JPC

AVRIL 1988

NUMERO 53

Le numéro 40 F

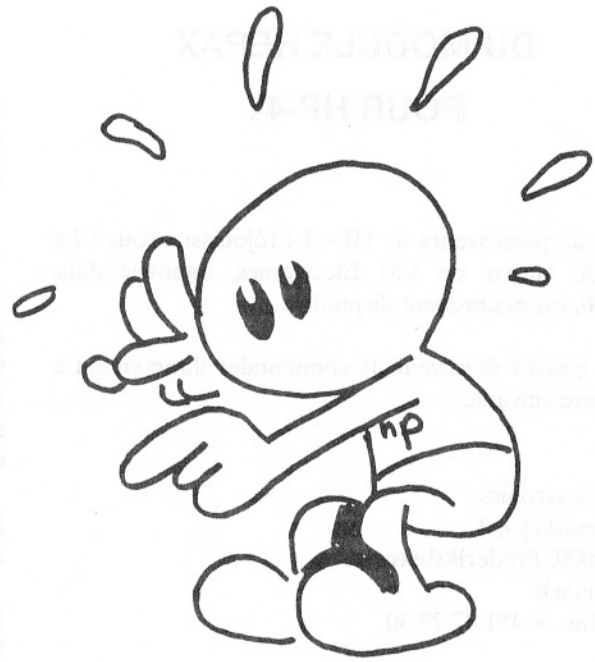
A PROPOS DU CLUB

HP28

HP75

HP71

EDITORIAL



Chers Amis,

Permettez-moi de souhaiter la bienvenue aux membres de PPC Lausanne, qui ont décidé de recevoir *JPC*. Nos amis ont en effet choisi de partager notre Journal, tout en conservant leur personnalité.

Cette coopération entre nos deux Clubs renforce ainsi *JPC* dans son objectif de liaison entre les utilisateurs. Sachez en profiter !

Autre nouvelle très importante : David Lin, le président de la firme Corvallis MicroTechnology, passera à Paris lors d'une de nos prochaines réunions (le 16 avril ou le 7 mai) pour présenter ses produits, et en particulier le nouvel ordinateur MC II. Cette réunion sera donc exceptionnelle. Ne la manquez surtout pas !

Allez, en attendant, faites de beaux rêves...

Pierre David (37)

DISPONIBILITE DU MODULE HEPAX POUR HP-41

Heureux possesseurs de HP-41 : réjouissez-vous ! Le module Hepax de VM Electronics, annoncé dans *JPC 50*, est maintenant disponible.

Vous pouvez désormais le commander directement à l'adresse suivante :

VM Electronics
Nyelandsvej 7, 1.
DK-2000, Frederiksberg
Danemark
Tél : Int. + 451 87 23 30

Rappelons que ce module offre une mémoire supplémentaire à votre HP-41 dans laquelle vous pouvez non seulement sauvegarder tous vos programmes, mais également les exécuter sans avoir besoin de les recopier en mémoire principale ! En cela, cette mémoire est radicalement différente de la classique mémoire étendue. 32000 octets au maximum peuvent être ainsi rajoutés.

Les commandes utilisées ont une syntaxe proche de celle des modules XFonctions.

Vous disposerez également d'un éditeur de programmes en langage machine rapide et performant, ainsi que de tous tous les outils nécessaires pour la programmation en assembleur de la HP-41.

Les prix des différents modules sont :

- Advanced Hepax : DKK 2475 (2300 FF)
- Standard Hepax : DKK 1800 (1700 FF)
- Hepax Memory : DKK 1475 (1400 FF)
- Hepax Double Memory : DKK 2050 (1900 FF)

Les prix en Francs sont donnés à titre purement indicatif.

Olivier Arbey (118)

COURRIER DES LECTEURS

Alexandre Allien
5 avenue Salvador Allende
77200 Torcy

Sos !

Quand j'ai reçu *JPC 50*, je me suis précipité sur l'article *Lex sur HP-28C* de Paul Courbis, d'un doigt allègre, j'ai tapé le programme... Horreur ! Les routines en langage machine ne se trouvent pas sur la version 1CC.

Je pense que nous sommes beaucoup dans le même cas. Que devons-nous faire ?

J'apprécie énormément notre journal, longue vie à lui !

Alexandre Allien (459)

C'est une excellente question. La réponse se trouve dans JPC 47 *(page 10). Wlodek Mier-Jedrzejowicz a donné une version de* PEEK *qui devrait fonctionner sur les versions 1CC. La suite, c'est à toi de la trouver, à l'instar de Paul Courbis...*

S.O.S.

Philippe Aspero
59 rue du Maréchal Foch
68300 Village-Neuf

Recherche :

Des programmes de calcul en multi-précision pour HP-71 (calcul de /, ×, -, +, π, e, n!, y^x, sin x,...). Tous les programmes sont bienvenus !

Serge Vaudenay
8 résidence des Chênes
94420 Le Plessis Trévise

Recherche :

Une méthode efficace pour trouver les racines d'un polynôme complexe, afin de programmer la fonction PROOT du HP-71 sur le HP-28. L'exposé mathématico-intuitif de HP sur la méthode employée (ici la méthode de Laguerre (???)) est incompréhensible...

Pierre-Marie Vougier
1 bis, rue du Gabon
83200 Toulon

Recherche :

Des programmes d'inversion de matrice pour HP-41, et aimerait savoir jusqu'à quel degré on peut monter en utilisant toute la mémoire disponible.

Des programmes de lissage de courbes sur des polynômes de degré élevé.

Guillaume Le Stum
12 impasse Houël
76600 Le Havre

Recherche :

Comment utiliser la fonction TAYLR du HP-28 et obtenir des coefficients rationnels (c'est à dire que les fractions ne soient pas effectuées) ?

Entre l'envoi de ce S.O.S. et le fabrication de ce JPC, Guillaume nous a envoyé la réponse. Nous vous la livrons donc :

J'ai trouvé la réponse suite à une erreur de manipulation... Le problème était d'obtenir avec la commande TAYLR des développements limités avec des coefficients non effectués. Il suffit de multiplier l'expression à développer par une variable non encore définie.

Par exemple, si A n'est pas présent dans le menu USER :

```
'A*ATAN(X)' [ENTER]
'X' [ENTER]
3 TAYLR
```

donnera : `'A*X-A*2/6*X^3'`. Il ne reste plus qu'à faire mentalement abstraction de A, et le tour est joué... Longue vie au Club !

COURRIER DU COEUR

Sébastien Lalande
12 rue de Seine
78290 Croissy sur Seine
Tél : (1) 39 76 27 41

Fait :

Pose de modules 4, 34 ou 64 Ko pour HP-28C ; accélération HP-28C et connexion entre HP-28C et ordinateurs.

Jean-Jacques Moreau
64 avenue de la Paix
93150 Le Blanc-Mesnil
Tél : (1) 48 67 33 04

Vend :
Lecteur de disquettes HP-9114A : 3000 F.

Janick Taillandier
335 rue Lecourbe
75015 Paris

Vend :
Module 32 Ko Ram + 32 Ko Eprom HHP pour HP-71 : 1500 F.

Alain Bochet
108 rue de Lagny
93100 Montreuil
Tél : 42 87 96 89 (Dom) ou 42 07 75 36 p45 (Bur)

Vend :
Lecteur de cassettes HP-82161A + 4 cassettes + sac de transport HP (peut contenir le drive + alimentation + HP-41 ou HP-71), le tout en excellent état: 1500 F.

Philippe Guez
56 rue J.J. Rousseau
75001 Paris
Tél : (1) 45 06 25 03 (Bureau)

Vend :

Imprimante ThinkJet HP-2225B : 3500 F, interface vidéo 32 colonnes HP-82163B : 500 F, lecteur de cartes magnétiques pour HP-71B : 1250 F, cartes magnétiques pour HP-71B : 1 F pièce, convertisseur HP-82166A : 1250 F, module HP-IL pour HP-41 : 1250 F, modules Ram 4 Ko pour HP-71B : 500 F pièce, module Games, Auto-duplication, et Math pour HP-41 : 150 F pièce, machine à écrire Praxis 35 interfaçable RS232 : 3000 F.

Henri Kudelski
Chemin de la Croix, 48
1052 Le Mont sur Lausanne
Suisse

Vend :
Deux HP-28C neuves : 1300 FF à débattre.

Guillaume Le Stum
12 impasse Houël
76600 Le Havre
Tél : 35 42 34 48

Vend :
HP-28C, état neuf, sous garantie encore 8 mois, 1600 F.

C. Becker
27 avenue Verdi
59110 La Madeleine
Tél 21 71 50 07

Vend :
Pour HP-71 un module HP-IL HP-82401AF, une imprimante HP-2225B, un module éditeur de textes HP-82485AF et un lecteur de cartes magnétiques HP-82400A. Prix à débattre.

DERNIERE MINUTE !

La firme suédoise bien connue Šø Ðvün¢æn a réalisé un module extraordinaire pour HP-41 : il s'agit d'un émulateur combiné HP-28S et HP-71. Surnommé *49.5 Emulator Pac* (NdlR : 49.5 est la moyenne de 28 et 71...), il inclut un coprocesseur compatible avec le HP-71 (et le HP-28S par la même occasion), et contient les 64 Ko de Rom du HP-71 et les 128 Ko du HP-28S, plus une mémoire autonome de 32 Ko.

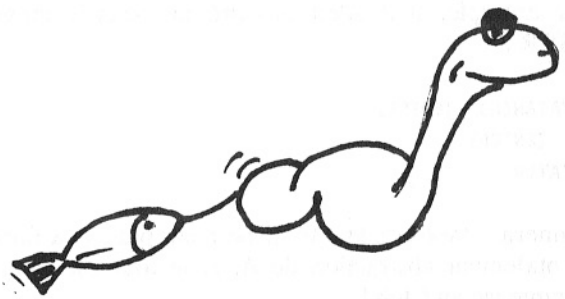
Ce module comprend deux fonctions :

- `HP28` (qui permet d'entrer dans le mode d'émulation HP-28, donc d'accéder aux fonctions de calcul symbolique et de calcul matriciel ou complexe),
- `HP71` (qui permet de même d'accéder au Basic très complet du HP-71).

Signalons de plus que ce co-processeur tourne environ quatre fois plus vite (4,71281 MHz) qu'un HP-71 ou un HP-28 de base.

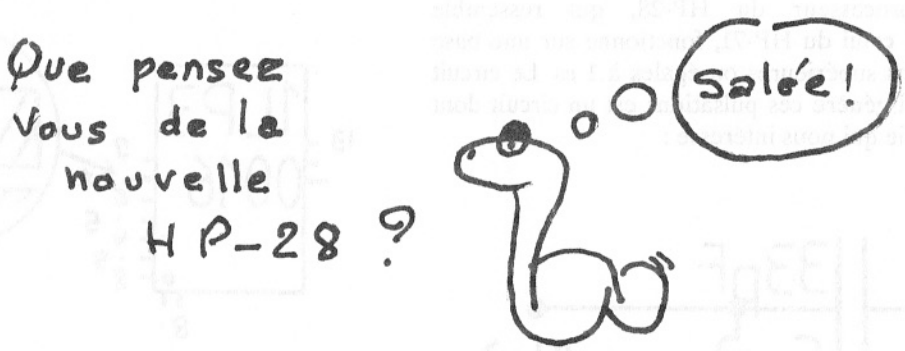
Ce module dispose enfin d'une sortie infra-rouge, et permet aux deux modes d'utiliser l'HP-IL du HP-41.

Gageons que nous reparlerons bientôt de ce fantastique module !



HP28

P. Courbis
S. Lalande



MODE HORLOGE SUR HP-28S

Le programme pour simuler une horloge sur le HP-28C paru dans *JPC 46* (Juillet-Août 1987) est maintenant utilisable sur le HP-28S (version 2BB).

Il suffit pour cela de modifier l'adresse utilisée pour le SYSEVAL. Tapez #11CA SYSEVAL au lieu de #123E SYSEVAL.

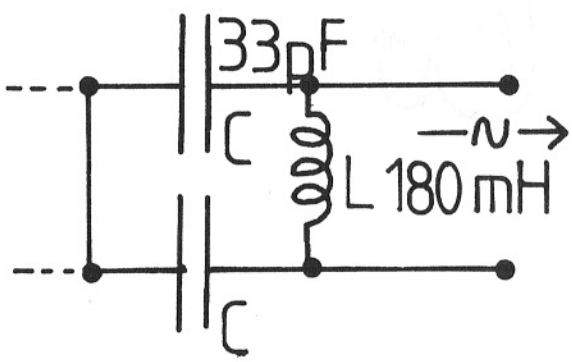
Paul Courbis (392)

HP-28 GTI

Comme vous avez déjà dû vous en rendre compte, le HP-28C possède trois défauts : le manque de mémoire (l'insupportable No Room to ENTER doit commencer à vous agacer...), la lenteur et le manque de communication avec l'extérieur. Je vous propose d'apporter la solution à vos problèmes.

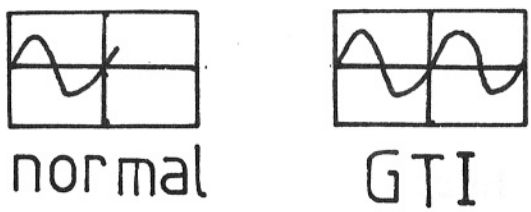
GTI

Le microprocesseur du HP-28, qui ressemble beaucoup à celui du HP-71, fonctionne sur une base de pulsations supérieures ou égales à 1 μs. Le circuit oscillant qui génère ces pulsations est un circuit dont voici la partie qui nous intéresse :



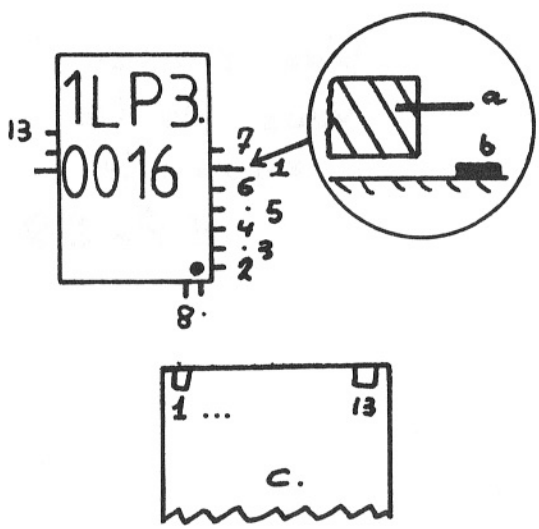
La valeur des condensateurs est à l'origine de 33 pF. Si on remplace cette valeur par 10 pF, on s'approche au mieux de la limite sans aucun risque. Il suffit donc de remplacer les deux condensateurs C (figure 1) par deux autres de 10 pF.

Résultat : 1,56 fois plus rapide (autant que le HP-28S). Pour voir la stupéfiante différence, faites l'auto-test ([ON] [←]) ou tracez la fonction sinus :



MEMOIRE

Après beaucoup de difficultés, j'ai réussi à déterminer les connexions de la Rom 1LP3 0016 (figure 1). Les *H.D.S.* du HP-71 m'ont fourni ceux de différents modules existants. Je vous propose le raccordement suivant qui vous permettra de mettre tous les modules que vous voudrez : 4, 32 ou 64 Ko. Ces connexions marchent parfaitement et les différents HP-28 que j'ai modifiés sont à faire pâlir d'envie la plupart des admirateurs du HP-28 normal. Plus jamais de No Room to ENTER ! Vous pouvez sauvegarder des pages écran, faire des animations ou des maxi-programmes en assembleur ! Assez ri, passons aux choses sérieuses :



(*a* est non connecté, le *1* est à connecter en b, le module (*c*) est présenté face sans composant)

Voici à quoi correspondent les différents raccords :

```
1 : VDD        2 : B3
3 : B2         4 : B1
5 : B0         6 : *STR
7 : *CD        8 : DIN
9 : DOUT      13 : GND
```

Le *9* sert pour le chaînage de plusieurs modules. Pour connecter plusieurs modules, il faut brancher le DIN du premier sur le DIN du circuit, le DOUT du premier sur le DIN du second, le DOUT du second sur le DIN du troisième etc.

Une fois les connexions réalisées, faites Memory Lost ([ON] [INS] [→]) puis l'auto-test ([ON] [INS] [←] ; si lors de la présentation de la Ram, la machine s'arrête au bout de 4 pages, ceci signifie que le module possède une mauvaise connexion avec le cicuit imprimé) et MEM. Epatant ! Remarque : cette opération est assez délicate pour quelqu'un qui ne l'a jamais faite et une erreur peut provoquer la destruction de la Rom ou du module.

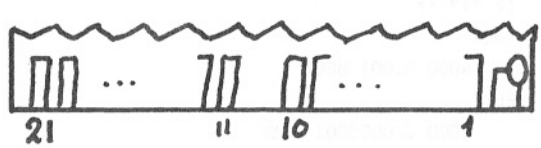
ALIMENTATION

Deux possibilités : ou bien vous vous fabriquez un connecteur branché sur un transformateur qui se loge à la place des piles ou bien vous vous branchez sur le gros condensateur aux bornes - et + (figure 1) et vous utilisez une petite prise de type mini-jack ou mini-broche que vous encastrerez dans le boîtier découpé à cet effet.

ENTREES

Première méthode

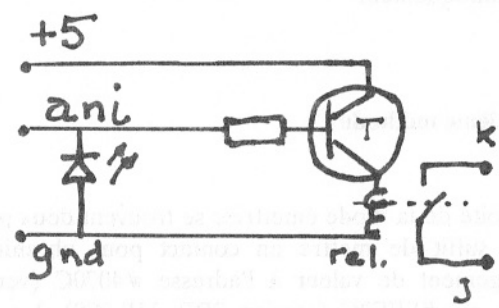
Celle-ci consiste à simuler l'appui des touches au clavier en mettant en contact deux des pistes au bas de la carte.



	4	7	10	13	15	18
21	INS	DEL	↑	↓	←	→
20	[ ]	MODE	TRIG	SOLV	USER	NEXT
19		ENTER	CHS	EEX	DROP	BACK
17	'	7	8	9	÷	
16	STO	4	5	6	×	
14	EVAL	1	2	3	-	
11		0	.	,	+	

	9	8	1	2	3	5
20	A	B	C	D	E	F
19	G	H	I	J	K	L
17	M	N	O	P	Q	R
16	S	T	U	V	W	X
14	Y	Z	#	{	[	(
11		Space	«	=	LC	α

Voici un exemple parmi d'autres utilisant la sortie joystick de l'Apple II. Il faut faire quatre circuits très simples (il y a quatre circuits identiques : remplacer successivement i, j et k par 0, 4, 21 puis par 1, 7, 21 puis par 2, 7, 14 et enfin par 3, 7, 11) R=4700 ohms, T=BC237, relais = ILS Günther 3753 1231 051, la diode est une LED ordinaire.



