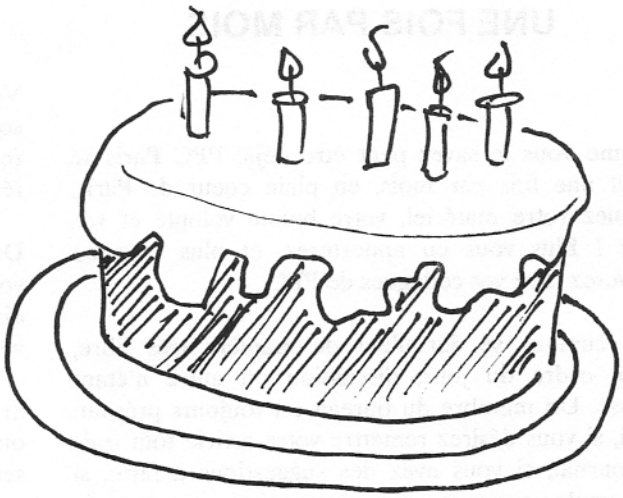
V.M. Electronics	Le module HEPAX	*12*
R. Dine	Portfolio	*15*

HP75

J. Y. Hervé	Récupération de fichiers avec MADLEX	*18*

HP71

J. Baudier	L'assembleur du HP-71 (acte V)	*20*
J. Y. Naour	Des caractères pour faire joli	*27*
J. Elhay	Evaluez vos polynômes	*28*
G. Toublanc	Arrondir dans la bonne base	*29*
B. Gil	Dérivées symboliques (acte I quater...)	*33*
O. Arbey	La guerre des noyaux (acte II)	*33*
E. Gengoux	Un catalogue ordonné	*34*
J. Maille	Ca date !	*35*
	Le coin des Lhex	*41*
PPC Paris	Sommaire des articles parus en 1987	*46*



EDITORIAL

Chers amis,

Je suis fier et heureux de vous présenter ce numéro 50 de JPC.

Notre Club a aujourd'hui cinq ans. A ce jour, cinquante numéros ont été publiés. Je vous laisse imaginer le travail que cela représente...

Nous recevons beaucoup de lettres d'encouragement, nous félicitant pour la qualité et la diversité du journal. En ce jour anniversaire, je voudrais remercier ici tous ceux qui, par leurs articles, font la qualité et la diversité du journal. Qu'ils soient chaudement encouragés, PPC Paris compte sur eux.

Nous voici repartis pour cinq nouvelles années. Qu'elles soient aussi riches et intéressantes que les précédentes.

Pierre David (37)

PPC PARIS SE REUNIT UNE FOIS PAR MOIS

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

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

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

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

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

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

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

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

Pierre David (37)

AH ! VOUS ECRIVEZ

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

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

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

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

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

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

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

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

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

Jean-Jacques Dhénin (177)

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

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

DES NOUVELLES DE JPC ROM

Vous vous souvenez de l'opération **JPC Rom** lancée il y a un an. Le but était de fournir aux possesseurs de HP-71 une Eprom contenant tous les Lex parus dans JPC depuis l'origine.

JPC Rom contient toutes les fonctions (utiles) parues dans JPC. En plus, ceux parmi vous qui ont acquis JPC Rom ont eû les dernières versions de ces fonctions, souvent avant même leur parution. Ainsi, les fonctions de date étaient déboguées longtemps avant leur parution. D'autres fonctions ont été placées dans JPC Rom avant leur parution, telles les fonctions de programmation structurée évoluées. D'autres ne seront jamais publiées. Ainsi, SYSEDIT et KA ne paraîtront jamais dans JPC.

Avec tout ceci, JPC Rom flirte avec les 22 Ko.

Rappelons que cette opération offrait deux choix :

- avoir une Eprom contenant JPC Rom, et le manuel,
- avoir une Eprom contenant JPC Rom et vos programmes, plus le manuel de JPC Rom.

La première solution vous apportait tous les Lex parus dans JPC, la plupart d'entre eux étant remaniés et améliorés.

Le deuxième choix correspondait à votre propre module d'application, comparable aux modules HP. En plus de JPC Rom, vous aviez *vos* programmes en mémoire morte constamment disponibles et n'occupant pas de place en mémoire vive.

Si vous n'avez pas été parmi les premiers à commander votre Eprom JPC, vous pouvez à présent le faire.

Il suffit de nous faire parvenir l'Eprom que vous aurez achetée (CMT 32 Ko ou 64 Ko Eprom), et éventuellement vos programmes si vous choisissez la deuxième option. Il vous en coûtera 600 F pour la première option, et 800 F pour la deuxième.

Puisqu'une Eprom est un module de mémoire morte, que l'on ne peut programmer qu'avec des outils spéciaux, il vous sera possible de nous faire parvenir votre module pour que nous le reprogrammions, moyennant 100 F.

A titre exceptionnel, ceux qui ont participé à la première opération peuvent bénéficier d'une première mise à jour gratuite.

Nous avons bénévolement accompli un énorme travail pour JPC Rom. Profitez-en !

Pierre David (37)
Jean-Jacques Dhénin (177)
Michael Markov (301)
Janick Taillandier (246)

COURRIER DU COEUR

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

Vend :
Lecteurs de cartes magnétiques HP82400A neufs pour HP-71 (dans leur boîte, avec documentation et 5 cartes magnétiques) : 500 F seulement.
1 lot de cartes magnétiques pour HP-41, HP-67, HP-97 et même HP-65 : 100 F seulement.

Janick Taillandier
335 rue Lecourbe
75015 Paris

Vend :
Lecteur de disquettes HP-9114A : 3000 F, imprimante ThinkJet HP-2225B : 2500 F.
Un module 32 Ko Ram + 32 Ko Eprom HHP pour HP-71 : 3000 F, module 32 Ko Ram CMT : 1500 F.

Jean-Jacques Dhénin
35 rue Boileau
92120 Montrouge
Tél : (1) 42 53 91 60

Vend :
HP-75 + accessoires et docs d'origine.

Pierre Picheret
Calmon Aiguefonde
81200 Mazamet

Vend :
Imprimante ThinkJet HP-2225B : 2500 F, imprimante HP-82905B : 1500 F.

René Fagnon
81 rue de Belfort
25000 Besançon
Tél : 81 88 55 32 à partir de 18h

Vend :
HP-75C : 4000 F, module 8 Ko Ram : 1000 F, module I/O : 500 F, module Text Formatter : 500 F, module VisiCalc : 1000 F.

HP-71B : 3000 F, module finance : 500 F, module HP-IL : 1000 F, lecteur de cartes : 1000 F, module Forth Assembleur : 600 F, module Text Formatter : 500 F.

Interface vidéo HP : 1000 F, lecteur de cassettes : 3500 F, imprimante thermique : 3000 F, moniteur : 1500 F, interface vidéo Pac Screen : 3500 F, imprimante ThinkJet + kit : 3500 F, lecteur de disquettes HP-9114A : 4000 F, convertisseur HPIL/HPIB HP-82169 : 1500 F.

Vincent Herlicq
3 rue Berteaux Dumas
92200 Neuilly sur Seine

Vend :
Imprimante ThinkJet : 2800 F, lecteur de disquettes HP-9114A : 2800 F, HP-71B + module HP-IL : 4000 F, module Forth : 700 F. Prix négociables. Pour tout achat, Vincent offre soit un module mémoire, soit une interface vidéo.

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

Vend :
Lecteur de cassettes HP-82161A + 4 cassettes : 2700 F, Interface HPIL / RS232C HP-82164A : 3500 F, module HPIL HP-82160 A pour HP-41, modules mémoire HP-82106A pour HP-41 : 200 F.

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

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

SOS

Claude Boeuf
1 rue des Côtes
Varney
55000 Bar le Duc

Cherche :

Des informations permettant d'utiliser une ThinkJet HP-IL depuis un Atari 1040 relié à la boucle par une interface HP-IL / GPIO. Le contrôleur de boucle est un HP-71. Le cablage utilisé entre GPIO et Atari :

Atari	HP82165A
BUSY	RDYO
STROBE	DAVI
DATA 1	DA 0
DATA 2	DA 1
DATA 3	DA 2
DATA 4	DA 3
DATA 5	DA 4
DATA 6	DA 5
DATA 7	DA 6
DATA 8	DA 8

et le programme :

```
10 SEND UNT UNL MLA TALK 2 ! 2 adresse GPIO
20 ENTER GPIO ; A$
30 SEND UNT UNL LISTEN 1 MTA ! 1 adresse ThinkJet
40 PRINT A$ @ GOTO 10
```

ne fonctionnent pas !

Laurent Vergnes
135bis rue de Paris
94220 Charenton le Pont

Se pose les questions suivantes à propos de la HP-28 :

- impossibilité d'afficher une courbe sur R, comportant une $\sqrt[3]{f(x)}$. En effet, le calculateur dans ce mode donne un nombre complexe et en effectuant `'-8^INV(3)'`, par exemple, on obtient une solution réelle ! Donc, comment privilégier les solutions réelles quand elles existent ?

- comment fonctionnent les commandes `DRAW` et `DRWΣ`, d'une façon plus détaillée, ou comment s'effectuent les calculs ?

Frédéric Vadez
2 rue du Colombier
94200 Ivry
(16 1) 46 70 05 96

Je vous propose un tableur pour le HP-71. Ce programme permet d'éditer un fichier HPAF et d'associer à chaque colonne une formule utilisant les données contenues dans les autres colonnes.

Les applications sont très nombreuses, depuis le suivi d'un portefeuille boursier, jusqu'à la gestion automatique de stock, en passant par le calcul prévisionnel de subventions.

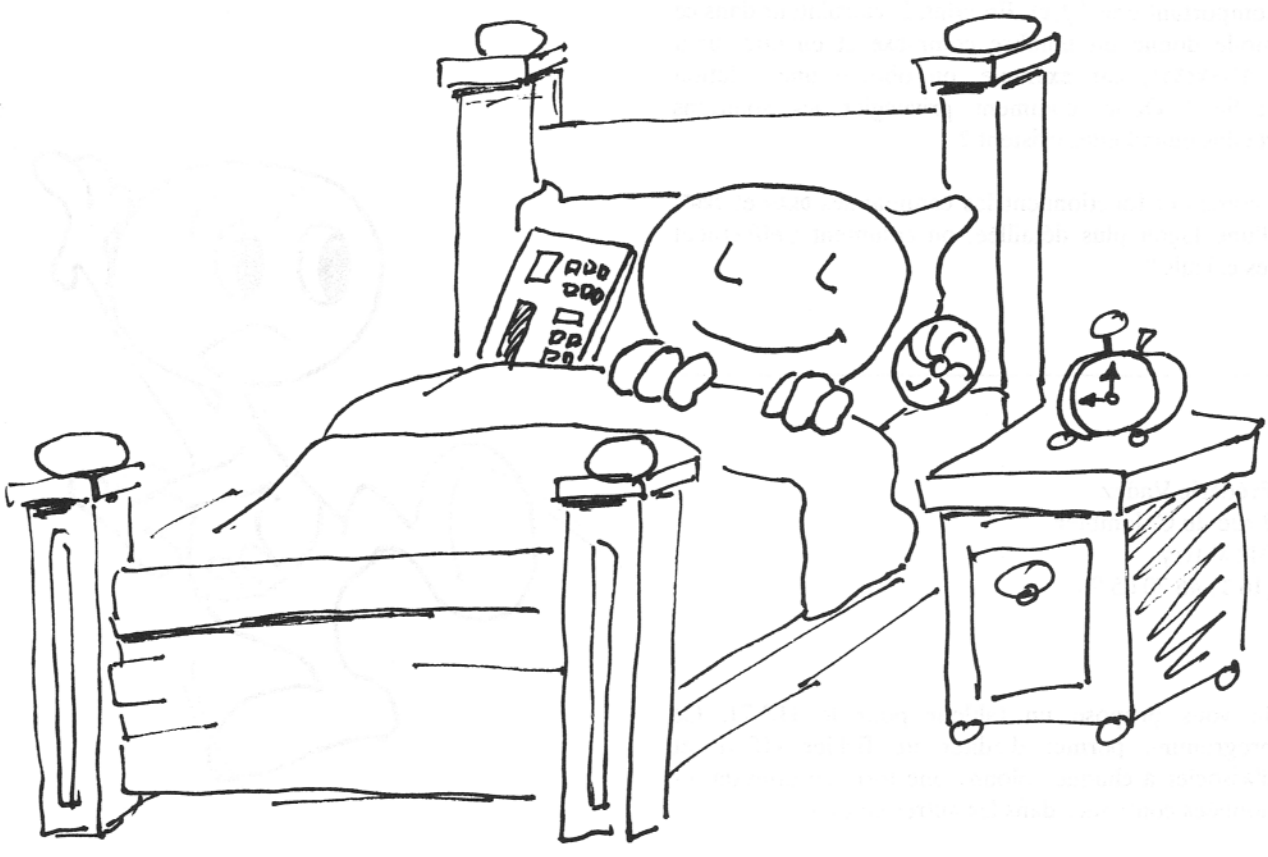
Ce programme peut fonctionner seul avec le Lex KEYWAIT, universellement répandu, mais il faut au moins posséder le programme CREATE et le Lex DATALEX du *Data Management Pac* de la User's Library de HP pour créer un fichier HPAF.

Je diffuse moi-même ce programme. Vous pouvez l'obtenir avec sa documentation et des exemples d'applications pour 399 francs.

Pour tous renseignements, écrivez moi ou appelez moi (plutôt le soir entre 21h et 22h30).

Frédéric Vadez (79)

Je ne sais pas ce qu'a cette machine
mais elle est de plus en plus
affectueuse



HP28

P. Courbis	Choc en retour	*8*
P. Courbis	Lex sur HP-28C	*8*
L. Istria	Réduisez vos matrices	*9*

CHOC EN RETOUR

Dans JPC 49 (novembre 1987), Eric Gengoux déplorait l'impossibilité de tracer la courbe de $\sin(x)/x$ sur la HP-28C. En fait, c'est possible grâce à l'instruction IFERR décrite à la page 237 du manuel de référence. Il suffit de rentrer le programme suivant dans la variable EQ :

```
« IFERR 'SIN(X)/X' →NUM
    THEN 1
  END
»
```

puis d'entrer les bornes du tracé, par exemple (-10,-.7) et (10,1), par [PMIN] et [PMAX], et enfin de tracer la courbe par [DRAW] (en mode radians de préférence).

Paul Courbis (392)

LEX SUR HP-28C

Il est possible de générer sur le HP-28C des programmes en langage machine (ou assembleur) directement utilisables.

Si vous souhaitez simplement utiliser les réalisations faites par d'autres, par exemple dans JPC, voici la marche à suivre.

Le but est d'entrer une suite de chiffres hexadécimaux qui représentent une routine en langage machine.

Première étape

La première étape est de se constituer un outil pour rentrer ces chiffres. Il faut utiliser le programme ci-dessous :

```
« → LM
  « HEX
    ""
    1 LM SIZE
    FOR X
      "#" LM
      X DUP2 1 + DUP SUB 3 ROLLD
      DUP SUB + + STR→ B→R CHR +
      2
    STEP
  »
»
```

et stockez-le sous le nom ASS.
Mettez ensuite la chaîne suivante dans la pile :

```
"76C205B3C169C20E10001431331791331411421648080C09F20"

[USER] [ASS]
```

Vous voyez maintenant une chaîne de caractères sans signification. Continuez :

```
→LEX [STO]
```

Faites ensuite :

```
[→LEX] #4FFCE SYSEVAL
```

A présent, vous voyez dans la pile deux System Object. Faites alors :

```
'→LEX' [STO]
```

Visitez ensuite le programme ASS, et rajoutez un appel à →LEX à la fin (entre STEP et le premier chevron fermant).

Deuxième étape

Maintenant que l'outil est forgé, il reste à l'utiliser. Les programmes en langage machine qui paraîtront dans JPC seront, en principe, sous la forme d'une chaîne.

Par exemple, le Lex suivant représente une routine d'inversion vidéo. Rentrez la chaîne, en une seule ligne, sans espace :

```
"76C2091C7069C208700013310310A80C0109271FFD2041371
 F870047F201FFD5041371F004047A1011980D011A11313114
 2164808C1531B9C151117713388B60013366EF09F20"
```

Puis exécutez :

```
[USER] [ASS]
```

Vous avez à présent deux `System Object` dans le premier niveau de la pile.

Il ne reste plus qu'a stocker le résultat dans une variable. Par exemple, notre routine va s'appeler `INVERSE`.

```
'INVERSE' [STO]
```

Vous pouvez maintenant utiliser `INVERSE` comme tout autre programme. Il suffit d'appuyer sur la touche `[INVERSE]` dans le menu `[USER]`.

Attention : ne faites jamais `[VISIT]` sur votre nouvelle fonction ainsi entrée. Votre programme serait détruit.

Nous verrons comment concevoir des Lex dans un prochain numéro. Le voile commence à se soulever doucement sur la structure interne de la HP-28C...

En attendant, bonnes inversions !

Paul Courbis (392)

REDUISEZ VOS MATRICES

Eh bien bonjour à tous ! Eh oui, me revoilà, après une absence bien longue, je rentre au bercail, parmi ceux que j'ai toujours aimés, vous, fidèles lecteurs de notre cher journal : JPC.

Bon. Trève de bavardages. Allons droit au fait qui nous intéresse, c'est à dire à l'article du jour : il s'agit d'un petit programme qui permet de réduire une matrice, j'entends par là lui ôter un certain nombre de lignes et de colonnes. En gros, parti d'une matrice $n \times n$ on obtient en fin une matrice $(n-1)\times(n-1)$.

Mais, me direz vous, à quoi peut servir ce doux programme ? Eh bien voilà : imaginons un instant que nous ayons un système 4×4 dont les matrices sont A et B. On a donc l'équation : AX=B.

Seulement voilà, il se trouve que le déterminant de la matrice A est nul. C'est à dire, en termes purement mathématiques, qu'au moins deux de ses vecteurs sont liés.

D'un point de vue plus terre à terre, on a alors une des équation qui fait double emploi. Ou encore, l'une des équation est, à des coefficients multiplicateurs près, la somme des autres. Et cela est bien embêtant car alors il manque une équation. On a une infinité de solutions toutes de la même forme, et dépendantes d'une variable que l'on peut fixer arbitrairement.

Voyons donc un exemple concret de la chose. Soit à résoudre le système :

```
2x + 3y - 2z +  t = 6
3x -  y + 2z -  t = 3
 x +  y + 3z - 2t = 4
 x - 3y - 4z + 3t = -5

  |2  3 -2  1|    | 6|
A=|3 -1  2 -1|  B=| 3|        (1)
  |1  1  3 -2|    | 4|
  |1 -3 -4  3|    |-5|
```

On voit tout de suite (n'est ce pas que cela saute aux yeux ?) que la dernière équation est *combinaison linéaire* de la seconde et de la troisième. En effet, en soustrayant 2 fois la troisième à la seconde, on obtient la quatrième. Une des variables peut alors être fixée arbitrairement sans nuire à la véracité du système.

