JPC

FEVRIER MARS AVRIL 1986

NUMERO 31-32-33

Le numéro 35 FF.

A PROPOS DU CLUB

Jean-Jacques Dhénin	Editorial	*1*
Pierre David	PPC-Paris se réunit	*2*
	Courrier des lecteurs	*3*
	Courrier du coeur	*5*

HP-41

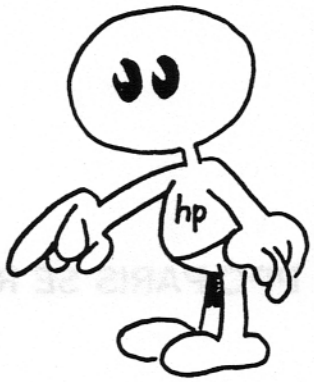
Serge Vaudenay	Polynômes	*10*

HP-71

Alain Herreman	Allez, les GOTO !	*18*
Jean-Jacques Moreau	Manipulations de complexe	*20*
Jean-Jacques Moreau	Pour contraster	*21*
Jean-Jacques Moreau	Cherchez le type !	*22*
Jean-Jacques Moreau	Amnésique ?	*24*
Jean-Jacques Moreau	Trouvez vos chaînes...	*25*
Jean-Jacques Moreau	L'exécution sommaire...	*29*
Jean-Jacques Moreau	Safety first !	*31*
François Le Grand	Prenez date	*35*
Janick Taillandier	Programmation structurée	*38*
Jean-Jacques Moreau	Assemblez sans bruit	*44*
Jean-Jacques Moreau	More...	*50*
Jean-Jacques Moreau	Once again !	*51*
C. Marcoin	Instructif	*53*
Jean-Pierre Bondu	Edith et copy (v.o.)	*54*
Jean-Pierre Bondu	Table des équivalences	*58*
Jean-Marie Simon	Copie de disque	*62*
Philippe Davase	La Troisième colonne	*62*
Jean-Marie Simon	:Port(folio)	*63*
Jean-Jacques Dhénin	Le coin des Lhex	*73*

Edité par PPC-PC : rue Jean-Jacques Rousseau - 75001 PARIS

EDITORIAL



Chers co-lecteurs,

Nous sommes heureux de sortir enfin ce volumineux numéro triple qui fera date. A l'heure où nous écrivons ces lignes, il nous tarde de vous rencontrer pour vous faire part des dernières nouvelles rapportées de Corvallis par M. Martinet.

Notre ami et néanmoins co-llègue est allé au cours du mois de février à une réunion internationale dans les locaux de l'usine HP de l'Oregon. Parmi les informations les plus importantes, nous pouvons citer :

La HP-41 et ses accessoires devraient prochainement voir leur prix baisser. Ainsi, la 41CX descendrait en dessous de 70$. La raison : l'arrivée d'un nouveau contrôleur de boucle HPIL (le HP-72C), un "remake" du HP-71.

Ce micro-ordinateur, toujours avec la même rom système, mais déboguée et élargie, tournerait sur un CPU de 20 bits (réels !) avec une fréquence d'horloge de 10 MHz. On parle d'une extension des roms internes à 256Ko, et d'une capacité d'adressage portée à 2Mo, dont 640Ko (comme les compatibles IBM-PC) d'origine.

D'autre part, le HP-72C est équipé d'un mini-écran (HPIL) plat à cristaux liquides de 16 lignes de 40 caractères. Cet écran est rabattable, détachable et livré d'origine avec la machine.

Prix total : aux environs de 500$. Philippe Guez est depuis peu en possession d'un prototype, et le banc d'essai de cette fabuleuse machine paraîtra dans le prochain numéro.

Toute l'équipe se joint à nous pour vous souhaiter de bonnes heures de programmation...

Jean-Jacques Dhénin et Pierre David

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 péride 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 Stransbourg 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 19 avril 1986
Samedi 3 mai 1986
Samedi 7 juin 1986

Pierre DAVID

COURRIER DES LECTEURS

M. CANTERI Gilbert
Groupe scolaire Linas, 20/2/1986
Place E.PILLON
91310 LINAS

J'aimerai utiliser une calculatrice programmable HP41CV pour faire de l'acquisition de données à partir de signaux analogiques 0-5 volts.

Pourriez-vous me dire s'il existe des systèmes de convertion analogiques-digitaux (ainsi que leurs références) pour cette calculatrice et s'il existe des programmes d'acquisition.

Cher ami,

l'acquisition de données avec ton HP41 est possible. Cependant jusqu'à présent j'ai plutôt vu des applications faisant de l'aquisition digitale. La règle à calcul (Bld Saint-Germain à Paris) a présenté une boucle HPIL contenant une balance (type pharmacie) au Sicob. De plus, il me semble que la Société Kristal à Grenoble a travaillé sur des applications analogues.

JJ DHENIN

BILLEX
94700 MAISONS ALFORT 11/01/1986

Le manuel du Forth/Assembleur me semble un peu chiche en explication, particulièrement sur l'assembleur. Aussi j'aimerai connaître la réponse à 2 ou 3 questions.

-A combien (où comment) trouve-t-on les 3 volumes des I.D.S. internal design specification ?

-Quel est le tome le plus intéressant ?

-Existe-t-il un livre que vous recommandez dans le domaine de l'assembleur ?

A la relecture des éditions du JPC, il m'a semblé que le forth était un peu délaissé, à ce propos, n'y a-t-il pas un livre correct pour l'apprentissage du FORTH ?

Concernant l'écriture des programmes en assembleur, est-il nécessaire de définir des labels qui ne servent qu'une fois ? ex :

```
=BF2DSP  EQU   #01C0E   | pourrait s'écrire
         GOSBVL =BF2DSP | GOSBVL #01C0E
```

Bien que le manuel ne parle pas de cette possibilité.

Par ailleurs, pourrait-on envoyer des articles sous forme carte magnétique pour aider à la parution du journal ?

Cher ami,
reprenons dans l'ordre.

- Les IDS sont un ensemble de documentations comprenant 5 volumes :

* HP-71 Software Internal Design Specification
  Volume I : Detailed Design Description
  Volume II : Entry Point and Poll Interfaces
  Volume III : Operating Systeme Source Listing
* HP-71 Hardware Specification
* HP-71 HP-IL Module Internal Design Specif.
  Volume I : Detailed Design and Entry Point
  Volume II : Source Listing

Pour obtenir ces documents, il convient de s'adresser à HP France. Cependant on peut te prêter l'un ou l'autre de ces volumes si tu en exprimes le souhait au cours d'une de nos réunions du Samedi. Tu pourras ainsi te rendre compte par toi-même de leur intéret.

Si tu n'es pas rompu au travail en assembleur il ne me semble pas qu'un livre puisse te guider en tous cas en ce qui concerne le HP-71 qui est équipé d'un CPU 64 BITS. Cependant la lecture des sources (IDS) et la réalisation de tes propres fonctions te conduiront à une maitrise de plus en plus performante. On parle également d'un groupe qui se réunirait à Paris tous les samedis. Mais qui serait intéressé ?

En ce qui concerne le FORTH, JPC est le reflet de la composition des membres du Paris-Chapter. Seuls les articles qui nous sont envoyés

composent le journal et personne n'est rétribué. En conséquence tant que le FORTH ne trouve que peu d'amateurs, il sera délaissé, en effet.
Des livres ce n'est pas ça qui manque en la matière. Par contre il semble difficile te t'en conseillé un. Tout dépend de ton niveau. Va faire un tour dans une librairie, parcours quelques ouvrages tels que Débuter en FORTH ou Programmer le FORTH R.V. LOO Ed. Marabout (30F).

Enfin, et c'est important, tu as tout à fait raison, nous préférons recevoir les articles sur support magnétique (cartes, cassettes, ou disquettes) et cela même pour le courrier qu'il faut bien retaper pour la mise en page.

JJ DHENIN

DANIEL CONNAN PPC PC 143
152 Avenue J.Jaurès 5/12/85
93500 PANTIN

Une petite note discordante dans le concert d'autosatisfaction dont donne l'impression la lecture de JPC.

Attention, je ne mets pas en doute la bonne foi et le dévouement de la poignée de personnes qui se décarcassent pour le faire "sortir".

Mais je dois reconnaitre en toute subjectivité que je n'y trouve pas ce que j'y cherche, et que jusque maintenant, j'ai cotisé pour rien. Je ne pense pas être le seul dans mon cas, car il est maintenant de notoriété publique que la micro-informatique d'amateur marque le pas, sinon régresse. Cela commence à se savoir qu'à part l'usage professionnel un micro ordinateur ne sert à rien.

Pressentant la chose, j'ai voulu tâter prudemment le terrain avec un HP34C, une Casio FX702P, et enfin un HP15C et je me suis fixé comme but l'étude des mathématiques.

Une chose est acquise, les calculateurs ne sont pas en cause, ce sont de merveilleux outils fiables et tout et tout.

Avec eux, tant bien que mal, j'ai eu accès à des notions mathématiques qu'il m'aurait été impossible d'acquérir autrement, mais c'est là où le bât blesse, quand une question se présente à mon esprit où trouver la réponse ?

Certains livres sont tellement abstraits que je les trouve illisibles. Question de niveau ? bien sur, mais dans certains livres je comprends les séries de Fourier, les polynômes de Lagrange, et dans d'autres la notion de dérivée est camouflée soigneusement sous un langage barbare. Je ne suis arrivé à la comprendre qu'en recoupant cinq à six livres différents.

J'aurais bien aimé trouver au club des "cours magistraux" adaptés aux machines sur telle ou telle question que chacun aurait choisi à son tour. Les "réunions informelles" du samedi c'est bien, mais les a-parté à voix basse n'apprennent pas grand chose.

Merci quand même à Pierre DAVID qui lors de ces réunions a bien voulu me consacrer deux heures à faire tourner sur HP71B un programme coefficients FOURIER et à m'en fournir le listing... en BASIC alors que le propre de HP était justement la notation polonaise inverse qui a bien été laissée en plan la pauvre. Depuis j'ai mis sur pied un programme calculant les dits coefficients et restituant le graphe d'après ces coefficients, sur 36 points, le maximum permis par le nombre de mémoires,

Encore une chose, j'ai vu quelque part que HP avait daigné établir des relations avec PPC sur des bases "professionnelles". Qu'est-ce à dire ?

Jusqu'à nouvel ordre, PPC est un club régi par la loi de 1901, c'est à dire à but non lucratif. Je pense que le professionalisme d'HP à sens unique a pour but de VENDRE le maximum de matériel. Pas de mystère.

Et bien soit, soyons professionel. J'en suis toujours au même point : je cherche à faire tourner sur HP15C le programme PI en multiprecision de Science et Vie N 789 de décembre 1980. (déjà !). Naïvement j'avais offert une médaille à votre concours de programmes (dont on n'entend plus parler), visiblement personne ne s'y interesse.

Voyons les capacités des professionnels . Sont-ils au top niveau et concurrentiels ?

Je lance ici ce qu'on appelle un appel d'offres, c'est à dire que je suis prêt à payer pour avoir la solution de mon problème. Je serais également intéressé par un "topo" et un programme calculant toutes les racines réelles et complexes de l'équation du 5ème degré. Pas de démonstration abstraite mais un programme qui tourne sur HP15C.

J'attends les propositions. Je rappelle aussi que j'ai proposé 2 fois de venir aider à la parution de JPC, j'ai envoyé ma contribution pour le stand du SICOB, j'ai offert du temps pour aider à le tenir : pas de réponse.

Tout cela est peu encourageant. Je pressens pourtant que le HP71B est un outil formidable, mais je ne ferai l'effort de l'acquérir que quand j'aurai la certitude de trouver réponse aux innombrables questions qu'il ne manquera pas de susciter, ce qui apparemment n'est pas demain la veille.

Et puis, le graphisme en couleur de l'Amstrad de 256 ou 512 Ko n'est pas mal non plus pour les jeux, la simulation (voler en spitfire !) etc...etc...

Conclusion : à défaut de cette lettre qui déparerait JPC pourriez-vous insérer (même à titre onéreux) l'appel d'offres suivant :

Je cherche à titre payant une personne pouvant faire tourner sur HP15C le programme Science et vie 789 de décembre 1980 et calculant PI en multiple précision.
Je suis acheteur également d'un programme calculant les solutions réelles et complexes de l'équation du 5ème degré. (sur HP15C)

Cher ami,

devant cette avalanche de compliments, je vais devoir prendre un peu la défense de mes petits camarades, et ce, point par point.

Le concert de satisfaction où tu mêle ta note discordante, est justifié à plusieurs titre :
-la bonne humeur, l'ironie font partie de notre club. Il est déjà assez difficile de résoudre les difficultés quotidiennes sans en rajouter par une attitude constipée.
-Nous étions partis à une dizaine dans cette aventure et nous sommes maintenant près de trois cents. -Cela a nécessité une grande régularité de l'effort des heures de travail, et par voie de conséquence nous pouvons bien ne pas nous prendre au sérieux.
-Nous sommes effectivement les meilleurs (sur HP71) il suffit, pour s'en convaincre, de lire les autres revues. Nous avons plus de 60 fonctions en assembleur à notre actif à ce jour.
-Tous cela n'aurait pas été possible sans un travail collectif de beaucoup. Il n'y pas de place pour l'individualisme dans notre association.

Nous sommes tous de bonne foi et dévoués, car si certains mettent en page le journal, d'autres se décarcassent pour écrire les articles.

Ce journal est, comme on dit, une auberge espagnole tu y trouves ce que tu apportes. Oh! j'entends déjà ta réponse, tu te proposes pour le Sicob et en contre partie tu veux des programmes pour ton HP15, et de plus tu t'étonnes de ne pas avoir de réponse pour ta participation au SICOB. Fallait venir. On avait tellement de boulot qu'on n'avait même pas le temps de te répondre.

Des cours magistraux il y en a la FAC. Nous, nous réunissons pour échanger. La preuve P. David t'a donné de son temps pour résoudre ton problème. Cela dit puisque tu sembles de bonne volonté je te propose de mettre sur pied les réunions dont je parle dans la réponse précédente. Ce sera une bonne chose que nous ayons quelqu'un qui prenne en charge l'organisation de ce truc.

Je termine. Combien faut-il de temps pour recopier ta lettre et y répondre ?

JJ DHENIN (de mauvais poils)!

COURRIER DU COEUR

Jean-Jacques DHENIN
35 rue Boileau

92120 MONTROUGE
Vend :
1 module "Translator-pac" pour HP71B"
Prix neuf : 1830ff, soldé 1400ff.
J.-P. INDJEHAGOPIAN, 142 Grande Rue, 92380 Garches. Tel.: Bur. (1) 30 38 38 00 - Dom. (1) 47 41 21 97. Vends: HP75C (02/83 peu servi) avec module 8k à 3800FF, Imprimante 80 colonnes HP82905B IL (02/83 peu servi) à 3000FF, Moniteur Vidéo (22cm) HP82912A à 1000FF, Interface Vidéo HP82163B à 1100FF, Lecteur de cartes HP82104A (1984 jamais servi) pour HP41 à 1100FF, Lecteur de codes barres pour HP41 à 600FF, Module PLOTTER HP82184 pour HP41 à 500FF, des accessoires pour HP75: Text Form à 600FF, Visicalc à 1300FF, Maths à 900FF. Factures, modes d'emploi, emballages d'origine fournis.