L'Apple commande alors l'appui des deux premières touches du menu et le 1 et le 0 à l'aide de ces 4 circuits.

Soient *A*, *B*, *1*, *0* ces touches. Sur *A* vous placez un programme qui fera trois choses même temps : placer un # dans la ligne de commandes, un programme sur *A* annonçant la fin d'un octet et la mise en place d'un # dans la ligne de commandes et, enfin, sur *B* un programme de fin totale. Par exemple :

```
'DEBUT'
«
  "" 'STRING' STO
  { F-OCT FIN } ORDER
  HEX #4F038 "0181" POKE BIN
»
```

```
'F-OCT'
«
  B→R CHR STRING SWAP +
  'STRING' STO HEX #4F038
  "0181" POKE BIN
»
```

```
'FIN'
«
  B→R CHR STRING SWAP +
  'STRING' STO HEX
  STRING STR→ { DEBUT } ORDER
»
```

Du coté Apple, on effectue `POKE -16296,0` pour mettre AN0 OFF et -16295 pour ON, -16294 pour AN1 OFF ... -16289 pour AN3 ON. Le message commence donc par appuyer sur *A* c'est à dire AN2 ON puis OFF puis sur *1* et *0* selon le cas et par AN2 pour la fin de chaque octet et à la fin du dernier on appuie sur FIN.

Ainsi, vous entrez les code décimaux en page graphique ou en fichier, sauvegardez sur Apple, à la main ou par la sortie infra-rouge et les rappelez quand nécessaire pour les entrer dans le HP-28 automatiquement.

Deuxième méthode

A droite de la diode émettrice se trouvent deux pistes qu'il suffit de mettre en contact pour obtenir un changement de valeur à l'adresse #4070C (version 1BB) ou #FFF0C (version 2BB, HP-28S). Lorsque j'accélère et mets de la mémoire sur les HP-28C, je connecte à cet endroit un ILS : ces gélules de verre ferment un interrupteur en présence de champs magnétiques. Il suffit de placer un électro-aimant en face du port infra-rouge et d'envoyer par induction des signaux longs ou courts (1 ou 0) au HP-28.

Toisième méthode

Cette méthode, encore meilleure, consiste à baisser la tension d'alimentation de 4,5 V à 3 V par le petit couvercle du compartiment des piles ce qui ne requiert aucune modification de la machine. Ce changement est répercuté à une adresse mémoire de la machine. Comme précédemment, il suffit d'envoyer des signaux longs ou courts donnant des 1 ou des 0 (plus de détails dans un prochain JPC).

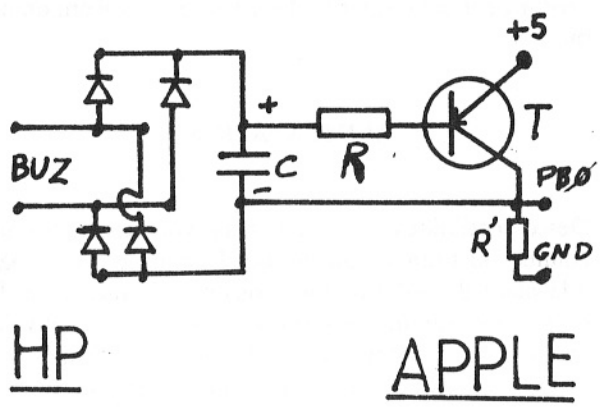
SORTIES

Première méthode

Une méthode archaïque consiste à afficher à l'écran de grosses taches noires et à les détecter avec un stylo optique.

Deuxième méthode

Il s'agit d'utiliser le buzzer. Le circuit à utiliser coté Apple est le suivant (C = 1 μF chimique, R = 47000 ohms, R' = 470 ohms, T = BC237) et vous vous branchez directement aux bornes du buzzer :



Du coté HP, il faut utiliser le programme suivant après avoir mis la chaîne à transmettre dans la pile :

```
«
  BIN 51 CF

  DUP SIZE R→B 32768 + →STR
  4 99 SUB
  1 15 FOR
    X DUP X X SUB
    IF "1" ==
    THEN
        4000 0.001 BEEP
    ELSE
        4000 0.0000001 BEEP
    END
  NEXT
```

```
  WHILE
    DUP SIZE
  REPEAT
    DUP2 OVER SIZE SUB
    SWAP NUM 512 + R→B →STR
    4 99 SUB
    1 8 FOR
      X DUP X X SUB
      IF "1" ==
      THEN
          4000 0.001 BEEP
      ELSE
          4000 0.0000001 BEEP
      END
    NEXT
    DROP
  END
  DROP
»
```

Du coté Apple :

```
 5 INPUT "NOM DU PRGM :";A$
10 HGR : POKE -16302,0
20 AD=8192 : L=0
30 FOR X=1 TO 15 : L=L+L : WAIT -16287,128
40 IF PEEK(-16287)<128 THEN NEXT : GOTO 80
50 L=L+1
60 IF PEEK(-16287)>127 THEN 60
70 NEXT
80 T=0 : FOR W=1 TO 8 : T=T+T : WAIT -16287,128
90 IF PEEK(-16287)<128 THEN NEXT : GOTO 130
100 T=T+1
110 IF PEEK(-16287)>127 THEN 110
120 NEXT
130 POKE AD,T : AD=AD+1 : IF AD<8192+L THEN 80
140 PRINT CHR$(4) "BSAVE";A$;",A$2000,L";STR$(L)
150 TEXT : END
```

C'est en utilisant cette méthode que j'ai transféré la Rom du HP-28C !

Troisième méthode

La meilleure sortie est, bien entendu, la diode infra-rouge. En effet, pour allumer la diode il suffit de faire : « #4070D "1" POKE » pour la version 1BB ou « #FFF0D "1" POKE » pour la version 2BB.

Il y a trois niveau d'allumage : "1" le plus fort, "2" le moyen, "4" le faible et "0" éteint. Lors des transformations sur le HP-28, vous pouvez placer, si vous le souhaitez, une petite diode rouge branchée en série avec l'émetteur infra-rouge : on simule ainsi un coeur, l'effet est saisissant !

Comme pour l'entrée, le circuit permettant de lier le HP-28 à un ordinateur quelconque par interface RS232 sera publié dans un prochain JPC.

Attention : une fois allumée, la diode infra-rouge le restera même après avoir éteint la machine. Il faut donc toujours veiller à ce qu'elle soit éteinte par « #4070D "0" POKE » ou par un arrêt système ([ON] [↑]). La consommation de la machine passe de 4,5 mA à 30 mA lorsque la diode est allumée et à ce rythme, les piles ne dureront pas longtemps.

OUVERTURE DE LA MACHINE

La compagnie *S.O.S.* aux Etats-Unis utilise une méthode qui revient à percer ou à fraiser le dos de la machine (attention au buzzer) puis de remettre un cache une fois les transformations effectuées. Cette méthode est peu esthétique et fragilise la machine.

La méthode proposée ici est meilleure puisque pratiquement invisible et ne fragilise pas la machine. Elle n'est détectable que par un oeil exercé : seules deux petites fissures de 1 mm par 0,5 mm sont visibles en haut à droite et à gauche du clavier.

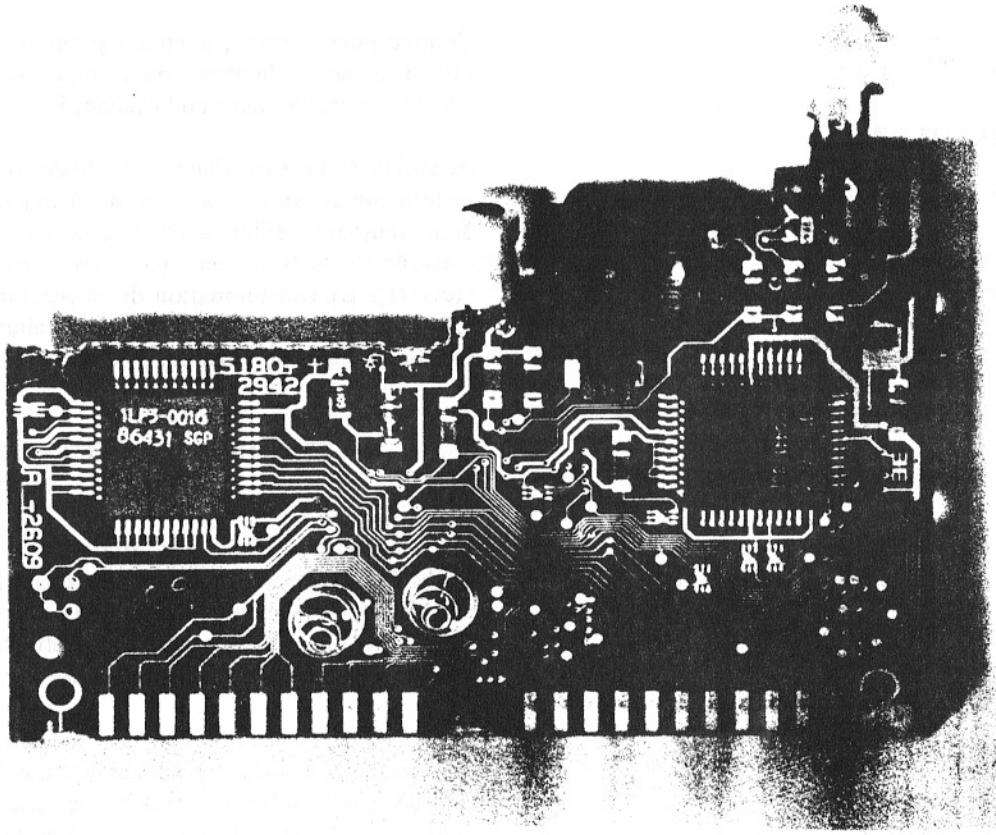
Il suffit de décoller doucement et sans l'abimer le cache qui couvre le clavier, de scier verticalement en haut du clavier, juste sous l'écran sur une profondeur de 2 mm et à l'aide d'une pince judicieusement choisie (type pince crocodile) de faire sauter le haut du HP en écartant la pince dans le compartiment des piles. On peut alors faire les transformations désirées, fraiser l'intérieur pour loger le module 32 Ko puis recoller l'ensemble. La machine est d'une solidité parfaite et d'une esthétique très peu altérée.

CONCLUSION

Ces transformations sont délicates et exigent une certaine expérience. Ayant acquis cette expérience en utilisant ma première machine comme cobaye (la pauvre...), je peux poser pour les interessés la mémoire supplémentaire, le turbo, une diode et un ILS sur leur machine préférée. Pour tout renseignement, me contacter.

Un HP-28C accéléré, muni de 34 Ko (les 2 Ko d'origine plus 32 Ko) et d'entrées / sorties permettant la sauvegarde de programmes sur disquettes est une véritable merveille !

Sébastien Lalande (442)



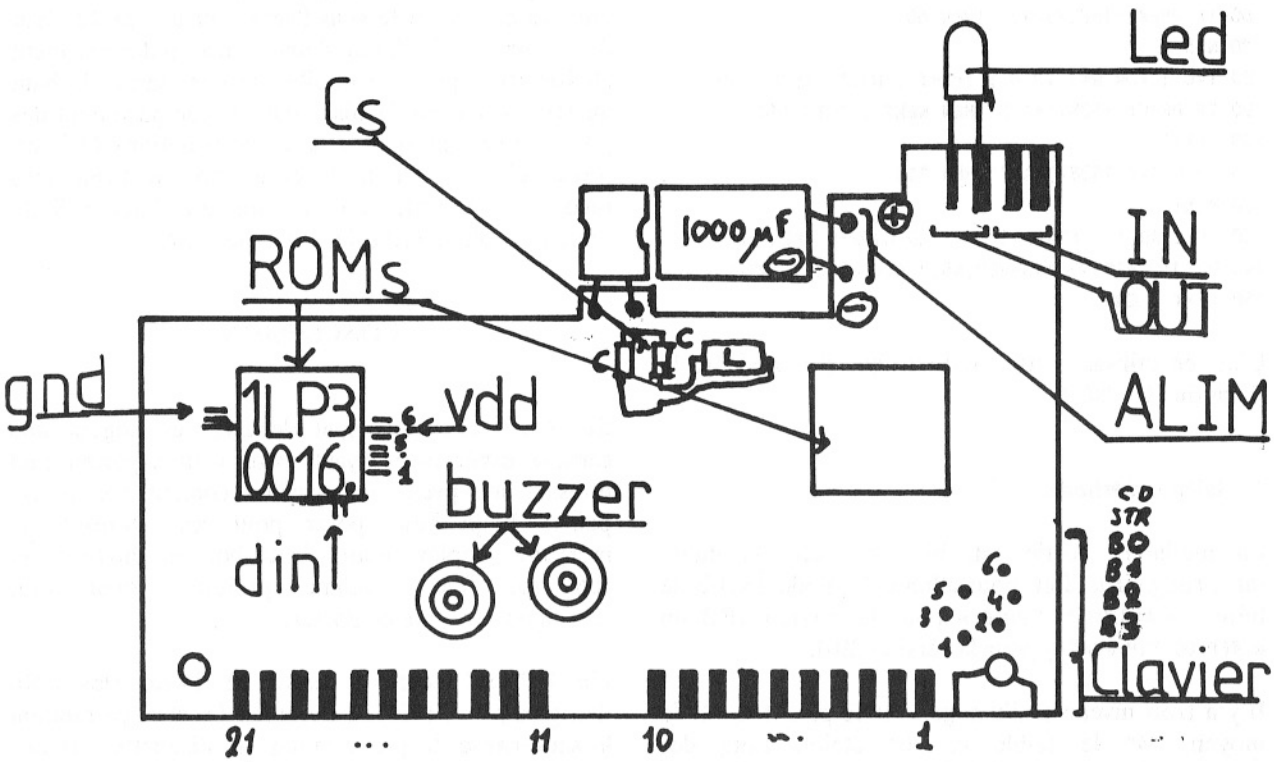


figure 1 - HP28C

HP75

UNE MEMOIRE D'ELEPHANT !

Au début de l'année, je vous avais signalé la possibilité de faire « brûler » vos logiciels favoris dans une Eprom de 32 Ko, et vous avais promis des détails dès que ceux-ci seraient disponibles... Ce qui est le cas depuis peu, puisque le Club vient de recevoir un courrier de Corvallis Micro Technology (CMT, producteur des Eprom 32 Ko et 64 Ko du HP-71), qui nous propose d'importantes remises (30%). Voici donc les tarifs pour commande directe (mais vous pouvez passer par EduCALC et régler par carte Visa ou autre...), et surtout la manière de procéder, que j'ai testée avec un logiciel de prévision de chiffre d'affaires (ou de toute autre série économique mensuelle) qui, joint à ses utilitaires, frisait justement les 32 Ko.

Tout d'abord, les tarifs ; ils s'entendent hors TVA et douane.

Désignation	Code EduCALC v	Prix EduCALC v	Prix CMT Club v
Eprom 32K vierge	#75-669	$ 229.95	$ 171.50
Programmation	#75-675	$ 60.00	$ 50.00
Frais d'expéd.	Voir selon voie		$ 15.00

Maintenant, les détails pratiques : l'Eprom se présente sous la forme d'un module d'application enfichable classique. Enfin, presque : il y a sur le dessus une fenêtre permettant l'effacement aux ultra-violets, plus trois contacts spéciaux dont le rôle reste mystérieux... voir dessin. Contrairement à ce que nous connaissons bien pour le HP-71, il n'est pas possible de brûler soi-même une telle Eprom, le brûleur n'étant pas disponible hors de CMT ; de plus, étant donnée la structure très particulière de la mémoire du HP-75, et notamment des modules enfichables (adresses 6000 à 7FFF hexa) en *pages* de 8 Koctets, ne pouvant comporter chacune qu'un seul fichier Lex et excluant certains mots-clés, voir "les pièges du PMS", il faut absolument :

- tester systématiquement l'ensemble du logiciel dont on envisage l'implantation en Eprom avec un PMS (voir plus bas l'essai de celui construit par Vincent Delorme),

- fournir sur disquette à la fois les fichiers eux-mêmes (Basic ou Lex) et les *images de pages Eprom* (fichiers particuliers de type R, obtenus avec le PMS et son Lex associé), et sur papier (lettre accompagnant votre commande) le catalogue de chacune des pages d'Eprom, dans l'ordre d'implantation (i.e. ":ROMA", ":ROMB", ":ROMC",...), *avec leurs checksums*.

Je tiens à la disposition de ceux qui envisageraient une telle implantation un modèle de lettre en anglais, et suis prêt à les assister pour finaliser leur projet. Et, puisque j'utilise maintenant un tel logiciel, d'abord implanté dans un PMS très musclé (voir ci-après), puis en Eprom, laissez-moi vous dire à quel point il est agréable de disposer d'une machine tout à la fois dédiée (logiciel résidant en permanence dans un module) et où la mémoire Ram reste complètement libre pour les données !

Puisque nous y sommes, rappelons que HP avait réalisé à l'intention des développeurs de logiciels un PMS de 16 Ko (les initiales PMS signifient Port Module Simulator, c'est à dire une mémoire Ram un peu spéciale simulant un module enfichable, mais pouvant bien entendu se charger à volonté à partir du HP-75, au moyen d'un Lex *ad hoc*), référencé HP-82173A, qui s'est vite révélé bien petit... On pouvait en brancher plusieurs sur un même HP-75, mais on devait alors retirer autant de modules, ce qui empêchait certaines combinaisons (par exemple VisiCalc + Math + Rom I/O + Eprom, ou deux modules seulement + deux PMS, dans le cas d'un logiciel de 32 Ko finaux). Certes, un montage expérimental de 48 Ko avait été présenté à la Conférence CHHU d'Atlanta, mais il avait fallu bricoler puissamment le HP-75 pour contourner la difficulté des trois ports... Or, depuis peu, un « Super-PMS » existe, qui sera très bientôt décrit dans JPC : notre ami Vincent Delorme a en effet réalisé un PMS de 32 Ko qui ne mobilise qu'un seul emplacement, donc permet le test en vraie grandeur d'Eproms de 32 Ko utilisant les fonctions de deux modules externes, sans acrobaties ni compromis. J'apprends d'ailleurs à l'instant même qu'il vient de le pousser à 96 Ko, moyennant modification du Lex PMS de HP pour pouvoir *switcher* le nombre maximum de pages de 8 Ko permis par la taille de la table ROMTAB, sans être gêné par celles correspondant aux modules déjà branchés sur la machine. Une très belle performance, d'autant plus que la documentation interne du PMS n'a pas été diffusée par HP (même en NOMAS), et qu'il a fallu désassembler le Lex PMS avec les moyens du bord, comprendre ce qu'il faisait et comment on pourrait le gonfler. A propos, si quelqu'un a des informations sur le fonctionnement interne du Pod, autre grand inconnu des NOMAS, nous sommes clients !