En fixant une des variables, on peut donc s'affranchir d'une des équations. Posons donc $t=4$. Le système peut alors se simplifier en :

```
  |2  3 -2|      |2 |
A=|3 -1  2|    B=|7 |        (2)
  |1  1  3|      |12|
```

et nous avons alors un système de trois équations à trois inconnues tout à fait soluble. La matrice A a, en fait, été réduite. C'est exactement ce à quoi sert le petit utilitaire que je me permets de vous soumettre ce mois-ci.

En voila le listing commenté par votre aimable serviteur :

```
« LIST→ DROP → l c          Ligne et colonne à
                            ôter
« ARRY→ LIST→ DROP → l1 c1  Ligne et colonne du
                            tableau
```

```
« l1 c1 * c - 1 + 1          Préparation de la
                             boucle quoi enlève
                             les colonnes. Il
                             faut sauter à chaque
                             fois la valeur d'une
                             ligne
FOR i
     i ROLL DROP             ôte la colonne
     c1 NEG                  Saute la ligne
STEP
c1 1 - 'c1' STO              Une colonne de moins
l1 l - 1 + c1 *              Où est le premier
                             élément de la ligne?
DUP
c1 - 1 +                     Et le dernier?
FOR i
     i ROLL DROP             Idem pour une ligne
     1 NEG
STEP
l1 1 -                       Une ligne de moins
c1                           La colonne
2 →LIST                      Taille du tableau
→ARRY                        Dans le tableau
» » »                        Fini
```

Dans l'exemple ci dessus, il suffit de placer la matrice A(4*4) du système (1) au sommet de la pile, puis de mettre {4,4} et d'appuyer sur red. on obtient alors la matrice A(3,3) du système (2).

On trouve alors la solution d'une simplicité enfantine (croyez vous que ce soit voulu ?) : x=1, y=2, z=3, et, bien entendu, t=4

Et bien voilà, j'espère avoir un peu contribué au développement de la HP-28 à l'aide de ce petit programme. A la prochaine fois.

Laurent Istria (3)

HP41

V.M. Electronics — Le module HEPAX *12*
R. Dine — Portfolio *15*

LE MODULE HEPAX POUR HP-41

VM Electronics est fière de vous présenter le module HEPAX.

HEPAX est une abréviation pour HEwlett-PAckard 41 eXpansion. Le module HEPAX se présente comme un module d'application, mais permet à l'utilisateur d'avoir jusqu'à 32000 octets de mémoire additionnelle. La seule différence entre les modules d'application et le module HEPAX est que l'utilisateur peut stocker ses propres programmes dans le module HEPAX. Ces programmes peuvent être modifiés à sa guise.

Le module HEPAX existe en version 8000 octets ou 16000 octets de mémoire supplémentaire, utilisable pour des programmes, des données, des fichiers texte ou du code machine.

C'est un module de taille standard, qui utilise un seul port et permet l'utilisation des 319 registres internes et des 600 registres de mémoire étendue.

UTILISATION DU MODULE HEPAX

Un programme peut être copié dans le module depuis la mémoire principale simplement en entrant son nom en ALPHA, puis en exécutant la fonction HSAVEP. Le programme est alors dans le module HEPAX, et peut être utilisé à partir de celui-ci. Il peut être effacé de la mémoire principale, libérant ainsi de la place pour d'autres utilisations.

Vous pouvez également créer vos fichiers texte et donnée, exactement comme ceux du module Extended Functions. Le module HEPAX contient les équivalents exacts de toutes les fonctions de traitement de fichiers du module Extended Functions. La différence est que ces fonctions sont précédées de la lettre H, par exemple, pour transférer un nombre dans un fichier HEPAX, il faut taper HSAVEX.

Le module HEPAX est aussi un système de type MLDL, c'est-à-dire que l'utilisateur peut développer ses propres fonctions en langage machine 41. Le module HEPAX standard et le module HEPAX evolué contiennent de nombreuses fonctions sophistiquées, parmi lesquelles :

- un désassembleur puissant pour désassembler le code machine de la HP-41.
- un éditeur hexadécimal pour le développement en code machine.
- un utilitaire permet le transfert rapide de blocs de données de 4K de et vers les unités de stockage de masse.

Les programmes et données résidant dans le module sont en parfaite sécurité. En effet un MEMORY LOST n'affecte pas le module HEPAX, et le module peut être protégé contre l'écriture pour prévenir d'éventuels effacements de programmes et de données. Les données sont conservées tant que le module est branché dans la machine.

LES DIFFERENTS MODULES HEPAX

Il existe quatre modules HEPAX différents :

- un module HEPAX standard contenant 8K de Ram ainsi que toutes les fonctions précitées et beaucoup d'autres.
- un module HEPAX évolué, contenant 16K de Ram et les fonctions supportées.
- un module mémoire HEPAX contenant 8K de Ram additionnelle, sans aucune fonction.
- un double module mémoire HEPAX contenant 16K de Ram.

Normalement, l'utilisateur commencera avec un module HEPAX standard, ou avec un module évolué, et ajoutera des modules mémoire au fur et à mesure de ses besoins. Cependant, les possesseurs de la ZENROM disposent déjà de quelques fonctions et se contenteront de modules mémoire.

DOCUMENTATION

Les modules sont vendus avec une documentation de 100 pages, en anglais, divisée logiquement en 4 parties, de telle sorte que vous n'ayez qu'à lire la partie se rapportant à votre module HEPAX. Le manuel explique en détail l'utilisation de chaque fonction, donne de nombreux exemples, et contient un résumé de toute la structure interne de la HP-41 et de son langage machine, afin que vous puissiez écrire des programmes dans ce langage.

INFORMATIONS SUPPLEMENTAIRES

Les fonctions du module HEPAX sont contenues dans une Rom de 16K. Afin de minimiser le besoin d'espace pour les adresses, cette Rom est divisée en 4 parties, dont une seule est active à la fois. A chaque allumage du calculateur, la Rom balaye la zone adressable du 41 et s'auto-adresse à un endroit vacant.

La Ram est située dans l'espace adressable réservé pour chaque port. Dans le cas d'un double module, elle occupe l'espace adressable de son voisin, et doit donc être installée à côté d'un élément ayant une adresse système ou d'un port vide. Les éléments ayant une adresse système sont : le module mémoire, le X-memory, le Time, l'HP-IL, et l'imprimante HP82143.

A chaque fois que le calculateur est allumé, le module teste automatiquement l'intégrité de la Ram, et la vide si elle a été altérée. Ceci n'arrivera que si le module a été ôté du calculateur, ou si les piles sont faibles depuis trop longtemps.

RESUME

Les modules HEPAX sont aussi simples à utiliser que le module Extended Memory, mais sont beaucoup plus puissants. Ils s'affranchissent par ailleurs de l'imperfection majeure du module Extended Memory : les programmes peuvent être exécutes directement dans le module HEPAX, sans avoir à les copier, effacer, etc...

Le module HEPAX vous fournit de nombreux outils de programmation en micro-code, de façon que vous puissiez écrire vos propres routines.

Les modules HEPAX ont une grande puissance pour un petit prix (NdT : c'est comme Mini Mir) :

- Le module HEPAX standard avec ses 8000 octets de mémoire, et toutes les fonctions supportées sera facturé DKK 1800, soit environ US$ 245.

Offre de lancement à : DKK 1440, soit environ US$ 195.

- Le module HEPAX évolué avec 16000 octets et toutes les fonctions supportées sera facturé DKK 2475, soit environ US$ 340.

Offre de lancement à : DKK 1980, soit environ US$ 270.

- Le module mémoire HEPAX avec uniquement ses 8000 octets de mémoire sera facturé DKK 1475, soit environ US$ 200.

Offre de lancement à : DKK 1180, soit environ US$ 160.

- Le double module mémoire HEPAX sera facturé DKK 2050, soit environ US$ 280.

Offre de lancement à : DKK 1640, soit environ US$ 220.

Pour tout renseignement complémentaire, contactez :

VM Electronics
Nyelandsvej 7, 1.
DK-2000 Frederiksberg
Denmark

RESUME DES FONCTIONS DU MODULE HEPAX

Ce qui suit est une liste non exhaustive des fonctions du module HEPAX. Pour chaque nom est donnée une courte description de la fonction. Veuillez noter que cette liste est sujette à des changements mineurs.

Fonctions système

Fonctions d'ordre général

`HCLFL` : efface le contenu d'un fichier texte ou données HEPAX.
`HEPDIR` : catalogue de tous les fichiers dans le fichier système.
`HEPDIRX` : catalogue d'un fichier dans le fichier système.
`HEPROOM` : donne le nombre de registres non utilisés dans le fichier système.
`HFLSIZE` : donne la taille de n'importe quel fichier.
`HPURFL` : efface un fichier dans le fichier système.
`HRCLPT` : rappelle le pointeur du fichier courant.
`HRCLPTA` : rappelle le pointeur de n'importe quel fichier.
`HREADFL` : lit n'importe quel fichier (HEPAX ou non) d'un support.
`HRENAME` : renomme un fichier dans le fichier système.
`HRESZFL` : redéfinit la taille d'un fichier texte ou données.
`HSEC` : protège tout fichier contre édition et effacement.
`HSEKPT` : place le pointeur dans le fichier courant.
`HSEKPTA` : place le pointeur dans le fichier spécifié.
`HUNSEC` : ôte la protection de tout fichier.
`HWRTFL` : écrit tout fichier HEPAX sur un support de masse.

Fonctions sur les fichiers de données

`HCRFLD` : crée un fichier HEPAX de données.
`HGETR` : prend tous les registres d'un fichier de données.
`HGETRX` : prend les registres suivant la valeur de X.
`HSAVER` : sauvegarde tous les registres d'un fichier de données.
`HSAVERX` : sauvegarde les fichiers suivant la valeur de X.
`HSAVEX` : sauvegarde le contenu de X.

Fonctions sur les fichiers texte

HAPPCHR : ajoute un caractère au fichier texte courant.
HAPPREC : ajoute un enregistrement au fichier texte courant.
HARCLRC : ajoute un enregistrement au registre ALPHA.
HASROOM : rend le nombre de caractères disponibles dans un fichier texte.
HCRFLAS : crée un fichier texte.
HDELCHR : efface un caractère d'un fichier texte.
HDELREC : efface un enregistrement d'un fichier texte.
HGETREC : écrit un enregistrement dans le registre ALPHA.
HINSCHR : insère un nombre de caractères dans un fichier texte.
HINSREC : insère un enregistrement dans un fichier texte.
HPOSFL : cherche une chaîne dans un fichier texte.

Autres fonctions

HGETA : prend un fichier "write-all" dans la mémoire HEPAX.
HGETK : prend un fichier d'assignation dans la mémoire HEPAX.
HSAVEA : copie un fichier "write-all" dans la mémoire HEPAX.
HSAVEK : copie le fichier d'assignations courant dans la mémoire HEPAX.
HSAVEMC : copie une fonction micro-code utilisateur dans la mémoire HEPAX.
PRIVATE : rend un programme privé dans la mémoire HEPAX.
ROMNAME : donne un nom à un bloc mémoire HEPAX.

Fonctions de programmation

Fonctions M-Code

CLRAM : efface un bloc dans la mémoire HEPAX.
CODE : code le registre ALPHA dans le registre X.
COPYROM : copie une Rom entière dans la mémoire HEPAX.
DECODE : décode le registre x dans le registre ALPHA.
DECODYX : décode une partie du registre Y sous contrôle de X.
DISSAS : désassembleur 41.
HEXEDIT : éditeur M-Code avancé.
HPROMPT : demande hexadécimale.

Autres fonctions

HEPAX : multifonction incluant 17 programmes utilitaires, ainsi que des fonctions supportées.
HEPAXA : la même avec entrées en ALPHA.
RAMTEST : teste la Ram HEPAX.
RAMTOG : inverse la protection d'écriture d'un bloc HEPAX.
READROM : lit l'image d'une Rom sur un support de masse.
ROMTEST : teste la somme de contrôle de n'importe quelle Rom.
WRTROM : écrit l'image d'une Rom sur un support de masse.
XF : multifonction incluant toute les fonctions du module Extended Functions (33 fonctions).
XFA : la même avec entrées en ALPHA.

Notes spéciales

HREADFL et HWRTFL

Ces fonctions sont compatibles avec les fichiers créés par l'HP-IL. Cela signifie que de tels fichiers peuvent être lus directement par le module HEPAX et vice-versa.

CRFLD

A l'aide de cette fonction, il est possible de créer des fichiers jusqu'à 5200 registres (avec 32K de mémoire HEPAX).

HSAVEMC

Conjointement avec la fonction HGETMC (dans la multifonction HEPAX), il devient possible de créer une libraire de vos propres fonctions microcodées. Les fonctions HGETMC et HSAVEMC sauvegardent et lisent des fonctions microcodées de et vers la mémoire HEPAX. Ces fonctions travaillent automatiquement, vous n'avez donc pas à vous préoccuper des FAT, sauts dans des ports dépendants, etc... ainsi, vous pouvez facilement faire une Rom à partir de la librairie HEPAX.

DISSAS

Désassemble correctement du M-code 41 dans toutes les conditions (inclusion de messages, entrées FAT, numéro de version etc...). Cette fonction utilise les mnémoniques et codes hexa de Jacobs/De Arras, mais peut aussi afficher des caractères ASCII, ou des caractères du HP-41. Elle s'utilise aussi bien avec l'écran qu'avec une imprimante.

HEXEDIT

Un éditeur de M-code simple et très puissant. Il contient des fonctions pour insérer, effacer et copier, ainsi que pour travailler dans différentes parties de la

Ram HEPAX. L'entrée de données est même plus facile qu'avec la ZENROM. Afin de ne pas trop consommer, cette fonction ne fait pas tourner le processeur tout le temps. Cependant, elle possède également un mode pour travailler avec les interruptions.

Le concept multifonction

Afin de donner à l'utilisateur la possibilité d'avoir accés à plus de 64 fonctions, des fonctions du module Extended Functions, ainsi que des fonctions évoluées ont été regroupées en multifonctions. Quand vous exécutez une multifonction, la machine attend un nom de fonction, ou un nombre de 2 chiffres. En mode programme, une multifonction apparaît sur 2 lignes, la première est la multifonction, la seconde représente le numéro de la fonction (ce nombre ne modifie en rien la pile).

Traduction : Laurent Istria (3)

PORTFOLIO

Voici encore un petit programme pour HP-41 sans prétention que j'ai établi pour mon plaisir.

Le portfolio est une loterie journalière organisée par *Le Figaro*. Les joueurs ont des cartes sur lesquelles figurent 8 numéros différents de la série des entiers de 1 à 44.

Chaque jour, le journal associe une valeur différente à chacun de ces numéros et publie le total des 8 valeurs d'une même carte pour qu'elle gagne. Ceci représente un travail fastidieux ; d'où l'idée de grouper dans un programme unique la surveillance et le calcul des résultats d'un certain nombre de cartes.

Le programme fourni prend en compte n = 23 cartes.

Pour l'utiliser, il faut charger le programme FOLIO, puis enregistrer les données dans les registres 65 à 110. Il faut enregistrer les données d'une carte sur deux registres. Par exemple, la première carte ayant les 8 valeurs 2, 4, 12, 17, 24, 33, 34 et 39 est enregistrée selon le format suivant :
R65 = 2041217
R66 = 24333439

On voit donc que le nombre de registres utilisés est $2n$, n étant le nombre de cartes.

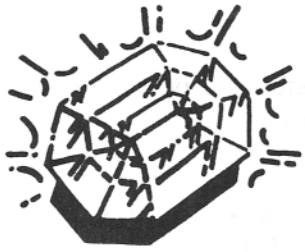
L'exécution du programme commence par la saisie des valeurs associées du jour à l'appel de leur numéro d'ordre (de 1 à 44). Ces valeurs sont stockées dans les registres 21 à 64. Les lignes 13 à 61 du programme calculent les valeurs des n cartes successives et les stockent dans les registres 1 à n (23 dans l'exemple). Le nombre de contrôle à la ligne 14 doit être 65,110 représentant une boucle de 65 à $65+2n-1$.

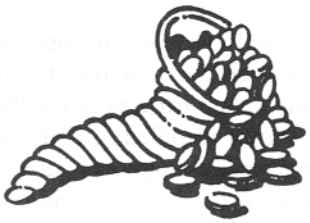
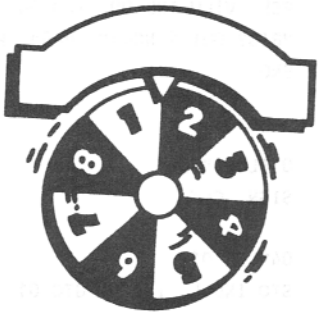
Les résultats sont lus sur l'écran du HP-41 et s'il y a lieu, imprimés sur l'imprimante ou l'écran vidéo par l'interface. On peut itérer cette dernière opération lorsque survient l'affichage "BIS? OUI=1 NON=0", en introduisant 1, puis [R/S].

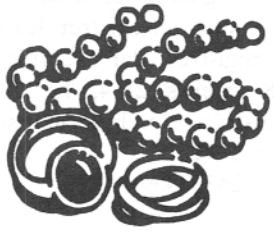
René Dine (293)

```
01*LBL "FOLIO"
FIX 0  21,064  STO 00

05*LBL 01
RCL 00  INT  20  -  STOP  STO IND 00  ISG 00
GTO 01  65,110  STO 00

16*LBL 04
RCL IND 00  1  E6  /  20  +  RCL IND X  X<>Y
XEQ A  R^  XEQ A  SF 02  R^

29*LBL A
FRC  100  *  20  +  RCL IND X  X<>Y  RDN  +
FC?C 02  RTN  X<>Y  FS?C 01  GTO 03  RDN  1
ST+ 00  RDN  SF 01  GTO 04

50*LBL 03
+  RCL 00  64  -  2  /  SIGN  RDN  STO IND L
ISG 00  GTO 04  BEEP  BEEP

64*LBL 06
1,023  STO 00

67*LBL 05
PSE  VIEW IND 00  ISG 00  GTO 05
"BIS? OUI=1 NON="  "⊢0"  PROMPT  X≠0?  GTO 06
END


01*LBL "TCL"
SIGN  CLX

04*LBL 01
STO IND L  ISG L  GTO 01  END
```

R65=	2041217,00	R66=	24333439,00
R67=	2071320,00	R68=	25293537,00
R69=	3071218,00	R70=	24303643,00
R71=	4102022,00	R72=	24323743,00
R73=	9111820,00	R74=	28303640,00
R75=	2051920,00	R76=	29303637,00
R77=	8111214,00	R78=	27324042,00
R79=	3051617,00	R80=	26283944,00
R81=	3051617,00	R82=	27324344,00
R83=	1061215,00	R84=	24333537,00
R85=	9111416,00	R86=	24324143,00
R87=	1061420,00	R88=	30334044,00
R89=	7091416,00	R90=	25323443,00
R91=	5061222,00	R92=	23253444,00
R93=	6072022,00	R94=	25304144,00
R95=	2041418,00	R96=	23323441,00
R97=	3051721,00	R98=	26323639,00
R99=	3061820,00	R100=	25293942,00
R101=	8091419,00	R102=	25334044,00
R103=	2071314,00	R104=	30323443,00
R105=	3081417,00	R106=	25333544,00
R107=	4091222,00	R108=	28303537,00
R109=	7101415,00	R110=	24274042,00





HP75

J. Y. Hervé — Récupération de fichiers avec MADLEX *18*

RECUPERATION DE FICHIERS
UTILISATION DE MADLEX

Récupérer un fichier purgé sur cassette ou sur disquette implique de connaître l'emplacement physique de l'entrée de ce fichier dans le catalogue de la mémoire de masse, puis de modifier à cet emplacement les octets concernés.