Vend lecteur de cartes pour HP-71. Etat neuf (Acheté le 01/02/85). Sous garantie. Contacter:
Jean-Jacques Moreau
64 ave de la paix
93150 le Blanc-Mesnil
Tel: 48-67-33-04 (Apres 20 heures)

Thierry BRAVIER, 60 Rue du Chateau d'eau, 75010 Paris. Tel.: (1) 42 02 28 90. Urgent, vends pour HP41: module HPIL (1983) quasiment jamais servi, Lecteur de codes barres, lecteur d'éproms - programmeur de microcodes avec une éprom préprogrammée pleine de fonctions, un lecteur de cartes à faire réparer et divers accessoires pour HP41 (cartes magnétiques, chargeur, étui pour piles à encastrer, grilles plastifiées), grosse documentation (Nombreux programmes, livres, manuels et de très nombreux numéros de divers PPC - US, Paris, Toulouse, Australie -) le tout vendu ensemble ou séparément. Faites vos offres (raisonnables).

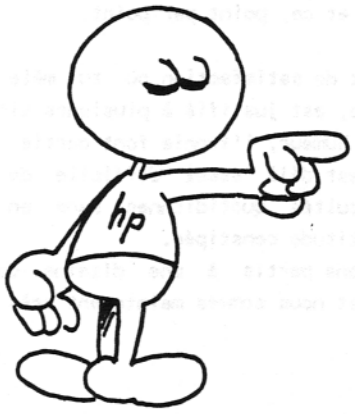
A. BOURG-BROC
13, rue Albert Samain
45000 ORLEANS
Vends :
1 HP41 CV (déc. 81)
1 Module X.FUNCTIONS
1 Module Extention-Mémoire
1 Module "financier"
1 Module "statistiques"
1 lecteur de cartes magnétiques
+ manuel d'utilisation de la 41CV
+ manuel d'applications
+ manuel de l'utilisateur expérimenté
+ manuel d'utilisation du lecteur
+ bilbliothéque "financière"
+ bibliothéque "statistique"
+ batterie cadmium neuve
clavier neuf

L'ensemble = 4100,00 ff
Contact = 10h/17h (1) 43 87 23 51
20/21h30 38 62 77 83

F. LEGRAND
Rue de la poterie
TREVIGNON
29128 TREGUNC
tel : 98 50 00 10
Vends :

1 HP41CV	1100
1 imprimante 82143	1000
1 lecteur de cartes 82104	800
1 modue time	400
1 module extention fonctions	400
2 modules extention mémoire	800
1 module navigation 14017	200
500 cartes magnétiques	400
1 clavier souple	0
1 convertisseur hpil 82166 neuf	1500
1 overlay kit	0
?... accessoires divers	0
Total	6500
bradé	6000

F. DEJEAN
921, rue du Dr. SHAFFNER
NOUIYELLES/SOULENS
62221 Nord Pas-de-Calais
Vends :
1 HP41
1 imprimante 82143
1 lecteur de cartes
4500 ff



D.B.E ingenierie

Monsieur Philippe GUEZ
P P C PARIS
56 rue J.J Rousseau
75001 PARIS

V/RÉF.

N/RÉF.
AB/HA 86095

BOULOGNE, le 25 février 1986

Monsieur,

Comme convenu lors de notre entretien téléphonique de ce jour, nous vous confirmons par la présente nos besoins :

Fonction : Analystes Programmeurs
Matériel : H.P 71
Langage : Assembleur 71 (et Basic)

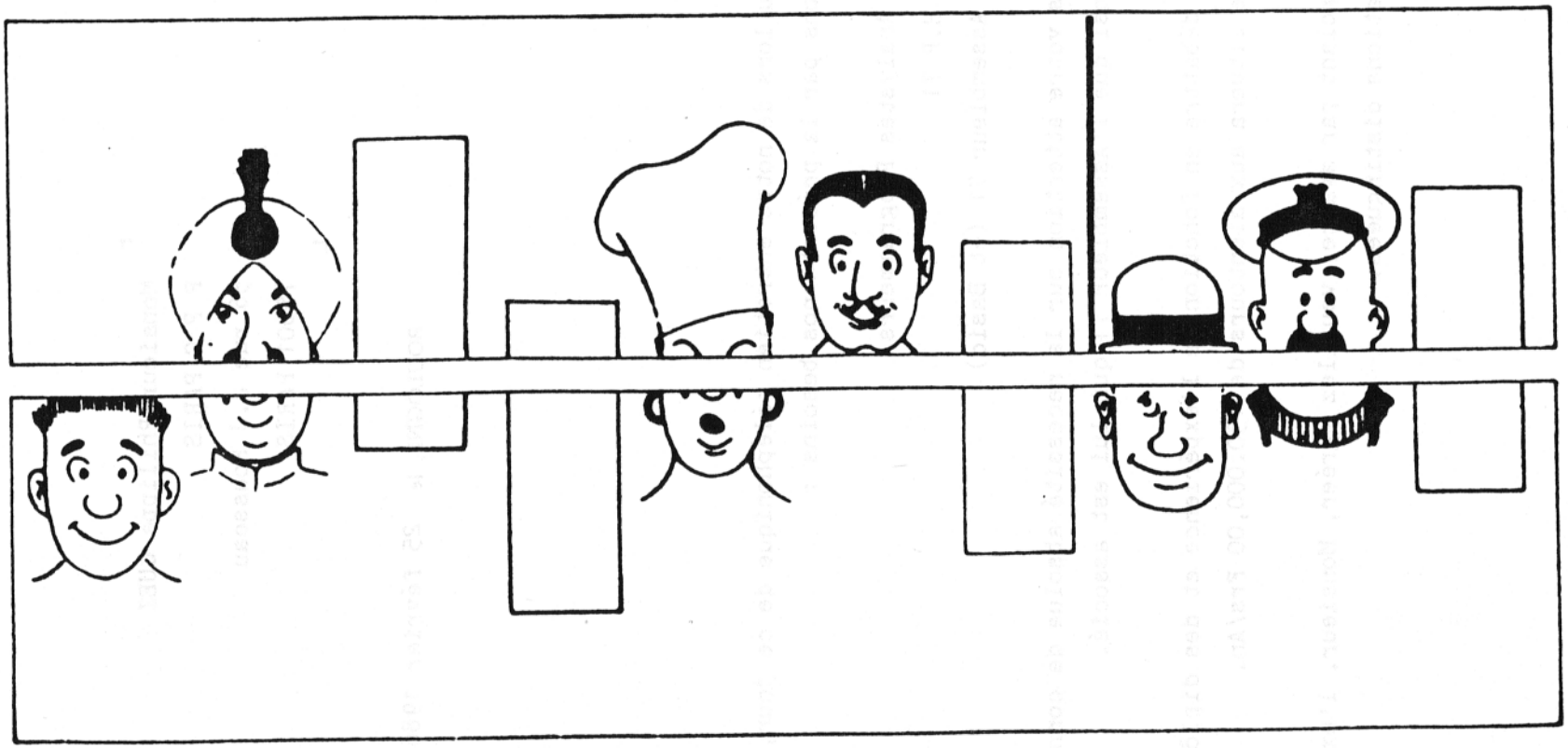
Nous attirons votre attention sur la necessité absolue de connaitre le H.P 71 ainsi que l'Assembleur 71 qui lui est associé.

Le salaire à débattre en fonction de l'expérience et des diplômes éventuels, se situera aux alentours de 130.000,00 Frs/An.

En vous remerciant par avance, veuillez agréer, Monsieur, l'expression de nos salutations distinguées.

Bernard ARROUA
Directeur Informatique

Découper les trois panneaux. Suivant la manière dont on les assemble, l'une des têtes disparaît.



POLYNOMES

Approximations polynômiales.

Je vous parlerai, aujourd'hui, d'un problème que n'importe quel scientifique peut rencontrer: l'observation d'un phénomène physique.

Que ce soit l'évolution du pH d'une solution en fonction du temps, la température ambiante d'une région en fonction de la date, le nombre des articles écrits par un adérant du club en fonction de l'éxitabilité de ses neurones, le nombre d'œufs pondus par une poule convenablement nourrie en fonction de son âge ou la pression de la vapeur d'eau en fonction de la température (exemple que j'ai retenu), nous pouvons mesurer ces phénomènes, mais à priori, nous ne pouvons pas trouver de loi générales.

On a fait des mesures, et on voudrais bien pouvoir évaluer sans trop d'erreurs ce qui se passe entre ces mesures. En suposant que le phénomène soit "monotone" entre les mesures, c'est à dire qu'il varie toujours dans le même sens, on peut trouver un polynôme dont la courbe coïncide parfaitement avec les mesures effectuées. La mémoire de votre HP41 limite à 14 mesures pour la recherche de ce polynôme. Le programme que je vous livre évalue donc, à l'aide du nouveau module de gestion de tableau, ce polynôme qui permettra de généraliser le phénomène dans les limites des mesures effectuées (il n'est donc pas question d'extrapoler).

Prenons l'exemple de la pression de la vapeur d'eau en fonction de sa température. Supposons que nous ayons effectué les six mesures suivantes :

-5 degrés 0.0025
0 degrés 0.0038
5 degrés 0.0054
10 degrés 0.0076
15 degrés 0.0107
20 degrés 0.0147

Pour entrez ces valeurs, faites :

XEQ "POL"

A la question

DEGRE ?

vous répondez :

5.00

(ce sera le degré du polynôme à chercher à partir des 5+1 mesures) puis [R/S]

Vous pouvez ensuite les températures des mesures :

X1,1=_ [5]
X1,1=5_ [CHS]
X1,1=-5_ [R/S]
etc..
X2,1=0
X3,1=5
X4,1=10
X5,1=15
X6,1=20

Vous entrez les pressions :

Y1,1=0.0025
Y2,1=0.0038
Y3,1=0.0054
Y4,1=0.0076
Y5,1=0.0107
Y6,1=0.0147

Après un certain temps de réflexion, la machine affiche :

X2,1=2.00 -08
X3,1=3.00 -07
X4,1=5.50 -06
X5,1=2.83 -04
X6,1=3.80 -03

Ce sont les coefficients du polynôme (ordre décroissant).

REG 47a52

y=R47*x^5 +R48*x^4 +R49*x^3 +R50*x^2 +R51*x +R52

Avec le POL, on a donc une approximation acceptable pour ce phénomène. Le programme POL résoud, en fait, un système de n+1 équations linéaires à n+1 inconnues (n étant le degré du polynôme).

POL est un programme dérivé du programme SYST, qui résoud n équations à n inconnues, par la méthode du pivot de Gauss dont on donne également le listing.

Exemple : soit le système suivant

2x - 3y + z = -6		2	-3	1	-6
5x - 6y = 3		5	-6	0	3
x + 4y + 9z = 2		1	4	9	2

J' ai écris à côté la matrice du système.

Faites XEQ'SYST', entrez le nombre d'inconnues (içi ordre 3), [R/S], puis la matrice, ligne par ligne. Le programme liste ensuite la matrice pour la vérifier, et vous demande si c'est bon (OK ?).

Vous répondez 0 pour oui, et la machine vous donne les solutions qui sont içi :

x=8.0943 y=6.2453 z=-3.4528 (registres 17 à 19)

Note : un système singulier admet une infinité de solutions, et un système impossible n' admet aucune solution.

Serge Vaudenay

PROGRAMME POL

Même si d'un point de vue strictement mathématique ce programme n'est pas satisfaisant, du fait que si le nombre de mesures est important on obtient un polynôme inommable, ce programme offre l'intérêt de mettre en évidence les qualités du nouveau module de gestion de tableau (PANAME) fabriqué par des membres du club PPC français et disponible avec un manuel de près de 200 pages à la Règle à Calcul ou auprès de votre club préféré.

```
01■LBL "POL"
02 "DEGRE ?"
03 PROMPT
04 1
05 +
06 5
07 RCL Y
08 1
09 +
10 RCL Z
11 X<>Y
12 CHS          X contient le numéro du premier
                registre du tableau. Le signe -
                prépare la construction d'un
13 BLDPT        pointeur de tableau.
14 RCL X
15 CLINC        En effaçant l'incrément, on
                désigne la totalité du tableau.
16 RGNb         Calcul du nombre de cellules.
17 RCL Z
18 +
19 6
20 +
21 SIZE?        Le module de gestion de tableau
                contient les fonctions SIZE et
                PSIZE.
22 X<>Y
23 X>Y?
24 PSIZE
25 R^
26 STO 04
27 R^
28 STO 00
29 BRKPT        Eclatement du pointeur en ses
                3 composantes dans X, Y et Z.
30 RDN
31 1
32 +
33 RCL 04
34 1
35 +
36 -1
37 BLDPT        Reconstruction du nouveau
                pointeur.
38 STO 03
39 "X"
40 XEQ 02
41 RCL 04
42 1
43 -
44 RCL 00
45 COLPT        Calcule le pointeur de la colonne
46 RCL 03
47 1 E-3
48 -
49 RGCOPY       Copie la colonne toute entière.
50 STO 02
51 1
52 RCL 04
```

```
53 RCL 00
54 COLPT
55 RGINIT       Remise à zéro de la colonne.
56 RG+Y
57 2
58 RCL 04
59 X=Y?
60 GTO 00
61 2
62 -
63 STO 01

64■LBL 07
65 RCL 01
66 INT
67 RCL 00
68 COLPT
69 RCL 02
70 RGCOPY
71 RCL 01
72 INT
73 1
74 +
75 RCL 00
76 COLPT
77 RG*          Produit terme à terme des deux
                colonnes
78 DSE 01
79 GTO 07

80■LBL 00
81 RCL 03
82 1 E-3
83 -
84 SORT         Effectue le tri.
85 RCL X
86 1
87 +

88■LBL 08
89 RCL IND X
90 ST- IND Z
91 RDN
92 RCL IND Y
93 X#0?
94 GTO 00
95 "SINGULIER"
96 AVIEW
97 RTN

98■LBL 00
99 RDN
100 RCL X
101 ISG X
102 GTO 08
103 "Y"
104 XEQ 02
105 RCL 04
106 1
107 +
108 RCL 00
109 COLPT
110 RCL 03
111 1 E-3
112 -
113 RGCOPY
114 RCL 04
115 1
116 -
117 STO 01

118■LBL 03
119 1
120 RCL 01
121 0
122 BLDPT
123 STO 02

124■LBL 04
125 RCL 02
126 INT
127 RCL 01
128 1
129 +
130 RCL 00
131 LC-AD       Calcule le numéro du registre
                où se trouve la valeur
                Y=ligne, X=colonne -> Registre
132 RCL IND X
133 X=0?
134 GTO 05
135 X<>Y
136 RCL 04
137 +
138 1
139 +
140 RDN
141 RCL IND T
142 X#0?
143 GTO 00
144 RCL 02
145 INT
146 RCL X
147 1
148 +
149 RCL 00
150 LINPT       Calcule le pointeur de ligne.
151 X<>Y
```

```
152 RCL 00
153 LINPT
154 CHS          Le changement de signe indique
155 RGCOPY       un échange terme à terme des
                 deux lignes.
156 GTO 04

157■LBL 00
158 /
159 CHS
160 RCL 02
161 INT
162 1
163 +
164 RCL 00
165 LINPT
166 RCL 03
167 X<>Y
168 RGCOPY
169 RG*Y         Multilie chaque terme de la
                 ligne par la constante
                 constante contenue dans Y.
170 RCL 02
171 INT
172 RCL 00
173 LINPT
174 X<>Y
175 RG+-         Somme terme à termes des 2
                 lignes.
176■LBL 05
177 ISG 02
178 GTO 04
179 DSE 01
180 GTO 03
181 1
182 RCL 04
183 .
184 BLDPT
185 STO 01
186 1 E-3
187 ST- 03

188■LBL 06
189 RCL 01
190 INT
191 1
192 -
193 X=0?
194 GTO 00
195 1
196 +
197 1
198 RCL 00
199 LC-AD
200 RCL 01
201 INT
202 RCL X
203 1
204 -
205 RCL 00
206 LC-AD
207 .
208 BLDPT
209 RGSUM         Calcule la somme des éléments

210■LBL 00
211 RCL 01
212 INT
213 RCL 04
214 1
215 +
216 RCL 00
217 LC-AD
218 RDN
219 RCL IND T
220 -
221 CHS
222 RCL 01
223 INT
224 RCL X
225 RCL 00
226 LC-AD
227 RDN
228 RCL IND T
229 /
230 1
231 RCL 01
232 INT
233 RCL 03
234 LC-AD
235 X<>Y
236 STO IND Y
237 RCL 01
238 INT
239 RCL 00
240 COLPT
241 RG*Y
242 ISG 01
243 GTO 06
244 RCL 03
245 "X"
246 2 E-7
247 +
248 CHS
249 RGVIEW       Visualise tous les éléments
                 de la matrice .
250 CHS
251 BRKPT
```

```
252 "REG "
253 RDN
254 X<>Y
255 APPX            Affiche la partie entière.
256 "|-a"
257 X<>Y
258 APPX
259 CLST
260 AVIEW
261 RTN

262■LBL 02
263 RCL 03          Rappel du pointeur matrice
                    à saisir.
264 CLINC           Indique tous les éléments.
265 RGINIT          Remise à zéro de tous les
                    éléments.
266 9.997 E-4       Prépare l'affichage pour
267 -               RGVIEW : X i,j=
268 CHS             Arrêt sur première valeur.
269 RGVIEW          Cette seule instruction pour
                    saisir la totalité de la
                    matrice. Merci.
270 .END.
```