Ajoutons pour conclure que le super-PMS dont je vous parle, réalisé en technique *wrapping*, reste très accessible à la fois en prix de revient et en facilité de construction. Le niveau de montages est comparable à ce qui a été récemment publié dans la revue *Elektor*. A bientôt donc, le HP-75 n'est pas encore mort !

Eric Gengoux (108)

EXPLORER LES LEX

Pour connaître la liste des mots-clés contenus dans un Lex résidant en Ram (pour ceux contenus dans un module, c'est un peu plus compliqué), il fallait jusqu'ici utiliser un autre Lex, HELP (HP User's Library #00180-75-8). Cet outil offre l'avantage de donner la syntaxe de chacun des mots-clés, mais présente deux inconvénients : celui d'être payant, et de donner *tous* les mots-clés de *tous* les Lex et modules d'application, y compris les Roms système, sans aucun filtrage possible.

Le programme Basic joint, qui nécessite seulement la Rom I/O et MEMLEX (liste jointe), ne traite *que* le Lex dont le nom est spécifié, s'il a été au préalable chargé en Ram. Ainsi, pour MEMLEX, il aurait imprimé :

```
MEMLEX  422 octets  Id:13000  PEEK
                              POKE
                              ADDR
                              ROMPEEK
                              DIRECT
```

Pour aller un peu plus loin, voici un autre utilitaire, toujours en Basic, LEXCAT, qui effectue le désassemblage de l'en-tête du fichier Lex et imprime le résultat. On notera l'utilisation de la fonction REPL$ de la Rom I/O en ligne 520, pour éviter les sauts de ligne ou de page intempestifs dûs aux caractères ASCII (10) et (12).

A la fin, on obtient un listing de la forme :

```
     CATALOGUE: MEMLEX
34158  60 149              Adresse
34160 166   1              Longueur: 422
34162 141                  Accès
34163  76                  Type: L
34164 192  83 253 163      Date de création
34168  77  69  77  76  69 MEMLEX
34173  88  32  32
                           EN-TETE
38204 200  50              Lexid
38206  12   0              Runtab: 38218
38208  38   0              Asciis: 38242
38210  24   0              Partab: 38230
38212  64   0              Errmsg: 38268
38214 131   0              Intcpt: 38335
                           RUNTAB
  . . . . . . . . . . . . . . . . . . . . . . .
                           ASCIIS
38242  80  69  69 203      #1: PEEK
38246  80  79  75 197      #2: POKE
  . . . . . . . . . . . . . . . . . . . . . . .
                           ERRMSG
38269  97 100 100 114 101 #249:address not in RAM
38274 115 115  32 110 111
38279 116  32 105 110  32
38284  82  65 205
38287 119 114 111 110 103 #250:wrong ROM address
  . . . . . . . . . . . . . . . . . . . . . . .
38332 255                  Byt
38333 225                  Version: a
38334  23                  Attribute
```

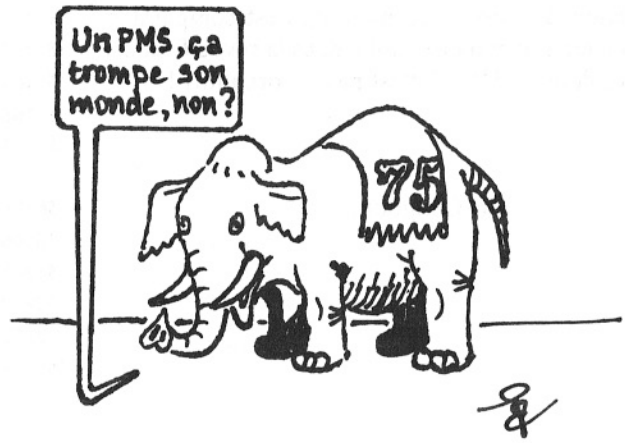
Pour aller encore plus loin, c'est à dire désassembler le Lex proprement dit, il vous faudra disposer d'un... désassembleur ! Ca y est ! Monsieur de La Palisse a encore frappé ! Celui écrit par Janick Taillandier est excellent, et sera très prochainement publié ici même.

A bientôt et... bonne exploration !

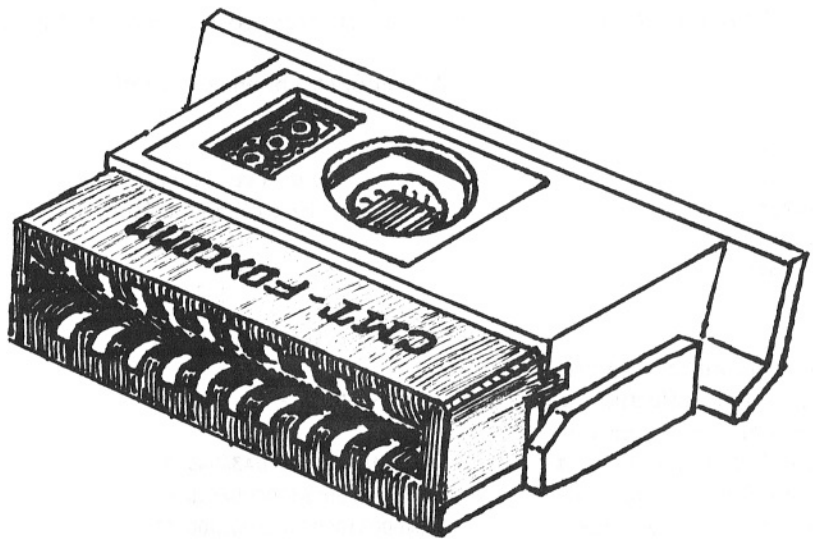
Jean-Yves Hervé (450)
Eric Gengoux (108)

```
MEMLEX L 440 Bytes ROMid:13000

Line ------- Data ------ Check

  1 39AEA6018D4CC053FDA34D45 B1
  2 4D4C45582020C8320C002600 A4
  3 180040008300000B600CA00 5D
  4 03012B019B0100009B009C00 05
  5 9B009B009B00FFFF504545CB 79
  6 504F4BC5414444D2524F4D50 8C
  7 4545CB4449524543D4FFF961 EE
```

```
 8 646472657373206E6F742069 83
 9 6E205241CD77726F6E672052 91
10 4F4D20616464726573F3696E FD
11 76616C696420524F4D206E6F 1F
12 AE656E64206F6620524F4D54 40
13 41C2FFE1179E4206E36DA8B4 92
14 6E21A16D06E54CE442E59E6C EE
15 06E30AE59E9E42B1A38202C6 F9
16 84005DA802CE64125E0AE2F0 0E
17 E6CE994CF99E102DCECD3ECE 1B
18 6146FA7D5E9321A4CE8A0160 92
19 0AE59EA1CE8B3ECECD3E7020 34
20 A15EA9FF7FCE8A01996028A3 49
21 30C5FA02F0CB5EA90060CE8A 71
22 01996028C330C5FB02F0BA60 E6
23 30A1CE6146FA7D7E21A69E18 BD
24 2DCE774ACE92245EB1B1825C E3
25 A9FF7F985E1CC5CE8A016028 E4
26 A35EA9FF7FCE8A01CEDF3CCE 3F
27 D7379E202DCE8B3E6691F20A 88
28 C94900FB05C92000FB07CE99 69
29 4CFB9EF0CA6612A253A885CE 0E
30 CD3E7020A15EA9FF5FCE8A01 00
31 996030C5FB0D5EA9FF7FCE8A D9
32 01996030C5FB07CE994CFA9E 42
33 F0D16030A1CE6146FA7D5012 46
34 A05113A0CEAC47F60E5E9321 80
35 A4CE1D48CE8A01600AE59ECE F0
36 1D48CE994CFC9E182DCE774A 8B
37 CE92245E18A1F0C8          57
38 C8E0
```



EG87113
"Le HP75 a une mémoire d'éléphant!"



15-11-87

ASSEMBLEUR

BASIC

LE COIN DES LHEX

PROGRAMMATION STRUCTUREE (ACTE II)

Comme promis le mois dernier, voici la suite de la programmation structurée sur HP-71.

INTRODUCTION

Lors de l'acte I, nous avons vu les mots-clefs permettant de programmer « proprement », c'est à dire avec des boucles n'ayant qu'une seule entrée et sortie, ou des structures de choix plus faciles à manipuler que les structures standard.

En essayant ces nouveaux mots-clefs, vous avez dû constater qu'il était difficile de relire vos programmes. En effet, les ordres LIST et PLIST ne pratiquent pas l'*indentation*. Vous ne retrouvez pas sur le papier de votre imprimante les beaux décalages de l'article précédent.

C'est l'objet de cet acte II. Le Lex que nous vous présentons aujourd'hui comprend deux nouveaux mots-clefs pour mieux présenter vos oeuvres : DBLIST liste (et PBLIST imprime) un programme Basic en indentant automatiquement les structures.

HISTORIQUE

La première apparition de ce Lex date d'octobre 1986 (*JPC 38*). Jean-Pierre Bondu nous présentait le Lex le plus volumineux jamais paru dans *JPC*.

Jean-Pierre s'était inspiré d'un programme Basic nommé JPCLISTE qui servait (jusqu'à un passé très récent) à lister les programmes Basic dans *JPC*, et l'avait transformé en assembleur.

A l'apparition de STRUC2, le Lex de Jean-Pierre a commencé à prendre un coup de vieux. L'indentation des structures devenait indispensable. D'où l'article de ce mois-ci.

Notons enfin que Jean-Pierre avait également inclus un mot-clef RENUMREM destiné à renuméroter un programme en tenant compte des lignes de remarques. Nous avons repris ce mot-clef tel quel, et nous renvoyons le lecteur intéressé à JPC 38 pour plus de détails.

SYNTAXE

Les syntaxes de DBLIST et PBLIST sont identiques :

Syntaxe classique

DBLIST
DBLIST *départ*
DBLIST *départ* , *fin*
DBLIST *fichier*
DBLIST *fichier* , *départ*
DBLIST *fichier* , *départ* , *fin*

- premier cas : le fichier courant est listé dans sa totalité,
- deuxième cas : seule la ligne *départ* du fichier courant est listée,
- troisième cas : le fichier courant est listé de la ligne *départ* à la ligne *fin*,
- quatrième cas : le fichier spécifié est listé dans sa totalité,
- cinquième cas : seule la ligne *départ* du fichier spécifié est listée,
- sixième et dernier cas : le fichier spécifié est listé de la ligne *début* à la ligne *fin*.

En plus de ces six possibilités, vous pouvez ajouter deux options extrêmement intéressantes.

L'indentation

La première de ces options est l'indentation. Si vous complétez une des formes précédentes par :

DBLIST ... INDENT *largeur*

Votre programme sera automatiquement listé en indentant les structures. La *largeur* correspond au nombre d'espaces ajoutés à chaque nouvelle structure rencontrée.

Par défaut, la *largeur* d'indentation est nulle, ce qui veut dire que le programme n'est pas indenté.

La redirection

Si maintenant vous faites suivre une des syntaxes précédentes par :

DBLIST ... TO *fichier*

Le listage n'ira pas à l'écran, mais dans le fichier spécifié. C'est ce qui nous permet, par exemple, d'imprimer le journal avec des beaux listings structurés.

Vous pouvez naturellement cumuler toutes ces options et ces syntaxes. Par exemple, vous pouvez taper des lignes telles que :

```
DBLIST 100,150 INDENT 4 TO EXEMPLE
```

ou encore :

```
PBLIST ESSAI,1000,9000 TO X INDENT 2
```

REGLES D'INDENTATION

Pour bien utiliser DBLIST et PBLIST, il faut connaître les règles d'indentation. Rassurez-vous, elles sont naturelles et pas trop nombreuses...

Pas de GOTO !

Vous pouvez utiliser des GOTO dans vos programmes, mais l'indentation ne peut plus être garantie... Bah ! De toute manière, vous n'utilisez plus cette instruction depuis longtemps, n'est-ce pas ?

Indentations des boucles

Toutes les boucles (WHILE, LOOP, REPEAT, mais aussi FOR...NEXT) sont indentées en décalant le corps de la boucle. Par exemple :

```
100 X=1
110 WHILE X<5
120     DISP X
130     X=X+1
140 END WHILE
150 DISP "Fini"
160 END
```

Les instructions de contrôle de la boucle ne sont pas décalées par rapport au reste du programme.

Indentations des structures de choix

La structure IF multi-lignes est indentée comme suit :

```
100 INPUT X
120 IF X<0 THEN
130     DISP "X négatif"
140     BEEP 1400,.075
150 ELSE
160     R=SQRT(X)
170 END IF
180 DISP "Racine =";R
190 END
```

Les structures de choix multi-lignes (SELECT...CASE) sont indentées en décalant les CASE et les clauses :

```
100 K$=KEYWAIT$
110 SELECT K$
120     CASE "A" TO "Z"
130         DISP "Lettre"
140     CASE "#46"
150         DISP "Touche [RUN]"
160     CASE ELSE
170         DISP "Autre"
180 END SELECT
190 BEEP
200 END
```

Indentation des DEF FN

Seules les DEF FN multi-lignes sont indentées. C'est exactement la même indentation que pour les boucles.

Traitement des DATA et des labels

Les blocs de DATA et les labels sont précédés d'une ligne blanche. Les blocs de DATA sont également suivis d'une ligne blanche.

Traitement des SUB

Les sous-programmes SUB sont précédés d'un trait horizontal pour bien marquer la séparation avec le reste du programme. En plus, leur décalage est remis à 0 avant de lister le corps du sous-programme qui, lui, est décalé.

CONCLUSION

STRUC2 et BASICLEX constituent un outil fondamental pour aborder les problèmes informatiques : la programmation structurée a prouvé son efficacité. Non, elle n'est pas réservée aux seuls professeurs, et bien souvent, elle est le moyen de ramener la complexité des problèmes à un niveau abordable.

Voilà. Nous sommes sûrs que vous saurez en faire bon usage.