Pour qu'une telle opération soit possible, il faut bien évidemment que le fichier concerné n'ait pas été écrasé entretemps par l'écriture intempestive d'un autre fichier. Il faut donc proscrire, dans la mesure du possible, tout nouvel ordre d'écriture sur la mémoire de masse avant l'opération de récupération.

Recherche du fichier purgé

L'utilitaire DIRPUR permet d'obtenir la liste de tous les fichiers purgés présents sur un support magnétique et, si l'on veut, de récupérer l'un d'eux. Il donne le numéro d'ordre séquentiel *N* des catalogues concernés sur le support et le nom des fichiers purgés correspondants.

Ce programme pourra être modifié en fonction des besoins de chacun, par exemple, en imposant la recherche d'un fichier déterminé par son nom.

Calcul de l'emplacement des octets à modifier

Quand le numéro d'ordre *N* du fichier purgé est repéré, DIRPUR calcule alors où se trouvent les octets indiquant au système d'exploitation que ce fichier doit être considéré comme purgé (octets caractérisant le type dans l'entrée de catalogue), soit :

- son numéro d'enregistrement (*record*) : `R=IP(N/8)+2`

- l'octet dans l'enregistrement : `O=MOD(N*32+10,256)`

Modification des octets

Pour modifier les deux octets concernés, l'utilitaire n'a plus alors qu'à exécuter l'ordre d'écriture directe `FWRITE$` du fichier MADLEX, en donnant les valeurs suivantes aux deux octets :

Premier octet :
caractère de code 0 pour les fichiers ASCII/LIF1 (type I)
caractère de code 224 pour tous les autres fichiers

Deuxième octet :
caractère de code 136 pour les fichiers BASIC (B)
caractère de code 82 pour les fichiers TEXT (T)
caractère de code 83 pour les fichiers APPT (A)
caractère de code 137 pour les fichiers LEX (L)
caractère de code 1 pour les fichiers ASCII/LIF1 (I)

Exemple

Recherche d'un fichier nommé TEST, purgé. L'utilitaire indique 54 comme numéro d'ordre. La position des octets à modifier sera donc :

```
R = IP(54/8)+2 = 8
O = MOD(54*32+10,256) = 202
```

Comme il s'agit d'un fichier Basic (type B), l'utilitaire enverra l'ordre suivant :

```
FWRITE CHR$(224)&CHR$(136),":M1",8,202
```

Un `CAT "TEST:M1"` permetttra de vérifier que le fichier a effectivement été récupéré, et qu'il s'agit bien d'un fichier Basic.

NDLR : L'auteur a publié l'an passé ce Lex (Micro-Revue 14, pages 43 à 46), pour que les possesseurs de HP-75 puissent enfin créer et exploiter de grands fichiers de données en accès direct sur disquette. Il nous donne ici quelques illustrations des possibilités offertes, avec la récupération d'un fichier purgé.

Jean-Yves Hervé

```
0010  ! DIRPUR avec MADLEX (c) J.Y.HERVE
0020  ! Récupération de fichier purgé
0030  DIM C$[12]
0040  DELAY 30
0050  ! Recherche des fichiers purgés
0060  N=0 @ R=2 @ O=0
0070  C$=FREAD$(12,":M1",R,O)
0080  IF C$[11,11]=CHR$(255) THEN 120
0090  IF C$[12,12]=CHR$(0) THEN DISP "N";N,C$[1,8]
0100  O=O+32 @ IF O>230 THEN O=0 @ R=R+1
0110  N=N+1 @ GOTO 70
0120  DELAY 0
0130  ! Calcul des octets à modifier
0140  INPUT "N du fichier ? ";N
0150  INPUT "Type ? ","B";C$
0160  R=POS("BTALI",C$) @ IF R=0 THEN 150
0170  C$=CHR$(224) @ I R=5 THEN C$=CHR$(0)
0180  FOR I=1 TO R @ READ O @ NEXT I
0190  DATA 136,82,83,137,1
0200  C$=C$&CHR$(O) @ R=IP(N/8)+2
      @ O=MOD(N*32+10,256)
0210  ! Modification des octets
0220  FWRITE$ C$,":M1",R,O
```

ASSEMBLEUR

J. Baudier	L'assembleur du HP-71 (acte V)	*20*
J. Y. Naour	Des caractères pour faire joli	*27*
J. Elhay	Evaluez vos polynômes	*28*
G. Toublanc	Arrondir dans la bonne base	*29*

BASIC

B. Gil	Dérivées symboliques (acte I quater...)	*33*
O. Arbey	La guerre des noyaux (acte II)	*33*
E. Gengoux	Un catalogue ordonné	*34*
J. Maille	Ca date !	*35*
B. Gil	Programme "DIFF"	*36*
E. Gengoux	Programme "FASTCAT"	*39*
J. Maille	Programme "GREGOR"	*40*
J. Maille	Programme "DIFFDATE"	*41*

LE COIN DES LHEX

41

L'ASSEMBLEUR DU HP-71
(ACTE V)

Chers amis, bonjour... Cette fois-ci, nous allons continuer notre progression dans l'étude des fonctions écrites en assembleur. Lors de l'acte IV, nous avons vu comment faire pour écrire un Lex apportant une fonction sans paramètre qui renvoie une constante, comme par exemple les fonctions HPI et MYPPC.

Pour continuer sur cette lancée, je vous propose le petit Lex suivant qui a l'avantage d'utiliser l'accès mémoire, comme nous l'avons vu dans l'acte III, et une fonction du style de celle de l'acte IV. Le fichier source CONTLEXS qui apporte une nouvelle fonction CONTRAST qui renvoie la valeur actuelle du contraste.

Quelque part dans sa mémoire, le HP-71 réserve un quartet pour y stocker la valeur du contraste actuel. Ce quartet se trouve à l'adresse #2E3FE. Cette valeur peut donc varier de 0 à 15 (F en hexadécimal). Jusqu'à présent, nous disposions de l'ordre (et non de la fonction) CONTRAST de base qui permet de modifier ce contraste. Ainsi CONTRAST 5 met le contraste à 5 en chargeant 5 dans le quartet d'adresse #2E3FE. On peut également faire la même chose, si l'on bidouille, en utilisant l'ordre POKE "2E3FE","5". Pour récupérer la valeur du contraste, jusqu'à maintenant, la seule solution était de faire PEEK$("2E3FE",1). Voici donc le Lex qui comble une lacune du Basic HP ! Les explications le suivent.

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

DCONTR  EQU     #2E3FE    Display contrast quartet
HDFLT   EQU     #1B31B    Convert HEX integer
         *                to DEC floating point
FNRTN2  EQU     #0F219    Retour avec PC dans A[A]

        ENTRY   CONTe
        CHAR    #F
        KEY     'CONTRAST'
        TOKEN   6
        ENDTXT

        NIBHEX  00        Pas de paramètre
CONTe   ADOEX             Sauvegarde de D0
        R0=A              dans R0
        D0=(5)  DCONTR
        A=0     A
        A=DAT0  1         Chargement du quartet
        GOSBVL  HDFLT     Conversion en nbre flottant
        C=A     W         Copie dans C pour FNRTN1
        A=R0              Met PC dans A[A]
        GOVLNG  FNRTN2

        END
```

Explications :

Comme pour les fonctions vues jusqu'à présent on indique l'absence de paramètre d'appel de notre fonction par un classique NIBHEX 00. En entrant dans un Lex, on se retrouve avec un certain environnement que nous laisse le système. En particulier, comme nous l'avons vu la dernière fois, D0 et D1 contiennent des valeurs que l'on se doit de restituer en sortie si l'on veut éviter d'avoir des problèmes. D0 contient le PC (Program Counter) et D1 contient l'adresse du haut de la Math Stack. Mais nous verrons plus tard ce que sont PC et Math Stack. Comme l'on désire accéder à la mémoire du HP-71 pour lire le contraste, on va être amené à se servir soit de D0 soit de D1 (voir l'acte III). J'ai choisi de prendre D0, mais le choix de prendre D1 aurait été aussi valable. On va charger dans D0 la valeur DCONTR, c'est à dire l'adresse #2E3F2. A ce moment on dit que D0 pointe DCONTR (que je noterai D0 ^ DCONTR). On met ensuite le champ A du registre A à 0. Puis on charge le quartet désiré avec A=DAT0 1. A ce moment, on se retrouve avec A[A]=0000*n*, où *n* est la valeur en hexadécimal du contraste. Il va falloir maintenant mettre ce nombre sous la forme habituelle des nombres dits en *représentation virgule flottante*, c'est à dire la forme utilisée lors de l'article précédent avec un quartet de signe, 12 de mantisse et 3 d'exposant. Comme d'habitude, HP a bien fait les choses puisqu'une routine appelée HDFLT réalise la conversion hexadécimal vers flottant. Cette routine convertit donc A[A] hexadécimal en A[W] flottant. Si l'on regarde les *I.D.S.* (II ou III) on voit que cette routine utilise A,B,C et D. C'est pour cela que l'on a sauvegardé D0 dans R0 et non dans C par exemple.

En sortie de cette routine, il ne reste plus qu'à mettre A dans C pour l'appel de FNRTN2. Ensuite, on récupère la valeur de PC sauvegardée dans R0. Pour finir, on pouvait soit remettre PC dans D0 et appeler FNRTN1 soit laisser PC dans A[A] et appeler FNRTN2. J'ai donc pris cette deuxième solution qui nous fait économiser 3 quartets. Voilà, il reste alors à assembler et à essayer. Vous vous demandez peut être comment le HP-71 fait-il pour reconnaître cette nouvelle fonction alors qu'il possède déjà un ordre CONTRAST. Ceci s'explique par le fait que le HP-71 cherche d'abord dans les Lex les fonctions ou ordres avant d'aller voir dans ses Roms. On peut ainsi réécrire ou détourner les instructions du Basic.

LES FONCTIONS AVEC DES PARAMETRES

Jusqu'à ce moment, nous n'avons vu que les fonctions renvoyant un nombre et ne prenant pas de paramètre. Voyons alors comment on peut en passer à des fonctions du style de SGN et SQR par exemple. C'est à dire des fonctions acceptant un paramètre numérique en entrée. Comme pour celles vues jusqu'à présent, on doit placer devant le point d'entrée d'une nouvelle fonction plusieurs quartets qui indiquent le nombre valide d'arguments et leurs types. Ainsi pour les fonctions sans argument, on mettait NIBHEX 00. Tout de suite avant le point d'entrée, on place un quartet indiquant le nombre maximum d'arguments. Juste au dessus, on indique le nombre minimum. Et en continuant en remontant, on place pour chaque paramètre un quartet indiquant son type. Ainsi on met 8 pour un nombre, 4 pour une chaîne de caractères, A (8+2) pour un tableau de nombres et 6 (4+2) pour un tableau de chaînes de caractères. Par exemple, pour les fonctions suivantes, on aura :

Pour SIN(n) :

```
NIBHEX 8  :paramètre numérique
NIBHEX 11 :on a un paramètre obligatoire
```

Pour POS(a$,b$[,c]) :

```
NIBHEX 8  :3ème paramètre : num
NIBHEX 4  :2ème : alpha
NIBHEX 4  :1er : alpha
NIBHEX 23 :2 paramètres minimum et 3 maximum
```

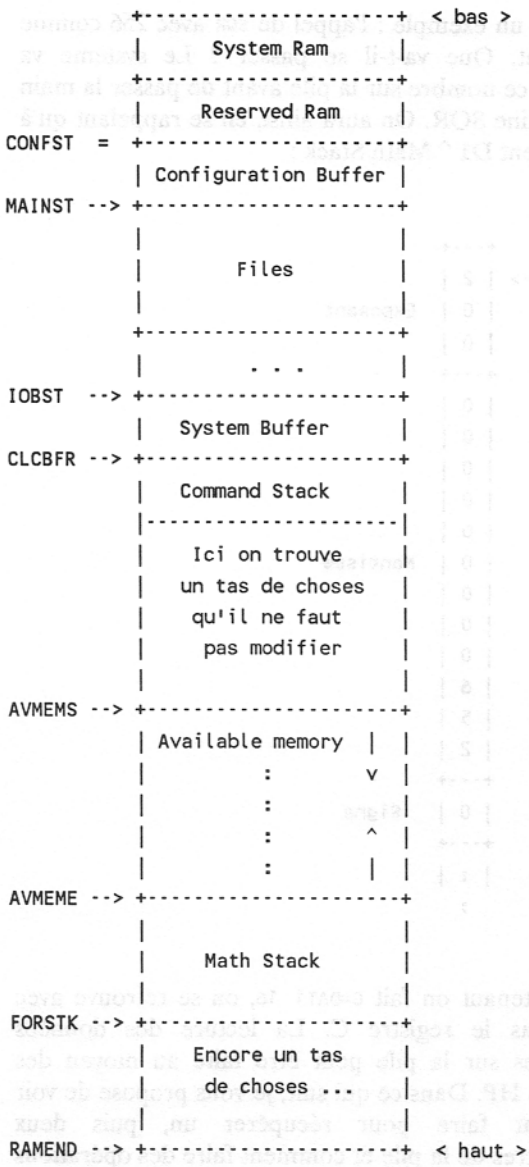
On peut bien entendu écrire ce dernier exemple sous la forme NIBHEX 84423.

Lors de l'appel de fonction à partir du clavier ou d'un programme, les arguments, s'ils existent, sont placés dans un endroit spécial de la mémoire appelé la Math Stack. Cette Math Stack est une pile (stack = pile en anglais) qui sert d'intermédiaire entre le système qui y dépose des choses et les routines des fonctions appelées qui les récupère.

Etudions plus en détail la place en mémoire et le fonctionnement de la Math Stack.

La Math Stack

Tout d'abord la Math Stack est placée en Ram comme l'indique le schéma suivant. Pour ce qui est du reste de ces zones, nous verrons plus tard.



Ce schéma détaille la partie de mémoire située entre la Ram système et RAMEND (fin de mémoire vive). En fin de mémoire (adresses les plus hautes) se trouvent tout un tas de piles dont la Math Stack. Comme on peut s'en douter, tout ce petit monde n'arrête pas de jouer à l'ascenseur. Ainsi AVMEMS (AVailable MEMory Start = début de mémoire disponible) descend lors du chargement ou la création d'un fichier et remonte (ouf !) lors de l'utilisation de PURGE. La Math Stack, quant à elle, monte vers AVMEMS. Lorsque l'on dépose des choses sur la pile, il faut s'assurer que AVMEME (AVailable MEMory End = fin de mémoire) ne remonte pas au dessus de AVMEMS. RAMEND, quant à lui, est fixe et n'est modifié que par l'ajout d'un module de Ram.

Ainsi, lors de l'appel d'une fonction, le HP-71 dépose sur la Math Stack le ou les arguments. Les habitués des piles des HP-41 et autres vont pouvoir s'amuser !

Prenons un exemple : l'appel de SQR avec 256 comme argument. Que va-t-il se passer ? Le système va déposer ce nombre sur la pile avant de passer la main à la routine SQR. On aura ainsi, en se rappelant qu'à ce moment D1 ^ Math Stack :

```
< bas >    +---+
   D1 --> | 2 |
          | 0 |  Exposant
          | 0 |
          +---+     -
          | 0 |
          | 0 |
          | 0 |
          | 0 |
          | 0 |
          | 0 |  Mantisse
          | 0 |
          | 0 |
          | 0 |
          | 6 |
          | 5 |
          | 2 |
          +---+     -
          | 0 |   Signe
          +---+
< haut >  | : |
            :
```

Si maintenant on fait C=DAT1 16, on se retrouve avec 256 dans le registre C. La lecture des données présentes sur la pile peut être faite au moyen des routines HP. Dans ce qui suit, je vous propose de voir comment faire pour récupérer un, puis deux paramètres de la pile et comment faire des opérations dessus. Pour cela, nous allons étudier plusieurs fonctions qui sont toutes regroupées dans un même Lex dont le source est placé placé à la fin de cet article.

CHS

Comme son nom l'indique peut-être, CHS est une fonction qui renvoie l'opposé de l'argument. Ainsi CHS(-2) renvoie 2 et CHS(INF) -Inf. Comment allons nous écrire cette fonction ? En entrée de notre Lex, D1 pointe le haut de la Math Stack ceci est parfait car la routine POP1N va se charger de dépiler la donnée (pop en anglais) et la mettre dans le registre A. On doit maintenant changer le signe du nombre contenu dans A. Pour cela, on va d'abord le mettre dans C et utiliser le registre de 4 bits P comme intermédiaire. Une fois le travail terminé, on utilisera la routine FNRTN1 déjà vue. Le corps de cette fonction est donc :

```
        NIBHEX 8          le paramètre est numérique
        NIBHEX 11         on aura un seul paramètre