PROGRAMME SYST

Résolution de systèmes par la méthode de Gauss au moyen du module gestion de tableau dit PANAME parce qu'il a été conçu à Paris. Vous apprécierez les ordres BLDPT, BRKPT, COLPT, LGNPT, RGCOPY, RGINIT ..etc. Il y a comme ça plus de 120 fonctions dans un module de 8 Ko

```
01■LBL "SYST"
02 "ORDRE ?"
03 PROMPT
04 5
05 RCL Y
06 1
07 +
08 RCL Z
09 X<>Y
10 CHS
11 BLDPT
12 RCL X
13 CLINC
14 RGNb
15 RCL Z
16 +
17 6
18 +
19 SIZE?
20 X<>Y
21 X>Y?
22 PSIZE
23 R^
24 STO 04
25 R^
26 STO 00
27 BRKPT
28 RDN
29 1
30 +
31 RCL 04
32 1
33 +
34 -1
35 BLDPT
36 STO 03
37 RCL 00
38 CLINC
39 RGINIT

40■LBL 02
41 RCL 00
42 3 E-7
43 +
44 CHS
45 "A"
46 RGVIEW
47 "VERIFICATION"
48 AVIEW
49 "A"
50 CHS
51 1 E-7
52 -
53 PSE
54 RGVIEW
55 "OK ?"
56 Y/N
57 GTO 00
58 GTO 02

59■LBL 00
60 RCL 04
61 1
62 -
63 STO 01

64■LBL 03
65 1
66 RCL 01
67 0
68 BLDPT
69 STO 02
```

```
 70■LBL 04
 71 RCL 02
 72 INT
 73 RCL 01
 74 1
 75 +
 76 RCL 00
 77 LC-AD
 78 RCL IND X
 79 X=0?
 80 GTO 05
 81 X<>Y
 82 RCL 04
 83 +
 84 1
 85 +
 86 RDN
 87 RCL IND T
 88 X#0?
 89 GTO 00
 90 RCL 02
 91 INT
 92 RCL X
 93 1
 94 +
 95 RCL 00
 96 LINPT
 97 X<>Y
 98 RCL 00
 99 LINPT
100 CHS
101 RGCOPY
102 GTO 04

103■LBL 00
104 /
105 CHS
106 RCL 02
107 INT
108 1
109 +
110 RCL 00
111 LINPT
112 RCL 03
113 X<>Y
114 RGCOPY
115 RG*Y
116 RCL 02
117 INT
118 RCL 00
119 LINPT
120 X<>Y
121 RG+-

122■LBL 05
123 ISG 02
124 GTO 04
125 DSE 01
126 GTO 03
127 1
128 RCL 04
129 .
130 BLDPT
131 STO 01
132 1 E-3
133 ST- 03

134■LBL 06
135 RCL 01
136 INT
137 1
138 -
139 X=0?
140 GTO 00
141 1
142 +
143 1
144 RCL 00
145 LC-AD
146 RCL 01
147 INT
148 RCL X
149 1
150 -
151 RCL 00
152 LC-AD
153 .
154 BLDPT
155 RGSUM

156■LBL 00
157 RCL 01
158 INT
159 RCL 04
160 1
161 +
162 RCL 00
163 LC-AD
164 RDN
165 RCL IND T
166 -
167 CHS
168 RCL 01
169 INT
170 RCL X
171 RCL 00
172 LC-AD
```

```
173 RDN
174 RCL IND T
175 SF 25
176 /
177 FC?C 25
178 GTO 07
179 1
180 RCL 01
181 INT
182 RCL 03
183 LC-AD
184 X<>Y
185 STO IND Y
186 RCL 01
187 INT
188 RCL 00
189 COLPT
190 RG*Y
191 ISG 01
192 GTO 06
193 RCL 03
194 "X"
195 2 E-7
196 +
197 CHS
198 RGVIEW
199 CHS
200 BRKPT
201 "REG "
202 RDN
203 X<>Y
204 APPX
205 "|-a"
206 X<>Y
207 APPX
208 CLST
209 AVIEW
210 RTN

211■LBL 07
212 X<>Y
213 "IMPOSSIBLE"
214 X=0?
215 "SINGULIER"
216 AVIEW
217 CLST
218 .END.
```

HP-71

HP-71 Forth/Assembleur

Alain Herreman	Allez, les GOTO !	*18*
Jean-Jacques Moreau	Manipulations de complexe	*20*
Jean-Jacques Moreau	Pour contraster	*21*
Jean-Jacques Moreau	Cherchez le type !	*22*
Jean-Jacques Moreau	Amnésique ?	*24*
Jean-Jacques Moreau	Trouvez vos chaînes...	*25*
Jean-Jacques Moreau	L'exécution sommaire...	*29*
Jean-Jacques Moreau	Safety first !	*31*
François Le Grand	Prenez date	*35*
Janick Taillandier	Programmation structurée	*38*
Jean-Jacques Moreau	Assemblez sans bruit	*44*
Jean-Jacques Moreau	More...	*50*
Jean-Jacques Moreau	Once again !	*51*
C. Marcoin	Instructif	*53*
Jean-Pierre Bondu	Edith et copy (v.o.)	*54*
Jean-Pierre Bondu	Table des équivalences	*58*

HP-71 Basic

Jean-Marie Simon	Copie de disque	*62*
Philippe Davase	La Troisième colonne	*62*
Jean-Marie Simon	:Port(folio)	*63*

HP-71 Lhex

Jean-Jacques Dhénin	Le coin des Lhex	*73*

ALLEZ, LES GOTO!

Les quelques mots que je vais vous présenter sont à proscrire de toutes vos réalisations en FORTH ! En effet j'ai écris pour le langage le plus structuré les fonctions les moins structurantes, je veux parler des GOTO numériques (j'espére que C. Moore n'est pas abonné à JPC !).

Pour introduire cette hérésie et pour éviter que vous ne bruliez la revue, je vais vous dire quelques mots sur les structures de contrôle en FORTH (control structures pour les initiés).

Commençons par du vague ; les structures de contrôle sont toutes ces "fonctions" : DO ... LOOP,

DO ... +LOOP, IF ... ELSE ... THEN, IF ... THEN, BEGIN ... UNTIL, BEGIN ... WHILE ... REPEAT, CASE ... OF ... ENDOF ... ENDCASE, LEAVE.

En fait le noyau commun à ces mots se résume à deux mots, je crois communément appelés BRANCH et OBRANCH .

BRANCH est un branchement inconditionnel, il déplace le pointeur de N octets où N se trouve dans la suite des codes compilés juste aprés le code de BRANCH (E5D99 en hexa.). OBRANCH lui est un branchement conditionnel, il ne déplace le pointeur de N octets que si le test qui le précède est négatif, c'est-à-dire si la valeur du dessus de la data-stack est nulle.

Mais là se pose un problème. Prenons l'exemple de IF ... THEN .

Lors de la compilation, IF va être transformé en un OBRANCH (de code E5D86), mais on a vu précédemment qu'aprés le OBRANCH devait se trouver la longueur du saut en octet, ici, si le test est négatif le pointeur doit enchaîner sur THEN, mais on ne sait pas encore où se trouve le THEN . La solution à ce problème est en fait très naturelle .

IF lors de la compilation va déposer dans la data-stack 2 nombres : son adresse et un numéro qui le caractérise et donc quand arrive le moment de compiler le THEN correspondant à notre IF, on devra avoir sur le dessus de la data-stack le numéro qui caractérise IF et son adresse. Si THEN ne trouve pas ce numéro (par exemple si vous avez mal manipulé le contenu de la data-stack avec des mots immédiats) vous serez avertis par un "conditionals not paired". Si par contre (ce qui est le cas le plus fréquent) ce test est vérifié, il est alors possible de calculer la longueur du saut, elle correspond à la différence entre HERE (adresse actuelle du pointeur) et l'adresse qui se trouve sur le dessus de la pile et cette différence on la stocke à cette mème adresse, c'est à dire juste après le code de OBRANCH. Voilà, toutes les structures de contrôle pendant le compile-time fonctionnent sur ce mode ci.

Revenons sur les points importants.

Il faut bien noter que les nombres laissés sur la pile vont par paire : le numéro caractéristique et l'adresse du branchement. La structure L.I.F.O. de la pile (Last In First Out) fait que se trouve sur le dessus de la pile les codes concernant la structure qui est en train d'être compilée. Quatre messages d'erreur concernent ces structures :

"conditionnals not paired " (voir ci-dessus), "no DO before LEAVE" qui apparait simplement si LEAVE ne trouve pas le numéro caractéristique de DO sur la pile des données, "illegal CASE structure" qui apparait pour des raisons semblables et "definition not finished" qui apparait si lors de la compilation le nombre de données de la data-stack a varié, ce test permet de vérifier que les structures de contrôle ont bien marché, c'est-à-dire que chaque IF ait eu son THEN, BEGIN son REPEAT avec le WHILE ou son UNTIL ...

Le fonctionnement de ce test est simple : au début de la compilation d'un mot (:) on retient le nombre de valeurs dans la data-stack, à la fin (;) on regarde si ce nombre n'a pas changé, s'il a changé il est possible que se soit l'adresse ou le numéro caractéristique d'une structure de contrôle qui soit en trop, la définition ne serait donc pas finie mais il se pourrait très bien que vous ayez rajouté à l'aide de mots immédiats des données dans la pile, mais ça on ne veut pas le savoir !

On peut aussi remarquer que certains de ces

mots ont une fonction pendant l'exécution (run-time) et d'autres pas. Par exemple LOOP pendant la compilation devra calculer l'adresse de son DO comme il est expliqué ci-dessus, mais devra pendant l'exécution tester les valeurs de la boucle pour savoir s'il faut continuer à tourner ou passer à la suite. Par contre THEN lors de l'exécution ne fait rien, se qui se comprend très bien vu qu'il n'est qu'un point de branchement . Vous pouvez lister les codes d'un mot contenant un IF ... THEN, vous verrez qu'il n'y a aucune trace du THEN.

Voici comme promis les numéros qui caractérisent les différentes structures de contrôle (d'aprés les I.D.S. HP-71 FORTH/Assembleur ROM) :

DO (3), LEAVE (6), LOOP (aucun), +LOOP (aucun), IF (2), ELSE (2), THEN (aucun), BEGIN (1), WHILE (4), REPEAT (aucun), UNTIL (aucun), CASE (8), OF (5).

On peut maintenant comme application de ce qui précède créer des GOTO numériques. L'avantage (?) de ces GOTO est qu'ils ont une structure non bijective, c'est-à-dire qu'à un label peut correspondre autant de GOTO qu'on le désire. Nous allons créer des labels de 0 à 9, dont les numéros caractéristiques irons respectivement de 10 à 19 afin de ne pas interférer avec ceux des autres structures de contrôle (qui vont de 1 à 8). Du fait qu'un label peut servir plusieurs fois on ne peut comme avec les autres structures de contrôle perdre son adresse dés qu'un GOTO lui fait référence, il nous faut donc créer une "grosse variable" VLBL qui contiendra les adresses des différents labels.

Voyons maintenant le fonctionnement en détail.

Sur la HP-41 on faisait GTO 01 sur le 71 on fera [1] GOTO et pour le label [1] LBL.
Explication de [X] GOTO :

Pour GOTO deux situations se présentent : si son label a déjà ét "compilé", il n'y a qu'à compiler le code de BRANCH (E5D99) et calculer la distance qui le sépare du label à l'aide de HERE et de l'adresse du label qui se trouve dans VLBL.

Dans le cas contraire (label non encore "compilé") le GOTO doit déposer sur la pile son numéro caractéristique (X+10) et son adresse, mais attention, pour ne pas gèner les autres structures de contrôle, ces nombres doivent être déposés non au dessus mais en dessous de la pile des données ! GOTO doit aussi compiler le code de BRANCH avec pour l'instant un saut de longueur nulle.

Explication de [X] LBL :

De même pour LBL deux situations se présentent : si LBL a déjà ét appelé, il doit calculer et stocker toutes les distances qui le séparent des GOTO qui lui sont adressés, c'est à dire qu'il doit chercher dans la pile des données tous les numéros caractéristiques qui lui correspondent, une fois ce (ou ces) numéro trouvé (en fait il suffit de tester un nombre sur deux car tous les nombres vont par paire) le nombre d'en dessous sera l'adresse du GOTO et alors il peut calculer les distances dont il a besoin. Mais LBL doit aussi stocker son adresse dans VLBL pour les GOTO potentiels futurs.

Si aucun GOTO n'a appelé le LBL avant qu'il ne soit "compilé", seul la dernière étape subsiste (stockage de l'adresse de LBL, c'est-à-dire HERE, dans VLBL).