A bientôt donc,

Pierre David (37)
Janick Taillandier (246)

```
       LEX    'BASICLEX'

=AVMEMS EQU    #2F594
=BASCHA EQU    #07741
=BSERR  EQU    #0939A
=CK"ON" EQU    #076AD
=CKINFO EQU    #18542
=COMCK  EQU    #036CD
=CPL#10 EQU    #07887
=CRETF+ EQU    #084C4
=CURIND EQU    #2F8CA
=CURREN EQU    #2F56C
=CURRL  EQU    #2F7E8
=CURRST EQU    #2F55D
=D0=PCA EQU    #09B37
=DISPt  EQU    #00000
=EOLCK  EQU    #02A7E
=EOLXC* EQU    #052EC
=EOLXCK EQU    #05405
=EXPEXC EQU    #0F186
=EXPRDC EQU    #05922
=FILDC* EQU    #05759
=FINDA  EQU    #023E3
=FINDF+ EQU    #09F63
=FINDL  EQU    #0FFE4
=FINDL0 EQU    #0FFFD
=FSPECe EQU    #02F02
=FSPECp EQU    #03CC5
=FSPECx EQU    #09F2D
=FUNCR0 EQU    #2F89B
=GETPR1 EQU    #06BFB
=GETPR0 EQU    #06BEE
=GETSTC EQU    #07726
=IVAERR EQU    #0E920
=IVPARe EQU    #02E3F
=InhEOL EQU    #00004
=LDCM10 EQU    #04F6F
=LIN#DC EQU    #05115
=LISTDC EQU    #05839
=MEMCKL EQU    #012A5
=MFERR  EQU    #09393
=MLFFLG EQU    #2F870
=MOVED2 EQU    #1B104
=MOVEUA EQU    #1B168
=MPY    EQU    #0ECBB
=NTOKEN EQU    #0493B
=NTOKNL EQU    #048E6
=NUMCK  EQU    #0369D
=NXTLIN EQU    #10031
=NXTSTM EQU    #08A48
=OBCOLL EQU    #01435
=OUT1TK EQU    #02CEB
=OUT3TK EQU    #02D15
=OUTBS  EQU    #2F58F
=OUTBYT EQU    #02CE8
=OUTNBC EQU    #05423
=PCADDR EQU    #2F679
=PRINTt EQU    #00001
=PRSC00 EQU    #07B93
=RENSUB EQU    #1A753
=RESPTR EQU    #03172
=RNDAHX EQU    #136CB
=RPLLIN EQU    #013F7
=SENDEL EQU    #17DC1
=SNDWD+ EQU    #17E1F
=STMTR0 EQU    #2F871
=STMTR1 EQU    #2F881
=STUFF  EQU    #1B0B2
=SWPBYT EQU    #17A24
=SYNTXe EQU    #02E2B
=TRFMBF EQU    #2F8C5
=eFEXST EQU    #0003B
=eSTMNF EQU    #0001E
=fBASIC EQU    #0E214
=lEOL   EQU    #00002
=oBSsod EQU    #00011
=oFLENh EQU    #00020
=oFLSTr EQU    #00031
=oFTYPh EQU    #00010
=t!     EQU    #000FC
=t@     EQU    #000F4
=tCOMMA EQU    #000F1
=tDATA  EQU    #000C6
=tDEF   EQU    #000B9
=tELSE  EQU    #000F5
=tEND   EQU    #000DA
=tENDDF EQU    #000BA
=tENDSB EQU    #000C2
=tEOL   EQU    #000F0
=tFOR   EQU    #000C3
=tIF    EQU    #000DF
=tLBLST EQU    #000F6
=tLINE# EQU    #0000F
=tNEXT  EQU    #000C4
=tREM   EQU    #000E6
=tSUB   EQU    #000C1
=tTO    EQU    #000F3
=tXWORD EQU    #000EF


=id     EQU    #E1
 t      EQU    90        Token de base


=tINDENT EQU   200


       CON(2) =id
       CON(2) t          Lowest token
       CON(2) 2+t        Highest token
       REL(5) nxtlex     pour INDENT et INTO

       NIBHEX F          No speed table
       REL(4) 1+TxTbSt
       CON(4) 0          No message
       CON(5) 0          No poll handler
```

```
*************************************************
* Definitions empruntees a STRUC2
*************************************************

* Quartets de reconnaissance
=qENDL  EQU   0         END LOOP
=qENDS  EQU   1         END SELECT
=qENDI  EQU   2         END IF

* Tokens de STRUC2
=tEND2   EQU  66
=tWHILE  EQU  67
=tREPEAT EQU  68
=tUNTIL  EQU  69
=tLEAVE  EQU  70
=tLOOP   EQU  96
=tSELECT EQU  97
=tCASE   EQU  98
=tIF2    EQU  99
=tELSE2  EQU  100

* Main table
        CON(3) (TxEn01)-(TxTbSt)
        REL(5) =DBLSTe
        NIBHEX D        Statement
        CON(3) (TxEn02)-(TxTbSt)
        REL(5) =PBLSTe
        NIBHEX D        Statement
        CON(3) (TxEn03)-(TxTbSt)
        REL(5) =RENUMe
        NIBHEX D        Statement
* Text Table
 TxTbSt
 TxEn01 CON(1) 11
        NIBASC 'DBLIST'
        CON(2) t
 TxEn02 CON(1) 11
        NIBASC 'PBLIST'
        CON(2) 1+t
 TxEn03 CON(1) 15
        NIBASC 'RENUMREM'
        CON(2) 2+t
        NIBHEX 1FF      End of text table

 nxtlex CON(2) =id
        CON(2) =tINDENT
        CON(2) =tINDENT
        CON(5) 0
        NIBHEX F        No speed table
        REL(4) 1+TxTbS2
        CON(4) 0        No message
        CON(5) 0        No poll handler
* Main table # 2
        CON(3) (TxE201)-(TxTbS2)
        CON(5) 0
        NIBHEX 0        Mot non accessible
* Text Table # 2
 TxTbS2
 TxE201 CON(1) 11
        NIBASC 'INDENT'
        CON(2) =tINDENT
        NIBHEX 1FF


*************************************************
* ROUTINES DE PARSE
*************************************************

 PRSETO EQU    (=FUNCR0)+0
 PRSEIN EQU    (=FUNCR0)+1

*************************************************
* DBLSTp, PBLSTp
*
* But : parse quelque chose de tres complique...
* Historique :
*   86/05/..: JPB     reecriture d'apres I.D.S.
*   88/01/10: PD & JT documentation
*   88/01/31: PD & JT correction de PBLIST TOTO
*************************************************

=DBLSTp
=PBLSTp
* Mettre a 0 les flags PRSETO et PRSEIN
* qui interdisent de mettre deux fois la meme
* option.
        A=0    B        Le 88/01/31 : gagne quelques
        CDOEX           quartets
        D0=(5) PRSETO
        DAT0=A B        Pas encore vu de TO/INDENT
        D0=C
* [<file>] [,<li#> [,<li#>]] [TO <file>][INDENT <n>]
* Autrefois inspire de LISTP (#03B92 / HP71B)
        GOSUB  eolckx
        GOC    resptr   tEOL trouve
        GOSUB  CKINDT
        GOC    LSTP60   tINDENT trouve
        GOSUB  resptr
*   <file>    ou bien encore    TO
* ^                             ^
        GOSBVL =FSPECp  specificateur valide ?
        GONC   LSTP12   oui
* si S7=1
*   alors reserved word in A
*         (TO, ALL, KEYS, INTO, CARD)
*   sinon bad file parse
* fin si
        ?ST=0  7        Bad file parse ?
        GOYES  LSTP20   Oui : peut-etre <line#>
* reserved word in A. Est-ce tTO ?
        D0=D0- 2        Revient sur tTO
        LC(2)  =tTO
        ?A=C   B
        GOYES  LSTP40

 fspece GOVLNG =FSPECe  Illegal file spec
```

```
* <file>   , ...
*        ^
 LSTP12 GOSBVL =COMCK   Comma ?
        GONC   LSTP30   Non : TO ou INDENT ou vraie fin

* <file> ,   <line> ...
*          ^

 LSTP20 GOSUB  ck2li#   <line1> [ , <line2> ]
* ...   TO <file>
*     ^
 LSTP29 GOSBVL =NTOKEN  Pour avancer
 LSTP30 GOSUB  CKTO
        GOC    LSTP40
        GOSUB  CKINDT
        GOC    LSTP60
 resptr GOVLNG =RESPTR

* Traitement de TO <file>
 LSTP40 LC(5)  PRSETO
        GOSUB  CKSEEN
        GOSBVL =OUT1TK
        GOSBVL =FSPECp  specificateur valide ?
        GONC   LSTP29   Oui : on continue
        GOC    fspece   Non : erreur (B.E.T.)

* Traitement de INDENT <expression>
 LSTP60 LC(5)  PRSEIN
        GOSUB  CKSEEN
        GOSBVL =OUT3TK
        GOSBVL =NUMCK
        GOTO   LSTP30

*****************************************************
* eolckx
*
* But : idem EOLCK, mais avec NTOKEN au lieu de
*   WRDSCN.
* Entree :
*   - D1 = ^ flot ascii
* Sortie :
*   - Cy = 1 si tEOL trouve
* Appelle : NTOKEN, FINDA
* Niveaux : 3
* Abime : A, B, C, P, D0, D1, R0, S0-3, S11
* Historique :
*   88/01/31: PD & JT recodage apres suppression
*****************************************************

 eolckx GOSBVL =NTOKEN
        GOSBVL =FINDA
        CON(2) =tEOL
        REL(3) rtnsc
        CON(2) =t@
        REL(3) rtnsc
        CON(2) =t!
        REL(3) rtnsc
        CON(2) =tELSE
        REL(3) rtnsc
        NIBHEX 00
        RTNCC           Cy = 0 : non trouve
 rtnsc  RTNSC           Cy = 1 : trouve

*****************************************************
* RENUMp
*
* But : parse RENUMREM [<l1>[,<l2>[,<l3>[,<l4>]]]]
* Historique :
*   86/05/..: JPB     reecriture d'apres I.D.S.
*   88/01/10: PD & JT documentation
*****************************************************

=RENUMp GOSUB  eolck+
        GOC    resptR   EOL : sortir
        GOSUB  ck2li#
        ?ST=0  9        2 <line#> trouves ?
        GOYES  rtncc    Non : sortir
        GOSBVL =COMCK   Comma ?
        GONC   resptR   Non : sortir
        GOSUB  ck2li#
 rtncc  RTNCC           Ok

 eolck+ GOSBVL =EOLCK   fin de commande ?
        RTNC            oui : Cy=1
 resptR GOVLNG =RESPTR  non : on revient

*****************************************************
* CKSEEN
*
* But : verifier que le token TO ou INDENT n'a pas
*   deja ete vu durant la parse.
* Entree :
*   - C(A) = adresse du flag (PRSETO ou PRSEIN)
* Sortie :
*   - si deja vu, alors erreur
*     sinon flag mis a jour, Cy = 0
* Niveaux : 0
* Utilise : C(S)
* Historique :
*   88/01/10: PD & JT conception & codage
*****************************************************

 CKSEEN CD0EX
        C=DAT0 S
        ?C#0   S        Deja vu ?
        GOYES  CKS10    oui : erreur
        C=C+1  S
        DAT0=C S
        CD0EX
        RTN
 CKS10  GOVLNG =SYNTXe  "Syntax Error"

*****************************************************
* CKTO, CKINDT
*
* But : tester tTO ou tINDENT
```

```
* Entree :
*   - A = token a tester
* Sortie :
*   Cy = 1 : token trouve
*   Cy = 0 : token cherche non trouve
* Niveaux : 0
* Utilise : C
* Historique :
*   88/01/10: PD & JT conception & codage
*****************************************************

 CKTO   LC(2)  =tTO
        ?A=C   B
        RTNYES
        RTN

 CKINDT LC(6)  (=tINDENT)~(=id)~(=tXWORD)
        P=     5
        ?A=C   WP
        GOYES  CKIN10
 CKIN10 P=     0
        RTN

*****************************************************
* ck2li#
*
* But : parse   "  <line#1> [ , <line#2> ]  "
* Entree :
*   - D1 = ^ blancs optionnels avant
* Sortie :
*   - S9 = 1 si les deux <line#> ont ete reconnus
* Appelle : ck1li#, COMCK, NTOKNL, OUT3TK
* Niveaux : 4 (NTOKNL)
* Utilise : A-C, P, D0, D1, R0, S0-S3, S11
* Historique :
*   86/05/..: JPB     reecriture d'apres I.D.S.
*   88/01/10: PD & JT separation & documentation
*****************************************************

 ck2li# ST=0   9
        GOSUB  ck1li#   <line1>
        GOSBVL =COMCK   Comma ?
        GONC   resptR   Non : sortir
        ST=1   9        On reconnait les 2 <line#>

 ck1li# GOSBVL =NTOKNL  Idem NTOKEN, autorise line#
        LC(2)  =tLINE#
        ?A#C   B
        GOYES  LSTPE    Pas <line#> : erreur
        LC(2)  =tCOMMA
        A=C    B
        GOVLNG =OUT3TK

 LSTPE  GOVLNG =IVPARe  INVALID (MISSING PARM)

*****************************************************
* ROUTINES DE DECOMPILE
*****************************************************
*****************************************************
* DBLSTd, PBLSTd
*
* But : decompiler DBLIST, PBLIST, RENUMREM
* Historique :
*   86/05/.. : JPB     conception & codage
*   88/01/10 : PD & JT nouvelle syntaxe
*****************************************************

=DBLSTd
=PBLSTd ST=0   8        Premiere fois : pas de ','
* Boucle de decompilation des elements
* A l'entree, D1 pointe sur le token a decompiler.
* Les alternatives sont :
*   <file spec>
*   [ , <line #> ]    (S8=1 si affichage ',')
*   TO <file spec>
*   INDENT <exp num>
 LSTD00 GOSBVL =EOLXC*  No return if end of statem.
        GOSBVL =FINDA
        CON(2) =tCOMMA  <line#>         (LSTDC+)
        REL(3) LSTD10
        CON(2) =tTO     TO <file>
        REL(3) LSTD20
        CON(2) =tXWORD  INDENT <exp num>
        REL(3) LSTD30
        NIBHEX 00
* <file spec>
        GOSBVL =FILDC*
 LSTD05 ST=1   8
        GOTO   LSTD00

* [ , ] <line#>
 LSTD10 ?ST=0  8        Besoin de sortir ',' ?
        GOYES  LSTD12   Non
        LCASC  ','      Affichage de ','
        GOSBVL =OUTBYT
 LSTD12 D1=D1+ 2        Passer tCOMMA
        GOSBVL =LIN#DC  Envoyer <line#>
        GOTO   LSTD05

* TO <file spec>
 LSTD20 GOSUB  OUTSPC
        LCASC  ' OT'
        P=     3*2-1
        GOSBVL =OUTNBC
        D1=D1+ 2
        GOSBVL =FILDC*
        GOTO   LSTD05

* INDENT <exp num>
 LSTD30 GOSUB  OUTSPC
        LCASC  ' TNEDNI'
        P=     7*2-1
        GOSBVL =OUTNBC
        D1=D1+ 6
        GOSBVL =EXPRDC
        GOTO   LSTD05
```

```
*************************************************
* OUTSPC
*
* But : afficher un espace si TO ou INDENT ne
*   suivent pas directement D/PBLIST.
* Entree :
*   - S8 = 0 si directemenbt derriere D/PBLIST
* Sortie :
*   - un blanc dans le flot ASCII
* Appelle : OUTBYT
* Utilise : A(B), C(B), D0
* Niveaux : 1
* Historique :
*   88/01/10: PD & JT conception & codage
*************************************************


 OUTSPC ?ST=0  8
        RTNYES
        LCASC  ' '
        GOVLNG =OUTBYT


*************************************************
* RENUMd
*
* But : decompiler RENUMREM
* Historique :
*   86/05/.. : JPB     conception & codage
*************************************************

=RENUMd GOVLNG =LISTDC

*************************************************
* ROUTINES D'EXECUTION
*************************************************

 INDVAL EQU    00+=TRFMBF  5 q : valeur indentation
 CURIND EQU    05+=TRFMBF  5 q : indentation cour.
 OUTYPE EQU    10+=TRFMBF  1 q : 0: fichier, 1: std
 OUTADR EQU    11+=TRFMBF  5 q : adresse ds fichier
 OUTHDR EQU    16+=TRFMBF  5 q : header du fichier

 LDEB   EQU    21+=TRFMBF  5 q : ligne de debut
 LFIN   EQU    26+=TRFMBF  5 q : ligne de fin

 INFILE EQU    31+=TRFMBF  5 q : adresse ds fichier
 INEND  EQU    36+=TRFMBF  5 q : fin du fichier

 INDARM EQU    41+=TRFMBF  1 q : 0:std, 1:DATA, 2:!

 INDCOU EQU    42+=TRFMBF  2 q : valeur courante
 INDLIG EQU    44+=TRFMBF  2 q : en debut de ligne

 SAUTAV EQU    46+=TRFMBF  1 q : sauter ligne avant
 SAUTAP EQU    47+=TRFMBF  1 q : sauter ligne apres

 TMP    EQU    48+=TRFMBF  au moins 2 q.

 inDATA EQU    1        pour INDARM
 inREM  EQU    2        pour INDARM


 clRIEN EQU    0
 clNDDF EQU    1
 clSUB  EQU    2
 clDEF1 EQU    3
 clDEF2 EQU    4
 clDATA EQU    5
 clLBL  EQU    6
 clREM  EQU    7
 clIF   EQU    8

*************************************************
* PBLIST / DBLIST
*
* But : produire un listing structure sur
*   l'imprimante ou dans un fichier.
* Detail :
*   initialiser les variables globales
*   initialiser les options par defaut
*
*   tant que non fin de ligne tokenisee
*     faire
*       selon token
*         tCOMMA :
*           LDEB, LFIN := numeros de ligne
*           tester la validite
*         tTO :
*           evaluer le specificateur de fichier
*           chercher le fichier
*           si il existe
*             alors "File Exists"
*           fin si
*           stocker les parametres pour creation
*         tXWORD :
*           INDVAL := evaluer l'expression
*         autre :
*           evaluer le fichier
*           chercher le fichier
*           si il n'existe pas
*             alors "File Not Found"
*           fin si
*           verifier type et protections
*           INFILE := ^ fichier
*       fin selon
*   fin tantque
*   creer le fichier de sortie si necessaire
*   pour toutes les lignes du fichier
*     decompiler et lister
* Historique :
*   86/05/.. : JPB     conception & codage
*   88/01/10 : PD & JT ajout indentation & fichier
*************************************************


        REL(5) =PBLSTd
        REL(5) =PBLSTp
```

NOUS EN AVONS

La coopérative du Club vous propose :

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

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

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

VOUS EN VOULEZ

Nom :
Prénom :
No de membre :
Adresse :

Commande :

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

Préciser éventuellement les
numéros de JPC commandés :

PPC PARIS

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

BULLETIN D'ADHESION

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

Profession ____________________
Intérêts ____________________

Matériel HP en votre possession ____________________

Autre matériel informatique ____________________

Comment avez-vous connu PPC Paris ? ____________________

Que recherchez-vous au sein de PPC Paris ? ____________________

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

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

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

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

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

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

AH ! VOUS ECRIVEZ

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

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

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

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

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

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

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

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

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

Jean-Jacques Dhénin (177)

```
DEF KEY 'fW', CHR$(197);      (é)
DEF KEY 'fE', CHR$(193);      (ê)
DEF KEY 'fR', CHR$(201);      (è)
DEF KEY 'fY', CHR$(203);      (ù)
DEF KEY 'fU', CHR$(195);      (û)
DEF KEY 'fI', CHR$(209);      (î)
DEF KEY 'fO', CHR$(194);      (ô)
DEF KEY 'f/', CHR$(92);       (\)
DEF KEY 'fA', CHR$(192);      (â)
DEF KEY 'fS', CHR$(200);      (à)
DEF KEY 'fD', CHR$(205);      (ë)
DEF KEY 'fJ', CHR$(207);      (ü)
DEF KEY 'fK', CHR$(221);      (ï)
DEF KEY 'f*', CHR$(124);      (|)
DEF KEY 'fC', CHR$(181);      (ç)
```

PPC PARIS SE REUNIT
UNE FOIS PAR MOIS

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

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

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

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

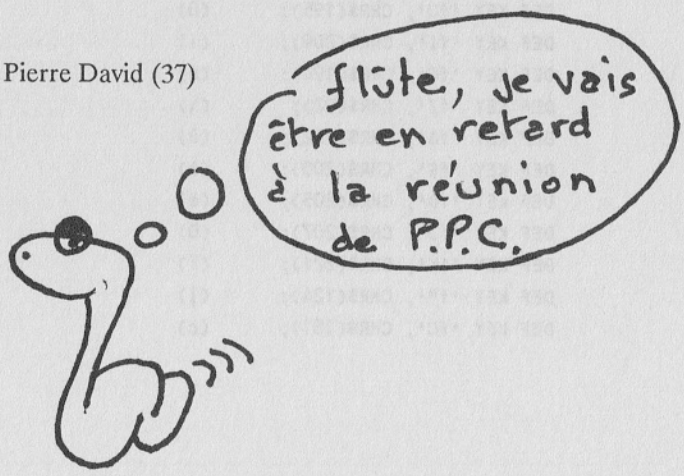
et en montant au deuxième étage, vous entendrez des éclats de rire et des discussions passionnées vers la salle 215. Attention, toutefois, de venir entre 16 et 19h.