CHSe    GOSBVL POP1N      on dépile la donnée
        C=A    W          on la met dans C[W]
        P=C    15         met le signe (C[S]) dans P
        ?P=    0          signe positif ?
        GOYES  neg        si oui : saut à neg
pos     P=     0          on met un signe +
        GOTO   fin
neg     P=     9          on met un signe -
fin     C=P    15         on le remet dans C[S]
        GOVLNG FNRTN1     on s'en va ...
```

Ceci n'est pas bien compliqué, n'est-ce pas ? Le passage par le registre P est bien pratique et plus simple pour manipuler et faire des tests sur un quartet. Après avoir dépilé la donnée et l'avoir mise dans C, on recopie le chiffre de signe C[S] (quartet 15) dans P. P est-il égal à zéro ? Si oui on saute (instruction GOYES) au label neg où l'on met 9 dans P, sinon on continue gentiment en mettant 0 dans P et en sautant vers fin pour éviter l'instruction P= 9. A ce moment P contient le nouveau signe. On s'empresse alors de remettre cette valeur dans C[S] et de repartir avec FNRTN1. On pourrait de la même manière réécrire la fonction ABS qui se chargerait de mettre C[S] à zéro.

ADDONE

La fonction ADDONE est une fonction très utile (?) puisqu'elle permet d'ajouter un à un nombre. Comme CHS, il s'agit d'une fonction qui ne prend qu'un seul argument. La syntaxe est donc ADDONE(*n*) qui renvoie n+1. Pour écrire cette fonction nous allons utiliser la routine HP dont le nom est ADDONE.

Mais avant de voir comment réaliser une telle fonction, je vous propose un petit rappel sur la représentation interne des données numériques dans les registres du processeur du HP-71. Nous avons déjà vu dans les articles précédents la représentation avec 12 chiffres de mantisse. Cependant, il existe une autre représentation interne qui permet, en utilisant deux registres, d'avoir 15 chiffres de mantisse. Ceci permet, lors des calculs internes, d'avoir une plus grande précision. La représentation 15 chiffres occupe donc deux registres comme suit :

Premier registre :

```
15                          0
+-+-------------------------+
|S|                   | Exp |
+-+-------------------------+
 1         10            5      = 16
```

Deuxième registre :

```
15                          0
+-+-------------------------+
| | Mantisse                |
+-+-------------------------+
 1           15               = 16
```

Dans les *I.D.S.* la forme 12 chiffres est souvent appelée *forme externe* et la forme 15 chiffres *forme interne*. Bien entendu, il existe dans la Rom système des routines permettant de faire les conversions d'une forme vers l'autre. Ainsi la routine SPLITA transforme le nombre contenu dans A sous forme externe en une forme interne dans les registre A et B, SPLITC transforme C (12 chiffres) en C et D (15 chiffres) et la routine SPLTAC fait ces deux transformations d'un coup. Enfin, la routine uRES12 se charge de remettre le contenu des registres A et B dans C sous forme 12 chiffres. Pour simplifier l'écriture, on appelle le nombre contenu dans les registres A et B AB, et celui contenu dans C et D CD. En plus de ces deux registres, les routines de calcul du HP-71 en utilisent deux autres : XM (eXternal Module missing bit) et SB (Sticky Bit) pour indiquer si le résultat est exact (SB=0) ou approché (SB=1) ou donne un résultat exceptionnel (division par zéro, opération invalide, etc.. XM=1). Généralement, les routines de calcul initialisent SB et XM à 0 avant de commencer. Mais ADDONE ne le faisant pas, c'est à nous de le faire avant de l'appeler. Ainsi la routine uRES12 pourra, en sortie de ADDONE, en fonction de XM et SB ainsi qu'en fonction des valeurs des trappes, générer le message d'erreur ou d'avertissement.

Pour écrire notre fonction, nous utiliserons donc POP1N qui prend l'argument de la Math Stack pour le mettre dans A, SPLITA qui met A dans AB, ADDONE qui réalise AB = AB + 1 et enfin nous appellerons uRES12 qui transforme AB en C (forme externe) et quitterons la fonction par un appel à FNRTN1 comme nous en avons l'habitude maintenant.

Ce qui nous donne donc :

```
      NIBHEX 811        Un paramètre numérique

ADDe  GOSBVL POP1N      Récupère le paramètre
      GOSBVL SPLITA     On le met sous forme interne
      XM=0              Met 0 dans XM
      SB=0              Et dans SB
      GOSBVL ADDONE     On ajoute 1
      GOSBVL uRES12     Mise sous forme 12-chiffres
      GOVLNG FNRTN1     Retour avec résultat dans C
```

INVERSE

La fonction INVERSE renvoie l'inverse du paramètre que l'on lui donne. Ainsi, INVERSE(2) renvoie 0.5. Cette fonction, tout comme ADDONE, accepte un seul paramètre numérique. Je ne vais pas détailler la routine de INVERSE car elle est quasiment identique à celle de ADDONE. La seule différence réside dans le fait que l'on utilise la routine UN/X15 (qui renvoie AB=1/AB) au lieu de ADDONE. On a donc :

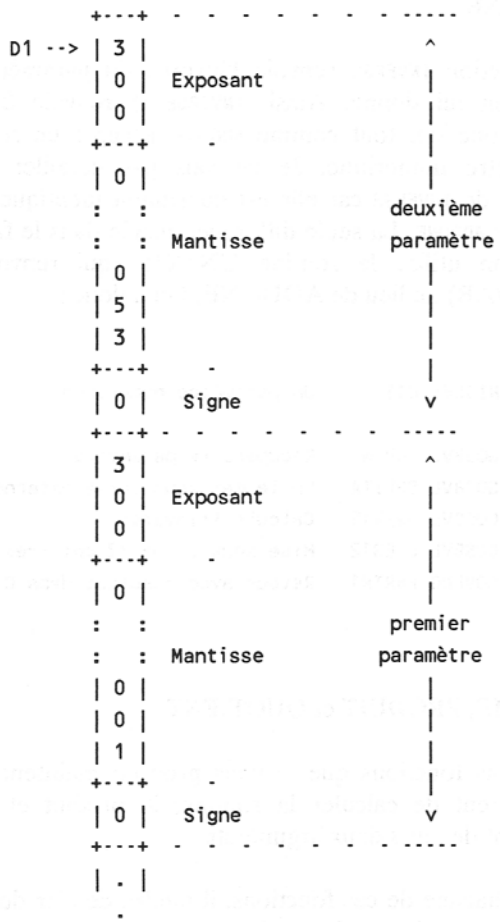
```
      NIBHEX 811        Un paramètre numérique

INVe  GOSBVL POP1N      Récupère le paramètre
      GOSBVL SPLITA     On le met sous forme externe
      GOSBVL UN/X15     Calcule l'inverse
      GOSBVL uRES12     Mise sous forme 12 chiffres
      GOVLNG FNRTN1     Retour avec résultat dans C
```

SOMME, PRODUIT et QUOTIENT

Les trois fonctions que je vous propose maintenant permettent de calculer la somme, le produit et le quotient de leurs deux arguments.

Pour chacune de ces fonctions, il faudra dépiler deux nombres de la Math Stack. Ceci peut se faire en utilisant POP1N deux fois de suite, ou bien en utilisant un autre point d'entrée : POP2N qui renvoie les deux arguments sous forme 12-chiffres dans A et C.

Lorsqu'on appelle une fonction, le HP-71 empile les arguments sur la Math Stack dans l'ordre de lecture, c'est-à-dire de gauche à droite. Ainsi, le premier argument se retrouve en bas de la pile et les suivants au dessus avec D1 pointant donc sur le dernier d'entre eux. Ce qui donne dans le cas de l'appel de MAX(100,350) :



Si l'on appelle maintenant POP2N, on aura A = 100 et C = 350. En utilisant alors SPLTAC, on aura bien 100 dans AB et 350 dans CD. Pour écrire nos trois nouvelles fonctions, on va donc faire appel à ces deux routines POP2N et SPLTAC. Ensuite, on utilisera AD2-15, MP2-15 et DV2-15 qui renvoient respectivement AB = AB+CD, AB = AB.CD et AB = AB/CD. Enfin, comme pour les fonctions vues auparavant, on se servira de uRES12 et FNRTN1. Ceci nous donne donc :

```
        NIBHEX 8811     Deux paramètres numériques

SOMe    GOSBVL POP2N    Récupère les paramètres
        GOSBVL SPLTAC   On le met sous forme int.
        GOSBVL AD2-15   On calcule AB = AB+CD
        GOSBVL uRES12
        GOVLNG FNRTN1

        NIBHEX 8811     Deux paramètres numériques

PROe    GOSBVL POP2N
        GOSBVL SPLTAC
        GOSBVL MP2-15   On calcule AB = AB.CD
        GOSBVL uRES12
        GOVLNG FNRTN1

        NIBHEX 8811     Deux paramètres numériques

QUOe    GOSBVL POP2N
        GOSBVL SPLTAC
        GOSBVL AD2-15   On calcule AB = AB/CD
        GOSBVL uRES12
        GOVLNG FNRTN1
```

Avant de finir cette article, je voudrais résumer ce que nous avons vu sur les fonctions numériques :
- Les fonctions vont lire leur paramètres sur la Math Stack (on utilise les routines POP1N, POP2N et leurs cousines (cf. *I.D.S.*)).
- Pour avoir une meilleure précision, on convertit les arguments dans une forme 15 chiffres (routines SPLITA, SPLITC et SPLTAC).
- Pour effectuer les calculs, on dispose d'un ensemble complet de routines comme AD2-15, MP2-15 etc ... (cf. *I.D.S II* chapitre 13 : System Math Fonctions).
- Les résultats de calculs peuvent être réutilisés comme arguments pour d'autres opérations ou remis sous la forme externe (routine uRES12).
- Le retour peut se faire par l'appel aux routines du type de FNRTN1.

De plus, et d'une manière générale, lors de l'écriture d'un Lex il faut faire attention (si l'on veut éviter le Memory Lost) à rendre D0 (=PC) et D1 (^ Math Stack) ce qui veut dire qu'il faudra sauvegarder les valeurs qu'ils contiennent si l'on a l'intention d'utiliser ces registres (pour faire des accès mémoire par exemple comme pour la fonction CONTRAST vue au début de cet article). Il faut également, lorsqu'on utilise une routine HP, veiller à bien passer les bons paramètres (sans rien oublier) et vérifier que la dite routine n'utilise pas des registres ou des ressources dans lesquelles on a sauvegardé quelque chose.

Je vous laisse donc étudier le fichier source NUMLEXS dans lequel, pour réduire la taille et augmenter la lisibilité, j'ai utilisé des sous-routines. Vous y verrez que les nouvelles instructions Basic qu'apporte ce Lex sont classées dans l'ordre alphabétique. Ceci est indispensable au bon fonctionnement de ce Lex et permet au HP-71 de faciliter ses recherches.

Enfin la documentation suivante vous permettra d'en savoir plus sur ce que nous avons abordé aujourd'hui dans cet article :

Dans les *I.D.S.* :
Volume I, Chapitre 3 : Memory Structure :
- 3.2 sur le quartet de contraste et compagnie.
- 3.3.4 sur l'emplacement de la Math Stack en Ram.
- 3.5.7 sur les choses que l'on peut trouver sur la Math Stack.

Volume I, Chapitre 7 : Statement Parse, Decompile, and Execution :
- 7.4.2 et 7.4.3 toujours sur la Math Stack.

Volume I, Chapitre 9 : Utilities :
- 9.3.2 sur les routines permettant de récupérer des objets sur la Math Stack.
- 9.6 sur les routines mathématiques et de conversion disponibles.

Volume I, Chapitre 13 : Internal Data Représentation :
- 13.1 et 13.2 sur les formats 12 et 15-chiffres.
- 13.4 sur le même sujet.

Volume I, Chapitre 14 : Numeric Computation Algorithms :
- 14.1 sur des généralités à propos du calcul interne.

Volume II, Chapitre 5 : Conversion Utilities :
- 5.3 sur HDFLT.

Volume II, Chapitre 13 : System Math Functions :
- 13.2 sur 1/X15 (ou UN/X15).
- 13.1 sur ADDONE.
- 13.3 sur AD2-15.
- 13.4 sur MP2-15.
- 13.5 sur DV2-15.

Volume II, Chapitre 14 : Math Stack Utilities :
- 14.2 sur POP1N.
- 14.1 sur POP2N.

Volume II, Chapitre 15 : System Level Math Utilities :
- 15.14 sur SPLITA.
- 15.17 sur SPLTAC.
- 15.19 sur uRES12.

Volume II, Les points d'entrée suivants :

```
HDFLT  (#1B31B) 1/X15  (#0C33E) ADDONE (#0C330)
AD2-15 (#0C363) MP2-15 (#0C43A) DV2-15 (#0C4AC)
POP1N  (#0BD1C) POP2N (#0BC8C)  SPLITA (#0C6BF)
SPLTAC (#0C934) uRES12 (#0C994) FNRTN1 (#0F216)
```

Dans JPC :

JPC 25 (juin 85) :
- Les variables et la Math stack de Michel Martinet page 48.
- A propos de combinaisons et d'arrangements de Laurent Istria page 50.
- Heures-minutes-secondes ou les nombres réels de Michel Martinet (encore lui !) page 52.)

JPC 41 (février 87) :
- Combinaisons et arrangements de Guy Toublanc page 32.

Nous voici arrivés au terme de cet article qui ferme la série de ceux consacrés aux fonctions numériques. J'espère avoir été suffisamment clair. Sinon, n'hésitez pas à me faire parvenir vos questions (elles seront les bienvenues) ou vos remarques pour, s'y besoin est, approfondir encore le sujet.

A partir de la prochaine fois, nous aborderons les fonctions alphanumériques. A bientôt donc...

Jacques Baudier (192)

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

      TITLE  diverses routines - J.B. octobre 87
       *     pour exemple dans ACTE V

************ TABLES DES EQUIVALENCES *************

POP1N  EQU    #0BD1C   dépile un argument de la MS
POP2N  EQU    #0BC8C   dépile deux arguments
SPLITA EQU    #0C6BF   transforme A en AB
SPLTAC EQU    #0C934   transforme A en AB, C en CD
ADDONE EQU    #0C330   AB = AB + 1
UN/X15 EQU    #0C33E   AB = 1/AB
AD2-15 EQU    #0C363   AB = AB + CD
MP2-15 EQU    #0C43A   AB = AB . CD
DV2-15 EQU    #0C4AC   AB = AB / CD
uRES12 EQU    #0C994   met AB (15) dans C(12)
FNRTN1 EQU    #0F216   évalue C et termine le Lex

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

      ENTRY  ADDe     point d'entrée de ADDONE
      CHAR   #F       fonction
      ENTRY  CHSe     point d'entrée de CHS
      CHAR   #F       fonction
      ENTRY  DIVe     point d'entrée de DIVISE
      CHAR   #F       fonction
      ENTRY  INVe     point d'entrée de INVERSE
      CHAR   #F       fonction
      ENTRY  PRODe    point d'entrée de PRODUIT
      CHAR   #F       fonction
      ENTRY  SOMe     point d'entrée de SOMME
      CHAR   #F       fonction

      KEY    'ADDONE'
      TOKEN  7
      KEY    'CHS'
      TOKEN  8
      KEY    'DIVISE'
      TOKEN  9
      KEY    'INVERSE'
```

```
       TOKEN  10
       KEY    'PRODUIT'
       TOKEN  11
       KEY    'SOMME'
       TOKEN  12

       ENDTXT

*************************************************
*                                               *
* CHS     : renvoie l'opposé du nombre          *
*                                               *
*************************************************

       NIBHEX 8        un paramètre numérique
       NIBHEX 11       obligatoire
CHSe   GOSBVL POP1N    recupère le paramètre
       C=A    W        le met dans C[W]
       P=C    15       met le signe (C[S]) dans P
       ?P=    0        signe positif ?
       GOYES  neg
pos    P=     0        on met un signe +
       GOTO   ok
neg    P=     9        on met un signe -
ok     C=P    15       on remet dans C[S]
       GOTO   fnrtn1   on s'en va ...

*************************************************
*                                               *
* ADDONE  : ajoute un à un nombre               *
*                                               *
*************************************************

       NIBHEX 811      un paramètre numérique
ADDe   GOSUB  take1    met le paramètre dans AB
       GOSBVL ADDONE   ajoute 1
       GOTO   fin

*************************************************
*                                               *
* INVERSE : renvoie l'inverse du nombre         *
*                                               *
*************************************************

       NIBHEX 811      un paramètre numérique
INVe   GOSUB  take1    met le paramètre dans AB
       GOSBVL UN/X15   calcule l'inverse
       GOTO   fin

*************************************************
*                                               *
* SOMME   : renvoie la somme des deux arguments *
*                                               *
*************************************************

       NIBHEX 8        1er  paramètre : numérique
       NIBHEX 8        2éme paramètre : numérique
       NIBHEX 22       deux paramètres

SOMe   GOSUB  take2    1er paramètre dans AB et
         *             2éme dans CD
       GOSBVL AD2-15   AB = AB + CD
       GOTO   fin

*************************************************
*                                               *
* PRODUIT : renvoie le produit des deux         *
*           arguments                           *
*                                               *
*************************************************

       NIBHEX 8
       NIBHEX 8
       NIBHEX 22
PRODe  GOSUB  take2
       GOSBVL MP2-15   AB = AB . CD
       GOTO   fin

*************************************************
*                                               *
* DIVISE  : renvoie le quotient des deux        *
*           arguments                           *
*                                               *
*************************************************

       NIBHEX 8
       NIBHEX 8
       NIBHEX 22
DIVe   GOSUB  take2
       GOSBVL DV2-15   AB = AB / CD
       GOTO   fin

*************************************************
* take1 : sous-programme                        *
*                                               *
* BUT   : depile un nombre de la MATH STACK     *
*         le met sous la forme 15 chiffres dans *
*         AB                                    *
*                                               *
*************************************************

take1  GOSBVL POP1N
       GOSBVL SPLITA
       RTN

*************************************************
* take2 : sous-programme                        *
*                                               *
* BUT   : depile deux nombres de la MATH STACK  *
*         le met sous la forme 15 chiffres      *
*         dans AB et CD                         *
*                                               *
*************************************************

take2  GOSBVL POP2N
       GOSBVL SPLTAC
       RTN
```

```
*************************************************
* fin   :                                       *
*                                               *
* BUT   : met AB sous la forme 12 chiffres dans *
*         C et retourne via FNRTN1              *
*                                               *
*************************************************

fin    GOSBVL uRES12
fnrtn1 GOVLNG FNRTN1   fin ...

       END
```

DES CARACTERES POUR FAIRE JOLI

Voici déjà mon second article pour JPC. Il s'agit d'un petit utilitaire qui permet de générer facilement trois jeux de caractères étendus.

Il s'agit en fait de caractères étendus peu originaux, puisqu'ils dérivent tous les trois du jeu de caractères primaire ou principal du HP-71. Le premier jeu (fonction BOLDSET$) correspond au jeu primaire, mais en caractères gras. Le second jeu (fonction INVSET$) est un jeu de caractères en vidéo-inverse alors que le troisième jeu (fonction UNDSET$) est un jeu de caractères soulignés.

Pour obtenir chaque caractère modifié, il faut ajouter 128 au code du caractère désiré. Il existe pour ce faire une fonction nommée HIGH$ du Lex LEXTEND publiée par Michel Martinet en page 31 de JPC 21.