Une dernière chose reste à faire si vous ne voulez pas avoir de Memory Lost (je sens que la suite vous intéresse !). Il faut absolument remettre à zéro le contenu de VLBL avant la compilation de mots contenant des GOTO . Pour cela deux solutions sont possibles. Soit vous le faites "à la main" à chaque fois que vous compilez de tels mots, pour cela il suffit d'exécuter le mot suivant : OVLBL VLBL 32 30 NFILL ; IMMEDIATE (en hexa.). Soit et cela me semble préférable, vous redéfinissez : de manière à ce qu'il le fasse systématiquement, vous n'avez qu'à rentrer ce qui suit :

```
: G [COMPILE] : VLBL 32 30 NFILL ; IMMEDIATE -D
' G NFA 2+ +! tout cela en hexa. et en sachant
```

que NFA donne le NFA du mot dont le CFA est sur le dessus de la pile des données.

Voilà, j'en ai fini avec mes GOTO et LBL en FORTH et j'espère bien ne jamais voir dans JPC des mots les contenant !

Alain Herreman.

```
-----------------------------------------------

       LEX    'DEMOCOMP'
       ID     #5C
       MSG    0
       POLL   0

       ENTRY  Demo
       CHAR   15
       KEY    'CSWAP'
       TOKEN  255

*
*
* Ce LEX a été fait pr vs montrer comment
* réaliser une fonction travaillant sur
* des nombres complexes. Il n'est toutefois
* utilisable qu'avec le module MATHS
* (du - tant que ns n'avons pas crée de fonctions
* de substitution).
*  Le principe du LEX est simple: on veut
* réaliser 1 fonction qui inverserait
* les parties réelles et imaginaires d'1 nb
* complexe. Pr ce faire, on prend le
* nombre qui se trouve sur la pile, on regarde si
* c'est un complexe ou un réel
* on inverse parties im. et ré.; il suffit alors
* de mettre le résultât sur la
* pile, et le tour est joué.
*
*

=POP1N  EQU   #0BD1C
=MEMCHK EQU   #012C7
=BSERR  EQU   #0939A
=EXPR   EQU   #0F23C

       ENDTXT

       NIBHEX 811        --- CSWAP( (A,B) ou R )
Demo   GOSBVL =POP1N     Retire le nb de la pile
       GOC    Complx     Est-il complex?
       C=0    W          Non; il faut annuler la
*             pseudo-partie imaginaire ajoutée
       R0=C              par =POP1N
Complx R1=A

*
* A ce moment:
*   Nombre= (R0,R1)
*

       GOC    Swap       Si le nb est complexe il
*             est nécessaire de vérifier s'il
       CD1EX             il y a suffisemment de
*            place sur la pile pr y placer 1
       D=C    A          résultat complexe;
*                        sauvegarde D1 ds D(A)
       A=C    A          Stack pointer -> A(A);
       C=0    A
       P=     0
       LCHEX  12         Longueur prise, sur la
*           pile, par la partie im. et l'en-
*                        tête d'un nb complexe
       GOSBVL =MEMCHK
       CDEX   A          Sauvegarde l'éventuel
*                       numéro d'erreur ds D(A)
       D1=C              Restaure D1
       GONC   Swap       Y a-t-il suffisemment de
*                        place?
       C=D    A          Non; récupére le code
*                        d'erreur
       GOVLNG =BSERR

Swap   A=R1              Partie ré. -> A(W)
       AR0EX               -> R0;
       R1=A              Partie im. -> R1;

*
*
* A ce moment:
*  P=0
*  HEX mode
*  Result=(R1,R0)
*  D1=déjà décrémenté de 16
*    On s'est assuré qu'il y a suffisemment de
*    pace sur la pile
*
*

       A=R1              Partie ré. -> A(W)
       DAT1=A W             -> Stack;
       D1=D1- 16
       A=R0              Partie im. -> A(W)
       DAT1=A W             -> Stack;
       D1=D1- 2
       LCHEX  0E         En tête de nb complexes
       DAT1=C B             -> Stack;
       XM=0              Désarme XM
       GOVLNG =EXPR      Exprime le résultat
       END
```

```
-----------------------------------------------

       LEX    'CONTRAST'
       ID     #01
       MSG    0
       POLL   0

       ENTRY  Lcd
       CHAR   #D
       KEY    'CONTRAST'
       TOKEN  23

*
*
*  L'ordre CONTRAST sert à modifier le contraste
* du 71 tout en voyant le résultat
* de cette modification.
*  2 touches sont actives (et a répétition):
*    + : permet d'augmenter la valeur du
*        contraste jusqu'à un maximum de 15
*    - : permet de diminuer la valeur du
*        contraste jusqu'à un minimum de 0
*
*  2 touches sont inactives:
*     <- et ->

*  Ttes les autres touches font sortir de notre
*  routine. Elles sont ôtées du
* buffer de touches si programme s'éxéwcute;
* laissées sinon.
*
*  Sachez que ce LEX est si pratique qu'il ne
* quitte plus ma machine.
*
*

=OUTELA EQU    #05303
=SCRLLR EQU    #0212E
=DSPCNT EQU    #2E3FE
=BF2DSP EQU    #01C0E
=CRLFND EQU    #0229E
=POPBUF EQU    #010EE
=NXTSTM EQU    #08A48
=RPTKY  EQU    #152BA
=FINDA  EQU    #023E3
=NOSCRL EQU    #14C8A
key+    EQU    #00038
key-    EQU    #0002A

       ENDTXT
```

```
Lcdd   GOVLNG =OUTELA
Lcdp   RTNCC
       REL(5) Lcdd
       REL(5) Lcdp      --- CONTRAST ---
Lcd    GOSUB  Pop       Reprend l'éxécution au
*                       label Pop
       NIBASC 'Set cont'  Début de la chaîne que
*                         l'on va afficher
       NIBASC 'rast:[+ '  .
       NIBASC '- KEY]'    .
       NIBHEX FF        Fin du buffer

Pop    C=RSTK
       D1=C             D1 @ début de la chaine
*                       précédente
       GOSBVL =BF2DSP   On l'affiche...
       GOSBVL =CRLFND   ...Sans delay

Cont   GOSBVL =POPBUF   Prépare la tache à =RPTKY
       GOSBVL =RPTKY    L'action de la touche
*                       doit-elle se répéter?
       A=B    A         (Mets le code de cette
*                       touche en A(A))
       GOC    Cat
       GOSBVL =SCRLLR   Déroule l'affichage;
*             retourne avec le code d'une touche
        *              en A(A)
Cat    GOSBVL =FINDA    Se branche sur une routine
*                       suivant le code en A(A)
       CON(2) key+      Key +
       REL(3) Read+
       CON(2) key-      Key -
       REL(3) Read-
       CON(2) 0         Fin de la table des sauts
       GOSBVL =NOSCRL   RESET NEEDSC
       ?ST#1  13        Retirons le code de la
*                 touche sur laquelle on vient d'
       GOYES  Contex    appuyer si un programme
*                      est en train d'être éxécuté
       GOSBVL =POPBUF   (C'est ce que l'on fait)
Contex GOVLNG =NXTSTM   Sinon Main Loop saura quoi
*                       en faire

Read+  D0=(5) =DSPCNT
       A=DAT0 S         Valeur du contrast en A(S)
       A=A+1  S         Peut-on l'augmenter sans
*                       dépasser 15?
       GOC    Cont      Non
Write  DAT0=A S          Définit la nouvelle valeur
*                       du contraste
       GONC   Cont      (B.E.T)

Read-  D0=(5) =DSPCNT
       A=DAT0 S
```

```
      A=A-1  S       Peut-on diminuer la valeur
*                    du contraste sans risque?
      GOC    Cont    Non
      GONC   Write   (B.E.T.)  Oui


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



       LEX    'SCANLEX'

      TITLE  Utilitaires \ 01/05/86 \ J-J Moreau

       ID     #E1
       MSG    0
       POLL   0

       ENTRY  Entry
       CHAR   #F
       ENTRY  ScanF
       CHAR   #F
       ENTRY  Type
       CHAR   #F

       KEY    'ENTRY$'
       TOKEN  72
*
* ENTRY$("KEYWORD"[,n]):
*   Renvoie l'adr à laquelle commence la routine
*  définissant la n-ième
*  occurence du mot-clé KEYWORD.
*

       KEY    'TOKEN'
       TOKEN  73
*
* TOKEN("KEYWORD"[,n]):
*   Renvoie le TOKEN de la n-iéme occurrence de
*   KEYWORD sous la forme décimale:
*        1000*ID + TK
*


       KEY    'TYPE'
       TOKEN  74
*
* TYPE("KEYWORD"[,n]):
*   Renvoie le type de la n-ième occurrence du
*   mot-clè KEYWORD.
*   Le type d'un mot clef est définit + bas.
*

=REVPOP EQU    #0BD31
=REV$   EQU    #1B38E
=RESPTR EQU    #03172
=EXPR   EQU    #0F23C
=HEXASC EQU    #17148
=STRHDR EQU    #0F09A
=NTOKEN EQU    #0493B
=RESCAN EQU    #04A4C
=A-MULT EQU    #1B349
=HDFLT  EQU    #1B31B
=D1C=R3 EQU    #03047
=FNRTN2 EQU    #0F219
=CSLW5  EQU    #0ED3D
=RNDAHX EQU    #136CB
=ARGERR EQU    #0BF19
tXFN    EQU    #000B3
tXWORD  EQU    #000EF

       ENDTXT



Find   P=C    15       Nb d'arguments de la
*                     fonction dans P
       A=0    A        Mets a zero le paramètre n
*                     (l'occurrence d'un mot-clé)
       ?P=    1        N'y a-t-il qu'un paramètre?
       GOYES  Find1
       GOSBVL =RNDAHX  Non; second paramètre ->
*                     A(A);
       GONC   Argerr   2nd parm <0 IF Argerr
       D1=D1+ 16       D1 @ en-tête de la chaine
*                     à tokenizer
       A=A-1  A        A(A) -> n (l'occurrence)
       GOC    Argerr   2nd parm 0= IF Argerr
Find1  R1=A            Sauvegarde n, qui servira
*                     de compteur, ds R1
       P=     0
       GOSBVL =REVPOP  Mets la chaîne à tokenizé
*                     ds le bon sens
       CD1EX
       D1=C            D1@ 1er char.
       C=C+A  A
       D=C    A        D(A)=(FORSTK)
       GOSBVL =CSLW5   Sauvegarde D0&D1 ds R3
       CD0EX
       D0=C
       R3=C
       ST=1   0        On arme ST0 car on désire
*                  connaitre la type du mot-clé
       GOSBVL =NTOKEN  Mets A(?-0)=TOKEN,
*          B(A)=adr d'éxécution (si elle existe)
Findx  ?ST=1  11       Voulait-on tokenizer 1
*                     variable?
       GOYES  Argerr
       ?A#0   A        Ou bien GO?
       GOYES  Find2    Ni l'un ni l'autre
```

```
Argerr GOVLNG =ARGERR  (Remarquez que =IVAERR
*                      s'écrit: GOVLNG =ARGERR)

Find2  AR1EX           n (notre compteur) -> A(A);
       A=A-1  A        n-1 -> A
       AR1EX           Sauvegarde le nv compteur
       RTNC            Retourne si l'on a fait
*                      n-boucles
       GOSBVL =RESPTR  Sinon restaure divers
*            pointeurs pr que l'on puisse de nv
       ST=1   0        Utiliser =NTOKEN
       A=C    A        Information nécessaire à
*                      =NTOKEN -> C(A);
       GOSBVL =RESCAN  A peu près équivalent à
*                      GOSBVL =NTOKEN
       GOTO   Findx    Boucle


       NIBHEX 8412
ScanF  GOSUB  Find     Tokenize
       ST=0   4


*
*
*   Tokens: Avant et après  transformation
*   FN    :     Tk , 0Tk00       (Function)
*   STMT  :     Tk , 0Tk00      (Statement)
*   XFN   : TkIdB3 , 0TkId      (XFunction)
*   FFN   : TkIdB4 , 0TkId (Funny Function)
*   XSTMT : TkIdEF , 0TkId     (XStatement)
*   WORD  : TkIdEF , 0TkId           (Word)
*
*


       LC(2)  =tXWORD
       ?A=C   B
       GOYES  Schi++   Go if XWORD/XSTMT
       LC(2)  =tXFN
       ?A=C   B
       GOYES  Schi++  Go if XFN
       C=C+1  B
       ?A=C   B
       GOYES  Schif+   Go if FFN
       ASL    A        Ici si FN ou STMT
       A=0    M       Effectue la transformation
*                      décrite + haut
       ASL    A
       GONC   Schift   (B.E.T.)
Schif+ ST=1   4        Indique que l'on tokenize
*                      une FFN
Schi++ ASR    W       Effectue la transformation
*                      décrite + haut
       ASR    A
Schift R0=A            Sauvegarde le TOKEN
       A=0    M
       A=0    XS       A(A)=Id
       C=0    A
       LC(3)  1000
       GOSBVL =A-MULT  A(A)=Id*1000
       C=R0            C(A)=0TkId
       CSR    A
       C=0    M
       CSR    A        C(A)=Tk
       A=A+C  A        A(A)=Id*1000 + Tk
Fnrtn  GOSBVL =HDFLT   A(A)=Resultat en décimal
       B=A    W        Sauvegardons-le ds B(W)
       GOSBVL =D1C=R3  Mets: D1=Stack pointer,
*C(A)=PC
       A=B    W        Resultat -> C(W);
       ACEX   W        PC -> A(A);
       ?ST=0  4        A-t-on tokenizé une FFN?
       GOYES  Pos      Non
       P=     9        Oui; indiquons-le en
*                      renvoyant 1 nb négatif
       CPEX   15
Pos    GOVLNG =FNRTN2


       NIBHEX 8412
Entry  GOSUB  Find     Calcule l'adr d'éxécution
*                      de KEYWORD
       GOSBVL =D1C=R3  Stack pointer -> D1;
       D0=C            PC -> D0;
       A=B    A        Mets l'adr en question ds
*                      A(A)

*
*
*   Ci-suit 1 routine HP; elle convertit 1 nb en
* hexa ds A(A) en 1 chaine
* alphanumérique qu'elle place sur la pile.
*
*


       C=0    W
       P=     4
       CPEX   15       Indicate 4+1 nibs to
*                      convert in ASCII
       D=C    W        Save counter in D(W)
       GOSBVL =HEXASC
       D=D+1  S
       D=D+D  S
       DSLC
       C=D    A        Compute output string
*                      length
       GOSBVL =STRHDR  Prepare header for string
*                      in B(W)
       P=C    0
```

```
        P=P-1           String length-1 to P
        A=B     W
        DAT1=A WP       Write string
        D1=D1- 16       Move to header
        P=      0
        GOSBVL =REV$    Reverse string
        GOVLNG =EXPR

        NIBHEX 8412
Type    GOSUB  Find     Détermine le type de
*                         KEYWORD

*
*
* Type:
*   FN/FFN/XFN/WORD : 0             (En fait F pr
*   1 fonction)
*   STMT/XSTMT      : n  (0<n<F)
*
*


        A=0     A
        C=ST            Type
        A=C     P          -> A(A);
        ST=0    4        Indique qu'on veut 1
*                        résultat positif
        GOTO    Fnrtn


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



      LEX    'CONFIGLX'

      TITLE  Utilitaire \ 01/12/86 \ J-J Moreau

      ID     #E1
      MSG    0
      POLL   Polhnd
      ENDTXT

*
*  Le poll-handler qui suit intercepte le poll
* pCLDST,qui est envoyé après
* chaque MEMORY LOST (Ce qui, tout programmeur en
* assembleur le sait bien,
* arrive des dizaines de fois par jour, si ce
* n'est plus. Je m'arrête là car on
* va me reprocher de vous effrayer). Car il est
* insupportable de devoir mettre
* un contraste de 7, éteindre le sonnette, passez
* en RADIANS/USER, charger
* quelques LEX (Comme REPEAT, CONTRAST...)... le
* tout manuellement. Aussi j'ai
* créée CONFIGLX qui fait tout cela pour moi et
* peut, qui plus est, afficher