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

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

Les dates des prochaines réunions sont :
Samedi 16 avril 1988
Samedi 7 mai 1988
Samedi 4 juin 1988

Pierre David (37)



```
=PBLSTe LC(2)  (=PRINTt)*16+#F
        GOTO   LIST05

 bserr  GOVLNG =BSERR

        REL(5) =DBLSTd
        REL(5) =DBLSTp
=DBLSTe LC(2)  (=DISPt)*16+#F
* Initialiser la sortie standard
 LIST05 D1=(5) =MLFFLG  Destination du xBLIST
        DAT1=C B
* Initialisation des variables globales
*   INDVAL = 0 (pas d'indentation)          5 q
*   CURIND = 0 (indentation courante = 0)   5 q
*   OUTYPE = 0 (sortie std et pas fichier)  1 q
        D1=(5) INDVAL
        C=0    W
        DAT1=C 11
* LDEB = ligne de debut
* LFIN = ligne de fin
        D1=(2) LDEB
        C=C+1  A
        DAT1=C A        LDEB := 0001 par defaut
        D1=D1+ (LFIN)-(LDEB)
        LCHEX  9999
        DAT1=C A        LFIN := 0999 par defaut

        D1=(2) INDARM   INDARM, INDCOU, INDLIG,
        C=0    W        ... SAUTAV et SAUTAP
        DAT1=C 7        ... := 0
* Chercher l'adresse du fichier courant
        D1=(4) =CURRST
        C=DAT1 A
        D1=(4) INFILE
        DAT1=C A        INFILE := ^ debut fichier

 loop   A=DAT0 B        A(B) := token courant
        GOSBVL =EOLXCK
        GONC   loop10
        GOTO   endlop   Sortie si EOL trouve

 loop10 GOSBVL =FINDA
        CON(2) =tCOMMA  <line#>
        REL(3) LIST10
        CON(2) =tXWORD  INDENT <exp num>
        REL(3) LIST20
        CON(2) =tTO     TO <filespec>
        REL(3) LIST30
        NIBHEX 00
* <filespec>
        GOSBVL =FSPECx  specificateur valide ?
        GOC    bserR    non
        GOSBVL =FINDF+  trouve le fichier
        GOC    bserR    non trouve
        CD1EX           C(A) := ^ file start
        D1=(5) INFILE
        DAT1=C A        INFILE := ^ file start
        GOTO   loop

 bserR  GOTO   bserr
* <line#> [ , <line#> ]
 LIST10 D1=(5) LDEB
        GOSUB  getli#
        DAT1=A A        <line# 1>
        D1=D1+ (LFIN)-(LDEB)
        DAT1=A A        <line# 2>
        GOSUB  getli#
        GOC    loop     pas de <line# 2>
        DAT1=A A
        D1=D1- (LFIN)-(LDEB)
        C=DAT1 A        C(A) := <line# 1>
        ?C<=A  A        1 <= 2
        GOYES  loop     oui : ok
 invarg GOVLNG =IVAERR  non (ah bon ?)

* INDENT <exp num>
 LIST20 D0=D0+ 6        sauter tXWORD id tINDENT
        GOSBVL =EXPEXC
        GOSBVL =RNDAHX
        GONC   invarg   negatif
        D1=(5) INDVAL
        DAT1=A A
        GOTO   loop

* TO <filespec>
 LIST30 D0=D0+ 2
        GOSBVL =FSPECx  specificateur valide ?
        GOC    Bserr    non
* Cy = 0 : mainframe recognisable file specifier found
* A(W) = file name
* D(S) = #F  if device not specified
* D(B) = details sur le port
        D1=(5) =STMTR0  sauvegardes
        C=D    W        Sauvegarde de D(W) (pour CRETF+)
        DAT1=C W
        D1=D1+ 16
        DAT1=A W        Sauvegarde du nom du fichier

        GOSBVL =FINDF+  Find file
* A(W) = B(W) = file name
* Cy = 0 : file found
* Cy = 1 : file not found
        GOC    LIST32   fichier non trouve : ok
        LC(4)  =eFEXST  "File Exists" / Beeeeeep !
 Bserr  GOTO   bserr

 LIST32 D1=(5) OUTYPE
        LC(1)  1
        DAT1=C P        Sortie := fichier
        GOTO   loop

* Verification du type et des protections du
* fichier a lister.
 endlop D0=(5) INFILE
        C=DAT0 A
        D1=C
        D1=D1+ (=oFTYPh)-1
```

```
        A=DAT1 A
        ASR    A        A(A) := file type
        D1=D1- (=oFTYPh)-1
        GOSBVL =BASCHA  BASIC ?
        GOC    bsErr    Non : "Invalid File Type"
        GOSBVL =GETPR1  PRIVATE ?
        GOC    bsErr    Oui : "File Protect"
* D1 = ^ file type

* Preparer les pointeurs
        D1=D1+ (=oFLENh)-(=oFTYPh)
        CD1EX
        D1=C
        A=DAT1 A        A(A) := REL(5) FiLeNd
        C=C+A  A
        D0=(2) INEND
        DAT0=C A
        D=C    A        D(A) := ^ fin fichier
        D0=(2) LDEB
        A=DAT0 A        A(A) := <line1 #>
        B=A    A
        D1=D1+ (=oBSsod)-(=lEOL)
        CD1EX
        D1=C
        D1=D1+ =lEOL    D0 = ^ line # (1ere ligne)
        A=0    A        La doc ne l'indique pas
        GOSBVL =FINDL0  Supporte !
        GOC    lst10
        ?ST=0  0        Il existe un numero de ligne
        GOYES  lst10    plus grand que <line1 #>
        GOTO   nxtstm

 bsErr  GOTO   bserr

 lst10  D0=(5) INFILE
        CD1EX
        DAT0=C A        INFILE := ^ line# courant
* Creation du fichier a partir des elements stockes
* dans STMTR0/1 pendant "TO <filespec>" si
* necessaire.
        D0=(2) OUTYPE
        A=DAT0 S
        ?A=0   S
        GOYES  nocret
        D1=(5) =STMTR0
        C=DAT1 W
        D=C    W        Restaurer pour CRETF+
        C=0    A
        LC(2)  37+4     header + EOF-mark
        GOSBVL =CRETF+  Le Lex ne doit pas bouger
        GOC    bsErr    Probleme a la creation
        C=R1            C(A) := adresse du header
        D1=(5) OUTHDR   Header du fichier de sortie
        DAT1=C A
        D1=(5) =STMTR1
        A=DAT1 W
        D1=C
        DAT1=A W        Nom du fichier
        D1=D1+ 16       D1 = ^ file type
        LCHEX  400001   text + copy code
        DAT1=C 6
        D1=D1+ 16
        D1=D1+ 5
* EOF-Mark
        C=0    A
        C=C-1  A
        DAT1=C 4        D1 := ^ fin du fichier


        CD1EX
        D1=(5) OUTADR
        DAT1=C A
 nocret

*************************************************
* lst100
*
* But : debut de la boucle principale de D/PBLIST
*   c'est a dire tests de sortie
* Entree : -
* Sortie :
*   si sortie necessaire
*     alors sortie par NXTSTM
*     sinon D1 = ^ <line #>
*   fin si
* Detail :
*   test 1 : EOF ?
*   test 2 : no ligne > no derniere ligne a lister
*   test 3 : [ATTN] ?
* Appelle : CK"ON", NXTSTM
* Historique :
*   88/01/10 : PD & JT conception & codage
*************************************************

 lst100
* 1 : atteint EOF ?
        D0=(5) INFILE
        A=DAT0 A        A(A) := ^ line#
        D0=(2) INEND
        C=DAT0 A        C(A) := ^ FiLeNd
        ?A>=C  A
        GOYES  nxtstm
* 2 : atteint ligne > ligne fin ?
        D1=A            D1 := ^ line#
        A=0    A
        A=DAT1 4        A(A) := line#
        D0=(2) LFIN
        C=DAT0 A
        ?A>C   A
        GOYES  nxtstm
* 3 : [ATTN] ?
        CD1EX
        GOSBVL =CK"ON"
        D1=C
        GOC    lst110   Ok on a passe tous les tests
 nxtstm GOVLNG =NXTSTM
```

```
*************************************************
* lst110
*
* But : analyse du premier statement et
*   decompilation de la ligne.
* Entree :
*   - D1 = ^ <line #>
* Sortie :
*   - (OUTBS..AVMEMS) = ligne decompile
*     et indentation faite
* Appelle : gettok, LDCM10
*************************************************



* Ok, on peut y aller
* D1 = ^ line#
 lst110 D1=D1+ 2        D1 := ^ Stlen - 2
        GOSUB  gettok



        B=A    B       B(B) := classe reconnue
* Analyse du premier statement pour voir s'il faut :
*   - indenter avant
*   - sauter une ligne avant
*   - marquer un SUB

* Indentation avant ?
        C=R1           C(A) := indentation avant
        ?C=0   A
        GOYES  lst120
* il faut indenter
        D0=(5) INDCOU  Indentation courante
        A=DAT0 B
        A=A-C  B
        GONC   lst118  Ok, indentation valide
        A=0    B       Indentation := 0
 lst118 DAT0=A B       Nouvelle indentation
* Cas autre que DATA ?
*
* Si token = DATA
*   alors
*     on est a l'interieur, on ne fait rien
*   sinon
*     si on etait avant a l'interieur de DATA
*       alors
*         on n'est plus dans des DATA (INDARM := 0)
*         sauter une ligne avant (SAUTAV := 1)
*       sinon
*         cas le plus frequent, on ne fait rien
*     fin si
* fin si
 lst120 A=B    A        A(B) := classe du token
        LC(2)  clDATA
        ?A=C   B
        GOYES  lst122  cas "DATA"

        D0=(5) INDARM
        A=DAT0 1       A(0) = 0, inDATA ou inREM
        LC(1)  inDATA  On avait DATA avant ?
        ?A#C   P
        GOYES  lst122  Non : on ne fait rien (ah..)

        C=0    A       A(0) := 0 (no inDATA)
        DAT0=C 1       INDARM := 0

        D0=(2) SAUTAV
        C=C+1  A       C(0) := 1
        DAT0=C 1

* Si (INDARM = inREM) && (classe # clREM)
*   alors
*     INDARM := 0
* fin si
 lst122 D0=(2) INDARM
        A=DAT0 P
        LC(1)  inREM
        ?A#C   P
        GOYES  lst125  INDARM # inREM
        LC(2)  clREM
        ?C=B   B
        GOYES  lst125  classe = clREM
        C=0    A
        DAT0=C P       INDARM := 0

*************************************************
* Y-a-t'il des cas particuliers a traiter ?
*************************************************

 lst125 A=B    A        A(B) := classe du token
        GOSBVL =FINDA
        CON(2) clSUB   SUB
        REL(3) l1SUB
        CON(2) clDEF1  DEF mono-ligne
        REL(3) l1DEF1
        CON(2) clDEF2  DEF multi-ligne
        REL(3) l1DEF2
        CON(2) clDATA  DATA
        REL(3) l1DATA
        CON(2) clLBL   <label>
        REL(3) l1LBL
        NIBHEX 00
        GOTO   lst130  Pas de traitement special

 l1SUB  D1=(5) INDCOU  Indentation := 0
        C=0    B
        DAT1=C B
* sauter une ligne
        GOSUB  sautln
* une ligne de '-'
        D1=(5) =OUTBS  Debut de la ligne
        C=DAT1 A
        D1=C           D1 := ^ debut ligne

        LCASC  '--------'
        A=C    W       A(W) := pattern
        C=0    A
```

```
        LC(2)  8*8*2    64 '-'
        GOSBVL =STUFF   Remplissage de 64 '-'
        CD1EX           C(A) := adresse de fin
        D1=(5) =AVMEMS
        DAT1=C A        AVMEMS := fin
        GOSUB  print
        GOTO   l1skip   et terminer en sautant

 l1DEF1 A=DAT1 B        A(B) := tEOL ?
        LC(2)  =tEOL
        ?A#C   B
        GOYES  lst130   DEF + autre chose
        D1=(5) SAUTAP   Sauter apres
        LC(1)  1
        DAT1=C 1        Sauter une ligne apres
        GONC   l1skip   B.E.T.

 l1DATA D1=(5) INDARM   DATA ou REM ?
        A=DAT1 1
        LC(1)  inDATA
        DAT1=C 1        INDARM := inDATA
        ?A#C   P
        GOYES  l1skip   Premier DATA
        GOTO   lst130   pas de saut

 l1DEF2
 l1LBL
 l1skip D1=(5) SAUTAV
        LC(1)  1        1 est different de 0
        DAT1=C P        Sauter une ligne
        GOTO   lst130

* Sauter une ligne ?
 lst130 D0=(5) SAUTAV   Sauter avant ?
        A=DAT0 S
        ?A=0   S        Sauter la ligne ?
        GOYES  lst140   Non
        GOSUB  sautln   Sauter une ligne

* Figer l'indentation courante
 lst140 D0=(5) INDCOU   Indentation courante
        A=0    W
        A=DAT0 B

        D0=(2) INDVAL   Valeur de l'indentation
        C=0    W
        C=DAT0 A

        GOSBVL =MPY     A,B,C := Ind. en octets

        C=C+C  A        Indentation en quartets
        D0=(2) TMP
        DAT0=C A        TMP := deplacement en q.
* On voudrait C(A) quartets en plus. On peut ?
        GOSBVL =MEMCKL  Memory Check with Leeway
        GONC   lst145   Oui : on continue
        GOTO   bserr    Non : Insufficient Memory
* B(A) := deplacement en quartets

 lst145 D0=(4) =OUTBS
        C=DAT0 A        C(A) := OUTBS original
        RSTK=C          RSTK := OUTBS original

        D1=C            D1 := start of area
        LCASC  '        '
        A=C    W        A(W) := pattern a stuffer
        C=B    A        C(A) := longueur en quartets
        GOSBVL =STUFF
        CD1EX           C(A) := pseudo OUTBS
* Laisser de la place pour le no de ligne
        D0=C            D0 := ^ pseudo OUTBS
        LCASC  '   '
        DAT0=C 6

* Remettre D1 sur la ligne (<line #>)
        D1=(5) INFILE
        C=DAT1 A
        D1=C            D1 := ^ <line #>
        A=0    A
        A=DAT1 4        A(A) := no de ligne
        LCHEX  01000
        ?A>=C  A
        GOYES  lst160
        CSR    A        C(A) := 100
        ?A>=C  A
        GOYES  lst151   1 blanc
        CSR    A        C(A) := 10
        ?A>=C  A
        GOYES  lst152   2 blancs
        D0=D0+ 2
 lst152 D0=D0+ 2
 lst151 D0=D0+ 2
 lst160 CD0EX
        D0=(5) =OUTBS
        DAT0=C A
* D1 = ^ line#
* OUTBS decale pour LDCM10
        GOSBVL =LDCM10  Comme pour LIST
* R0 = ^ past tEOL
* B(A) = longueur en octets
* OUTBS collapsed
        C=RSTK          C(A) := OUTBS original
* Decaler le numero de ligne et le remettre
* avant l'indentation
        D=C    A        D(A) := OUTBS original
        D1=(5) TMP      juste avant no ligne
        A=DAT1 A
        A=A+C  A        A(A) := ^ no ligne
        D0=A            D0 := ^ no ligne
        D1=C            D1 := ^ OUTBS original
        A=DAT0 8        Lire no de ligne
        LCASC  '    '
        DAT0=C 8
        DAT1=A 8        Nouveau no ligne
        C=D    A        C(A) := OUTBS original
* Placer AVMEMS a la bonne valeur
        D1=(5) =OUTBS   OUTBS modifie
```

```
        A=DAT1 A
        DAT1=C A        OUTBS original
        A=A+B  A
        A=A+B  A
        D1=D1+ 5        =AVMEMS
        DAT1=A A        nouveau AVMEMS
* Traitement du premier token a nouveau
* - !
* - indentation "apres"
        D0=(5) INFILE
        A=DAT0 A
        D1=A            D1 := ^ <line #>
        D1=D1+ 2
        GOSUB  gettok   A nouveau le 1er token
        B=A    B        B(A) := classe du token
* Cas particulier du IF ?
        LC(2)  clIF
        ?A#C   B
        GOYES  lst165   Non : on continue
        GOTO   lst700   Ignorer le reste de la ligne

 lst165 C=R2            C(A) := indentation "apres"
        ?C=0   A
        GOYES  lst170   Pas d'indentation

        D0=(2) INDCOU
        A=DAT0 B        A(B) := indentation courante
        A=A+C  B
        DAT0=A B
* "!" ?
 lst170 LC(2)  clREM
        ?B#C   B
        GOYES  lst190   Rentrer dans la boucle
* Sauver D1 pour etre remis apres
        CD1EX
        RSTK=C          RSTK := D1
* D1 := ^ no ligne dans la chaine decompilee
        D1=(5) =OUTBS
        C=DAT1 A
        D1=C            D1 := ^ no ligne
* Mettre d'office 4 blancs
        LCASC  '    '
        DAT1=C 4*2
* Mettre un '-' ?
        D1=D1+ 3*2      D1 := ^ ('-' eventuel)
        D0=(2) INDARM
        A=DAT0 P        A(0) := 0, inDATA ou inREM
        LC(1)  inREM
        DAT0=C P        INDARM := inREM
        ?A=C   P        Deja une REM ?
        GOYES  lst180   Oui : ne rien faire
        LCASC  '-'      Non : ajouter un '-'
        DAT1=C B
 lst180 LCASC  '!'
 lst182 A=DAT1 B
        ?A=C   B
        GOYES  lst185
        D1=D1+ 2
        GONC   lst182   B.E.T.
 lst185 D1=D1+ 2+2      D1 := ^ debut du commentaire
        AD1EX           A(A) := start of source
        D1=A
        D1=D1- 2+2      D1 := start of dest
        GOSBVL =MOVEUA  AVMEMS := end of source

        D0=(5) =AVMEMS
        CD1EX           C(A) := end of dest
        DAT0=C A
        C=RSTK          restaurer D1
        D1=C

 lst190

************************************************
* Debut de la boucle "dans la ligne"
*
* But : explorer les statements entre le deuxieme et
*   le dernier de la ligne courante.
* Entree :
*   - D1 = ^ Stlen - 2 du deuxieme Statement
* Sortie :
*   - D1 = ^ tEOL de la ligne courante
* Historique :
*   88/01/24 : PD & JT conception & codage
************************************************

* Ce lst530 est une ruse (voir en fin de boucle)
 lst530
 lst500 GOSUB  gettok
        B=A    B        B(A) := classe du token
        GOC    lst900

        LC(2)  clIF     IF standard ?
        ?A=C   B
        GOYES  lst700   Ignorer le reste de la ligne
* Calculer la nouvelle indentation
        D0=(5) INDCOU
        C=DAT0 B        C(A) := indentation courante

        A=R1            indentation avant (<= 0)
        C=C-A  B
        GONC   lst510   Pas < 0
        C=0    B
 lst510 A=R2            indentation apres (>= 0)
        C=C+A  B
        DAT0=C B
* Cas particuliers ?
        LC(2)  clSUB
        ?B#C   B
        GOYES  lst520
* SUB
        DAT0=A B        indentation := R2
        GONC   lst530   B.E.T.
 lst520 LC(2)  clNDDF
        ?B#C   B
        GOYES  lst530
```

```
* END DEF
        A=DAT1 B        A(B) = tEOL ?
        LC(2)  =tEOL
        ?A#C   B
        GOYES  lst530
        LC(1)  1
        D0=(2) SAUTAP
        DAT0=C 1        Sauter apres := vrai

* Le lst530 qui suit a ete deplace en debut de la
* boucle pour eviter le GOYES -> GOTO intempestif.
* lst530
        GOTO   lst500

 lst700 D1=(5) INFILE
        A=DAT1 A        A(A) := ^ <line #>
        D1=A
        GOSBVL =NXTLIN  D1 := ^ past tEOL
        GONC   lst910   B.E.T.
* Attention : le code continue en sequence !!!

*************************************************
* Fin de la boucle "dans la ligne"
*
* But : recuperer le pointeur dans le flot tokenise
*   pour la ligne suivante, et iterer
* Entree :
*   - D1 = ^ tEOL de la ligne courante
* Sortie :
*   - INFILE = ^ <line #> de la ligne suivante
* Historique :
*   88/01/24 : PD & JT conception & codage
*************************************************

 lst900 D1=D1+ 2        D1 := ^ <line #> ligne suiv.
 lst910 CD1EX           C(A) := ^ <line #>
        D1=(5) INFILE
        DAT1=C A        Nouvelle ligne
* Prepare la ligne suivante
* Reporter le "saut apres" sur "saut avant" et
* effacer "saut apres"
        D1=(5) SAUTAP
        A=DAT1 S
        C=0    S
        DAT1=C S
        D1=(2) SAUTAV
        DAT1=A S
        GOSUB  print
        GOTO   lst100

*************************************************
* SOUS-PROGRAMMES
*************************************************

*************************************************
* sautln
*
* But : imprimer une ligne vide
* Entree : -
* Sortie : -
* Abime : A-D, R3
* Appelle : OBCOLL, print (tombe dedans)
* Niveaux :
* Historique :
*   88/01/24 : PD & JT conception & codage
*************************************************

 sautln GOSBVL =OBCOLL
* Attention : le code continue !!!

*************************************************
* print
*
* But : imprimer la ligne comprise entre OUTBS et
*   AVMEMS sur le DISPLAY, le PRINTER ou dans un
*   fichier selon la valeur de OUTYPE.
* Entree :
*   - (OUTBS..AVMEMS) = la ligne
* Sortie : -
* Abime : A-D, R3
* Appelle : CKINFO, SNDWD+, SENDEL, SWPBYT, RPLLIN,
*   MOVED2
* Niveaux :
* Detail : l'execution de sautln continue en direct
* Historique :
*   88/01/24 : PD & JT isolement dans un sub
*************************************************

 print  D0=(5) OUTYPE
        A=DAT0 S        0 : std, 1 : fichier
        ?A#0   S
        GOYES  prnt50
* Impression sur DISPLAY ou PRINTER
        GOSBVL =CKINFO  Prepare HPIL pour l'envoi
        D0=(4) =AVMEMS
        A=DAT0 A
        D0=D0- 5        OUTBS
        C=DAT0 A        C(A) := ^ chaine
        A=A-C  A        A(A) := longueur en quartets
        B=0    W
        B=A    A
        BSRB            B(A) := longueur en octets
        ST=1   =InhEOL  Inhibit EOL pour le cas ou
        GOSBVL =SNDWD+  Envoi proprement dit
        GOVLNG =SENDEL  EOL

* Impression dans un fichier
 prnt50 D0=(4) =AVMEMS
* Insertion de la longueur LIF
        A=DAT0 A        A(A) := end of source
        D1=A
        D1=D1+ 4        D1 := end of dest
        D0=D0- 5        D0=(5) OUTBS
        C=DAT0 A        C(A) := start of source
        GOSBVL =MOVED2
* Calculer le padding LIF
```

```
        D0=(5) =AVMEMS
        A=0    W
        A=DAT0 A
        D0=D0- 5        OUTBS
        C=DAT0 A
        A=A-C  A        Longueur en quartets
        ASRB            Longueur en octets
        B=A    A        Sauvegarde
        GOSBVL =SWPBYT
        C=DAT0 A        C(A) := (OUTBS)
        D1=C            D1 := ^ LIF length
        DAT1=A 4
        SB=0
        BSRB
        A=0    A        Longueur a ajouter
        ?SB=0
        GOYES  prnt60   Longueur paire
        A=A+1  A        1 octet a ajouter
 prnt60 C=0    A
        LC(1)  4
        A=A+A  A
        A=A+C  A        Longueur totale a ajouter
        D0=D0+ 5        AVMEMS
        C=DAT0 A
        C=C+A  A
        DAT0=C A        AVMEMS + 4 ou 6
* Insertion dans le fichier
        C=0    A
        R3=C            R3 := longueur vieille ligne
        D0=(5) OUTHDR
        C=DAT0 A        C(A) := ^ file header
        D0=(2) OUTADR
        A=DAT0 A        A(A) := ^ end of old line
        GOSBVL =RPLLIN  Ne peut pas bouger
        GOC    bSerr
* A(A) = end+1 of replaced line in file
        D0=(5) OUTADR
        DAT0=A A
        RTN
 bSerr  GOTO   bserr

************************************************
* getli#
*
* But : evaluer un numero de ligne optionnel
* Entree :
*   - D0 = ^ tCOMMA precedant <line #>
* Sortie :
*   Cy = 1 : pas de <line #>
*   Cy = 0 :
*     - A(A) = numero de ligne en BCD
*     - D0 reactualise
* Niveaux : 0
* Utilise : C(B), A(A)
* Historique :
*   86/05/.. : JPB     conception & codage
*   88/01/10 : PD & JT modifications & documentation
************************************************

getli# A=DAT0 B
       LC(2)  =tCOMMA
       ?A#C   B
       RTNYES
       D0=D0+ 2
       A=0    A
       A=DAT0 4
       D0=D0+ 4
       RTN             Cy = 0 a cause de D0=D0+ 4

************************************************
* gettok
*
* But : classer le token pointe par D1
* Entree :
*   - D1 = ^ Stlen - 2 du token a classer
* Sortie :
*   Cy = 1 : EOL trouvee
*   Cy = 0 : EOL non trouvee
*     - D1 = ^ Statement terminator
*     - A(B) = classe du token
*   R1 = indentation avant
*   R2 = indentation apres
* Niveaux : 1
* Utilise :
* Appelle : FINDA
* Historique :
*   88/01/23 : PD & JT & JJD conception & codage
************************************************

 gettok C=0    W
        R1=C
        R2=C
        A=DAT1 B
        LC(2)  =tEOL
        ?A=C   B
        RTNYES
        D1=D1+ 2        D1 := ^ Stlen
        C=0    A
        C=DAT1 B        C(A) := Stlen
        AD1EX
        C=C+A  A        C(A) := ^ Stlen - 2 suivant
        D1=A            D1 := ^ Stlen du courant
        RSTK=C          RSTK := ^ tEOL ou t@ ou ...
        D1=D1+ 2        D1 = ^ token courant
        A=DAT1 B

        GOSBVL =FINDA
        CON(2) =tXWORD  XWORD
        REL(3) gtXWRD
        CON(2) =tFOR    FOR
        REL(3) gtFOR
        CON(2) =tNEXT   NEXT
        REL(3) gtNEXT
        CON(2) =tSUB    SUB
        REL(3) gtSUB
        CON(2) =tDEF    DEF FN
        REL(3) gtDEF
```