J'allais omettre le mode d'emploi : chaque jeu est créé par l'une des trois fonctions précitées (BOLDSET$, INVSET$ ou UNDSET$) et est ensuite chargé par l'ordre CHARSET du HP-71. Par exemple, pour obtenir le jeu de caractères vidéo-inverse, faire :

```
CHARSET INVSET$
```

L'avantage de ce Lex est sa petite taille et sa rapidité. Il permettra, je pense, d'agrémenter facilement la présentation de vos programmes.

Jean-Yves Naour (313)

```
       LEX    'XSETLEX'
       ID     #5C
       MSG    0          Pas de messages
       POLL   0          Pas de pollhandler
*
ADHEAD EQU    #181B7     Ajoute en-tête de chaîne
ASCII  EQU    #0079B     Adresse des caractères
D=AVMS EQU    #1A460     D(A) pointe début mémoire
MEMERR EQU    #0944D     Erreur de mémoire
*
       ENTRY  bold
       CHAR   #F
       ENTRY  inv
       CHAR   #F
       ENTRY  und
       CHAR   #F
       KEY    'BOLDSET$'
       TOKEN  230
       KEY    'INVSET$'
       TOKEN  231
       KEY    'UNDSET$'
       TOKEN  232
       ENDTXT
*
* Envoie l'erreur :
*     "Insufficient Memory"
*
memer  GOVLNG MEMERR
*
* Routine commune :
*   Cette routine fait une copie du jeu de
*   caractères primaires sur la mathstack.
*
cst    GOSBVL D=AVMS   D(A) pointe début mémoire
       D1=C            Restore D1 (aprés avoir
       *                appelé D=AVMS, C(A) con-
       *                tient une copie de D1)
       R1=C            Sauve D1 dans R1(A)
       A=C    A        A(A) = D1
       P=     0
       C=0    A
       LCHEX  610      C(A) = place nécessaire
       *                sur la mathstack.
       C=A-C  A
       ?C<D   A        Assez de mémoire ?
       GOYES  memer     non : --> erreur
       CDOEX            oui : Sauve D0 ...
       RSTK=C             ... dans return-stack.
       D0=(5) ASCII    D0 pointe le début de la
       *                table des car. primaires.
       LCHEX  7F4      Initialise les compteurs
       CSRC            C(W) = 4.............7F
       B=C    W        B(S) = compteur de boucle
       *                pour 1 caractère. Il y a
       *                5 octets à transférer.
       *               B(B) = compt. de carctères

loop   C=B    S        Réinitialise le compteur
```

```
*
* Routine de transfert de la table des caractères
*   Départ : D1 pointe la mathstack
*            D0 pointe la table primaire
*
loop1  D1=D1- 2        \  Transfère les 5 premiers
       C=DAT0 B         | nibbles d'un caractère
       DAT1=C B         | vers la mathstak.
       D0=D0+ 2         | (5 fois 2 nibbles)
       C=C-1  S        /  et décrémente pointeur
       GONC   loop1    Boucle si besoin
       D1=D1- 2        Place pour la colonne
       *                blanche de séparation
       C=0    A
       DAT1=C B        Place la sixième colonne
       *                d'un caractère (vide).
       B=B-1  B        décrémente pointeur de car.
       GONC   loop     Caractère suivant si existe
       C=RSTK          }
       D0=C            } Restitue D0
       D1=A            D1 pointe le début de
       *                chaine créée.
       RTNCC           Fin de cette routine.
*
* Début de la routine de création du jeu de
*   caractères video inverse. Il suffit d'inverser
*   tout les bits de la chaine créée sur la MSTK.
*
       NIBHEX 00       Aucun paramètre
inv    GOSUB  cst      Charge le jeu primaire
       C=0    A
       LCHEX  2FF      Nbre d'octet à traiter - 1
       B=C    A        B(A) = 002FF : compteur
inv0   D1=D1- 2        \
       C=DAT1 B         | Inversion de chaque
       C=-C-1 B         |  bit de chaque octets.
       DAT1=C B        /
       B=B-1  A        Décrémente le pointeur
       GONC   inv0     Octet suivant si existe
adhead GOVLNG ADHEAD   Place l'en-tête de la
       *                chaine et retour à BASIC.
*
* Début de la routine de création du jeu de
*   caractères soulignés. Il faut ajouter 80(hex)
*   à chaque octet de la chaine créée sur la MSTK.
*
       NIBHEX 00       Aucun paramètre
und    GOSUB  cst      Charge le jeu primaire
       C=0    A
       LCHEX  2FF      Compteur d'octets à traiter
       B=C    A        B(A) = 002FF  : compteur
       LCHEX  80       Code de la barre
       *                de soulignement
       A=C    B        A(B) = 80
und0   D1=D1- 2        \  Rajoute la barre de
       C=DAT1 B         | soulignement à
       C=C!A  B         | chaque octet.
       DAT1=C B        /
       B=B-1  A        Décrémente le pointeur
       GONC   und0     Octet suivant si existant
adheaD GOC    adhead   B.E.T. (retour)
*
* Début de la routine de création du jeu de
*   caractères gras. Ici, il faut compléter le
*   contenu de la troisième colonne par le contenu
*   de la seconde et compléter le contenu de la
*   seconde par le contenu de la première. Noter
*   par exemple que la lettre 'I' ne change pas.
*
       NIBHEX 00       Aucun paramètre
bold   GOSUB  cst      Charge le jeu primaire
       LCHEX  7F       Nbre de car. à traiter - 1
       B=C    A        B(A) = 0007F  : compteur
bld0   D1=D1- 2        }
       A=DAT1 B        } A(B) = première colonne
       D1=D1- 2        }
       C=DAT1 B        }    Fabrication de
       C=C!A  B        }    la colonne 2.
       A=DAT1 B        } \
       DAT1=C B        }  |
       D1=D1- 2           | Fabrication de
       C=DAT1 B           | la colonne 3
       C=C!A  B           |
       DAT1=C B          /
       D1=D1- 6        Ne rien faire aux
       *                colonnes 4, 5 et 6
       B=B-1  B        Décrémente pointeur
       GONC   bld0     Caractère suivant si existe

       GOC    adheaD   B.E.T. (retour)

       END
```

EVALUEZ VOS POLYNOMES

Soit un polynôme P(x). Pour l'évaluer, nous pouvons calculer :

$$P(x)=a_0+a_1x+a_2x^2+...+a_{13}x^{13}$$

Or, c'est une très mauvaise méthode numérique, car elle conduit à des erreurs d'arrondi assez graves dûes à l'élévation à la puissance.

En revanche, le schéma de Horner n'est pas sujet à de telles erreurs, puisqu'aucune exponentiation n'est faite :

$$P(x)=a_0+x(a_1+x(a_2+x(a_3+x(...(a_{12}+a_{13}x)...))))$$

La fonction suivante évalue donc un polynôme de degré jusqu'à 13, pour une variable x donnée. La syntaxe est :

```
INTERP( a0, a1, x )
INTERP( a0, a1, a2, x )
INTERP( a0, a1, a2, a3, x )
INTERP( a0, a1, a2, a3, ..., a13, x )
```

Bonnes évaluations,

Jack Elhay

```
       LEX    'HORNER'
       ID     #5C
       MSG    0
       POLL   0
       ENTRY  hORNER
       CHAR   #F
       KEY    'INTERP'
       TOKEN  13
FUNCD1 EQU    #2F8C0   un endroit sûr pour sauver D1
POP1R  EQU    #0E8FD   dépile réel, pas complexe
* SPLITA transforme un nombre réel « normal » (sur
* 12 chiffres) en représentation interne sur 15
* chiffres. L'entrée est dans A, le résultat est
* dans A et B.
SPLITA EQU    #0C6BF
MP2-15 EQU    #0C43A   multiplie 2 nb en 15 chiffres
AD2-15 EQU    #0C363   ajoute 2 nb en 15 chiffres
* uRES12 est l'inverse de SPLITA. A partir de la
* forme « 15 chiffres » utilisée pour les calculs
* intermédiaires, convertit en représentation
* « 12 chiffres » pour être renvoyé à l'utilisateur.
uRES12 EQU    #0C994   15 chif. A/B -> 12 chif. C
FNRTN1 EQU    #0F216   retour de la fonction
       ENDTXT
* Syntaxe :
* INTERP( a0, a1 [, a2 [, ...[, a13]...]] , x )
* Tous les paramètres sont numériques
       NIBHEX 888888888888888
       NIBHEX 3F        Au moins 3, au plus 15
hORNER R0=C             C(S) = nb de paramètres
       CD1EX            sauver D1
       D1=(5) FUNCD1    ... dans FUNCD1
       DAT1=C A
       CD1EX
       CD0EX
       R2=C             sauver D0 en R2
       CD0EX
       C=R0             Décrémente le nb de
       C=C-1  S         paramètres de 3, car 3 est le
       C=C-1  S         nb minimum de paramètres
       C=C-1  S
       R3=C             sauver le degré dans R3
       GOSBVL POP1R     dépile l'argument x (de P(x))
       GOSBVL SPLITA    -> 15 chiffres
       R0=A             sauvés en R0 et R1
       A=B    W
       R1=A
       D1=D1+ 16        placer D1 pour le suivant
       GOSBVL POP1R     le coefficient suivant
       GOSBVL SPLITA
loop   C=R1             rappeler l'argument x
       D=C    W
       C=R0
       GOSBVL MP2-15    et le multiplier
       C=B    W         copie (A,B) dans (C,D)
       D=C    W
       C=A    W
       D1=D1+ 16        coefficient suivant
       GOSBVL POP1R
       GOSBVL SPLITA
       GOSBVL AD2-15    ajouter
       CR3EX            R3 contient le compteur
       C=C-1  S         qui est maintenant
       CR3EX            décrémenté et remplacé
       GONC   loop      Cy=1 si plus de coefficients
       GOSBVL uRES12    résultat dans C pour FNRTN1
       D1=(5) FUNCD1    restaurer D1 pour la sortie
       A=DAT1 A         et le placer dans A(A)
       D1=A
       A=R2
       GOSBVL FNRTN1    sortie...
       END
```

ARRONDIR DANS LA BONNE BASE

Michel Martinet avait créé quatre mots Basic contenus dans HMSLEX (JPC 52).

Il a été constaté que les mots HMS, HMS+ et HMS- sont générateurs de gags du genre :

```
HMS(1/3) -> .196 soit 1/3 d'heure -> 19'60"

HMS+(.2,.2) -> .396 soit 20' + 20' -> 39'60"
```

Ces réponses sont exactes, mais ce n'est pas celles que l'on attendait...

La raison de cela : la routine interne uRES12 fait un arrondi de 15 digits en 12 digits mais en décimal et non en sexagésimal.

Voici une nouvelle version de HMSLEX débarassée de cette bogue et comportant les modifications suivantes :

HMS+ accepte jusqu'à 10 paramètres au lieu de 2.

HP n'admettant pas en Basic les mots avec signes, HMS+ et HMS- deviennent HMSADD et HMSSUB.
La fonction HMSMUL est ajoutée pour permettre les multiplications ou divisions d'un nombre sexagésimal par un nombre décimal.

MODE D'EMPLOI

HR

conversion d'heures ou de degrés sexagésimaux en heures ou degrés décimaux.

syntaxe : HR (*n*)

HMS

conversion d'heures ou de degrés décimaux en heures ou degrés sexagésimaux donc heures ou degrés et minutes, secondes, fraction décimale de seconde.

syntaxe : HMS (*n*)

HMSADD

somme algébrique de i paramètres en heures ou degrés sexagésimaux.

syntaxe : HMSADD (n_1, n_2, ..., n_i)
avec 2 <= i <= 10

HMSSUB

différence de deux paramètres n1, n2 en heures ou degrés sexagésimaux.

syntaxe : HMSSUB (n_1, n_2)

HMSMUL

produit algébrique d'un paramètre décimal *k* par un paramètre *n* en heures ou degrés sexagésimaux.

syntaxe : HMSMUL (*k*, *n*)

EXEMPLES

```
HR(.2) -> .333333333333
HR(.4545) -> .7625
HMS(1/3) -> .2
HMSADD(.2,.2) -> .4
HMSADD(.2,.4,10,-.151515) -> 10.444485
HMSSUB(3,1.454545) -> 1.141455
HMSMUL(3,-.2) -> -1
HMSMUL(1/3,1.45) -> .35
```

division d'un cercle en 7 parties égales, mesure angulaire d'un secteur :

```
HMSMUL(1/7,360) -> 51.254286
```

donc 51°25'42" et 86 centièmes

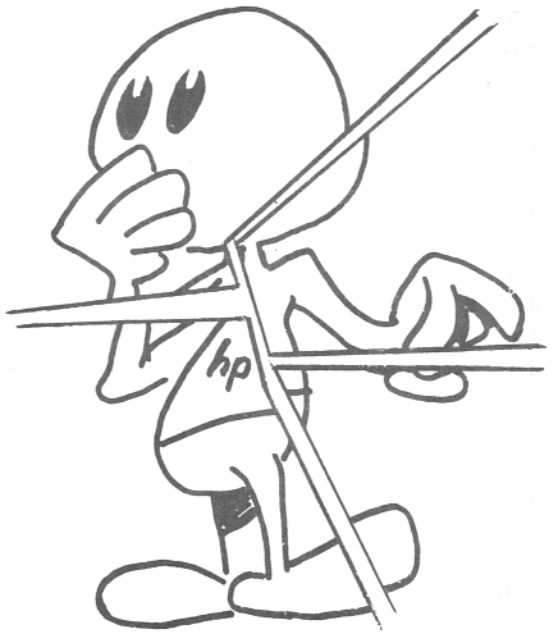
Guy Toublanc (276)

```
        LEX    'HMSLEX'
        CON(2) #E1
        CON(2) 28
        CON(2) 28
        REL(5) HMSLEX
        NIBHEX F
        REL(4) (TXT5)+1
        CON(4) 0
        CON(5) 0
        CON(3) 0
        REL(5) HM*
        CON(1) 15
TXT5    CON(1) 11
        NIBASC 'HMSMUL'
        CON(2) 28
        NIBHEX 1FF

        NIBHEX 8822
HM*     ST=1   3
        ST=1   2
        GOLONG STT

HMSLEX  CON(2) #E1
        CON(2) 22
        CON(2) 25
        CON(5) 0
        NIBHEX F
        REL(4) (TXT1)+1
        CON(4) 0
        CON(5) 0
        CON(3) 0
        REL(5) HM+
        CON(1) 15
        CON(3) (TXT2)-(TXT1)
        REL(5) HM-
        CON(1) 15
```

```
        CON(3) (TXT3)-(TXT1)
        REL(5) HM
        CON(1) 15
        CON(3) (TXT4)-(TXT1)
        REL(5) HR
        CON(1) 15
TXT1    CON(1) 11
        NIBASC 'HMSADD'
        CON(2) 22
TXT2    CON(1) 11
        NIBASC 'HMSSUB'
        CON(2) 23
TXT3    CON(1) 5
        NIBASC 'HMS'
        CON(2) 24
TXT4    CON(1) 3
        NIBASC 'HR'
        CON(2) 25
        NIBHEX 1FF
AD2-12  EQU    #0C35F
AD2-15  EQU    #0C363
ADDONE  EQU    #0C330
ARGPRP  EQU    #0E8EF
CLRFRC  EQU    #0C6F4
DV2-15  EQU    #0C4AC
EXAB2   EQU    #0D40E
FNRTN4  EQU    #0F238
MP2-15  EQU    #0C43A
RCCD1   EQU    #0D3F5
RCCD2   EQU    #0D41C
SPLITA  EQU    #0C6BF
SPLITC  EQU    #0C940
STAB1   EQU    #0D3D9
STAB2   EQU    #0D400
uRES12  EQU    #0C994

        NIBHEX 811
HR      ST=0   0
        GONC   ST1

        NIBHEX 811
HM      ST=1   0
ST1     ST=0   1
        ST=1   2
        ST=0   3
AOW     A=0    W
        B=0    W
        GOSBVL STAB1
POPN    GOSBVL ARGPRP
        ?ST=1  2
        GOYES  CONV
        A=-A-1 S
        ST=1   2
CONV    GOSUB  stab2
        GOSUB  rccd1
        GOSBVL STAB1
        GOSUB  exab2
        A=A+1  A
        A=A+1  A
        ?ST=1  0
        GOYES  six
        GOSUB  stab2
six     GOSUB SIX
        ?ST=0  0
        GOYES  exa
        GOSUB  stab2
exa     GOSUB  exab2
        ?ST=1  0
        GOYES  si
        GOSUB  SIX
si      GOSUB  SIX
        GOSBVL RCCD2
        GOSUB  ad2-15
        A=A-1  A
        A=A-1  A
        GOSUB  rccd1
        ?ST=1  1
        RTNYES
        GOSBVL uRES12
        ?ST=1  0
        GOYES  RNDTST
OUT     GOVLNG FNRTN4

RNDTST  R0=C
        R4=C
        A=R0
        A=0    S
        C=0    W
        P=     14
        LCHEX  6
        C=C-1  X
        R1=C
        A=A+1  X
        A=A+1  X
        A=A+1  X
        A=A+1  X
        GOSUB  IP
        GONC   FP
        GOSUB  DV/100
        GOC    NOFP
        GOSUB  TEST
NOFP    GOSUB  DV/100
        GOC    FP
        GOSUB  TEST
        R0=A
FP      C=R0
        A=R4
        C=A    S
        GOTO   OUT

        NIBHEX 8822
HM-     ST=0   2
        GONC   ST

        NIBHEX 88888888882A
HM+     ST=1   2
```

```
ST      ST=0   3
STT     ST=1   1
        ST=0   0
        C=C-1  S
        C=C-1  S
        R4=C
        GOSUB  AOW
STA     GOSBVL STAB1
        D1=D1+ 16
        ?ST=1  3
        GOYES  MUL
        GOSUB POPN
        C=R4
        C=C-1  S
        R4=C
        GONC   STA
        GOC    S
MUL     GOSBVL ARGPRP
        GOSBVL RCCD1
        GOSUB  mp2-15
S       ST=1   0
        C=0    W
        R0=C
        R1=C
        ST=0   1
        GOTO   CONV
SIX     C=0    W
        P=     14
        LCHEX  6
        D=C    W
        C=0    P
        ?ST=0  0
        GOYES  dv
        A=A-1  A
mp2-15  GOVLNG MP2-15
dv      A=A+1  A
        GOVLNG DV2-15

stab2   GOSBVL STAB2
clrfrc  GOVLNG CLRFRC

exab2   GOSBVL EXAB2
frac15  DAT1=A W
        C=B    W
        D=C    W
        GOSUB  clrfrc
        A=-A-1 S
        C=DAT1 W
        GOTO   ad2-15

rccd1   GOSBVL RCCD1
ad2-15  GOVLNG AD2-15

DV/100  A=A-1  X
        A=A-1  X
        R0=A
IP      GOSBVL SPLITA
        GOSUB  clrfrc
```

```
A=B     A=B    M
        RTN

TEST    AROEX
        GOSBVL SPLITA
        GOSUB  frac15
        A=B    M
        C=R1
        ?A#C   W
        GOYES  NO60
        A=0    W
        AROEX
        GOSBVL SPLITA
        GOSBVL ADDONE
        A=B    M
        AROEX
NO60    C=R0
        GOSBVL AD2-12
        GOTO   A=B

        END
```



CHOC EN RETOUR
DERIVEES SYMBOLIQUE
(ACTE I QUATER)

Le programme de dérivation publié dans JPC 44 (mai 1987) m'avait beaucoup intéressé. C'est pourquoi je propose aujourd'hui mes corrections :

- il y avait malheureusement plus de 3 erreurs dans le tableau des dérivées. J'ai remédié au problème en regardant mon cours de maths.