* Bonjour: c'est extraordinaire ! En un mot, le
* poll pCONFG est envoyé; CONFIGLX
* l'attrape au vol; éxécute le sous-programme
* BASIC: CONFIG (Merci à Mr Hernest
* Planque !); retourne à l'assembleur; rends le
* bienheureux poll; et voilà mon
* 71 qui me dis bonjour, avec un contraste de
* 7,...
*  Nous n'étonnerons personne en préçisant que le
* sous-programme CONFIG doit
* être ds un port indépendant, ainsi que le LEX:
* CONFIGLX (Ou mieux: ds 1 EEPROM HHP)
*  Un mot encore: sachez que si vs terminez votre
* ss-prgm CONFIG par ENDALL, le
* fameux poll pCONFG ne sera pas envoyé à d'autre
* lexs. Par ailleurs, le
* procédé que j'ai développé ds mon LEX pr le
* poll pCONFG peut-être étendu à
* tous les polls rapides (Pour les autres
* référez-vs aux IDS vol 1 (un peu
* avant le chapitre sur la tokenization)). Sur,
* ce bon MEMORY LOST.
*
*


=CALBIN EQU   #18D8C
=FORSTK EQU   #2F59E
=POPUPD EQU   #08F3E
=PSHUPD EQU   #08F0D
PgmRun  EQU   #0000D
=lPOLSV EQU   #0003E
=pCLDST EQU   #000FF
=tCALL  EQU   #000F9
=tLITRL EQU   #000C4
=tPRMEN EQU   #000F8
=tEOL   EQU   #000F0

Polhnd LC(2)  =pCLDST
       ?B=C   B         Est-ce le poll de départ à
*                       froid ?
       GOYES  Confg
       RTNSXM           Malheureusement non !

Confg  C=D    A         Sauvegarde ce sacré D(A)
*                       ds la pile de retour BASIC
       A=C    A         (C'est celle qu'utilise
*                       les ordres GOSUB et POP)
       GOSBVL =PSHUPD   (C'est cette routine qui
*                      se charge de tt le travail)
       C=RSTK           Sauvegarde l'adr de retour
```

```
*                    vers la routine =FPOLL ds la
      A=C    A       même pile
      GOSBVL =PSHUPD
      C=RSTK         On recommence avec l'adr
*                   de retour à la routine =CLDST
      A=C    A
      GOSBVL =PSHUPD
      ST=1   PgmRun  Puis on allume le drapeau
*                   13 qui indique au routines
       *             internes qu'on éxécute un
*                    programme (BASIC ou BIN)
      GOSBVL =CALBIN Et l'on se branche sur
*                    cette routine qui, à peu de
       *             choses près celle qui est
*                   chargée d'éxécuter, en BASIC
       *             un sous-programme. Ci suit
*                   la description du dit ss-prog
      CON(2) 20      Longueur du chaînage
      CON(2) tCALL   Tokenization de CALL CONFIG
      CON(2) tLITRL   .
      NIBASC 'CONFIG'  .ONFIG
      CON(2) tPRMEN   .
      CON(2) tEOL     .

*
*
*  A ce point du LEX on espere que l'utilisateur
*  n'a pas trop perturbé le 71
* pour que tout aille comme il faudrait (en
* faisant des POP, ou des POKE ...
*
*

      GOSBVL =POPUPD  Remettons ds la pile de
*                   retour (interne) les 2 adrs
      C=D    A        stockées ds la pile de
*                    retour BASIC
      RSTK=C
      GOSBVL =POPUPD  (C'est cette routine qui
*                  fait le travail, car ns sommes
      C=D    A        bien fatigués aujourd'hui)
      RSTK=C
      GOSBVL =POPUPD  Puis on retaure D(A), que
*                   tout bon chasseur de FPOLL se
       *             doit d'attraper puis, en
*                    bon chasseur qu'il est,
       *             remettre intact à sa place)
      C=-C-1 A        Désarmons carry
      RTNSXM          Et renvoyons le poll en
*                   faisant croire que ns n'avons
       *             fait (XM#0) (Un bon
*                   chasseur ne doit pas se faire
       *             remarquer). Disons, +
*                    sérieusement, qu'il ne faut
       *
*                          EN AUCUN CAS
       *                   retourner avec XM=0. Le
*                  poll doit atteindre TOUS les LEX.


*
*
* Petit exemple de ss-prog:
*  Exemple 1:
*     EDIT CONFIG:PORT(0)
*       10 BEEP OFF @ RADIANS @ USER @ CONTRAST 7
*       20 END SUB
*
*  Exemple 2:
*     EDIT TOOTOO:PORT(.05)
*       10 SUB CONFIG @ CONTRAST 5 @ STACK 16
*       15 DISP "Bonjour..."
*       20 RESTORE IO @ COPY REPEAT:TAPE(1)
*       25 COPY TOOOTOOO:HP71
*
*
```

```
      LEX    'FINDLEX'

      TITLE  Utilitaires BASIC
*01/05/86-01/07/86 \ J-J Moreau

      ID     #E1
      MSG    Msgtbl
      POLL   0
      ENTRY  Find
      CHAR   #7
      KEY    'FIND'
      TOKEN  75

*
*
*  L'ordre FIND recherche une chaîne de
* caractères dans le programme BASIC
* courant. La recherche se fait entre la ligne
* qui suit la ligne courante et la
* dernière ligne du programme. Si la chaîne
* recherchée est trouvée, la ligne de
* plus petit numéro qui la contienne est
* affichée; elle devient la ligne
* courante. Dans le cas contraire, on renvoie le
* message d'erreur:
*       "Not Found"
```

```
* La ligne courante n'est pas changée.
*
*  L'ordre FIND a été conçu pour faciliter la
* mise au point de programme BASIC.
* Il est surtout utile à des personnes qui n'ont
* pas l'editeur de textes et/ou
* peu de place dans leur machine.
*
* Exemple d'utililisation de l'ordre FIND. Tapez:
*   EDIT TOTO
*    10 A=B @ DESTROY A @ GOTO 20
*    20 U=V @ PRINT #1;A$[1,97]&C$&"Il pleut"
*   L'instruction: FIND '$&"' affiche la ligne
* 20, car elle il y est ecrit:
*         C$&"Il pleut
*
*


=AVE=D1 EQU   #18B88
=BF2DSP EQU   #01C0E
=BLDDSP EQU   #01898
=BSERR  EQU   #0939A
=CK"ON" EQU   #076AD
=CSLW5  EQU   #0ED3D
=CSRW5  EQU   #0ED2C
=CURRL  EQU   #2F7E8
=CURSRT EQU   #096C1
=DSPCNO EQU   #09716
=EXPEXC EQU   #0F186
=FINDL  EQU   #0FFE4
=GETPRO EQU   #06BEE
=GETSTC EQU   #07726
=LDCM10 EQU   #04F6F
=LDCOMP EQU    #04F69
=LDCSPC EQU   #2F6C1
=MAIN30 EQU   #0037E
=MFERR  EQU   #09393
=NULLP  EQU   #07999
=NXTLIN EQU   #10031
=NXTSTM EQU   #08A48
=OUTBS  EQU   #2F58F
=REVPOP EQU   #0BD31
=S-R1-2 EQU   #2F88B
=S-R1-3 EQU   #2F890
=STREQL EQU   #1B1EF
=STRNGP EQU   #0379D
=eFTYPE EQU   #0003F

        ENDTXT


eNSTRG  EQU   (#5E)*256+1  Not Found
Msgtbl CON(2) 1         Numero du 1er message ds
*                       ce LEX
       CON(2) 1         Numero du dernier message
*                       ds ce LEX
       CON(2) 24        Saut relatif jusqu'au
*                       message suivant
       CON(2) 1         Numero de ce message: 1
       CON(1) 9-1       Longueur du message (en
*                       quartets) -1
       NIBASC 'Not Foun'
       NIBASC 'd'
       CON(1) 12        Marqueur de fin message
       NIBHEX FF        Marqueur de fin de table
*                       de message


Strngp GOVLNG =STRNGP
       REL(5) Strngp     --- FIND "Search$" ---
Find   GOSBVL =EXPEXC  Place la chaîne (que l'on
*                 nommera: search$) sur la pile
       GOSBVL =REVPOP  Inverse l'ordre de ses
*               caractères; mets sa longueur en A
       ?A=0   A        Len(search$)=0 ?
       GOYES  NStrng
       D0=(5) =S-R1-2  Len(search$)
       DAT0=A A          -> S-R1-2;
       CD1EX           Start(search$)
       D1=C
       C=C+A  A
       C=C-1  A
       C=C-1  A         (C @ dernier char. de
*                       search$ )
       D0=D0+ 5
       DAT0=C A        -> S-R1-3;
       GOSBVL =AVE=D1  Actualise AVMEME d'après
*                 D1 (Il ne faut pas détruire
*                    la chaîne que l'on
*              recherche (son nom sera Object$)

       GOSBVL =GETPRO  Le fichier courant est-il
*                      privatisé?
       GOC    Mferr
       GOSBVL =GETSTC  Est-ce un fichier BASIC?
       GONC   Find20
       LC(2)  =eFTYPE
Mferr  GOVLNG =MFERR
Find20 GOSBVL =NULLP   Est-ce un fichier vide?
       GONC   Find30   Non
NStrng  P=     0
       LC(4)  eNSTRG
       GOVLNG =BSERR

Find30 D1=(5) =CURRL   CURRL
       C=0    A
       C=DAT1 4          -> C(A);
       GOSBVL =FINDL   Calcule l'adresse du début
```

```
*                       de la ligne courante
        GONC    NStrng
        CD1EX           C(A) @ #ligne (2 quartets
*                       apres l'EOL (OF) )
        B=C     A       Sauvegarde l'adr de la
*                       ligne courante dans B(A)
        CD1EX           Restore D1
        GOSBVL =NXTLIN  Recherche la ligne qui
*                       suit la ligne courante
        ?C<D    A       Existe-t-elle?
        GOYES   Find15  Oui
        A=B     A       Non; on commence la
*                       recherche à partir de CURRL
        D1=A
Find15  C=D     A       C(A) @ fin du fichier
        GOSBVL =CSLW5
        CD1EX           C(A) @ #ligne
        D1=C            D1 @ #ligne

Findx   R3=C            Sauvegarde Findindex dans
*                       R3
        GOSBVL =LDCM10  Decompile la ligne pointée
*                       par D1

*
*
* A ce moment:
*   (OUTBS) @ debut de la ligne decompilée
*   (c'est: object$)
*   B(A) = len(object$)
*   P = ?
*
*   S-R1-2 = len(search$)
*   S-R1-3 = start(search$)
*   R3(9-5)@ #ligne
*   R3(4-0)@ Find du fichier
*
*


        P=      0
        A=B     A       Len(object$)
        A=A+A   A         -> A(A);
        D0=(5) =LDCSPC  Start(object$)
        C=DAT0  A
        C=C+A   A
        R2=C              -> R2
        D1=C              -> D1;
        D0=(5) =S-R1-3  Start(search$)
        C=DAT0  A
        R1=C              -> R1;
        D0=D0- 5
        C=DAT0  A
*
```

```
*
* Ce qui suit est la réplique de la fonction HP:
* POS.
* On va chercher à savoir si search$ est incluse
* ds object$.
* POS commence par comparer les derniers
* caractères des 2 chaines, et progresse
* vers les adr decroissantes.
*
*

        A=A-C   A       Len(object$)-len(search$)
*                       0> IF Pos5
        GOC     Pos5
        B=0     M
        B=A     A
        BSRB
*     (len(object$)-len(search$))/2 -> Scanindex;
        C=C-1   A       Len(search$)-1
        P=C     0       RMD(Len(search$)-1,16) ->
*                       Partword;
        CSR     A       DIV(Len(search$)-1,16)
        D=C     A         -> Wordcount;
        GONC    Pos3    (B.E.T.)

Pos2    C=B     A
        RSTK=C          Scanindex -> Stack;
        CD1EX
        RSTK=C          Scanpt -> Stack;
        CD1EX
        D0=D0+ 2
        D1=D1+ 2
        C=D     A
        B=C     A       Wordcount -> B;
        GOSBVL =STREQL  Search$ Object$ INCLU IF

        C=RSTK
        D1=C            Scanpt -> D1;
        C=RSTK
        B=C     A       Scanindex -> B;
        GONC    Pos6
        B=B-1   A       Scanindex-1 <0? IF Pos5
Pos3    C=R1            Start(search$)
        D0=C              -> D0;
        GOC     Pos5
        A=DAT0  B       1er char. de search$ ->  A;
Pos4    D1=D1-  2       Scanpt-2 -> Scanpt;
        C=DAT1  B       Object$[Scanpt,Scanpt] ->
*                       C;
        ?A=C    B       IF Pos2 ;
        GOYES   Pos2
        B=B-1   A       Scanindex-1 >=0? IF Pos4;
        GONC    Pos4
Pos5    C=R2
```

```
       D1=C            Start(Object$) -> Scanpt;
Pos6   C=R2
       A=0    W
       A=C    A
       CD1EX           Scanpt -> C;
       A=A-C  A        Start(Object$)-Scanpt -> A;
       ASRB            A/2 -> A;

*
*
* Routine principale:
*   Elle definit la conduite à tenir:
*    1: Afficher une ligne qui contienne search$;
*    2: Passer a la ligne suivante, a condition
*    qu'elle existe;
*    3: Renvoyer un message d'erreur ds le cas
*       contraire
*
*

       P=     0
       C=R3            Findindex -> C;
       D1=C            D1 @ #ligne
       ?A=0   A        POS(Object$,Search$) NOT
*                      IF Find60
       GOYES  Find60

*
*
* Option 1: on affiche une ligne.
*
*

       GOSBVL =LDCOMP  Decompile la ligne @ D1;
*                      elle devient la ligne courante
       D0=(5) =OUTBS
       C=DAT0 A
       D0=C            D0 @ 1er char. de line$
*                      (i.e. object$)
       D1=(5) =LDCSPC
       A=0    M
       A=DAT1 A        A @ l'espace qui suit le
*                      #ligne
       A=A-C  A
       ASRB            Len(line$)-Len(line#)
       R0=A              -> R0;
       C=B    A        2*Len(line$)
       RSTK=C            -> RSTK;
       GOSUB  Find*    Envoie la sequence
*                      d'échappement suivante:
       NIBHEX B1C3
       NIBASC '>'
       NIBHEX B1E3
       NIBHEX FF

Find*  C=RSTK
       D1=C
       GOSBVL =BF2DSP
       C=RSTK          Récupère le # de char.
       B=C    A
       GOSBVL =DSPCNO  Affiche le buffer de sortie
       C=R0            # de fois dont il faudra
*                     déplacer le curseur à droite
       GOSBVL =CURSRT  CURSFL, puis à droite (cf
*                      ci-dessus)
       GOSBVL =BLDDSP  Construit l'affichage
       GOVLNG =MAIN30  Equivalent à: GOVLNG
*                      =SCRLLR

*
*
* Decide laquelle des options 2 & 3 l'on va
* choisir
*
*

Find60 GOSBVL =CSRW5   Récupère l'adr de la fin
*                      du fichier
       D=C    A          -> D;
       GOSBVL =NXTLIN  Essaie de trouver la ligne
*                    suivant celle que l'on vient
       P=     0         d'analiser
       ?C<D   A         Existe-t-elle?
       GOYES  Find70
       GOTO   NStrng     Non; on a donc scruté
*                        ttes les lignes

*
*
* Option 2:
*   On actualise les pointeurs et l'on recommence
*
*

Find70 C=R3            Findindex -> C;
       CD1EX           Actualise
*                      Findindex[Findpt,Findpt]
       R3=C            Sauvegarde Findindex ds R3
*                      jusqu'au prochain test
       GOSBVL =CK"ON"
       C=R3
       D1=C            D1 @ #ligne
       GOC    Findx+   N'a-t-on pas appuyé sur la
*                      touche ATTN?
       GOVLNG =NXTSTM  Si; il faut arrêter la
*                 rechercher, allumer l'indicateur
*                      SUSP, ce qui sera très
*                 bien fait par la routine =NXTSTM
Findx+ GOTO   Findx    Mais si tout va bien
```

NDLR : Ce fichier LEX ne respecte pas les messages précédemment écrits, ni la règle du premier message.