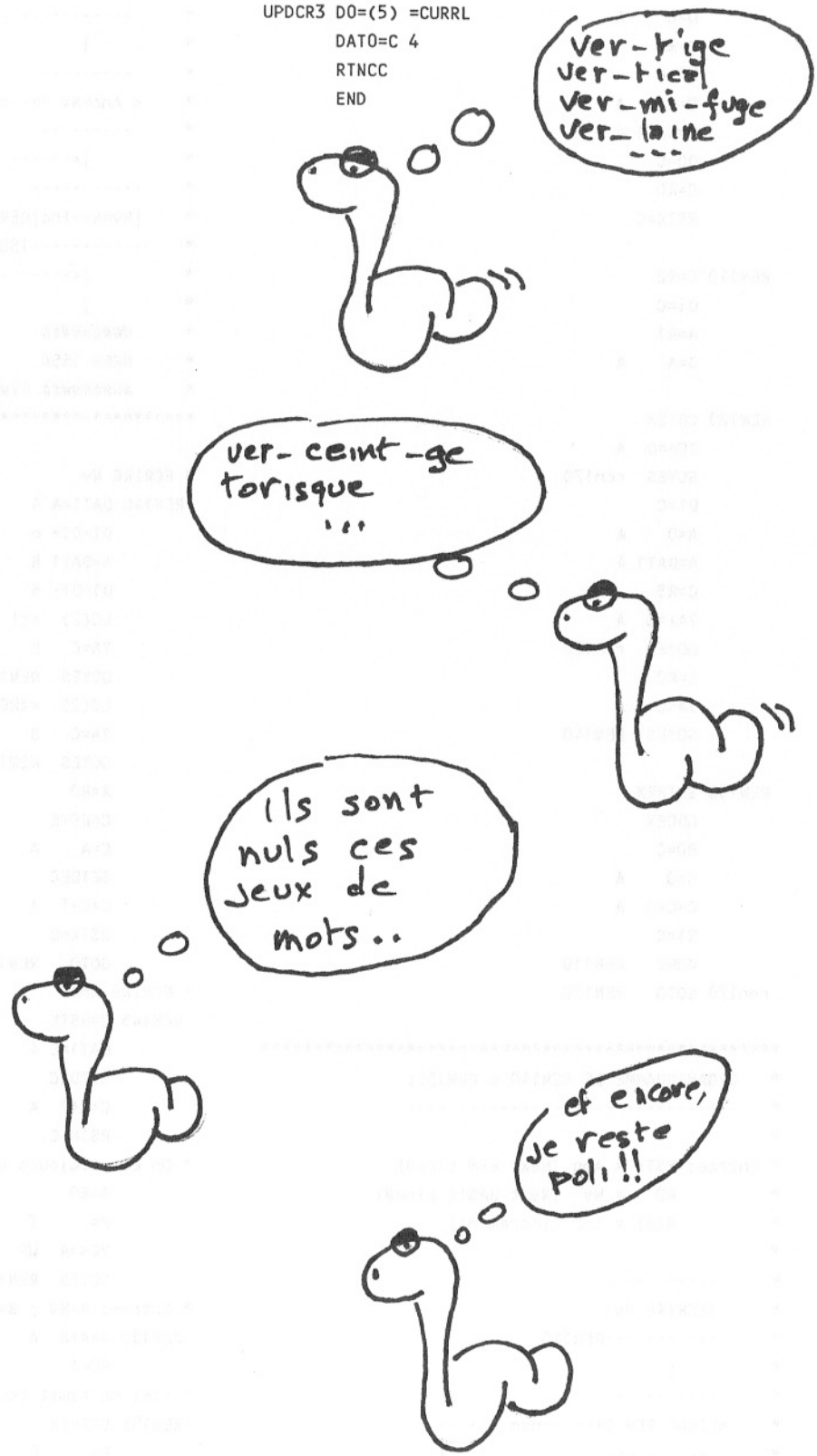
```
        CON(2) =tDATA   DATA
        REL(3) gtDATA
        CON(2) =tLBLST  '...':
        REL(3) gtLBL
        CON(2) =tEND    END
        REL(3) gtND
        CON(2) =tENDDF  END DEF
        REL(3) gtNDDF
        CON(2) =tENDSB  END SUB
        REL(3) gtNDSB
        CON(2) =t!      !
        REL(3) gtREM
        CON(2) =tIF     IF mono-ligne (standard)
        REL(3) gtIF
        NIBHEX 00


gtrien GOTO   gtRIEN


gtXWRD D1=D1+ 2
       A=DAT1 B
       LC(2)  =id
       ?A#C   B
       GOYES  gtrien


       D1=D1+ 2        D1 := ^ toooooken
       A=DAT1 B        A(B) := token

       GOSBVL =FINDA
       CON(2) =tWHILE  WHILE
       REL(3) gtWHIL
       CON(2) =tLOOP   LOOP
       REL(3) gtLOOP
       CON(2) =tREPEAT REPEAT
       REL(3) gtREPT
       CON(2) =tUNTIL  UNTIL
       REL(3) gtUNTL
       CON(2) =tIF2    IF jpcrom
       REL(3) gtIF2
       CON(2) =tELSE2  ELSE jpcrom
       REL(3) gtELS2
       CON(2) =tSELECT SELECT
       REL(3) gtSEL
       CON(2) =tCASE   CASE
       REL(3) gtCASE
       CON(2) =tEND2   END IF/LOOP/SELECT/WHILE
       REL(3) gtENDj
       NIBHEX 00
       GOTO   gtRIEN

************************************************
* indentation avant :  0
* indentation apres : +1
*
* FOR, WHILE, LOOP, REPEAT, IF
************************************************

 gtFOR
 gtIF2
 gtLOOP
 gtREPT
 gtWHIL C=0    A
        C=C+1  A
        R2=C           indentation apres := 1
        GOTO   gtRIEN  Classe : normale

************************************************
* indentation avant : -1
* indentation apres : +1
*
* ELSE, CASE
************************************************

 gtCASE
 gtELS2 C=0    A
        C=C+1  A
        R1=C           indentation avant := 1
        R2=C           indentation apres := 1
        GOTO   gtRIEN  Classe : normale

************************************************
* indentation avant :  0
* indentation apres : +2
*
* SELECT
************************************************

 gtSEL  C=0    A
        LC(1)  2
        R2=C           indentation apres := 2
        GOTO   gtRIEN  Classe : normale

 gtENDj D1=D1+ 2        D1 = ^ q de reconnaissance
        A=DAT1 B        A(B) = token suivant
        GOSBVL =EOLXCK
        GOC    gtND     END WHILE
* A(0) = quartet de reconnaissance
        LC(1)  =qENDS
        ?A=C   P
        GOYES  gtNDSL   END SELECT
* END LOOP / IF
* Attention : le code continue !!!

************************************************
* indentation avant : -1
* indentation apres :  0
*
* UNTIL, NEXT, END LOOP, END WHILE,
* END IF, END, END SUB
************************************************

 gtND
 gtNDSB
 gtNEXT
 gtUNTL C=0    A
```

```
        C=C+1  A
        R1=C            indentation avant := 1
        GOTO   gtRIEN   Classe : normale

*****************************************************
* indentation avant : -2
* indentation apres :  0
*
* END SELECT
*****************************************************

 gtNDSL C=0    A
        LC(1)  2
        R1=C            indentation avant := 2
        GOTO   gtRIEN   Classe : normale

*****************************************************
* indentation avant : -1
* indentation apres :  0
*
* Cas particulier

* END DEF
*****************************************************

 gtNDDF C=0    A
        C=C+1  A
        R1=C            indentation avant := 1
        LC(2)  clNDDF   Classe END DEF
        A=C    B
        GOTO   gtok99

*****************************************************
* indentation avant :  0 (en fait : cas particulier)
* indentation apres : +1
*
* Cas particulier
*
* SUB
*****************************************************

 gtSUB  C=0    A
        C=C+1  A
        R2=C            indentation apres := 1
        LC(2)  clSUB    Classe SUB
        A=C    B
        GOTO   gtok99

 gtDEF  D1=D1+ 7
        A=DAT1 B        A(B) := 0 si multi-ligne
        ?A=0   B
        GOYES  gtDEF2   Multiligne
* Attention : le code continue
* DEF FN mono-ligne

*****************************************************
* indentation avant :  0
* indentation apres :  0
*
* Cas particulier
*
* DEF mono-ligne
*****************************************************

 gtDEF1 LC(2)  clDEF1   DEF mono-ligne
        A=C    B
        GOTO   gtok99

*****************************************************
* indentation avant :  0
* indentation apres :  1
*
* Cas particulier
*
* DEF multi-ligne
*****************************************************

 gtDEF2 C=0    A
        C=C+1  A
        R2=C            Indentation apres := 1
        LC(2)  clDEF2   DEF multi-ligne
        A=C    B
        GOTO   gtok99

*****************************************************
* indentation avant :  0
* indentation apres :  0
*
* Cas particulier
*
* DATA
*****************************************************

 gtDATA LC(2)  clDATA   DATA
        A=C    B
        GOTO   gtok99

*****************************************************
* indentation avant :  0
* indentation apres :  0
*
* Cas particulier
*
* '...':
*****************************************************

 gtLBL  LC(2)  clLBL    label
        A=C    B
        GOTO   gtok99

*****************************************************
* indentation avant :  0
* indentation apres :  0
*
* Cas particulier
*
```

```
* !
************************************************

 gtREM  LC(2)  clREM    !
        A=C    B
        GOTO   gtok99


************************************************
* indentation avant :  0
* indentation apres :  0
*
* Cas particulier
*
* IF mono-ligne (standard)
************************************************

 gtIF   LC(2)  clIF     IF standard
        A=C    B
        GOTO   gtok99

 gtRIEN A=0    B
 gtok99 C=RSTK
        D1=C
        RTNCC

************************************************
* RENUMREM
*
* But : renumeroter un programme Basic en tenant
*   compte des remarques.
* Note : RENUMREM 100 , 10 , 200 , 500
*                 -v-   -v-  -v-   -v-
* registres ->     R0   R1   @R2    R3
* Historique :
*   86/05/.. : JPB     conception & codage
************************************************

        REL(5) =RENUMd
        REL(5) =RENUMp
=RENUMe CD0EX
        R0=C
        GOSUB  CHKPSF
        GOSUB  GETSTe
        GOSBVL =PRSC00

 REN005 ST=1   1
        ST=1   2
        GOSBVL =RENSUB
        GOC    REN010
        D1=(5) =PCADDR
        C=R2
        DAT1=C A
        GOSUB  UPDCRL
        LC(2)  =eSTMNF  "Statement Not Found"
        GOTO   mferr

 REN010 A=R0
        D0=A
        GOSUB  LINE#1
        R2=A
        C=0    A
        C=C+1  A
        CSL    A
        R0=C
        R1=C
        CSL    A
        CSL    A
        CSL    A
        R3=C
        GOSUB  GETLN#
        GOC    REN100
        R0=A
        GOSUB  GETLN#
        GOC    REN100
        R1=A
        GOSUB  GETLN#
        GOC    REN100
        C=A    A
        AD0EX
        R2=A
        GOSBVL =FINDL
        ?ST=0  0
        GOYES  REN020
        GOTO   nxtstm

 REN020 CD1EX
        CR2EX
        D0=C
        GOSUB  GETLN#
        GOC    REN100
        SETDEC
        P=     3
        A=A+1  WP
        P=     0
        SETHEX
        GOC    REN100
        C=A    A
        GOSBVL =FINDL
        A=0    A
        A=DAT1 4
        ?ST=1  0
        GOYES  REN100
        R3=A

 REN100 GOSUB  LINE#1
        CD1EX
        A=R2
        ?A=C   A
        GOYES  REN105
        D0=A
        D0=D0- 4
        GOSBVL =CPL#10
        C=0    A
        C=DAT1 4
```

```
        A=R0
        ?A>C   A
        GOYES  REN105
        GOLONG invarg

REN105  D1=(5) =CURREN
        C=DAT1 A
        D=C    A
        C=R2
        D1=C
        C=0    A
        C=DAT1 4
        D0=C
        C=R0
        RSTK=C

REN110  C=R2
        D1=C
        A=R1
        B=A    A

REN120  CD1EX
        ?C>=D  A
        GOYES  ren170
        D1=C
        A=0    A
        A=DAT1 4
        C=R3
        ?A>=C  A
        GOYES  ren170
        A=R0
        ?A<C   A
        GOYES  REN140

REN130  SETHEX
        CD0EX
        R0=C
        C=0    A
        C=C+1  A
        R1=C
        GONC   REN110
ren170  GOTO   REN170

************************************************
*   ORGANIGRAMME DE REN140 a REN155:
*   --------------------------------
*
* Entree: RSTK = Anc [Next REM Line#]
*         R0   = Nv  [Next BASIC Line#]
*         B[B] = Inc [Increment]
*
*     -----------
*     |ECRIRE Nv|
*     -----------REN140
*         |
*     -----------
*    <LIGNE REM ?>-----non-------
*     -----------                |
*         |oui                   |
*     ------------               |
*     |ECRIRE Anc|               |
*     ------------REN145         |
*         |                      |
*     -----------          ----------
*     |Anc=Anc+1|          |Anc=Nv+1|
*     -----------          ----------
*         |                      |
*     ---------                  |
*    < Anc>NV ?>--non--          |
*     ---------       |          |
*         |<---------------------
*    -----------      |
*    |Nv=Nv+Inc|REN   |
*    -----------150   |
*         |<-----------
*         |
*     #########
*     #REN 155#
*     ######### FIN
************************************************

* ECRIRE Nv
 REN140 DAT1=A 4        LINE# = Nv
        D1=D1+ 6
        A=DAT1 B        A[B] = token
        D1=D1- 6
        LC(2)  =t!
        ?A=C   B        Ligne REM ?
        GOYES  REN145   oui
        LC(2)  =tREM
        ?A=C   B        Ligne REM ?
        GOYES  REN145   oui
        A=R0            non: Anc = Nv+1
        C=RSTK
        C=A    A
        SETDEC
        C=C+1  A
        RSTK=C
        GOTO   REN150
* ECRIRE Anc
 REN145 C=RSTK          C=Anc
        DAT1=C 4        Line# = Anc
        SETDEC
        C=C+1  A        Anc = Anc+1
        RSTK=C          Sauvegarde Anc
* On est toujours en DEC
        A=R0            A=Nv
        P=     3
        ?C<=A  WP       Anc <= Nv ?
        GOYES  REN155   oui alors FIN
* Entree: A=Nv ; B=Inc ; DEC
 REN150 A=A+B  A        Nv = Nv+Inc
        R0=A            Sauve Nv
* FIN: on remet tout en ordre pour REN160
 REN155 SETHEX
        P=     0
```

```
        D1=D1+ 4        D1 @ tLEN
        C=0    A

REN160  C=DAT1 B
        AD1EX
        A=A+C  A
        D1=A
        A=DAT1 B
        D1=D1+ 2
        ?A#0   P
        GOYES  REN160
        GOTO   REN120

REN170  C=RSTK
        ST=1   1
        ST=0   2
        GOSBVL =RENSUB
        GOLONG nxtstm

GETLN#  A=DAT0 B
        D0=D0+ 2
        LC(2)  =tCOMMA
        ?A#C   B
        RTNYES
        A=0    A
        A=DAT0 4
        D0=D0+ 4
        RTNCC

LINE#1  D1=(5) =CURREN
        C=DAT1 A
        D=C    A
        D1=D1- 15
        A=DAT1 A
        C=0    A
        LC(2)  =oFLSTr
        A=A+C  A
        D1=A
        RTN

CHKPSF  GOSUB  GETSTe
        GOSBVL =GETPRO
        GOC    mferr
        ?SB=0
        RTNYES
        GONC   mferr

GETSTe  GOSBVL =GETSTC
        RTNNC
mferr   GOVLNG =MFERR

UPDCRL  D1=(5) =CURRST
        C=DAT1 A
        D1=C
        D1=D1+ (=oFTYPh)-1
        A=DAT1 A
        ASR    A
        LC(5)  =fBASIC
        ?A#C   A
        GOYES  UPDCR1
        GOSBVL =D0=PCA
        GOSBVL =CPL#10
        C=DAT1 A
        GONC   UPDCR3
UPDCR1  C=0    A
UPDCR3  D0=(5) =CURRL
        DAT0=C 4
        RTNCC
        END
```