- il manquait le u' à la dérivée de $u(x)^n$. J'ai donc modifié la ligne 1590, qui devient :

```
F1$=FNK$(V$,3)&"*"&FNK$(U1$,3)&"*"&FNK$(U$,5)&"^"
```

Le programme complet est listé à la fin, avec les autres programmes Basic. Le sous-programme MINPRIO (en Assembleur ou en Basic) est inchangé, voir JPC 44.

N'ayant pas cité toutes les corrections, il y en a peut-être d'autres, mais normalement, le programme joint tourne correctement.

Bruno Gil (332)

LA GUERRE DES NOYAUX
(ACTE II)

Voyons, comme promis dans notre dernière livraison, la structure du programme MARS71 paru dans JPC 48.

Que ceux que le sujet rebute me pardonnent, mais en prenant la peine de suivre ces modestes explications, ils auront acquis au bout du compte quelques notions du fonctionnement d'un compilateur / interpréteur.

Prenons le problème à sa source. Le but à atteindre est, à partir de lignes de programmes en *Redcode*, de générer des instructions compréhensibles par le HP-71. Ce traitement a lieu en deux étapes : l'entrée du programme *Redcode* proprement dit puis son exécution par le superviseur *MARS*.

Ce traitement a été séparé en deux phases essentiellement pour des raisons de rapidité. Les lignes *Redcode*, en fait des chaînes de caractères, sont "compilées" sous forme numérique au cours de la première phase. En effet, pendant la phase d'exécution, le traitement de ces chaînes est plus rapide à partir d'une forme numérique convenue à l'avance. Pour s'en convaincre, examinons un exemple.

Soit à analyser (les espaces sont intentionnels) :

```
MOV      #0      @-2
```

Cette instruction, une fois codée par le compilateur (voir le sous-programme COMP du programme *MARS*), sera mémorisée sous la forme : 102000998. Dans le cas d'une mémoire circulaire de 1000 cases, la case -2 a également pour adresse 998.

Ainsi, les espaces redondants seront repérés et supprimés, les modes d'adressage traduits et les adresses converties selon la taille de la mémoire circulaire, et tout cela une fois pour toutes ! Imaginez le temps d'exécution si ce travail devait être effectué à chaque fois que cette instruction était rencontrée, dans une boucle par exemple !

Cela, c'est le travail du compilateur. Une fois votre programme *Redcode* lancé, le traitement est inverse. A partir d'un nombre, tel 102000998, le superviseur (programme en Basic HP-71) devra effectuer le travail demandé, soit ici copier la valeur immédiate 0 à l'adresse indirecte relative -2.

C'est le sous-programme EXEC qui a pour rôle d'interpréter la case mémoire de rang *I*. Pour en saisir le fonctionnement, un mot sur le code fourni par le compilateur.

Celui-ci retourne pour chaque ligne de *Redcode* un nombre de la forme *CdDDDaAAA*, où *C* est le code de l'instruction (de 1 pour MOV à 8 pour CMP, voir JPC 48 pour la liste des ordres), *d* et *a* représentent le mode d'adressage (0 pour immédiat, 1 pour direct et 2 pour indirect) des arguments *DDD* et *AAA*, exprimés, eux, modulo le nombre de cases mémoires demandés ; ce codage explique la limitation de -999 à 999 pour les arguments. Dans le cas où un seul argument est nécessaire (ordre JMP), le champ correspondant au second argument est mis à zéro.

Un cas particulier à considérer, celui de l'ordre DAT, de code *C* égal à 0. Son code généré sera *AAA*, soit un nombre de 3 chiffres, représentant la valeur de la donnée fournie modulo le nombre de cases mémoires. Une autre option aurait été de ne pas prendre le modulo de la donnée, de manière à laisser la possibilité à un programme *Redcode* de modifier

son propre code en écrivant des données représentant des instructions exécutables... Le choix ici effectué oriente plutôt l'emploi de DAT comme pointeur. Mais si le coeur vous en dit, modifiez la ligne 1160 en :

```
1160 IF NO$='DAT' THEN N1=N$ @ GOTO 1310
```

Attention cependant à ne pas employer de constantes supérieures à 10^9 car vous auriez une instruction différente de DAT !

Voilà, vous connaissez maintenant l'essentiel des principes ayant conduit à la réalisation du superviseur *MARS*. Détaillons maintenant le programme.

Les principales variables utilisées sont :
- M : nombre de positions mémoire de la mémoire circulaire ;
- T : tableau à une dimension représentant la mémoire circulaire ;
- I : pointeur de l'instruction à exécuter du programme 1 ;
- J : idem pour le programme 2 ;
- N$: chaîne à compiler ;
- A, B : arguments A et B ;
- A0, B0 : modes d'adressage des arguments A et B.

Le sous-programme EXEC(T(),I,M) exécute l'instruction *Redcode* de rang I à l'intérieur du tableau T, I étant passé par valeur et pouvant représenter le pointeur du programme 1 ou 2. La décomposition du code fourni par le compilateur a lieu aux lignes 310 à 360, où les arguments et modes d'adressage se mettent en place, N3 contenant le code de l'instruction pour le branchement par ON ... GOSUB.

Le sous-programme DONNEE(T(),A,A0,I,M) retourne en A la donnée correspondant à l'argument A de mode d'adressage A0.

Le sous-programme ADR(T(),A,A0,I,M) procède de même mais pour le calcul d'une adresse de branchement relatif (instructions de saut).

Le sous-programme COMPIL(T(),I,M) effectue le travail de compilation du programme *Redcode* que vous entrez sous forme de chaînes de caractères.

Enfin, le sous-programme DATSEEK(T(),I,M) est un utilitaire retournant en I l'adresse de la première instruction exécutable suivant les éventuelles instructions DAT pouvant se trouver en début de votre programme.

Les différents drapeaux utilisés ont pour fonction :

- 1 : exécution d'une instruction du programme 1 ;
- 2 : exécution d'une instruction du programme 2 ;
- 5 : traitement d'un branchement ; signale que l'adresse I retournée est celle de la prochaine instruction à exécuter et qu'il ne faut pas l'incrémenter comme dans le cas des instructions sans branchement ;
- 6 : exécution de deux programmes *Redcode* simultanément ;
- 7 : fin d'exécution.

Désormais, avec ces quelques explications et les remarques se trouvant dans le programme *MARS*, vous devriez être à même d'en suivre le fonctionnement ligne à ligne. Dans le cas contraire, si un point vous parait obscur, n'hésitez à me le faire savoir et je vous éclairerai...

Pour vous récompenser de votre patience (si vous m'avez supporté jusqu'ici !), voici un exemple de programme autodéplaçant, se recopiant 100 cases mémoires au-delà de sa position initiale et y lançant son exécution :

```
DAT         0     ; pointeur d'origine
DAT        99     ; pointeur d'arrivée
MOV  @-2   @-1    ; copie
CMP   -3   #9     ; si les dix copiées...
JMP    4          ; ...alors sortir de la boucle
ADD   #1   -5     ; adresse suivante à copier
ADD   #1   -5     ; arrivée suivante
JMP   -5
MOV  #99   93     ; adresse d'arrivée correcte
JMP   93          ; exécution nouvelle copie
```

Pour conclure, en avant-première du prochain article, méditez un peu sur les possibilités qu'offrirait la faculté, pour un programme *Redcode*, de lancer plusieurs programmes *Redcode* en parallèle...

A bientôt pour la suite et à vos combattants préférés.

Olivier Arbey (118)

UN CATALOGUE ORDONNE

Un autre utilitaire pour démontrer les étonnantes possibilités de JPC Rom et en particulier de RREC$ et WREC.

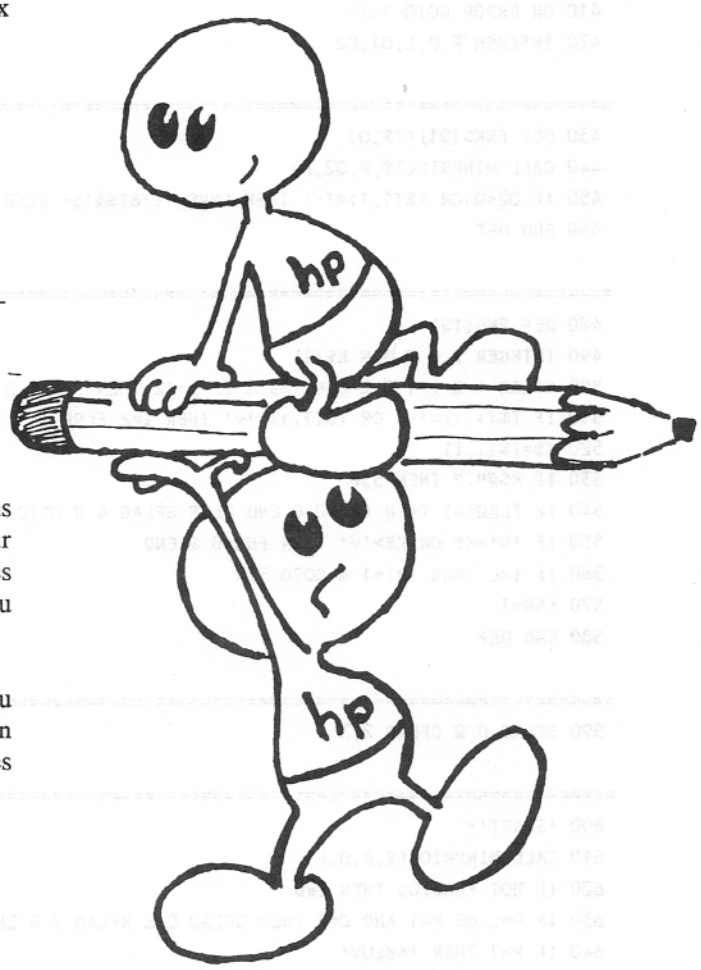
Ceux qui connaissent MS-DOS (est-il mal élevé, le bougre !) savent qu'il est facile d'éditer le catalogue d'un support trié par ordre alphabétique croissant des noms de fichiers... Me sentant un peu frustré de ce côté-là, et ayant besoin d'un outil simple pour connaitre le contenu de mes nombreuses disquettes pour chasser le doublon, j'ai réalisé le petit utilitaire FASTCAT.

Il n'imprime que les noms des fichiers (ni type, ni longueur, ni date), sous forme d'un tableau rectangulaire, avec indication du nom du support (label) et autres mentions utiles. La technique de tri utilisée (Shell) permet à ce petit Basic de rester compétitif, même pour un catalogue assez long.

A noter que, s'il restait des fichiers purgés sur votre support, ce programme ne les éliminera pas : ils seront édités comme les autres. Ne vous étonnez donc pas si vous voyez deux fois le même nom : l'un d'eux aura été purgé mais non encore recouvert.

Amusez-vous bien !

Eric Gengoux (108)



CA DATE !

Je prends la liberté de vous écrire pour vous communiquer un petit programme Basic pour débutants. Je pense qu'il intéressera quelques membres du Club. Il s'agit de la détermination du jour de la semaine d'une date donnée.

Ce calendrier perpétuel n'est valable qu'à partir du Vendredi 15 Octobre 1582 qui est la date d'entrée en vigueur du calendrier Grégorien. Voici quelques dates de référence :

19/5/1643, bataille de Rocroy : mardi
11/5/1745, bataille de Fontenoy : mardi
18/6/1815, bataille de Waterloo : dimanche
24/6/1859, bataille de Solférino : vendredi
21/2/1916, bataille de Verdun : lundi
6/6/1944, bataille de Normandie : mardi

J'ajoute un autre programme qui donne le nombre de jours entre deux dates.

Jean Maille (392)