D'autre part, le fichier source qui nous a été transmis n'était pas sur 50 colonnes, ce qui explique l'aspect parfois négligé de la présentation.

```
      LEX    'ENDUPLEX'

      CON(2) #E1       Id
      CON(2) 16        Lowest Token
      CON(2) 19        Highest Token
      CON(5) 0
      NIBHEX F         No speed table
      REL(4) 1+TxTbSt  Text table offset
      CON(4) 0         Message table offset
      CON(5) 0         Poll handler offset

* Main table

      CON(3) (TxEn01)-(TxTbSt)
      REL(5) endst
      NIBHEX F

      CON(3) (TxEn02)-(TxTbSt)
      REL(5) endup
      NIBHEX D

      CON(3) (TxEn04)-(TxTbSt)
      REL(5) start
      NIBHEX F

      CON(3) (TxEn03)-(TxTbSt)
      REL(5) exec
      NIBHEX D

* Text table

TxTbSt
TxEn01 NIBHEX B
      NIBASC 'ENDUP$'
      CON(2) 16
TxEn02 NIBHEX 9
      NIBASC 'ENDUP'
      CON(2) 17
TxEn03 NIBHEX D
      NIBASC 'EXECUTE'
      CON(2) 19
TxEn04 NIBHEX F
      NIBASC 'STARTUP$'
      CON(2) 18

      NIBHEX 1FF      End of text table


*
* Exécute une ligne BASIC. EXECUTE fut écrit pr
* rendre programmable des
* fonctions non programmables.

* Exemple d'utilisation:
*   EDIT TOTO
*     10 DISP 'Running...'
*     20 EXECUTE "FREE:PORT(.01) @ PURGE ABC
*                 CONTRAST 7 @ RUN TOTO,30"
*     30 DISP 'Done'
*


=ADHEAD EQU   #181B7
=D=AVMS EQU   #1A460
=DELAYd EQU   #05493
=EXPEXC EQU   #0F186
=I/OALL EQU   #1197D
=I/ODAL EQU   #11A41
=I/OFND EQU   #118BA
=I/ORES EQU   #118FF
=LINEP+ EQU   #02626
=MEMERR EQU   #0944D
=MOVEU3 EQU   #1B177
=NXTSTM EQU   #08A48
=REVPOP EQU   #0BD31
=SFLAG? EQU   #1364C
=SFLAGS EQU   #135FA
=STKCHR EQU   #18504
=STRNGP EQU   #0379D
=STROVF EQU   #1411A
=XXHEAD EQU   #1A44E
=bSTART EQU   #00808
=flPWDN EQU   #000CF
=flTNOF EQU   #000CD
=pCONFG EQU   #000FB
=pPWROF EQU   #000FC
bENDUP  EQU   #00A00
bXEQ    EQU   #00A01


Polhnd LC(2)  =pCONFG
       ?B=C   B        Une reconfiguration
```

```
*                       est-elle en cours?
        GOYES   savebf
        LC(2)   =pPWROF
        ?B=C    B       La machine souhaite-elle
*                       s'endormir?
        GOYES   xeq
        RTNSXM          Non, rien de tout cela

out     C=B     A       Ces lignes, placées ici pr
*                       gagner de la place,
        D=C     A       restaurent le registre
*                       D(A) si chers aux FPOLL
savebf  LC(3)   bENDUP
        GOSBVL  =I/ORES Evite à bENDUP d'être
*                detruit à la prochaine reconfigu.
        LC(3)   bXEQ
        GOSBVL  =I/ORES Idem pr bXEQ
rtncc   A=-A-1  A       Désarme carry
        RTNSXM

xeq     LC(3)   bENDUP
        GOSBVL  =I/OFND Mets D1 @ le début du
*                     buffer (après l'en-tête)
        GONC    rtncc   (Du moins s'il existe)
        C=D     A       D(A)
        B=C     A         -> B(A);
        LC(2)   =flPWDN
        GOSBVL  =SFLAG? flPWDN est-il éteint?
        GONC    out
        LC(2)   =flTNOF
        GOSBVL  =SFLAG? flTNOF est-il allumé?
        GOC     out
        LC(2)   =flTNOF
        GOSBVL  =SFLAGS Allumons flTNOF que l'on
*                 n'intercepte pas pPWROF 1 2nd *
linep   CD1EX           D1 -> C(A) (D1 est
*                       inchangé depuis =I/OFND)
        GOVLNG  =LINEP+ Compile la ligne BASIC @
*                       par C(A), puis l'éxécute



strngp  GOVLNG  =STRNGP
delayd  GOVLNG  =DELAYd
        REL(5)  delayd
        REL(5)  strngp   --- ENDUP "BASIC line" ---
endup   GOSBVL  =EXPEXC Mets la ligne à éxécuter
*                       sur la pile
        C=0     A
        LC(2)   2*95+2  Longueur max de cette
*                       ligne + 2 quartets pr 0D
        R3=C              -> R3;
        LC(3)   bENDUP
        GOSUB   end007  Stocke la ligne BASIC ds
*                       bENDUP

nxtstm  GOVLNG  =NXTSTM

        REL(5)  delayd
        REL(5)  strngp   --- EXECUTE "BASIC line"
exec    GOSBVL  =EXPEXC Mets la ligne sur la pile
        C=0     A
        C=C-1   A       Longueur max de la ligne
*                       (65536) -> C(A)
        R3=C              -> R3;
        LC(3)   bXEQ
        GOSUB   end007  Stocke la dite ligne ds
*                       bXEQ
        GONC    nxtstm  Va là-bas si elle ne
*                       contient aucune instruction
        LC(3)   bXEQ
        GOSBVL  =I/OFND Mets D1 @ début de la ligne
        GOTO    linep   Exécute là


end007  R2=C            Sauvegarde l'ID du buffer
*                       de stockage
        GOSBVL  =XXHEAD Enlève l'en tête de la
*                       chaîne placée sur la pile
        LCHEX   0D
        D1=D1- 2
        DAT1=C B        Ajoute 1 CR derrière la
*                     ligne (il faudra se rappeller
        ST=1    0       qu'elle n'est pas
*                       tokenizée)
        GOSUB   adhead  Ajoute un nouvel en tête
        GOSBVL  =REVPOP Puis inverse l'ordre des
*                       caractères de la chaîne
        CD1EX           Pointeur de pile -> C(A)
        R0=C              -> R0;
        C=0     A
        C=C+1   A
        C=C+1   A       2 -> C(A);
        ?A>C    A       La ligne est-elle composée
*                       d'autre chose que d'un CR?
        GOYES   end010
        C=R2            Non; ID -> C(A);
        GOSBVL  =I/ODAL Détruit le buffer dont
*                       l'ID est ds C(A)
        RTNCC           Indique que le buffer
*                       n'existe plus

end010  B=A     A       Longueur de la ligne ->
*                       B(A);
        C=R3            Longueur max autorisée ->
*                       C(A);
        ?A<=C   A       La ligne n'est-elle pas
*                       trop longue?
        GOYES   end040  Non;
        GOVLNG  =STROVF Si.
```

```
end040 R1=A            Longueur de la ligne -> R1;
       C=R2            ID -> C(A);
       GOSBVL =I/OALL  Contracte ou agrandit le
*                    buffer pr qu'il ait la taille
       GOC    end050   spécifiée en A(A)
       GOVLNG =MEMERR  Erreur s'il n'y a pas
*                      suffisemment de place
end050 A=R0
       D0=A            D0 @ début de la ligne (ss
*                      forme tokenizée)
       C=R1            Longueur de la ligne -> R1;
       GOSBVL =MOVEU3  Déplace la ligne de la
*                      pile ds le buffer
       RTNSC           Indique que le buffer
*                      existe


       NIBHEX 00        --- STARTUP$ ---
start  LC(3)  =bSTART
       GONC   run      (B.E.T.)

       NIBHEX 00        --- ENDUP$ ---
endst  LC(3)  bENDUP
run    AD1EX           Sauvegarde D1 ds R1
       R1=A
       GOSBVL =I/OFND
       CD1EX
       CDOEX           D0 @ ce qui suit l'en-tête
       B=C    A        Sauvegarde ce pointeur ds
*                      B(A)
       GOSBVL =D=AVMS  Mets (AVMEMS) ds D(A)
       A=R1
       D1=A            D1 = (FORSTK)
       GONC   bf2s20   Va si la chaîne contenue
*                   ds le buffer a 1 longueur nulle
bf2s10 A=DAT0 B        Prends 1 char. ds le buffer
       LCHEX  0D
       ?A=C   B        En a-t-on atteint la fin?
       GOYES  bf2s20   Non
       C=A    B        Oui
       GOSBVL =STKCHR  Mets le car. sur la pile;
*                  vérifie qu'il y a suf de place
       D0=D0+ 2        Passe au char. suivant
       GONC   bf2s10   (B.E.T.)

bf2s20 C=B    A
       D0=C
adhead GOVLNG =ADHEAD  Ajoute un en-tête a la
*                      chaîne.
```