DERIVEES SYMBOLIQUES (ACTE I QUINTE)

L'intérêt suscité par le programme de Peter Ehrenberg et Volker Klann (JPC 44) est explicable parce qu'il représente une nouvelle façon de traiter des programmes écrits en Basic, la nouveauté consistant dans la reconnaissance d'une chaîne de caractères. La discussion qui suit suppose que vous avez tenu compte des observations faites par Dominique Marcaillou (JPC 47) et Thierry Besançon (JPC 49). L'observation de Bruno Gil (JPC 50) doit être considérée comme une variante de la ligne 1590 pour le cas ou U$ est différent de X, sinon on introduit une multiplication inutile avec l'unité.

La maladresse de la table des correspondances fonctions / dérivées est déterminée par le manque d'une instruction d'arrêt marquant qu'une certaine expression ne se trouve pas dans la table. Pour corriger cela, il faut introduire la ligne 1045 (voir programme) et, à ce moment, le bug mentionné par Thierry Besançon est éliminé. La présentation qui suit permettra aux intéressés de passer aux actes suivants.

Le trait essentiel du programme est que l'analyse d'une expression se fait caractère par caractère. On distingue une série de sous programmes et de fonctions qui ont le but suivant :

- SUCHEFKT contient la table des correspondances fonctions / dérivées. Il teste d'abord la présence dans la chaîne F$ d'une parenthèse ouvrante dans une position qui est différente de la première ou de la dernière de la chaîne F$. Puis on sépare la partie fonction V$ de la partie argument U$ et, comme généralement U$ peut être une fonction de X, on cherche d'abord la dérivée respective U1$ par un appel récursif (ligne 840). Si la fonction V$ est trouvée dans la table, on a sa dérivée F1$.

- A fait le produit de F1$ et U1$ en testant les 2 cas singuliers U1$=0 et U1$=1 (fonction U$ constante ou réduite à X)

L'indice O1 utilisé par SUCHEFKT permet d'identifier si l'expression de la dérivée doit être inversée ou prise avec le signe moins.
- Pour O1=1, on fait le rapport U1$/F1$.
- Pour O1=2, on fait le rapport avec signe changé.
- Pour O1=3, on fait le produit.
- Pour O1=4, on fait le produit avec signe changé.

- EF rétablit, pour la fonction et sa dérivée, le signe moins en première position (signe qui a été supprimé à la ligne 660)

- ABLUV est le sous programme appelé si on a positionné un opérateur algébrique. P étant le pointeur de l'opérateur algébrique dans la chaîne F$, on introduit une chaîne U$ *postopérateur* et une chaîne V$ *préopérateur*, U1$ et V1$ étant les dérivées respectives. Le sous programme relance l'appel de DIFF pour les deux sous chaînes et détermine donc les dérivées afférentes. En fonction de la valeur du pointeur O, on se branche aux sous programmes ADD, SUB, MUL, DIV ou POT qui associent les fonctions U$ et V$ et les dérivées suivant les règles connues :

$(u+v)'=u'+v'$ (+ ou -)
$(uv)'=u'v+uv'$
$(u/v)'=(u'v-uv')/v^2$
$(u^v)'=vv'u^{v-1}$

- FNK$(T$,O) restitue la chaîne T$ ou met cette chaîne entre parenthèses si le premier caractère de T$ est '-' ou si la chaîne T$ contient des opérateurs algébriques. Le paramètre O est un trieur des opérateurs. Ainsi O=3 permet de retenir uniquement '+' et '-' ; O=5 pour tout opérateur sauf '^' et pour la ligne 1430, il faut choisir O=6 afin qu'une puissance se trouvant au dénominateur soit mise entre parenthèses avant de prendre son carré.

- FNN(T$) est une fonction qui retourne une valeur 0 ou 1. La valeur 1 correspond à une chaîne T$ représentant une constante composée des chiffres. Dans ce cas, le programme débouche sur la ligne 710 qui donne une dérivée nulle. Pour tout autre cas, incluant les cas manifestes d'erreur syntaxique comme par exemple T$='5.97.', la fonction restitue la valeur 0. Il faut aussi remarquer que le premier caractère alpha de la chaîne T$ rend FNN=0 et l'exploration de T$ est terminée.

- MINPRIO analyse la présence des opérateurs algébriques et fait le test des correspondances des parenthèses ouvertes et fermées. La valeur du pointeur O est 1, 2, 3, 4 ou 5 en correspondance respective avec +, -, *, / ou ^. Un autre pointeur P fixe la position dans la chaîne de l'opérateur algébrique. L'analyse de l'expression se fait entièrement, mais un seul opérateur algébrique est retenu. La priorité est accordée à l'opérateur '+' ou '-', qui constitue pour la dérivation un séparateur longitudinal. Les opérateurs '*' ou '/' sont retenus pour la position la plus à droite dans la chaîne et l'opérateur '^' est retenu si la sous chaîne examinée ne contient pas d'autres opérateurs. Il est important de souligner, et ceci constitue une qualité

remarquable du programme, la recherche des opérateurs algébriques ne se fait pas à l'intérieur des parenthèses. Dans le cadre de l'exploitation séquentielle de l'expression donnée (F$=K$=T$), on trie d'abord l'opérateur algébrique d'après les règles précisées plus haut (le pointeur P marque cette position).

Le programme démarre à l'étiquette START et on appelle MINPRIO. A la sortie de ce sous programme, l'indice K=0 montre la compatibilité des parenthèses. A la ligne 630, on teste qu'en première ou dernière position de la chaîne, il ne se trouve pas d'opérateur algébrique. Maintenant il faut distinguer trois cas essentiels :

1) on a trouvé un opérateur algébrique (avec p>1) et dans ce cas on appelle ABLUV.
2) on a trouvé un '+' ou '-' en première position (la ligne 630 a exclu la possibilité de trouver d'autres opérateurs). Dans ce cas, on supprime le premier caractère de la chaîne F$ et on est ramené au cas trois.
3) on n'a pas d'opérateur algébrique et alors on passe au test de la ligne 700 qui appelle la fonction FNN(F$).

Le cas 3 est celui qui débouche sur l'appel du sous programme SUCHEFKT. Deux conditions doivent être remplies : la chaîne F$ ne doit pas contenir d'opérateur algébrique et elle doit contenir une parenthèse ouvrante positionnée (ligne 740). Il faut remarquer que la ligne 820 doit avoir le numéro 745, car la placer à l'intérieur du sous programme SUCHEFKT n'est pas logique.

La force majeure du programme est l'appel récursif de DIFF aux lignes 840, 1160 et 1180. Pour illustrer cet aspect et la façon dont le programme travaille, on va prendre comme exemple la fonction : F$=(5*X+X^2)*(2*X/X^3)

On introduit pour les sous chaînes *pré-* et *postopérateurs* U$ et V$ un indice *i* $(U\$)_i$ pour indiquer l'ordre des opérations séquentielles du programme. Les phases de la résolution sont :

1) Les parenthèses de l'expression étant négligées, MINPRIO choisit l'opérateur '*' (position 9). On appelle ABLUV et on a $(U\$)_1=(5*X+X^2)$. Maintenant intervient le premier appel récursif de DIFF (ligne 1160).
2) A cause des parenthèses, MINPRIO ne fait aucun tri et on passe à la ligne 740 qui fait P=1. Ensuite, à la ligne 760, les parenthèses sont éliminées, on a $(U\$)_1=5*X+X^2$ et on repasse à l'étiquette START.
3) MINPRIO choisit le '+' (position 4), ABLUV détermine $(U\$)_2=5*X$ et un nouveau appel récursif suit.
4) MINIPRIO détermine le '*' (position 2) et, d'une façon semblable à la procédure présentée plus haut, ABLUV donne $(U\$)_3=5$ et l'appel récursif établit cette fois (ligne 710) $(U1\$)_3=0$; après le END de la ligne 730, on revient à la ligne 1170 et on détermine $(V\$)_3=X$. L'appel récursif s'arrête cette fois à la ligne 680, ce qui donne $(V1\$)_3=1$ et après le END de la ligne 690, le retour se fait à la ligne 1190 et, parce que O=3 ('*'), on appelle MUL et on obtient $(U1\$)_2=5$, puis on passe à la ligne 1170.
5) Maintenant, on a $(V\$)_3=X^2$ et un nouveau appel récursif suit, avec passage par ABLUV avec $(U\$)_4=X$ et $(V\$)_4=2$. Comme O=5, on se branche sur POT et on obtient $(V1\$)_2=2*X^1$. En continuation, on revient à la ligne 1190 et, comme O=1, on va à ADD et on obtient $(U1\$)_1=5+2*X^1$. On revient de nouveau à 1170 et cette fois, on a $(V\$)_1=(2*X/X^3)$ et un autre appel récursif suit.
6) On élimine les parenthèses et, conformément aux règles énoncées plus haut, on trie l'opérateur '/' et ABLUV détermine $(U\$)_4=2*X$.
7) Suivant la même procédure, on a $(U\$)_5=2$, $(V\$)_5=X$, $(U\$)_6=X$, $(V\$)_6=3$ et on obtient :
$(V1\$)_1=(2*X^3-2*X^3-2*X*3*X^2)/(X^3)^2$
et on revient pour la dernière fois à la ligne 1190 avec O=3 et les 4 fonctions :
- $(U\$)_1=5*X-X^2$
- $(U1\$)_1=5+2*X^1$.
- $(V\$)_1=2*X/X^3$
- $(V1\$)_1$=voir plus haut

MUL donne le résultat final :

$$F'=(5+2*X^1)*2*X/X^3+(5*X+X^2)*(2*X^3-2*X*3*X^2)/(X^3)^2$$

Si vous avez une fonction trigonométrique, par exemple, SIN(5+A*X)/X, alors:

1) On choisit le '/' et ABLUV donne $(U\$)_1=SIN(5+A*X)$. Après l'appel récursif, on appelle d'abord SUCHEFKT et avant de chercher la correspondance fonctions dérivées, à la ligne 840 comme l'argument $(U\$)_2=5+A*X$ est une fonction de X, l'appel récursif de cette ligne retourne la dérivée $(U1\$)_2=A$. Ensuite, on détermine la fonction V$=SIN et la table (ligne 860) donne F1$=COS(5+A*X).
2) On passe au sous programme A et, à la ligne 1080, on obtient A*COS(5+A*X) ; on revient à la ligne 1170 avec O=4('/') et les quatre valeurs :
- U$=SIN(5+A*X), U1$=A*COS(5+A*X)
- V$=X, V1$=1

3) DIV donne le résultat final :

$$F'=(A*COS(5+A*X)*X-SIN(5+A*X))/X^2$$

Aurel Rottman (289)

Programme "LEXDIR" pour HP-75 (liste des mots-clefs d'un fichier Lex)

```
   - Programme "LEXDIR" (HP-75 avec Rom I/O et MEMLEX)
     Auteur : Jean-Yves Hervé
  10 PRINTER IS ":P1" @ PWIDTH 80
  20 PRINT @ INPUT "Nom du lex ? ";N$
  30 I=DIRECT(N$)+2 @ A=PEEK(I)+PEEK(I+1)*256
  40 I=ADDR(N$) @ L=PEEK(I)+PEEK(I+1)*256
  50 I=PEEK(I+4)+PEEK(I+5)*256+I-1
  60 PRINT N$;TAB(10);A;"octets";TAB(25);"Id:";STR$(L);
  70 N$=""
  80 I=I+1 @ A=PEEK(I) @ IF A=255 THEN GOTO 20
  90 IF A<128 THEN N$=N$&CHR$(A) @ GOTO 80
 100 N$=N$&CHR$(A-128) @ PRINT TAB(40);N$ @ GOTO 70
```

****

Programme "LEXCAT" pour HP-75 (désassemblage de l'en-tête d'un Lex)

```
   - Programme "LEXCAT" (HP-75 avec Rom I/O et MEMLEX)
     Auteur : Jean-Yves Hervé
  10 INTEGER N,J,L,I,K,V(25),Z,X,T,Y
  20 DIM N$[40]
  30 DELAY 0 @ PRINTER IS ":P1" @ PWIDTH 80
  40 INPUT "Nom du LEX ?"; N$ @ Z=ADDR(N$)
  50 PRINT TAB(17);"CATALOGUE: ";N$ @ I=DIRECT(N$) @ PRINT
  60 GOSUB 150 @ N$="Adresse" @ GOSUB 520
  70 GOSUB 150 @ V=V-Z @ N$="Longueur" @ GOSUB 160
  80 V(1)=PEEK(I) @ N$="Acces" @ GOSUB 520
  90 V(1)=PEEK(I) @ N$="Type: "&CHR$(V(1)) @ GOSUB 520
 100 FOR J=1 TO 4 @ V(J)=PEEK(I-1+J) @ NEXT J
 110 K=4 @ N$="Date de creation" @ GOSUB 520
 120 N$="" @ FOR J=1 TO 8 @ V(J)=PEEK(I-1+J)
 130 N$=N$&CHR$(V(J)) @ NEXT J @ K=8 @ GOSUB 520
 140 PRINT TAB(28);"EN-TETE" @ I=Z @ GOTO 170
 150 V(1)=PEEK(I) @ V(2)=PEEK(I+1) @ V=V(1)+V(2)*256+Z @ K=2 @ RETURN
 160 N$=N$&": "&STR$(V) @ GOTO 520
 170 GOSUB 150 @ N$="Lexid" @ GOSUB 520
 180 GOSUB 150 @ N$="Runtab" @ X,V=V+2 @ GOSUB 160
 190 GOSUB 150 @ N$="Asciis" @ T=V @ GOSUB 160
 200 GOSUB 150 @ N$="Partab" @ Y,V=V+2 @ GOSUB 160
 210 GOSUB 150 @ N$="Errmsg" @ GOSUB 160
 220 GOSUB 150 @ N$="Intcpt" @ GOSUB 160
 230 PRINT TAB(28);"RUNTAB"
 240 I=X @ X=(Y-X)/2
 250 FOR J=1 TO X
 260 GOSUB 150 @ N$="#"&STR$(J) @ GOSUB 160 @ NEXT J
 270 PRINT TAB(28);"PARTAB" @ I=Y
 280 FOR J=1 TO X
 290 GOSUB 150 @ N$="#"&STR$(J) @ GOSUB 160 @ NEXT J
 300 GOSUB 150 @ N$="Relmar" @ GOSUB 520
 310 PRINT TAB(28);"ASCIIS" @ I=T @ V=0
 320 K=1 @ V=V+1 @ N$="#"&STR$(V)&": "
 330 V(K)=PEEK(I+K-1) @ IF V(K)=255 THEN N$="Byt" @ GOSUB 520 @ GOTO 360
```

```
340 IF V(K)<128 THEN N$=N$&CHR$(V(K)) @ K=K+1 @ GOTO 330
350 N$=N$&CHR$(V(K)-128) @ GOSUB 520 @ GOTO 320
360 PRINT TAB(28);"ERRMSG" @ K=1 @ V(K)=PEEK(I) @ IF V(K)=255 THEN 390
370 V=V(1)-1 @ N$="Errn" @ GOSUB 520
380 K=1 @ V=V+1 @ N$="#"&STR$(V)&":"
390 V(K)=PEEK(I+K-1) @ IF V(K)=255 THEN N$="Byt" @ GOSUB 520 @ GOTO 411
400 IF V(K)<128 THEN N$=N$&CHR$(V(K)) @ K=K+1 @ GOTO 390
410 N$=N$&CHR$(V(K)-128) @ GOSUB 520 @ GOTO 380
411 PRINT @ K=1 @ V(K)=PEEK(I) @ N$="Version: "&CHR$(MOD(V(K),128)) @ GOSUB 520
412 K=1 @ V(K)=PEEK(I) @ N$="Attribute" @ GOSUB 520 @ END
420 IMAGE 5D,5D,17X,32A
430 IMAGE 5D,5D,4D,13X,32A
440 IMAGE 5D,5D,4D,4D,9X,32A
450 IMAGE 5D,5D,3(4D),5X,32A
460 IMAGE 5D,5D,4(4D),X,32A
470 PRINT USING 420 ; I-N,V(1),N$ @ RETURN
480 PRINT USING 430 ; I-N,V(1),V(2),N$ @ RETURN
490 PRINT USING 440 ; I-N,V(1),V(2),V(3),N$ @ RETURN
500 PRINT USING 450 ; I-N,V(1),V(2),V(3),V(4),N$ @ RETURN
510 PRINT USING 460 ; I-N,V(1),V(2),V(3),V(4),V(5),N$ @ RETURN
520 N=K @ I=I+N @ K=1 @ N$=REPL$(CHR$(12)," ",N$,"",1) @ N$=REPL$(CHR$(10)," ",N$,"",1)
530 IF N<6 THEN 560
540 GOSUB 510 @ N=N-5 @ N$=""
550 FOR J=1 TO N @ V(J)=V(J+5) @ NEXT J @ GOTO 530
560 ON N GOTO 470,480,490,500,510
```