Programme "DIFF" (dérivation symbolique, voir JPC 44)

```
 10 DESTROY ALL
 30 DIM K$[91]
 40 INPUT 'f(x)=',F1$;K$
 50 K$=UPRC$(K$) @ L=LEN(K$)
 60 DIM F$[L],F1$[MIN(91,10*L)]
 70 F$=K$ @ CALL DIFF(F$,F1$)
 80 IF FLAG(0) THEN BEEP 4000,.01 @ DISP "f'=";F1$
 85 IF NOT KEYDOWN THEN 85 ELSE 40
 90 BEEP 1400,.05
100 IF FLAG(1) THEN DISP "ERR:";MSG$(76)
110 IF FLAG(2) THEN DISP "ERR:";MSG$(11)
120 IF FLAG(3) THEN DISP "ERR:";MSG$(65)
130 F1$=K$ @ WAIT 3 @ GOTO 40

================================================================
380 SUB DIFF(F$,F1$)
410 ON ERROR GOTO 1670
420 INTEGER P,D,L,O1,O2

================================================================
430 DEF FNK$[91](T$,O)
440 CALL MINPRIO(T$,P,O2,L)
450 IF O2<O OR T$[1,1]='-' THEN FNK$='('&T$&')' ELSE FNK$=T$
460 END DEF

================================================================
470 DEF FNN(T$)
490 INTEGER I,L @ DIM K$[1]
500 CFLAG 4 @ I=1 @ L=LEN(T$) @ IF L>12 THEN FNN=0 @ END
510 IF T$[1,1]='-' OR T$[1,1]='+' THEN I=2 ELSE I=1
520 K$=T$[I,I]
530 IF K$#"." THEN 550
540 IF FLAG(4) THEN FNN=0 @ END ELSE SFLAG 4 @ GOTO 560
550 IF '0'>K$ OR K$>'9' THEN FNN=0 @ END
560 IF I<L THEN I=I+1 @ GOTO 520
570 FNN=1
580 END DEF

================================================================
590 SFLAG 0 @ CFLAG 2,3

================================================================
600 'START':
610 CALL MINPRIO(F$,P,O,L)
620 IF NOT FLAG(0) THEN END
630 IF P=L OR P=1 AND O>2 THEN CFLAG 0 @ SFLAG 2 @ END
640 IF P>1 THEN 'ABLUV'
660 IF O#6 THEN F$=F$[2] @ L=L-1
670 IF F$#"X" THEN 700
680 IF O=2 THEN F$='-X' @ F1$='-1' ELSE F1$='1'
690 END

================================================================
700 IF ('A'>F$ OR F$>'Z' OR L>1 AND F$#'PI') AND NOT FNN(F$) THEN 740
710 F1$='0'
```

```
720 IF O=2 THEN F$='-'&F$
730 END

============================================================
740 P=POS(F$,'(')
750 IF P#1 THEN 'SUCHEFKT'
760 F$=F$[2,L-1] @ L=L-2
770 IF O#2 THEN 'START'
780 CALL DIFF(F$,F1$)
790 IF F1$#'0' THEN F$='-'&FNK$(F$,3) @ F1$='-'&FNK$(F1$,3)
800 END

============================================================
810 'SUCHEFKT':
820 IF P=0 OR P=L THEN CFLAG 0 @ SFLAG 2 @ END
830 DIM U$[L-P],U1$[(L-P)*10],V$[P-1]
840 U$=F$[P+1,L-1] @ CALL DIFF(U$,U1$)
850 V$=F$[1,P-1]
860 IF 'SIN'=V$ THEN O1=3 @ F1$='COS('&U$&')' @ GOTO 'A'
870 IF 'COS'=V$ THEN O1=4 @ F1$='SIN('&U$&')' @ GOTO 'A'
880 IF 'TAN'=V$ THEN O1=1 @ F1$='COS('&U$&')^2' @ GOTO 'A'
890 IF 'LOG'=V$ THEN O1=1 @ F1$=FNK$(U$,5) @ GOTO 'A'
900 IF 'EXP'=V$ THEN O1=3 @ F1$='EXP('&U$&')' @ GOTO 'A'
910 IF 'SQR'=V$ THEN O1=1 @ F1$='2*SQR('&U$&')' @ GOTO 'A'
920 IF 'ASIN'=V$ THEN O1=1 @ F1$='SQR(1-'&FNK$(U$,5)&'^2)' @ GOTO 'A'
930 IF 'ACOS'=V$ THEN O1=2 @ F1$='SQR(1-'&FNK$(U$,5)&'^2)' @ GOTO 'A'
940 IF 'ATAN'=V$ THEN O1=1 @ F1$='(1+'&FNK$(U$,5)&'^2)' @ GOTO 'A'
950 IF 'COT'=V$ THEN O1=2 @ F1$='SIN('&U$&')^2' @ GOTO 'A'
960 IF 'SINH'=V$ THEN O1=3 @ F1$='COSH('&U$&')' @ GOTO 'A'
970 IF 'COSH'=V$ THEN O1=3 @ F1$='SINH('&U$&')' @ GOTO 'A'
980 IF 'TANH'=V$ THEN O1=1 @ F1$='COSH('&U$&')^2' @ GOTO 'A'
990 IF 'COTH'=V$ THEN O1=2 @ F1$='SINH('&U$&')^2' @ GOTO 'A'
1000 IF 'ASINH'=V$ THEN O1=1 @ F1$='SQR(1+'&FNK$(U$,5)&'^2)' @ GOTO 'A'
1010 IF 'ACOSH'=V$ THEN O1=1 @ F1$='SQR('&FNK$(U$,5)&'^2-1)' @ GOTO 'A'
1020 IF 'ATANH'=V$ THEN O1=1 @ F1$='(1-'&FNK$(U$,5)&'^2)' @ GOTO 'A'
1030 IF 'ACOT'=V$ THEN O1=2 @ F1$='(1+'&FNK$(U$,5)&'^2)' @ GOTO 'A'
1040 IF 'ACOTH'=V$ THEN O1=1 @ F1$='(1-'&FNK$(U$,5)&'^2)' @ GOTO 'A'

============================================================
1050 'A':
1060 IF U1$="0" THEN F1$='0' @ GOTO 'EF'
1070 IF O1<3 THEN 1110
1080 IF U1$#"1" THEN F1$=FNK$(U1$,3)&'*'&F1$
1090 IF O1=4 THEN F1$='-'&FNK$(F1$,3)
1100 GOTO 'EF'
1110 IF O1=1 THEN F1$=FNK$(U1$,3)&'/'&FNK$(F1$,5) ELSE F1$='-'&FNK$(U1$,3)&'/'&FNK$(F1$,5)

============================================================
1120 'EF': IF O=2 THEN F$='-'&F$ @ F1$='-'&FNK$(F1$,3)
1130 END

============================================================
1140 'ABLUV':
1150 DIM U$[P-1],V$[L-P],U1$[(P-1)*10],V1$[(L-P)*10]
1160 U$=F$[1,P-1] @ CALL DIFF(U$,U1$)
1170 IF NOT FLAG(0) THEN END
1180 V$=F$[P+1,L] @ CALL DIFF(V$,V1$)
1190 ON O GOTO 'ADD','SUB','MUL','DIV','POT'
```

```
============================================================
1200 'ADD': 'SUB':
1210 IF V1$='0' THEN 1280
1220 IF U1$='0' THEN 1260
1230 IF O=2 THEN F1$=U1$&'-'&FNK$(V1$,3) @ END
1240 F1$=U1$&'+' @ IF V1$[1,1]='-' THEN F1$=F1$&'('&V1$&')' ELSE F1$=F1$&V1$
1250 END

============================================================
1260 IF O=2 THEN F1$='-'&FNK$(V1$,3) ELSE F1$=V1$
1270 END

============================================================
1280 IF U1$='0' THEN F1$='0' ELSE F1$=U1$
1290 END

============================================================
1300 'MUL': 'DIV':
1310 IF V1$='0' THEN 1420
1320 IF U1$='0' THEN 1390
1330 IF U1$='1' THEN F1$=V$ ELSE F1$=FNK$(U1$,3)&'*'&FNK$(V$,3)
1340 IF V1$='1' THEN 1370
1350 F1$=F1$&CHR$(43+(O-3)*2)&FNK$(U$,3)&'*'&FNK$(V1$,3)
1360 GOTO 1430
1370 IF O=4 THEN F1$=F1$&'-'&FNK$(U$,3) ELSE F1$=F1$&'+'&U$
1380 GOTO 1430
1390 IF O=4 THEN F1$='-' @ U$=FNK$(U$,3) ELSE F1$=''
1400 IF V1$='1' THEN F1$=F1$&U$ ELSE F1$=F1$&U$&'*'&FNK$(V1$,3)
1410 GOTO 1430
1420 IF U1$='0' THEN F1$='0' ELSE IF U1$='1' THEN F1$=V$ ELSE F1$=FNK$(U1$,3)&'*'&FNK$(V$,3)
1430 IF O=4 AND F1$#'0' THEN F1$=FNK$(F1$,3)&'/'&FNK$(V$,5)&'^2'
1440 END

============================================================
1450 'POT':
1460 IF V1$='0' THEN 1580
1470 IF U1$#'0' THEN 1520
1480 IF U$='E' THEN F1$='E^'&FNK$(V$,6) @ GOTO 1500
1490 F1$=FNK$(U$,5)&'^'&FNK$(V$,6)&'*LN('&U$&')'
1500 IF V1$#'1' THEN F1$=F1$&'*'&FNK$(V1$,3)
1510 END

============================================================
1520 F1$=FNK$(U$,5)&'^'&FNK$(V$,6)&'*('
1530 IF V1$#'1' THEN F1$=F1$&FNK$(V1$,3)&'*'
1540 F1$=F1$&'LN('&U$&')+'
1550 IF U$=V$ THEN F1$=F1$&'1' ELSE F1$=F1$&FNK$(V$,5) @ IF U$#'1' THEN F1$=F1$&'/'&FNK$(U$,5)
1560 F1$=F1$&')'
1570 END

============================================================
1580 IF U1$='0' THEN F1$='0' @ END
1590 F1$=FNK$(V$,3)&'*'&FNK$(U1$,3)&'*'&FNK$(U$,5)&'^'
1600 IF FNN(V$) THEN 1620
1610 F1$=F1$&'('&V$&'-1)' @ END
1620 IF V$='1' THEN F1$='1' @ END
1630 IF V$='0' THEN F1$='0' @ END
```

```
1640 V1$=STR$(VAL(V$)-1)
1650 IF V1$='1' THEN F1$=V$&'*'&FNK$(U$,3) ELSE F1$=F1$&FNK$(V1$,3)
1660 END

============================================================
1670 IF ERRN=24 OR ERRN=37 THEN SFLAG 3 ELSE BEEP 1200,.1 @ DISP 'Err L';ERRL;ERRM$ @ WAIT 3
1680 CFLAG 0
1690 END SUB



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

Programme "FASTCAT" (catalogue trié des disquettes, utilise JPC ROM)


  - Fastcat (c) Eric Gengoux 87/01/14 V1.0(71)
 20 OPTION BASE 1 @ INTEGER I,J,D
 30 DIM B0$[256],B1$[256],A$[10],O$[2],L$[8],F1$[8],F2$[8],Z1$[100]
 40 PRINTER IS ':2225b'
 50 INPUT 'Unite ',':TAPE(1)';A$ @ A$=UPRC$(A$)
 60 IF A$[1,5]#':TAPE' THEN BEEP @ GOTO 50
 70 IF DEVAID(A$)#16 THEN BEEP @ GOTO 50
 80 B0$=RREC$(0,A$)
 90 L$=B0$[3,8] @ IF L$='      ' THEN L$='No Label'
100 PRINT
110 PRINT ESC$('&k3S')
120 D1$=DATE$ @ D$=D1$[7,8]&D1$[3,6]&D1$[1,2]
130 PRINT 'Catalogue suppport (Label= ';L$;') date '&D$[7,8]&D$[3,6]&D$[1,2]
140 WIDTH 132 @ PWIDTH 132 @ PRINT ESC$('&k2S')
150 DISP "Lecture DIR en cours..."
160 CREATE DATA "FLIST",MAXD(A$) @ ASSIGN #1 TO "FLIST"
170 GOSUB 'RDCAT'
180 DISP "Tri en cours..."
190 GOSUB 'TRIA'
200 GOSUB 'PRCAT'
210 PURGE "FLISTE" @ DISP "Termine" @ BEEP 400,.5 @ BEEP 300,.5 @ END

============================================================
220 'RDCAT': FOR I=2 TO INF
230 O$[1,2]=CHR$(0)&CHR$(I)
240 B0$=RREC$(I,A$)
250 FOR J=1 TO 8
260 D=32*(J-1) @ I9=8*(I-2)+J @ IF I9>300 THEN BEEP @ DISP 'Medium full' @ END
270 IF B0$[1+D,1+D]=CHR$(255) THEN GOTO 320
280 F1$=B0$[1+D,8+D]
290 IF F1$[LEN(F1$)]=' ' THEN F1$='' @ GOTO 290
300 PRINT #1,I9;F1$
310 NEXT J
320 IF B0$[1+D,1+D]=CHR$(255) THEN GOTO 340
330 NEXT I
340 I9=I9-1
350 RETURN

============================================================
360 'TRIA': M=I9
370 M=M DIV 2
380 IF M=0 THEN 540
```

```
 390 K=I9-M
 400 J=1
 410 I=J
 420 L=I+M
 430 READ #1,I;F1$ @ READ #1,L;F2$
 440 IF F1$<=F2$ THEN GOTO 510
 450 T$=F1$
 460 F1$=F2$
 470 F2$=T$
 480 PRINT #1,I;F1$ @ PRINT #1,L;F2$
 490 I=I-M
 500 IF I>=1 THEN GOTO 420
 510 J=J+1
 520 IF J>K THEN GOTO 370
 530 GOTO 410
 540 RETURN

============================================================
 550 'PRCAT':
 560 Z$='          ' @ Z1$=Z$
 570 FOR J=1 TO 9 @ Z1$=Z1$&Z$ @ NEXT J
 580 FOR J=0 TO 9 @ PRINT '    ';J;'   '; @ NEXT J
 590 PRINT @ PRINT
 600 FOR I=1 TO I9 STEP 10
 610 B0$=Z1$
 620 FOR J=0 TO 9
 630 ON ERROR GOTO 670
 640 READ #1,I+J;F1$
 650 F1$=RED$(F1$)
 660 B0$[1+10*J,1+10*J+LEN(F1$)]=F1$
 670 OFF ERROR
 680 NEXT J
 690 IF I=1 THEN PRINT '00x  '; ELSE PRINT STR$(10*(I DIV 10))&'x  ';
 700 PRINT B0$[1,100]
 710 NEXT I
 720 PRINT
 730 RETURN
```

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

Programme "GREGOR" (calendrier perpétuel)

```
  10 DISP CHR$(27)&"[" @ DESTROY ALL
  20 DISP "Calendrier Gregorien ( >= au 15 Octobre 1582)"
  30 DISP @ DISP
  40 INPUT "Jour :";J
  50 INPUT "Mois :";M
  60 INPUT "Annee:";Y
  70 DISP @ DISP
  80 S=INT(Y/100)
  90 A=Y-100*S
 100 D3=MOD(J-1,7)
 110 IF M=3 OR M=1 THEN D2=0
 120 IF M=6 THEN D2=1
 130 IF M=9 OR M=12 THEN D2=2
```

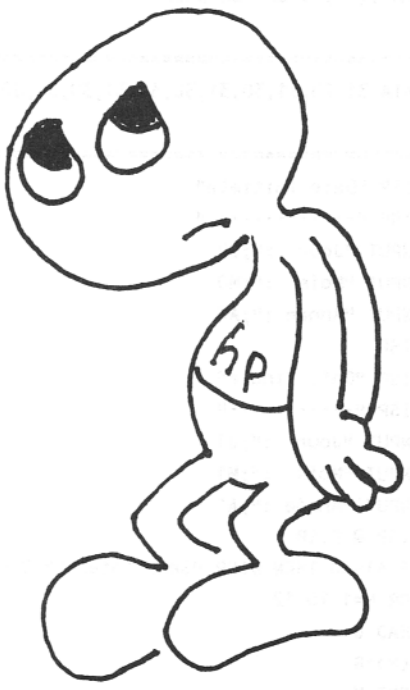
```
140 IF M=4 OR M=7 THEN D2=3
150 IF M=10 THEN D2=4
160 IF M=5 THEN D2=5
170 IF M=8 THEN D2=6
180 IF A=0 THEN 200
190 IF MOD(100*S+A,4)=0 THEN 240 ELSE 210
200 IF MOD(S,4)=0 THEN 240
210 IF M=1 THEN D2=4
220 IF M=2 THEN D2=0
230 GOTO 260
240 IF M=1 THEN D2=3
250 IF M=2 THEN D2=6
260 D1=MOD(A+INT(A/4),7)
270 D0=MOD(5*S+INT(S/4),7)
280 D=MOD(3+D0+D1+D2+D3,7)
290 DISP "--------"
300 IF D=0 THEN DISP "DIMANCHE"
310 IF D=1 THEN DISP "LUNDI"
320 IF D=2 THEN DISP "MARDI"
330 IF D=3 THEN DISP "MERCREDI"
340 IF D=4 THEN DISP "JEUDI"
350 IF D=5 THEN DISP "VENDREDI"
360 IF D=6 THEN DISP "SAMEDI"
370 DISP "--------"
```

****

Programme "DIFFDATE" (nombre de jours entre deux dates)

```
  10 A$=CHR$(27)
  20 DISP A$&"[" @ DISP A$&"j" @ DESTROY ALL
  30 DIM T(12) @ OPTION BASE 0

================================================================
  40 DATA 31,28,31,30,31,30,31,31,30,31,30,31

================================================================
  50 DISP "Date Initiale"
  60 DISP "------------"
  70 INPUT "Jour  :";J0
  80 INPUT "Mois  :";M0
  90 INPUT "Année :";A0
 100 DISP
 110 DISP "Date Finale"
 120 DISP "-----------"
 130 INPUT "Jour  :";J1
 140 INPUT "Mois  :";M1
 150 INPUT "Année :";A1
 160 DISP @ DISP
 170 IF A1=A0 THEN DISP "Sans Intérêt" @ END
 180 FOR M=1 TO 12
 190 READ B
 200 T(M)=B
 210 NEXT M
 220 FOR D=A0+1 TO A1-1
```

```
230 A=A+365
240 IF MOD(D,4)=0 THEN A=A+1
250 IF MOD(D,100)=0 AND MOD(D,400)#0 THEN A=A-1
260 NEXT D
270 FOR I=12 TO M0+1 STEP -1
280 M2=M2+T(I)
290 NEXT I
300 IF MOD(A0,4)=0 AND M0<2 THEN M2=M2+1
310 IF MOD(A0,100)=0 AND MOD(A0,400)#0 THEN M2=M2-1
320 FOR I=1 TO M1-1
330 M3=M3+T(I)
340 NEXT I
350 IF MOD(A1,4)=0 AND M1>2 THEN M3=M3+1
360 IF MOD(A1,100)=0 AND MOD(A1,400)#0 THEN M3=M3-1
370 J2=T(M0)-J0
380 IF MOD(A0,4)=0 AND M0=2 THEN J2=J2+1
390 J3=J1
400 Z=A+M2+M3+J2+J3
410 DISP
420 DISP "Nombre de Jours écoulés :  ";Z
430 END
```



LE COIN DES LHEX

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

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

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

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

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

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

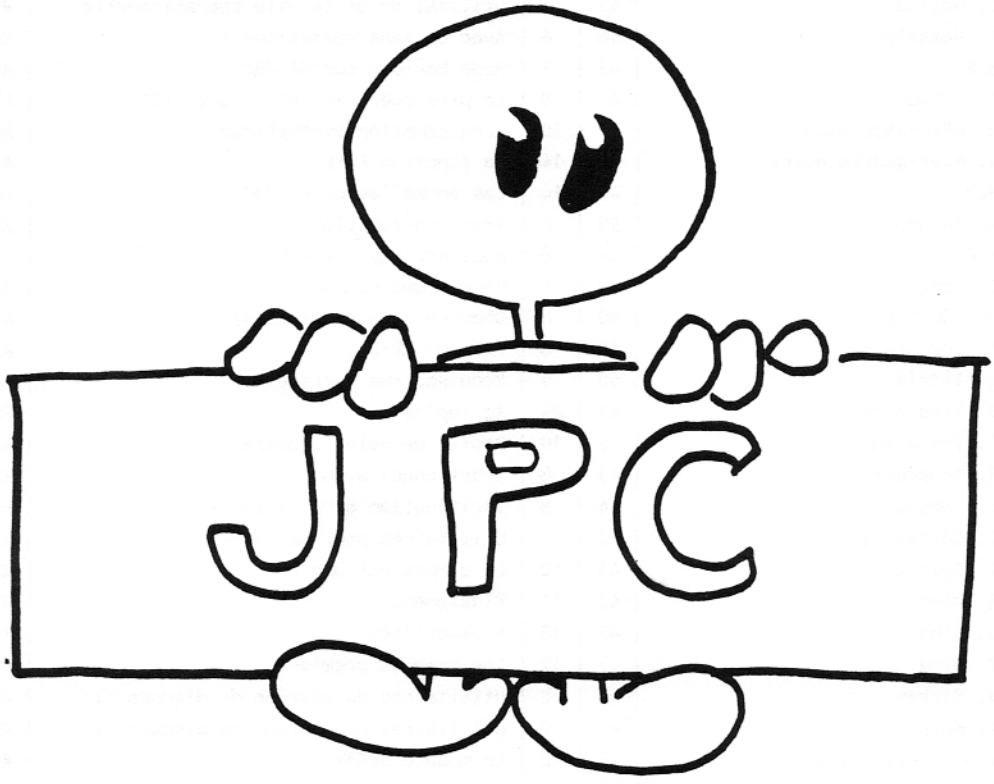
Attention : la fonction HMSMUL reprend le token de KSPEED, fonction récemment désallouée (l'effet de KSPEED est intéressant, mais pas son paramétrage).

```
CHARLEX

XSETLEX   BOLDSET$   XFN   92230        INVSET$    XFN   92231
          UNDSET$    XFN   92232

HORNER    INTERP     XFN   92013

HMSLEX    HMSADD     XFN   225022       HMSSUB     XFN   225023
          HMS        XFN   225024       HR         XFN   225025
          HMSMUL     XFN   225028
```