NDLR : Certaines chaînes ne sont pas exécutables avec l'ordre EXECUTE. Par exemple, essayez EXECUTE "BEEP"...

```
       LEX    'SECURELX'
       CON(2) 1
       CON(2) 79
       CON(2) 79
       REL(5) SEC/U    Comme les TOKENs de SECURE
*                    et de UNSECURE ne se suivent
       NIBHEX F        pas, on ne peut les
*                inclure ds le même LEX; mais comme
       REL(4) (Tx00)+1 ces 2 ordres utilisent les
*                       mêmes routines, il faut
       CON(4) 0        chaîner les LEX qui les
*                contiennent, ce qui ne peut pas
       CON(5) 0        faire en utilisant les
*                    pseudos-po habituelles (M.M.)

       CON(3) 0
       REL(5) Secure
       CON(1) 13
Tx00   CON(1) 11
       NIBASC 'SECURE'
       CON(2) 79
       NIBHEX 1FF

=BSERR  EQU   #0939A
=CNFFND EQU   #109AC
=CURRST EQU   #2F55D
=EOLCK  EQU   #02A7E
=EOLXCK EQU   #05405
=FILSK+ EQU   #06F1D
=FINDF+ EQU   #09F63
=FSPECe EQU   #02F02
=FSPECp EQU   #03CC5
=FSPECx EQU   #09F2D
=LOCADR EQU   #0A611
=MAINST EQU   #2F558
=NXTSTM EQU   #08A48
=POLL   EQU   #12337
=PURGDC EQU   #05745
=RAMROM EQU   #0A5F7
=RESPTR EQU   #03172
=ROMCID EQU   #00BFE
=ROMFND EQU   #1102F
=S-R0-0 EQU   #2F871
=S-R0-1 EQU   #2F876
=S-R0-2 EQU   #2F87B
=S-R0-3 EQU   #2F880
=eDVCNF EQU   #00040
=eFACCS EQU   #0003C
=lFNAMh EQU   #00010
```

```
=lFTYPh EQU   #00004
=oFLENh EQU   #00020
=pFPROT EQU   #0000B
=tALL   EQU   #000F8
=tKEYS  EQU   #000CF

       ENDTXT


*
*
* Routine de compilation des nouveaux ordres
*  UN/SECURE:
*  UN/SECURE [Nom de fichier][:<dev id>]/ALL/keys
*
*

Secp   GOSBVL =EOLCK   t(UN)SECURE suivit d'un
*                      EOL?
       GONC   Secp10   Non
Resptr GOVLNG =RESPTR  Oui; alors il n'y a rien à
*                      rajouter

Secp10 GOSUB  Resptr   Restaure D1
       GOSBVL =FSPECp  t(UN)SECURE suivit d'un
*                 Nom de fichier et/ou d'un type
       RTNNC           de mémoire?
       LC(2)  =tALL    Non; peut-être alors de
*                      ALL?
       ?A=C   B
       GOYES  Secp11   Si oui, laissons le
*                 système finir la décompilation
       LC(2)  =tKEYS   Ne serait-ce pas KEYS
*                      plutôt que ALL?
       ?A#C   B
       GOYES  Fspece   Non; l'utilisateur a donc
*                      écrit n'importe quoi
Secp11 RTNCC           Non; laissons le système
*                      agir

Fspece GOVLNG =FSPECe  (En définitive ns sommes
*                 plus rusé que l'utilisateur)


Purgdc GOVLNG =PURGDC



       STITLE Exécution de (UN)SECURE
************************************************
************************************************
**
** Nom:    Secure - Execute l'ordre SECURE
**          Uecure - Execute l'ordre UNSECURE
**
** Classe:   STEXEC
**
** But:      Execute l'ordre (UN)SECURE
**
** Contitions d'entrée:
**
**           D0 pointant après tSECURE ou
**           tUNSECURE
**           P=0
**
** Etat à la sortie:
**           via NXTSTM
**
** Registres/RAM utilisée(s):
**           A-D,D1,D0,R0-R3
**           STMTR0 (En entier), S0-S2,S6-S8,S11
**
** Ss-Pgm:
** CNFFND,EOLXCK,FILSK+,FINDF+,FSPECx,POLL
**           ROMFND
**
** Nb de registres de la pile de retour utilisés:
** 6
**
** Détails:    SECURE [Nom de fichier][:<dev
**                                 id/ALL/keys
**             UNSECURE [Nom de fichier][:<dev
**                                    id>]/ALL/keys
**
** Application(s) futures:
**      La modification de la seule routine Sec95
**      permettra d'étendre la
**      nouvelle syntaxe de (UN)SECURE à:
**                        EN/DISABLE, PURGE, HELP...
**      Les drapeaux S3-S6,S9-S10 sont inutilisés.
**
** Création: 17/01/86 J-J Moreau   D'aprés la
** routine CAT de N.N.
**
************************************************
************************************************

       REL(5) =Purgdc
       REL(5) =Secp
Secure ST=0   11       ST11=0 uniquement si l'on
*                        éxécute l'ordre SECURE
Sec-1  A=DAT0 B        Pour un commentaire
*             détailée de cette routine, qui n'a
       GOSBVL =EOLXCK  que peu d'intérêt en
*                 elle-même (On n'y fait que du
       GONC   Sec05    calcul) reportez-vs aux
*                      IDS vol.3 (Fonction CAT).
       D1=(5) =CURRST  Vs y trouverez un
```

```
*                    commentaire fort détaillé.
       C=DAT1 A
       D1=C
Sec00  GOSUB  Sec95    (Remarquez que l'on
*              n'arrive ici que s'il n'y a qu'un
       GOTO   Nxtstm   seul fichier à
*                      (dé)sécuriser)

Sec05  P=     15
       LCHEX  E
       P=     0
       D1=(5) =S-R0-3
       DAT1=C S
       LC(2)  =tALL
       ?A=C   B
       GOYES  Sec39
       C=0    S
       DAT1=C S
       GOSBVL =FSPECx
       GONC   Sec38
Bserr  GOVLNG =BSERR

Sec38  GOSBVL =FINDF+
       GONC   Sec00
       ?ST=1  6
       GOYES  Bserr
       ?C=0   S
       GOYES  Bserr
       C=D    S
       C=C-1  S
       GONC   Sec40
Sec39  GOTO   Sec10

Sec40  C=C-1  S
       GOC    Sec70
       GOSBVL =POLL
       CON(2) =pFPROT
       GOC    Bserr
       ?XM=0
       GOYES  Nxtstm
       D=D+1  S
       D=D+D  S
       GOC    Fspec+
       ?A#0   W
       GOYES  Sec42
Fspec+ GOTO   Fspece
Sec42  C=0    A
       LC(2)  =eFACCS
       GONC   Bserr    (B.E.T.)

Sec70  D=D+1  B
       GONC   Sec82
       D1=(5) =S-R0-3
       DAT1=C S
       GOSUB  Romchk
       GONC   Sec90
Sec34  C=0    A
       LC(2)  =eDVCNF
       GOTO   Bserr

Sec82  GOSUB  Romf-1
       GOC    Sec34
       CD1EX
       GOTO   Sec11


Secal?
Secalp D1=(5) =S-R0-3
       C=DAT1 S
       ?C#0   S
       GOYES  Sec84
Nxtstm GOVLNG =NXTSTM

Sec84  C=C+1  S
       GONC   Seccnt
Sec85  GOSBVL =ROMFND
Sec87  GOC    Nxtstm
Sec90  A=DAT1 B
       ?A=0   B
       GOYES  Sec85
       CD1EX
       GOTO   Sec11

Seccnt DAT1=C S
       GOSUB  Romchk
       GOTO Sec87

Sec10  D1=(5) =MAINST
       C=DAT1 A
Sec11  D1=(5) =S-R0-1
       DAT1=C A
       GOTO   Eoflc+
Eoflch A=C    A
       GOSBVL =FILSK+
Eoflc+ D1=C
       A=DAT1 B
       ?A#0   B
       GOYES  Eoflch

       D1=(5) S-R0-2
       DAT1=C A
       D1=(5) S-R0-1
       A=DAT1 A
       D0=A
       C=DAT0 B
       ?C=0   B
       GOYES  Secal+
       D1=D1- 5
```

```
Sec17  DAT1=A A
       D1=A
       GOSUB  Sec95
       D1=(5) S-R0-0
       A=DAT1 A
       C=0    A
       LC(2)  =oFLENh
       A=A+C  A
       D0=A
       A=DAT0 A
       CD0EX
       A=A+C  A
       D0=A
       C=DAT0 B
       ?C#0   B
       GOYES  Sec17
Secal+ GOTO   Secalp

*
*
*  Voici la routine qui se charge de
* (dé)sécuriser un fichier. C'est elle que
* l'on aura soin de modifier si l'on veut étendre
* le domaine d'action de
* certaines ordres comme ENABLE... (Cf ci-dessus)
*  On n'a en fait que 3 choses à savoir pr
* pouvoir modifier ce LEX:
*    1-  S3-S5, S9-S10 ne sont utiliser ni par
* Sec-1 ni par Sec95
*    2- D1 @ l'en-tête du fichier traité
*    3-  R0-R3 ne doivent pas être modifiés
*
*

Sec95  CD1EX          C(A) @ en-tête du fichier
*                     traité
       GOSBVL =LOCADR Détermine quel type de
*                     mémoire le contient
       GOSBVL =RAMROM Est de la mémoire vive
*                     (RAM ou IRAM)?
       GOC    Sec96
       GOTO   Nxtstm
Sec96  D1=D1+ lFNAMh  D1 @ champ S de l'en-tête
       D1=D1+ lFTYPh    .
       A=DAT1 B       (B sert à gagner de la
*                     place)
       LCHEX  1
       ?ST=0  11      Secure?
       GOYES  SecSe
       LCHEX  E       Enlève la protection du
*                     fichier
       A=A&C  P
       GONC   SecDn   (B.E.T.)
SecSe  A=A!C  P       Protège le fichier

SecDn  DAT1=A P
       RTN


Romchk P=     0       (D'après 1 routine HP)
       LC(2)  =ROMCID
       GOSBVL =CNFFND
       ?A=0   X
       RTNYES
       D1=D1+ 1
       B=A    X
       C=DAT1 8
       D=C    W
       D1=D1+ 3
       C=DAT1 3
       D1=D1+ 3
       C=DAT1 S
       C=C+1  S
       D=C    S
       D1=D1+ 4
       CSL    A
       CSL    A
       LCHEX  8
       AD1EX
       D1=C
       ASL    W
       ASL    W
       ASL    W
       A=B    X
       R1=A
       RTNCC


Romf-1 D=D-1  B
       C=D    A
       R3=C
       GOSUB  Romchk
       RTNC
       GONC   Romf-3  (B.E.T.)
Romf-2 GOSBVL =ROMFND
       RTNC
Romf-3 C=R3
       ?C#D   B
       GOYES  Romf-2
       RTNCC



*
*
* 2ème partie du LEX
*
*
```

```
SEC/U  CON(2) 1
       CON(2) 90
       CON(2) 90
       CON(5) 0          Plus aucun LEX n'est
*                        chainée à celui-ci
       NIBHEX F
       REL(4) (Tx01)+1
       CON(4) 0          Aucune table de messages
       CON(5) 0          Aucun routine
*                        d'interception de polls

       CON(3) 0
       REL(5) Uecure     Point d'entrée de l'ordre
*                        UNSECURE
       CON(1) 13
Tx01   CON(1) 15
       NIBASC 'UNSECURE'
       CON(2) 90
       NIBHEX 1FF

       REL(5) Purgdc
       REL(5) Secp
Uecure ST=1   11         ST11=1 lorsque l'on
*                        éxécute l'odre UNSECURE
       GOLONG Sec-1      Utilise la même routine
```

DATES

Suite à l'article de Laurent Istria (JPC numéro 28) à propos des dates, voici le même LEX légèrement retouché, auquel j'ai ajouté l'opération DATE+, une limite supérieure pour la date, ainsi qu'une erreur (Out of Range) quand les limites supérieures ou inférieures sont atteintes.

Cette limite supérieure est le 25 Novembre 2870, ce qui correspond à un nombre de jours machine égal à (hex FFFFF). Il est possible d'aller plus loin mais cela compliquerait le prgm. D'ici l'an 2870, notre cher Titan sera devenu une pièce de musée de l'ère préinformatique ! Si vous désirez faire des prévisions à très long terme, vous pouvez toujours me contacter !

L'utilisation de DATE+ se fait ainsi :

DATE+(D1,N) , D1 étant la date de "départ" exprimée dans le format choisi, N étant le nombre de jours à ajouter ou à retrancher, selon le signe de N.

Comment ais-je écris la routine DATE+ ? Tout simplement en utilisant la routine DAYYMD qui ne semble pas "bugger". En effet, un examen attentif de SETDATE, DATE et DATE$ permet de remarquer que si l'on ne travaille qu'avec des dates alphanumériques, il n'y a pas de problèmes, ce qui me laisse supposer que la routine boguée est YMDHMS qui est la routine qui renvoie la date sous la forme YYMMM.

François LE GRAND

NDLR : Actuellemen DATE+ écrase KBD qui précéde ce lex dans numérotation. Nous sollicitons vos avis pour ces questions de numérotation. Faut-il, par exemple supprimer KBD, le renuméroter ?

```
       LEX    'DATELX'
       ID     #E1
       MSG    msgtbl
       POLL   0
BSERR  EQU    #0939A
CSLC7  EQU    #1B42F
FLTDH  EQU    #1B223
DAYYMD EQU    #13335
ADHEAD EQU    #181B7
ARGERR EQU    #0BF19
CSLC4  EQU    #1B438
D=AVMS EQU    #1A460
FNRTN4 EQU    #0F238
HDFLT  EQU    #1B31B
IDIVA  EQU    #0EC6E
NXTSTM EQU    #08A48
OUTELA EQU    #05303
POP1N  EQU    #0BD1C
SFLAG* EQU    #135F3
SFLAG? EQU    #1364C
TBLJMC EQU    #02426
YMDDAY EQU    #13304
flDATE EQU    -27
mOoR   EQU    #0E101
       ENTRY  PLUS
       CHAR   #F
       ENTRY  DELTA
```

```
        CHAR    #F
        ENTRY   EUR
        CHAR    #D
        ENTRY   WEEK$
        CHAR    #F
        ENTRY   WEEK
        CHAR    #F
        ENTRY   AME
        CHAR    #D
        KEY     'DATE+'
        TOKEN   51
        KEY     'DDAYS'
        TOKEN   52
        KEY     'DMY'
        TOKEN   53
        KEY     'DOW$'
        TOKEN   54
        KEY     'DOW'
        TOKEN   55
        KEY     'MDY'
        TOKEN   56
        ENDTXT
msgtbl  CON(2)  1
        CON(2)  1
edate   EQU     00
        CON(2)  16
        CON(2)  00
        CON(1)  4
        NIBASC  'DATE '
        CON(1)  12
eoor    EQU     01
        CON(2)  31
        CON(2)  01
        CON(1)  11
        CON(1)  11
        NIBASC  'Out of R'
        NIBASC  'ange'
        CON(1)  12
        NIBHEX  FF
        NIBHEX  8822
PLUS    GOSBVL  POP1N
        GOSBVL  FLTDH
        ST=0    1
        GOC     AA
        ?XM=0
        GOYES   CB
arg2    GOTO    argerr
CB      ST=1    1
        A=-A    A
AA      R0=A
        D1=D1+  16
        GOSUB   FLTDB0
        GOSUB   sflag?
        C=B     A
        A=R0
        ?ST=0   1
        GOYES   BA
        ?A>C    A
        GOYES   arg2
        C=C-A   A
        GOTO    BC
BA      C=C+A   A
BC      GOSUB   TEST2
        GOSBVL  DAYYMD
        ?ST=0   0
        GOYES   AB
        BCEX    A
        CDEX    A
        BCEX    A
AB      ST=0    0
        C=D     W
        P=      1
        ?C=0    P
        GOYES   AC
        ST=1    0
AC      CSLC
        CSLC
        C=C+B   W
        GOSBVL  CSLC4
        C=C+A   W
        GOSBVL  CSLC7
        ?ST#0   0
        GOYES   AD
        CSLC
        C=C-1   X
AD      C=C+1   X
        GOTO    FIN2
        NIBHEX  811
WEEK$   ST=1    2
        GOTO    WEEKE
        NIBHEX  8822
DELTA   GOSUB   FLTDB0
        D1=D1+  16
        R0=C
        GOSUB   FLTDB0
        A=R0
        ?A>C    W
        GOYES   POS
        C=C-A   W
        A=C     W
        GOSBVL  HDFLT
        A=-A-1  S
        GOTO    FIN
POS     A=A-C   W
        GOSBVL  HDFLT
FIN     C=A     W
FIN2    GOVLNG  FNRTN4
        NIBHEX  811
```

```
WEEK    ST=0    2
WEEKE   GOSUB   FLTDBO
        A=C     W
        A=A-1   A
        C=0     A
        LCHEX   7
        GOSBVL  IDIVA
        ?ST=1   2
        GOYES   DOWTXT
        CSRC
        C=0     M
        C=0     XS
        CSRC
        GOTO    FIN2
DOWTXT  D1=D1+  16
        AD1EX
        R1=A
        AD1EX
        GOSBVL  TBLJMC
        REL(3)  DIM
        REL(3)  LUN
        REL(3)  MAR
        REL(3)  MER
        REL(3)  JEU
        REL(3)  VEN
        REL(3)  SAM
LUN     LCASC   'Lundi'
        P=      9
        GOTO    M10
MAR     LCASC   'Mardi'
        P=      9
        GOTO    M10
MER     LCASC   'Mercredi'
        P=      15
        GOTO    M16
JEU     LCASC   'Jeudi'
        P=      9
        GOTO    M10
VEN     LCASC   'Vendredi'
        P=      15
        GOTO    M16
SAM     LCASC   'Samedi'
        P=      11
        D1=D1-  12
        GOTO    STORE
DIM     LCASC   'Dimanche'
        P=      15
M16     D1=D1-  16
        GOTO    STORE
M10     D1=D1-  10
STORE   DAT1=C  WP
        GOSBVL  D=AVMS
        ST=0    0
        P=      0
        GOVLNG  ADHEAD
sflag?  P=      0
        LC(2)   flDATE
        SETHEX
        ST=0    0
        GOSBVL  SFLAG?
        GONC    S0
        ST=1    0
S0      RTN
FLTDBO  GOSBVL  POP1N
        B=A     W
        GOSUB   sflag?
        C=B     W
        ?C=0    P
        GOYES   S1
        CSLC
S1      CSLC
        CSLC
        D=C     W
        P=      1
        C=0     WP
        D=D-C   W
        CSLC
        CSLC
        B=C     W
        C=0     WP
        B=B-C   W
        GOSBVL  CSLC4
        A=C     W
        P=      3
        C=0     WP
        A=A-C   W
        ?ST=0   0
        GOYES   NON
        BCEX    A
        CDEX    A
        BCEX    A
NON     C=0     A
        P=      0
        LCHEX   12
        ?B>C    B
        GOYES   argerr
        LCHEX   07
        ?B>C    B
        GOYES   >JUIL
        LCHEX   2
        ?C=B    B
        GOYES   FEVR
        C=C-1   P
        C=C&B   P
        P=      1
        LCHEX   3
        GOTO    TEST
>JUIL   R1=A
```

```
        LCHEX  1
        A=C    P
        C=C&B  P
        C=A-C  P
        A=R1
        P=     1
        LCHEX  3
TEST    ?D>C   B
        GOYES  argerr
        GONC   TERM
FEVR    LCHEX  29
        ?D>=C  B
        GOYES  PROB
        GONC   TERM
argerr  GOVLNG ARGERR
PROB    ?D>C   B
        GOYES  argerr
        C=B    B
        R1=C
        R2=A
        C=0    A
        LCHEX  4
        SETDEC
        GOSBVL IDIVA
        ?B#0   A
        GOYES  argerr
        A=R2
        ?A#0   B
        GOYES  REST
        LCHEX  400
        GOSBVL IDIVA
        ?B=0   A
        GOYES  REST
        GOTO   argerr
REST    C=R1
        B=C    B
        A=R2
TERM    GOSBVL YMDDAY
TEST2   CAEX    W
        C=0    W
        P=     0
        LCHEX  #8D235
        ?A<C   W
        GOYES  AZ
        C=0    W
        C=C-1  A
*       LCHEX  #FFFFF
        CAEX   W
        ?C<=A  W
        RTNYES
AZ      C=RSTK
range   LC(4)  mOoR
        GOVLNG BSERR
        REL(5) WDC
        REL(5) WP
EUR     LC(2)  flDATE
        GOTO   SORTIE
        REL(5) WDC
        REL(5) WP
AME     LC(2)  flDATE
        P=     1
SORTIE  GOSBVL SFLAG*
        GOVLNG NXTSTM
WDC     GOVLNG OUTELA
WP      RTNCC
```

PROGRAMMATION STRUCTUREE

Le fichier LEX que je vous propose ce mois ci a pour but de conférer au Basic l'allure d'un langage un peu plus au goût du jour (comme Pascal) en le munissant des structures de programme WHILE <expression> ... END WHILE et REPEAT ... UNTIL <expression>. Ces mots définissent des blocs de code dont l'exécution dépend de la valeur de <expression> comme on le verra plus loin.