****

Programme "DIFF" (choc en retour, dérivées symboliques)

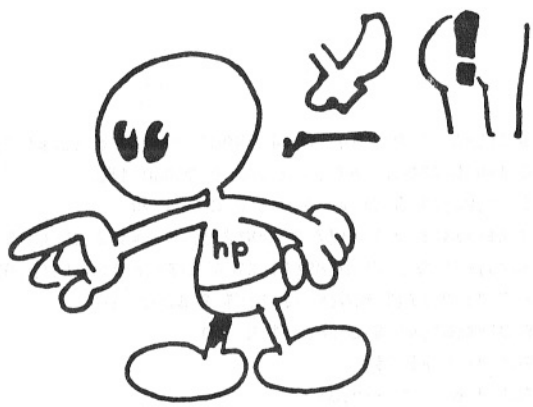
```
 10 DESTROY ALL
 30 DIM K$[91]
 40 INPUT 'f(x)=',F1$;K$
 50 K$=UPRC$(K$) @ L=LEN(K$)
 60 DIM F$[L],F1$[MIN(91,10*L)]
 70 F$=K$ @ CALL DIFF(F$,F1$)
 80 IF FLAG(0) THEN BEEP 4000,.01 @ DISP "f'=";F1$; @ PAUSE @ GOTO 40
 90 BEEP 1400,.05
100 IF FLAG(1) THEN DISP 'Paranthesis Omitted'
110 IF FLAG(2) THEN DISP 'Syntax in Formula'
120 IF FLAG(3) THEN DISP 'Formula too Complex'
130 F1$=K$ @ WAIT 3 @ GOTO 40
```

```
160 SUB MINPRIO(T$,P,O,L)
210     INTEGER I,K
220     DIM K$[1]
230     CFLAG 1 @ L=LEN(T$) @ P=0 @ K=0 @ O=6
240     FOR I=1 TO L
250         K$=T$[I,I]
260         IF K$='(' THEN K=K+1 @ GOTO 340
270         IF K$=')' THEN K=K-1 @ GOTO 340
280         IF K#0 THEN 340
290         IF K$='+' THEN O=1 @ P=I @ GOTO 340
```

```
300         IF K$='-' THEN O=2 @ P=I @ GOTO 340
310         IF K$='*' AND (O>2 OR P=1) THEN O=3 @ P=I @ GOTO 340
320         IF K$='/' AND (O>2 OR P=1) THEN O=4 @ P=I @ GOTO 340
330         IF K$='^' AND (O=6 OR P=1) THEN O=5 @ P=I
340     NEXT I
350     IF K#0 THEN CFLAG 0 @ SFLAG 1
360 END SUB

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

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$,5)&'^2)' @ GOTO 'A'
1045 PRINT 'Unknown function',V$ @ CFLAG 0 @ END

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$,6)&'^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$='0' @ END
1585 F1$=FNK$(V$,3)&'*'&FNK$(U$,5)&'^'
1590 IF U1$#'1' THEN F1$=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
```



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.

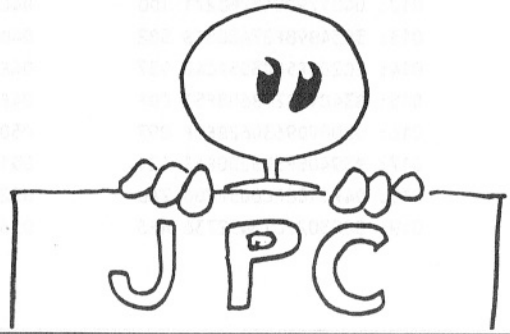
Vous trouverz donc le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !"), ainsi que le Lex de programmation structurée de ce mois-ci.

```
CHARLEX

BASICLEX  DBLIST    XWORD   225090          PBLIST    XWORD   225091
          RENUMREM  XWORD   225092
```

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

```
CHARLEX        624 octets

      0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E000000000000 68D
002: 5E4001EFF0000000 9FD
003: FE0000000800001F D57
004: F31BF961400032BF 0EA
005: 38F14A11DB10AD23 484
006: 07D532BFB8FD7911 837
007: 11AD754D7A101743 BBA
008: 11014D1CB15D0000 F25
009: 71450375FF864834 2A2
00A: 5655581008355654 5F9
00B: 5810070507701724 93F
00C: 7700775070077517 C92
00D: 2077040708364545 FE0
00E: 4A30000A49724000 333
00F: 0808094A2C180814 69C
010: A464242008355455 9F6
011: 581000054C714000 D3C
012: 0C3142404C700832 098
013: 41414A70002078A0 3F0
014: 2F300000000000000 71B
015: 0000000000000000 A2B
016: 0000000000000000 D3B
017: 0000000000000000 04B
018: 0000000000000000 35B
019: 0000000000000000 66B
01A: 0000000000000000 97B
01B: 0000000000000000 C8B
01C: 0000000000000000 F9B
01D: 0000000000000000 2AB
01E: 0000000000000000 5BB
01F: 0000000000000000 8CB
020: 0000000000000000 BDB
021: 000000000000080C F06
022: 1A28080008080A2C 270
023: 180008040E340800 5B9
024: 08001E3018000000 8F3
025: 0000000000000000 C03
026: 0000000000000000 F13
027: 0000000000000000 223
028: 0201000000010200 539
029: 0000000201020000 84E
02A: 0001000100000002 B62
02B: 0102010000000000 E76
02C: 0000000000000000 186
02D: 045E755142400101 4D2
02E: 0101010000000000 7E5
02F: 0000000000000000 AF5
030: 0000070507000000 E18
031: 000000000083444C4 156
032: 44400D7901112D70 4B6
033: 050D750509700000 800
034: 0D70000000384540 B43
035: 4020014E322E3140 E97
036: 084E794142400000 1E7
037: 000000000002E4559 525
038: 3200000000000000 83A
039: 0000000000000026 B52
03A: 5556587008365556 EB1
03B: 5810083645464830 202
03C: 0832414248700024 543
03D: 5655587008345655 8A0
03E: 5810083446454830 BEF
03F: 0C3042414C700024 F44
040: 5556587008355654 2A1
041: 5810083546444830 5F0
042: 0C3142404C700025 946
043: 5455587008355455 CA0
044: 5810083544454830 FEE
045: 0C3140414C700875 350
046: 14141870000A4972 6A1
047: 40000E3159454E30 A01
048: 0C7A0F7949400024 D79
049: 5554587000084A71 0D5
04A: 40000C523A262D10 436
04B: 0424587458400875 78D
04C: 1415187000094A70 ADD
04D: 4000083544454830 E21
04E: 0C3140414C300C74 189
04F: 5655545000054C71 4E0
050: 40000             5D9

BASICLEX       1819 octets

      0123456789ABCDEF sm

000: 2414359434C45485 364
001: 802E000000000000 693
002: A3E001EA5C526000 A18
003: F920000000000000 D49
004: 02D200DF000B200D 0C3
005: E1068B00DB4424C4 454
006: 943545A5B0524C49 7D5
007: 43545B5F2554E455 B5A
008: D42554D4C51FF1E8 F0D
009: C8C00000F7100000 269
00A: 0000000000000008 58B
00B: 94E44454E4458C1F 926
00C: FAE01361BB98F214 CCF
00D: 8134769042572114 01F
00E: 1775408F5CC305A1 3AA
00F: 867F1181313F9623 721
010: 38D20F208FDC6305 ABD
011: D07EE08FB394075C E71
012: 040179C04828D271 1DD
013: 3034B98F27A808FB 582
014: EC208F5CC305AC42 937
015: B34C98F27B608F51 CDF
016: D208FD963062BF8F 097
017: B39408F3E3200F61 421
018: 04F110CFC005F700 7AB
019: 0003027C10452736 AF6
01A: 0869018FDC630521 E74
01B: 7050038FE7A20400 1E4
01C: 8D27130136156494 544
01D: EE0B461544136018 8BB
01E: DB2E20313F962000 C38
01F: 135FE1E8C2591220 FC2
020: 20018497D008FDC6 350
021: 305DA8598F6E8403 6EB
022: 1F096601311FAEA8 A82
023: D51D208DF3E20848 E22
024: 8FCE2508F3E3201F 1D1
025: D103F630FE450008 554
026: F9575085866DF868 8F5
027: D031C28F8EC20171 C8B
028: 8F511506EDF7A403 029
029: 545F402258F32450 396
02A: 1718F957506BBF77 735
02B: 203D94E44454E445 AB7
02C: 022D8F324501758F E36
02D: 22950609F8680031 197
02E: 028D8EC208D93850 52C
02F: D4FFFCCDFF31F168 925
030: 108DA939043FFF3B CD6
031: DFF31F01F078F214 083
032: D1F5C8F2AF215DA1 44B
033: DADE614517433999 7E6
034: 91451DEEAF215D61 B92
035: ED55F1471E4E8F14 F49
036: 514A8F504505606E 2C6
037: D08F3E3201F830FE 672
038: 3603F080008FD2F9 9FF
039: 04D18F36F9043113 D7A
03A: 71F4E8F214565BF6 124
03B: 25F1FAD8F27C5614 4D7
03C: 11741417F4645914 838
03D: 11C41478BE788D02 BC9
03E: 9E01658F681F08FB F76
03F: C63157E1F5C8F214 318
040: 16F5F1618FD2F904 6BF
041: 721F178F2AFB1557 A62
042: 17F15178F36F904C DFB
043: 033B3006ACE1FFC8 1AD
044: F23011550671F1B4 51F
045: E8F214613517E143 89D
046: F41CE8F147704D48 C45
047: FBFB6043417F1371 FE0
048: 35143C2199E144D7 361
049: 19AD142D817E1371 6EC
04A: 35171D08FDFFF04F AAF
04B: 0860A063C0674E1B E33
04C: 4E8F213714419FC1 1C4
04D: 524948961F178F21 543
04E: 577AF7D231928F4C 8EE
04F: 48046C1191F5D8F2 C82
050: 1451F188F2153713 FF0
051: 5151717F35100004 33E
052: 15D517F174D2CE15 6DB
053: D31371F0D8F21451 A63
```

```
054: B4E8F2142199E146 DF8
055: 8BE72131D015B319 17E
056: FD1468B6211378FD 523
057: A6701354908D84A8 8A8
058: 01717CA4AE81198A C42
059: A811BFE8F214AB6A 009
05A: 550AE0148D431509 380
05B: 62521BEE8F215A03 716
05C: 0190621D215C0193 A75
05D: FE615C019EE15203 E0E
05E: 0290611317096180 154
05F: D21540D48F3E3202 4D9
060: 0D1030D60400A050 835
061: 080606900062A01F B92
062: FE8F2AE214D79031 F42
063: FF85F21471353FD2 2E7
064: D2D2D2D2D2D2D2AF 6B8
065: AD231088F2B0B113 A44
066: 71F495F214570D26 DC9
067: B3014B310F966041 139
068: F4F8F230115D05D1 4D1
069: 1FEE8F215B030115 864
06A: D09066065101F3F8 BE1
06B: F2301155063001B3 F37
06C: F8F2152494860776 2B8
06D: 21BFE8F2AF014A19 66F
06E: 5CAF21468FBBCE0C A48
06F: 6195F1448F5A2105 DC8
070: 606FFB1AF85F1460 178
071: 61353F0202020202 4BA
072: 020202AFAD98F2B0 857
073: B113713435020202 B9B
074: 15C51F4E8F214713 F29
075: 5D015B334000108B 28F
076: E91F68BEF0F68BE5 66F
077: 016116116113 61BF 9CA
078: 85F21448FF6F4007 D63
079: D71F5F8F2143CA13 110
07A: 013515A737020202 457
07B: 0215C71597DB1FF8 7FA
07C: 5F2143145C0C0174 B6B
07D: 1411B4E8F2142131 ED9
07E: 1717D42AE8318096 25C
07F: 6606BF011A8AAF01 5FA
080: 9FE14AA6A1483170 994
081: 96577137061FF85F D27
082: 2147135370202020 05E
083: 215D717519EE1520 3DA
084: 30215409029031D2 727
085: 14D311214B962801 A88
086: 7154F1731331311C DEA
087: 38F861B11B495F21 17B
088: 371440713576A1AE 4F3
089: 84473180962451BF 869
08A: E8F214E111B62550 BEF
08B: AE2112A6214C3120 F62
08C: 9658014851C31109 2C2
08D: 658B14B310F966CA 661
08E: 301194F15C06E9F1 9ED
08F: F4E8F21431318F13 D7B
090: 0015501711371F4E 0DA
091: 8F21451F4F8F2153 46E
092: 4AC215541D3F1514 7F0
093: 7B006A0C8F534101 B6F
094: BFC8F2152494C338 F14
095: F245811A495F1421 290
096: 84146EAAF1D881D8 646
097: 548FF1E718D1CD71 9FD
098: 1A495F1421311731 D5F
099: 841468F401B11B49 0DC
09A: 5F2AF0142184146E 464
09B: A81CD88F42A71146 805
09C: 135159382281DD08 B75
09D: 3240E4D2304C4CA1 F00
09E: 64146C2144D210B1 26D
09F: B5D8F2146190D142 5F8
0A0: 8F7F3104E01B0D8F 9A5
0A1: 2140016BE814A311 D0D
0A2: F96600161D015A31 07F
0A3: 6301AF210910A14B 3F5
0A4: 310F96200171D214 754
0A5: F133C21310617114 AAF
0A6: B8F3E320FE0403CA E62
0A7: 804C2C01C6E09B3F 209
0A8: 06C6116FC11AD9A0 5AA
0A9: ABBB02CF90CF311F 979
0AA: D911006E1117114B CE5
0AB: 311E966FE17114B8 077
0AC: F3E3203413006C20 3E0
0AD: 4472054F5036D104 747
0AE: 632016C202691024 A96
0AF: E20006FC0D2E610A E2F
0B0: 64C0D2E610910A66 1B5
0B1: B0D230210A6AA017 534
0B2: 114B8F504504A030 8A0
0B3: 1902D0D2E6109678 C20
0B4: 0D23021096B70D2E F9F
0B5: 61093110AEA6C60D 32E
0B6: 2E610A3120AEA6A5 6C6
0B7: 017614B968D03130 A2D
0B8: AEA6440D2E610A31 DC9
0B9: 40AEA62303150AEA 15F
0BA: 67203160AEA6C103 4DC
0BB: 170AEA61103180AE 865
0BC: A6600AE00713503A BDC
0BD: 17FF2C5FF1361087 F80
0BE: C027E128F39B7085 314
0BF: 18528F357A14B11F 6A1
0C0: 976F211A14577023 A0F
0C1: 1E16BF11101307FA D99
0C2: 1102D2E6F2108109 109
0C3: F2F2F210B7971436 499
0C4: 1007F61495101756 7F4
0C5: 14F4D61321028F4E B7D
0C6: FF08606061191371 EEA
0C7: 2A1347D314720523 24B
0C8: B142004491D68F4E 5D7
0C9: FF0D015B38705010 957
0CA: 37B211371128A232 CB5
0CB: 1301838F78870D21 028
0CC: 5F31108B6808CF07 3B8
0CD: F1FC65F2147D711A 765
0CE: 135D215F31341180 AC5
0CF: 611A135111D81378 E29
0D0: BFD2135D015B311B 1C3
0D1: 8BEC11108B281041 542
0D2: 36108D2E610954C6 8C1
0D3: 170159317514B1C5 C28
0D4: 31CF962C1316E962 FBA
0D5: 3111007D605E6066 31D
0D6: 9100715D305E6061 682
0D7: 102399E70C010004 9DE
0D8: 20173D214F133CA1 D58
0D9: 3114B17190CCE6E5 0F0
0DA: F078518428F357A1 478
0DB: 8C8C7F14A161311F 814
0DC: 96600D015A316303 B74
0DD: 1FC65F2147D71CE1 F22
0DE: 43D23113CA131017 288
0DF: 2108FEEB60441832 611
0E0: 005C08F627705008 97A
0E1: D393901FD55F2147 D0F
0E2: 13517E143F434412 075
0E3: E08A6618F73B908F 421
0E4: 78870147540D21B8 795
0E5: E7F215C303F A19
```



Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur de JPC, et son siège social est au 33, boulevard saint Martin, 75003 Paris.

La maquette de ce numéro a été préparée et réalisée par Olivier Arbey, Pierre David, Jean-Jacques Dhénin et Janick Taillandier, grâce à un système comprenant un HP71B, deux lecteurs de disquettes HP9114A, un HP9807A, un HP9153 un HP9134 et une imprimante LaserJet.

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

Imprimé par Copy-Express, 42 86 91 94.

ENGLISH SUMMARY

JPC 53 - APRIL 1988

First, something very important : the swiss club PPC Lausanne has decided to stop its own Journal and to share JPC with us. So, JPC is now the only journal for French speaking users of HP handhelds. We hope this will be profitable for both club.

An other important news is the presence in France of David Lin, president of CMT, to present MC II and the other CMT products.

The HP-28 column begins on page 6 by an article by Paul Courbis giving the address to be used on the new HP-28S for the CLOCK program (see JPC 46).

The most important article about the HP-28C was written by Sébastien Lalande. He explains us with clear schematics how to increase the calculator memory or speed, how to interface it with other computers for output as well as for input ! A lot of things have already been made : nice achievement for a closed machine.

The HP-75 column begins on page 12 with an article by Eric Gengoux about memory expansions and CMT Eproms. He indicates that rebates on Eprom burning are available from CMT for our members. Also, Vincent Delorme has made a low cost, 32 Kb PMS.

Then, the article by Jean-Yves Hervé explains how to get a list of keywords from a Lex file and how to decode a Lex file header using MEMLEX.

The HP-71 column begins on page 16 with the second part of the article on structured programming announced last month. This time, the Lex BASICLEX allows you to list a program with control structures, either standard or not, automatically indented. The result is very nice !

Then Aurel Rottman concludes the first series of articles about symbolic derivation. He explains the algorithm used by Peter Ehrenberg and Volker Klann (Prisma article and JPC 44) and fixes some of the known bugs...

Until next month,

Happy Programming and JPC reading !