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

```
CHARLEX  ID#E1  624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E000571221178 6AF
002: 5E4001E000000000 9F3
003: FE00000000800001F D4D
004: F31BF961400032BF 0E0
005: 38F14A11DB10AD23 47A
006: 07D532BFB8FD7911 82D
007: 11AD754D7A101743 BB0
008: 11014D1CB15D0000 F1B
009: 71450375FF864834 298
00A: 5655581008355654 5EF
00B: 5810002455565870 93C
00C: 0026555658700836 C8E
00D: 5556581008364545 FE4
00E: 4A30000A49724000 337
00F: 0808094A2C180814 6A0
010: A464242008355455 9FA
011: 581000054C714000 D40
012: 0C3142404C700832 09C
013: 41414A70002078A0 3F4
014: 2F30000000000000 71F
015: 0000000000000000 A2F
016: 0000000000000000 D3F
017: 0000000000000000 04F
018: 0000000000000000 35F
019: 0000000000000000 66F
01A: 0000000000000000 97F
01B: 0000000000000000 C8F
01C: 0000000000000000 F9F
01D: 0000000000000000 2AF
01E: 0000000000000000 5BF
01F: 0000000000000000 8CF
020: 0000000000000000 BDF
021: 000000000000080C F0A
022: 1A28080008080A2C 274
023: 180008040E340800 5BD
024: 08001E3018000000 8F7
025: 0000000000000000 C07
026: 0000000000000000 F17
027: 0000000000000000 227
028: 0201000000010200 53D
029: 0000000201020000 852
02A: 0001000100000002 B66
02B: 0102010000000000 E7A
02C: 0000000000000000 18A
02D: 045E755142400101 4D6
02E: 0101010000000000 7E9
02F: 0000000000000000 AF9
030: 0000070507000000 E1C
031: 000000000083444C4 15A
032: 44400D7901112D70 4BA
033: 050D750509700000 804
034: 0D70000000384540 B47
035: 4020014E322E3140 E9B
036: 084E794142400000 1EB
037: 000000000002E4559 529
038: 3200000000000000 83E
039: 0000000000000026 B56
03A: 5556587008365556 EB5
03B: 5810083645464830 206
03C: 0832414248700024 547
03D: 5655587008345655 8A4
03E: 5810083446454830 BF3
03F: 0C3042414C700024 F48
040: 5556587008355654 2A5
041: 5810083546444830 5F4
042: 0C3142404C700025 94A
043: 5455587008355455 CA4
044: 5810083544454830 FF2
045: 0C3140414C700875 354
046: 14141870000A4972 6A5
047: 40000E3159454E30 A05
048: 0C7A0F7949400024 D7D
049: 5554587000084A71 0D9
04A: 40000C523A262D10 43A
04B: 0424587458400875 791
04C: 1415187000094A70 AE1
04D: 4000083544454830 E25
04E: 0C3140414C300C74 18D
04F: 5655545000054C71 4E4
050: 40000            5DD

XSETLEX  ID#5C  178 octets

     0123456789ABCDEF sm

000: 85355445C4548502 366
001: 802E001571221178 6B8
002: 86100C56E8E00000 A27
003: F9200000000000000 D58
004: 0E0100F3103B000F 0C3
005: 4201D000FF24F4C4 451
006: 44355445426ED94E 7DA
007: 465355445427ED55 B53
008: E444355445428E1F ED8
009: F8DD44908F064A11 27A
00A: 35109DA20D232016 5E5
00B: EE8B3CD136061BB9 9A1
00C: 700324F7816AF5AC D3D
00D: 91C114E14D161A4E 0D0
00E: 50F1C1D214DA6D5F 485
00F: D071341310300779 7D7
010: FD232FF2D51C114F B8F
011: BEE14DCD51F8D7B1 F6D
012: 8100707FD232FF2D 307
013: 53108AEA1C114F0E 6A4
014: 6A14DCD50F40D007 A45
015: 54F31F7D51C114B1 DDB
016: C114F0E6A14B14D1 177
017: C114F0E6A14D1C5A 525
018: 6D56D44CF        748

HORNER   ID#5C  122 octets

     0123456789ABCDEF sm

000: 84F425E454250202 36A
001: 802E001571221178 6BC
002: 8F000C5D0D000000 A2A
003: F710000000000000 D58
004: 092000FB94E44554 0D3
005: 2505D01FF8888888 468
006: 888888883F108137 7E5
007: 1F0C8F2145137136 B5E
008: 10A136118A4EA4EA EF9
009: 4E10B8FDF8E08FFB 2E0
00A: 6C0100AF410117F8 65D
00B: FDF8E08FFB6C0119 A34
00C: AF71188FA34C0AF9 DF5
00D: AF7AF617F8FDF8E0 1E3
00E: 8FFB6C08F363C012 592
00F: BA4E12B55C8F499C 957
010: 01F0C8F214313111 CC0
011: 28F612F0F        ECE

HMSLEX   ID#E1  363 octets

     0123456789ABCDEF sm

000: 84D435C454850202 36D
001: 802E002571221178 6C0
002: BD2001EC1C1E3000 A4E
003: F710000000000000 D7C
004: 0C1000FB84D435D4 10C
005: 55C4C11FF8822853 4A2
006: 8528C18101E61910 80D
007: 0000F23000000000 B38
008: 00000B5100FF0004 E90
009: 100FE10C4000F720 203
00A: A3000FB84D435144 584
00B: 44461B84D4353555 8F7
00C: 2471584D43581384 C5E
00D: 25911FF811840580 FCF
00E: 8118508418528 43A 332
00F: F0AF18F9D3D08FFE 71B
010: 8E087280BCC85273 AB2
011: 517D718F9D3D0725 E4B
012: 1E4E48706077317E 1D1
013: 01860607A2174318 528
014: 706078F074F08FC1 8BD
015: 4D07341CCCC74318 C57
016: 71008F499C087090 FCE
017: 8D832F010810C110 33C
018: AC0AF22E306A3E10 6E3
019: 9B34B34B34B34790 A70
01A: 15B179F04607B017 DEA
01B: EE04907001100118 144
01C: 114AC669AF882284 4DA
01D: 251188888888882A 856
01E: 852843851840A4EA BD9
```

```
01F: 4E10C76FE8F9D3D0 FA0
020: 17F8735170FE11CA 33E
021: 4E10C53E4418FFE8 6F2
022: E08F5F3D07820850 A85
023: AF210810984161DE E0E
024: AF22E306AF7A8286 1BE
025: 0B0CC8DA34C0E48D 582
026: CA4C08F004D08D4F 939
027: 6C08FE04D01517AF CE2
028: 9AF774EFBCC15776 0B1
029: A008F5F3D08D363C 45D
02A: 0A3CA3C1008FFB6C 81B
02B: 077BFAD4011208FF BC2
02C: B6C072BFAD411997 F72
02C: B6C072BFAD411997 F72
02D: 6C1AF01208FFB6C0 325
02E: 8F033C0AD4120118 6A2
02F: 8FF53C06DBF      94E
```



DIVERS

Mach.	Auteur	No	Pg	Titre	Langage
	J. Taillandier	41	22	Ragots, bruits et rumeurs...	
	J.J. Dhénin	42	2	L'Assemblée Générale	
	P. David	42	6	Nouveaux produits	
	P. David	43	4	Nouveaux produits	
	J. Taillandier	45	4	Module graphique pour HP-71	Assembleur
	P. David	47	3	La conférence de Copenhague	
	P. David	47	7	Nouveaux produits	
	PPC Paris	50	3	Des nouvelles de JPC Rom	

PROGRAMMES

Mach.	Auteur	No	Pg	Titre	Langage
HP-28	P. David	41	4	Le HP-28C	RPL
HP-28	W.C. Wickes	41	13	Le HP-28C par son père	RPL
HP-28	P. David	43	6	Développez vos expressions	RPL
HP-28	P. David	44	6	Des courbes et des paramètres	RPL
HP-28	D. Dalila	45	6	Utilisation de la pile opérationnelle	RPL
HP-28	P. Bassaler	46	6	Avec ou sans sommations !	RPL
HP-28	H.P.	46	7	Mode horloge sur HP-28C	Assembleur
HP-28	J.D. Dodin	46	8	La pile opérationnelle (acte II)	RPL
HP-28	W. Mier-Jedrzejowicz	47	10	Programmation Synthétique	Assembleur
HP-28	W. Mier-Jedrzejowicz	47	14	La fonction PEEK	Assembleur
HP-28	H.P.	47	14	Les anomalies du HP-28C	RPL
HP-28	M. Maupoux	48	6	Tracé de fonctions	RPL
HP-28	H.P.	49	6	Mode Horloge (acte II)	Assembleur
HP-28	E. Gengoux	49	6	Encore des courbes !	RPL
HP-28	P. Courbis	50	8	Choc en retour (courbes)	RPL
HP-28	P. Courbis	50	8	Lex sur HP-28C	Assembleur
HP-28	L. Istria	50	9	Réduisez vos matrices	RPL
HP-41	G. Tisserand	41	24	Agrippine	RPN
HP-41	G. Tisserand	42	10	Profil de voies urbaines	RPN
HP-41	G. Toublanc	43	8	Fractionnez aussi	RPN
HP-41	E. Gengoux	44	8	Tarification SNCF (acte II)	RPN
HP-41	C. Gottheimer	45	10	Utilitaires pour matrices	RPN
HP-41	T. Guilloux	46	10	Le compte est bon	RPN
HP-41	R. Dine	46	11	Classements	RPN
HP-41	J. Elhay	47	18	Probabilités	RPN
HP-41	R. Dine	47	19	Les nombres premiers	RPN
HP-41	M. Markov	48	8	Utilitaires de gestion de disques (I)	RPN
HP-41	M. Markov	49	10	Utilitaires de gestion de disques (II)	RPN
HP-41	V.M. Electronics	50	12	Le module HEPAX	Assembleur
HP-41	R. Dine	50	15	Portfolio	RPN
HP-71	G. Toublanc	41	32	Combinaisons et arrangements	Assembleur
HP-71	X. Bille	41	35	Annuaire électronique	Basic
HP-71	C. Bacquet	41	36	Masses atomiques	Basic
HP-71	X. Bille	42	16	Tracer toutes les fonctions	Forth
HP-71	S. Vaudenay	42	19	Jeu de la vie	Forth
HP-71	G. Toublanc	42	21	Réfractaire	Assembleur
HP-71	G. Toublanc	42	27	FracStory	Assembleur
HP-71	A. Gillet	42	28	Débuter en boucles et tests	Basic

PROGRAMMES (suite)

Mach.	Auteur	No	Pg	Titre	Langage
HP-71	A. Herreman	42	28	Agenda	Basic
HP-71	E. Gengoux	42	29	Dessine-moi un autre mouton	Basic
HP-71	P. David J. Taillandier	43	16	FINPUT	Assembleur
HP-71	A. Herreman	43	42	Anagrammes et permutations	Basic
HP-71	A. Gillet	43	43	Questionnaires à Choix Multiples	Basic
HP-71	J. Baudier	44	18	Etats d'âme	Assembleur
HP-71	F. Stengel	44	18	Les objets en Forth	Forth
HP-71	J.M. Kefaloucos	44	24	Horloges Astronomiques	Basic
HP-71	P. Ehrenberg V. Klann	44	25	Dérivées symboliques	Basic
HP-71	S. Tendon	44	28	Graphes de Connett	Basic
HP-71	X. Bille	45	12	Catalogue des mots et utilitaires	Forth
HP-71	J. Baudier	45	14	L'assembleur du HP-71 (acte I)	Assembleur
HP-71	M. Martinet	45	15	Encore les mémoires de masse !	Assembleur
HP-71	J. Taillandier	45	19	Trouvez vos chaînes (acte II)	Assembleur
HP-71	A. Goubault	45	24	To ken or not to ken	Basic
HP-71	F. Duret-Lamouroux	45	25	Programme de débutant	Basic
HP-71	M. Martinet	45	25	Le point sur les nuages	Basic
HP-71	J. Baudier	46	16	L'assembleur du HP-71 (acte II)	Assembleur
HP-71	T. Tarvainen	46	18	Conversion de caractères	Assembleur
HP-71	P. Nicodème	46	25	Généalogie Assistée par Ordinateur	Basic
HP-71	X. Bille	47	22	Tracé de courbes (acte II)	Forth
HP-71	P. David J. Taillandier	47	24	Assembleur pour HP71	Assembleur
HP-71	J. Buitenhuis F. van der Windt	47	24	Module de développement	Assembleur
HP-71	J. Baudier	47	26	L'assembleur du HP-71 (acte III)	Assembleur
HP-71	J. Elhay	47	29	Inversez vos drapeaux	Assembleur
HP-71	E. Gengoux	47	30	Retrouvez vos fichiers	Basic
HP-71	J. Baudier	48	16	L'assembleur du HP-71 (acte IV)	Assembleur
HP-71	J. Elhay	48	22	Deux fonctions pour comparer	Assembleur
HP-71	G. Toublanc	48	23	Choc en retour (DIVILEX)	Assembleur
HP-71	O. Arbey	48	27	La guerre des noyaux (acte I)	Basic
HP-71	A. Goubault	49	16	Tours de Forth	Forth
HP-71	PPC Paris	49	18	Récapitulation des tokens	Assembleur
HP-71	L. Guillou	49	19	Chronomètre	Assembleur
HP-71	J.Y. Naour	49	22	Oui ou non ? Il faut choisir !	Assembleur
HP-71	P. David J. Taillandier	49	24	Encore des dates...	Assembleur
HP-71	T. Besançon	49	34	Dérivées symboliques (acte I ter)	Basic
HP-71	G. Kossmann	49	34	Catalogue de mémoire de masse	Basic
HP-71	J. Baudier	50	20	L'assembleur du HP-71 (acte V)	Assembleur
HP-71	J.Y. Naour	50	27	Des caractères pour faire joli	Assembleur
HP-71	J. Elhay	50	28	Evaluez vos polynômes	Assembleur
HP-71	G. Toublanc	50	29	Arrondir dans la bonne base	Assembleur
HP-71	B. Gil	50	33	Dérivées symboliques (acte I quater)	Basic
HP-71	O. Arbey	50	33	La guerre des noyaux (acte II)	Basic
HP-71	E. Gengoux	50	34	Un catalogue ordonné	Basic
HP-71	J. Maille	50	35	Ca date !	Basic
HP-75	E. Gengoux	41	28	Les logiciels professionnels	Divers
HP-75	E. Gengoux	42	12	Compatibilité des Basic HP71/HP75	Basic
HP-75	E. Gengoux	43	12	Le PMS du HP75	Assembleur
HP-75	P. Picheret	44	12	Graphiques sous VisiCalc	VisiCalc
HP-75	E. Gengoux	46	14	Utilitaires VisiCalc	VisiCalc
HP-75	E. Gengoux	48	14	Larticle est-il au catalogue ?	Basic
HP-75	E. Gengoux	49	14	Le nouveau coin des Lhex	Assembleur
HP-75	J.Y. Hervé	50	18	Récupération de fichier avec MADLEX	Basic

PPC PARIS CHAPTER

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

BULLETIN D'ADHESION

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

Profession ______________________________________

Intérêts ______________________________________

____

Matériel HP en votre possession ______________________________________

____

Autre matériel informatique ______________________________________

____

Comment avez-vous connu PPC Paris ? ______________________________________

____

Que recherchez-vous au sein de PPC Paris ? ______________________________________

____

____

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

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

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

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

Le montant de la cotisation s'élève à 350.00 F pour un an.

Etudiants: 300.00 F (justificatif indispensable)

Paiement effectué le |_|_|/|_|_|/19|_|_| à l'ordre de PPC Paris Chapter,

par [ ] chèque bancaire, [ ] chèque postal ou [ ] mandat lettre

Veuillez envoyer toute correspondance à :

PPC Paris Chapter, BP 604, 75028 Paris Cedex 01, France

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

La maquette de ce numéro a été préparée et réalisée par Pierre David et Janick Taillandier, grâce à un système comprenant un HP71B, deux lecteurs de disquettes HP9114A, deux HP9807A, un HP9153, un HP9154 et une imprimante LaserJet.

Directeur de la publication : P. Guez
Numéro ISSN : 0762 - 381X

ENGLISH SUMMARY

JPC 50 - DECEMBER 1987 / JANUARY 1988

We are proud to present you our 50th issue of JPC. The documents enclosed with this special issue are about our general meeting which will take place in Paris, on January 16th, 1988. This meeting is very important for us as far as new regulations will be discussed : they are mainly intended to better meet French laws requirements. Also, a new Board will have to be elected. Pierre David is expected to be elected as president, Janick Taillandier as treasurer, Jean-Jacques Dhénin, Olivier Arbey and Eric Gengoux as secretaries. Philippe Guez will no longer be member of the Board.

The JPC Rom operation began just a year ago. Since that time, the project has been presented at the Copenhagen Conference and more and more people are interested in it. We have also prepared an English manual with Michael Markov. Some foreign Clubs have agreed to distribute it in their countries. This Rom is now available to our members on a permanent basis. A lot of new functions have been added. It is now about 22 KB.

A major article about the HP-28C appears on page 8. Paul Courbis describes us his method to load assembly language code in the HP-28C. An example is given, which inverts the LCD display. More articles will follow, explaining his discoveries about the HP-28C internals and its disassembling. Things are moving fast in this field, and we do our best to give all informations as fast as possible.

Another article in our large HP-28C columns describes a little program reducing arrays and systems of equations by removing redundant equations.

The HP-41 columns begin with a translation of the HEPAX module description. This module, from the Danish V.M. Electronics company, seems to be a great product, and we are eagerly awaiting to see one.

The « Portfolio », which is the theme of the article from René Dine, is a French game organized by *Le Figaro*, a French daily newspaper.

The HP-75 article is written by Jean-Yves Hervé. He gives us a program to recover purged files using its MADLEX Lex file.

The HP-71 columns begins with the fifth part of the series by Jacques Baudier teaching assembly language programming. Thanks to Jacques for this achievement ! Jean-Yves Naour gives us three functions in assembly to extend the character set on the LCD display : you have bold, reverse video, and underline character sets. They are derivated from the standard (internal) character set by bit manipulations. Our Australian friend, Jack Elhay initiates us with optional arguments in a function by providing a polynomial evaluation based on Horner algorithm. The last article in our assembly language section improves a previous realization by Michel Martinet. The old one had rounding problems.

One more correction to the symbolic differentiation article published in JPC 44 appears on page 33. The new, debugged version is listed on page 36. The next article analyses the structure of the MARS program, published in JPC 48. Then follows a program by Eric Gengoux to produce a sorted directory of a mass medium into a file.

As always in the December issue, we publish a complete list of articles published during the year.

Until next year,

Happy New Year, Happy Programming and Happy JPC reading !

ENGLISH SUMMARY

JPC 50 - DECEMBER 1987 / JANUARY 1988

We are proud to present you our 50th issue of JPC. The documents enclosed with this special issue are about our general meeting which will take place in Paris, on January 16th, 1988. This meeting is very important for us as for as new regulations will be discussed : they are mainly intended to better meet French laws requirements. Also, a new Board will have to be elected. Pierre David is expected to be elected as president, Janick Taillandier as treasurer, Jean-Jacques Dijkstra, Olivier Arbey and Eric Gengoux as secretaries. Philippe Ozier will no longer be member of the Board.

The JPC Rom operation began just a year ago. Since that time, the project has been presented at the Copenhagen Conference and more and more people are interested in it. We have also prepared an English manual with Michael Markov. Some foreign Clubs have agreed to distribute it in their countries. This Rom is now available to our members on a permanent basis. A lot of new functions have been added. It is now about 22 KB.

A major article about the HP-28C appears on page 8. Paul Combis describes us his method to load assembly language code in the HP-28C. An example is given, which inverts the LCD display. More articles will follow, explaining his discoveries about the HP 28C internals and its disassembling. Things are moving fast in this field, and we do our best to give all informations as fast as possible.

Another article in our large HP-28C columns describes a little program reducing arrays and systems of equations by removing redundant equations.

The HP-41 columns begin with a translation of the HEPAX module description. This module, from the Danish V.M. Electronics company, seems to be a great product, and we are eagerly awaiting to see one.

The « Portfolio », which is the theme of the article from René Donc, is a French game organized by Le Figaro, a French daily newspaper.

The HP-75 article is written by Jean-Yves Herec. He gives us a program to recover purged files using its MAGIDEX.Lex file.

The HP-71 columns begins with the fifth part of the series by Jacques Baudier teaching assembly language programming. Thanks to Jacques for this achievement ! Jean-Yves Nexon gives us three functions to assembly to extend the character set on the LCD display : you have bold, reverse video, and underline character sets. They are derivated from the standard (internal) character set by bit manipulations. Our Australian friend, Jack Erbay initiates us with optional arguments in a function by providing a polynomial evaluation based on Horner algorithm. The last article in our assembly language section improves a previous realization by Michel Martinet. The old one had rounding problems.

One more correction to the symbolic differentiation article published in JPC 44 appears on page 30. The new, debugged version is listed on page 30. The next article analyses the structure of the MARS program, published in JPC 48. Then follows a program by Eric Gengoux to produce a sorted directory of a mass medium into a file.

As always in the December issue, we publish a complete list of articles published during the year.

Until next year,

Happy New Year, Happy Programming and Happy JPC reading !