Il vous sera désormais possible d'écrire, par exemple:

```
WHILE KEYDOWN @ DISP A$ @ END WHILE
```

Ce qui est fonctionnellement equivalent à:

```
10 DISP A$ @ IF KEYDOWN THEN 10
```

cependant la première formulation est plus claire et rend non significative la numérotation des lignes.

Les mots fournis par ce LEX sont:

WHILE <expression>
Si <expression> vaut 0 on poursuit l'exécution à l'instruction suivant END WHILE (bloc non exécuté), sinon on continue en séquence.

END WHILE
Marque la fin du bloc d'instructions. Il y a retour au WHILE correspondant pour évaluer à

nouveau <expression>.

REPEAT
Marque le début du bloc.

UNTIL <expression>
Si <expression> vaut 0, on retourne après REPEAT pour executer à nouveau le bloc. On observe donc que ce bloc est exécuté au moins une fois alors que dans le cas de WHILE, le bloc peut ne pas être exécuté du tout.

LEAVE
Permet de sortir (définitivement) d'un bloc. Après exécution de LEAVE on se retrouve après le END WHILE ou le UNTIL correspondant au bloc où LEAVE est executé. Par exemple:

```
10 WHILE ...
20 IF .. THEN LEAVE
30 REPEAT ... UNTIL ...
40 END WHILE <- on se retrouve ici après LEAVE.
```

Note: dans la description qui précède, <expression> représente une expression numérique valant 0 (faux) ou autre chose (vrai) - cf IF.

Je n'essaierai pas d'expliquer par le menu le fonctionnement de l'ensemble, ceci serait fastidieux; quelques remarques cependant:

1- Les messages d'erreur sont les suivants:

'WHILE' (225003) devrait être 'no WHILE', malheureusement, le premier message d'erreur d'un LEX doit avoir une longueur (totale) multiple de 16 nibbles, ceci explique ce message sibyllin.
'Invalid LEAVE' (225005) est émis quand il n'y a pas eu de début de bloc (WHILE ou REPEAT) ou lorsqu'il n'y a pas de fin de bloc (END WHILE ou UNTIL).
'no END WHILE' (225006) et 'no REPEAT' (225004) s'expliquent d'eux memes.

2- Pour assurer le fonctionnement de l'ensemble des adresses sont stockées sur la "GOSUB stack", ceci interdit qu'un RETURN situé à l'intérieur d'un bloc renvoie à un GOSUB extérieur à celui-ci. En d'autres termes le code suivant:

```
10 GOSUB 100
20 END
100 A=0
100 WHILE 1 ! bloc se repetant indéfiniment
120 A=A+1 @ IF A=5 THEN RETURN
130 END WHILE
```

ne permet pas de revenir en ligne 20 (on obtient le message 'RETURN w/o GOSUB'. Ce comportement est conforme aux règles de programmation structurée qui imposent une seule entrée et une seule sortie pour un bloc de code. Si quelqu'un me démontre qu'il vaut mieux retourner "proprement" je suis prêt à étudier les modifications requises.

3- Ces mots réagissent parfaitement en mode TRACE FLOW (ce n'est pas le plus simple...).

4- De même qu'il n'est pas possible d'avoir des boucles FOR .. NEXT 'mal' imbriquées, il n'est pas possible d'avoir des blocs WHILE .. END WHILE ou REPEAT .. UNTIL se chevauchant. Par exemple:
WHILE ... REPEAT ... UNTIL ... END WHILE est correcte,
tandis que:
WHILE ... REPEAT ... END WHILE ... UNTIL ne l'est pas.

Pour conclure, je souhaite remercier P. David pour l'idée d'utiliser la 'GOSUB stack' (ce qui est beaucoup plus efficace que mes idées antérieures) et pour son debugger - dont l'utilisation pour ce LEX a été déterminante - ainsi que J.J. Dhénin pour son aide dans la mise au point.

```
        LEX     'STRUC1'
        TITLE   Structured programming statements

tENDW   EQU     66
tWHILE  EQU     67
tREPT   EQU     68
tUNTIL  EQU     69
tLEAVE  EQU     70

        CON(2)  LEXID     ID equivalent
        CON(2)  tENDW
        CON(2)  tLEAVE
        CON(5)  0
        NIBHEX  F
        REL(4)  1+TxTbSt
```

```
       MSG    ermsgs
       POLL   0

*
*M A I N   T A B L E
*
       CON(3) (TxEn01)-(TxTbSt)
       REL(5) ENDW
       CON(1) 13

       CON(3) (TxEn02)-(TxTbSt)
       REL(5) WHILE
       CON(1) 13

       CON(3) (TxEn03)-(TxTbSt)
       REL(5) REPT
       CON(1) 13

       CON(3) (TxEn04)-(TxTbSt)
       REL(5) UNTIL
       CON(1) 13

       CON(3) (TxEn05)-(TxTbSt)
       REL(5) LEAVE
       CON(1) 13

*
*T E X T   T A B L E
*

TxTbSt
TxEn01
       CON(1) 5
       NIBASC 'END'
       CON(2) tENDW

TxEn05
       CON(1) 9
       NIBASC 'LEAVE'
       CON(2) tLEAVE

TxEn03
       CON(1) 11
       NIBASC 'REPEAT'
       CON(2) tREPT

TxEn04
       CON(1) 9
       NIBASC 'UNTIL'

       CON(2) tUNTIL
TxEn02
       CON(1) 9
       NIBASC 'WHILE'
       CON(2) tWHILE

       NIBHEX 1FF

LEXID  EQU    #E1
tXWORD EQU    #EF
WHILEt EQU    9
REPTt  EQU    10
eNOWHL EQU    3
eNORPT EQU    4
eLEAVE EQU    5
eINVLD EQU    236
oFTYPh EQU    #10
fBASIC EQU    #0E214
eSYSER EQU    #17
bSTMT  EQU    #801

STMTDO EQU    #2F891
STMTD1 EQU    #2F896
PRGMEN EQU    #2F567
TRACEM EQU    #2F7B0
CURRST EQU    #2F55D

WRDSCN EQU    #02C2A
FIXP   EQU    #02A6E
TKSCN7 EQU    #08A99
FIXDC  EQU    #05493
REST*  EQU    #03035
EOLXC* EQU    #052EC
OUTNBC EQU    #05423
POPUPD EQU    #08F3E
EXPEXC EQU    #0F186
POP1N+ EQU    #0BD91
PSHGSB EQU    #08F13
NXTSTM EQU    #08A48
BSERR  EQU    #0939A
MFERR  EQU    #09393
RUNRT1 EQU    #074E7
STOPDC EQU    #05303
TRFROM EQU    #0FE59
TRTO+  EQU    #0FE7B
I/OFND EQU    #118BA
EOLSCN EQU    #08AA7
MGOSUB EQU    #1AF01

ermsgs CON(2) eNOWHL
       CON(2) eLEAVE

bNOWHL CON(2) (fNOWHL)-(bNOWHL)
       CON(2) eNOWHL
       CON(1) 4
       NIBASC 'WHILE'
       CON(1) #C
```

```
fNOWHL
bNORPT CON(2) (fNORPT)-(bNORPT)
       CON(2) eNORPT
       CON(1) 8
       NIBASC 'no REPEA'
       NIBASC 'T'
       CON(1) #C
fNORPT

bLEAVE CON(2) (fLEAVE)-(bLEAVE)
       CON(2) eLEAVE
       CON(1) 14
       CON(2) eINVLD   'Invalid ' building block
       CON(1) 4
       NIBASC 'LEAVE'
       CON(1) #C
fLEAVE
       NIBHEX FF

       STITLE END WHILE code
       REL(5) ENDWd
       REL(5) ENDWp
ENDW   CD0EX
       D0=(5) (=STMTD1)
       DAT0=C A
       CD0EX
       GOSUB  popupd
       GOC    ENDW01
       P=     15
       LC(1)  WHILEt
       P=     0
       ?C=D   S        compare with our type
       GOYES  ENDW02   Ok
ENDW01 LC(2)  eNOWHL
myerr  P=     2
       LC(2)  LEXID
       P=     0
       GOVLNG =BSERR
ENDW02 ?ST=1  13
       GOYES  ENDW04   from prgm
       LC(3)  bSTMT    from kbd, find buffer
       GOSBVL =I/OFND
       B=A    A        buffer length
       CD1EX           A[A] buffer start
       ?D<C   A        addr < buffer start
       GOYES  ENDW01
       C=C+B  A        A[A] past end of buffer
       ?D>=C  A
       GOYES  ENDW01
ENDW04 C=D    A        get expression adress
       D0=(5) (STMTD0)
       DAT0=C A        STMTD0 @ before expression
       CD0EX           D0 @ expression
       GOSUB  eval
       GOC    ENDW10
       D0=(5) (STMTD1)
       C=DAT0 A
       CD0EX
       GOTO   runrt1

ENDW10 CD0EX
       R2=C            save D0 after expression
*                      in R2
       D0=(5) (=STMTD0) push address before
*                      expression on GOSUB stack
       A=DAT0 A        .
       P=     15       .
       LC(1)  WHILEt   .
ENDW11 GOSUB  pshgsb   .
       C=R2            restore D0 after expression
       CD0EX
       ?ST=0  15       in TRACE mode
       GOYES  runrt1   no
       GOSUB  trflck   check if trace active
       GOC    runrt1   no
       CD0EX
       R2=C
       GOSBVL =TRFROM  FROM part of trace
       C=R2
       D0=C
       GOSBVL =TRTO+   TO part of trace
       A=R2
       D0=A            restore D0 and exit
runrt1 GOVLNG =RUNRT1

trflck ST=0   10       from mainframe @ 0FE18
       ?ST=0  15
       RTNYES
       ?ST=0  13
       RTNYES
       ?ST=1  10
       RTNYES
       D1=(5) TRACEM
       P=     0
       LCHEX  2
       A=DAT1 B
       A=A&C  P
       A=A-1  P
       D1=(5) =CURRST
       C=DAT1 A
       D1=C
       D1=D1+ (oFTYPh)-1
       A=DAT1 A
       ASR    A
       LC(5)  =fBASIC
       ?A=C   A
       GOYES  clrcy
       RTNSC
```

```
clrcy  RTNCC
       STITLE WHILE code
       REL(5) WHILEd
       REL(5) WHILEp
WHILE  GOSUB  saveD0   save D0 before expression
*                      in STMTD0
       CD0EX
       GOSUB  eval
       GONC   WHIL20   condition false
       CD0EX
       R2=C
       D0=(5) (STMTD0) get start of expr address
       A=DAT0 A
       P=     15
       LC(1)  WHILEt
       GOSUB  pshgsb
       C=R2            reset D0 for proper exit
       CD0EX
nxtstm GOVLNG =NXTSTM

* The condition is false, we have to find the END
* WHILE matching
* our WHILE. We have a counter in R2[A]
* incremented for a WHILE
* and decremented for END WHILE.

WHIL20 C=0    A        prepare counter
       C=C+1  A        R2[A] incremented for our
* WHILE
       R2=C
       ?ST=0  13       from keyboard
       GOYES  WHIL05
       D1=(5) PRGMEN   find end of program
       C=DAT1 A
       GOTO   WHIL06
WHIL05 LC(3)  bSTMT    find end of buffer
       GOSBVL =I/OFND
       CD1EX
       C=C+A  A
WHIL06 D=C    A        store end of scan in D[A]
WHIL21 P=     0
       LC(2)  tXWORD
       GOSBVL =TKSCN7
       GOC    WHIL22   if found
       LC(2)  #17
       GOVLNG =MFERR
WHIL22 D0=D0+ 2
       LC(4)  (tWHILE)*#100+(LEXID)
       P=     3
       A=DAT0 WP       read token stream
       ?A#C   WP
       GOYES  WHIL23   not a WHILE
       C=R2
       C=C+1  A        count one more WHILE
       R2=C
       GOTO   WHIL30
WHIL23 P=     2
       LC(2)  tENDW
       P=     3
       ?A#C   WP
       GOYES  WHIL30   not END WHILE
       C=R2
       C=C-1  A        record it is END WHILE
       ?C=0   A        was it ours ?
       GOYES  WHIL40   yes !
       R2=C
WHIL30 GOSUB  eol      go to tEOL or t@
       GOTO   WHIL21   go search again

WHIL40 D0=D0+ 4
       GOTO   ENDW10   process trace & go to
*                      nxtstmt

* Evaluate expression @ D0 returns with carry
* clear if 0
* use everything, D0 @ end of expression on rtn

eval   GOSBVL =EXPEXC
       GOSBVL =POP1N+
       SETHEX
       P=     14
       GONC   eval01
       ?A#0   WP
       GOYES  eval02
       A=R0
eval01 ?A#0   WP
       GOYES  eval02
       RTNCC
eval02 RTNSC

* go to end of line with D0 @ tXWORD

eol    D0=D0- 4        back to length byte
       A=0    A        go to end of this stmt
       A=DAT0 B        line length
       B=A    A
       ADOEX
       A=A+B  A        points to EOL or @
       D0=A
       RTN

pshgsb GOVLNG =PSHGSB
popupd GOVLNG =POPUPD

saveD0 CD0EX
       D0=(5) (STMTD0)
       DAT0=C A
       RTN
```

```
        STITLE REPEAT ... UNTIL code
        REL(5) WHILEd   parse & decompile = WHILE
        REL(5) WHILEp
UNTIL   CD0EX
        D0=(5) (=STMTD1)
        DAT0=C A
        GOSUB  popupd
        GOC    UNTI01
        P=     15
        LC(1)  REPTt
        P=     0
        ?C=D   S        compare with our type
        GOYES  UNTI02   Ok
UNTI01  LC(2)  eNORPT
        GOTO   myerr
UNTI02  ?ST=1  13
        GOYES  UNTI04   from prgm
        LC(3)  bSTMT    from kbd, find buffer
        GOSBVL =I/OFND
        B=A    A        buffer length
        CD1EX           A[A] buffer start
        ?D<C   A        addr < buffer start
        GOYES  UNTI01
        C=C+B  A        A[A] past end of buffer
        ?D>=C  A
        GOYES  UNTI01
UNTI04  C=D    A        get expression adress
        D0=(5) (STMTD0)
        DAT0=C A        save address after REPEAT
        D0=D0+ 5        D0 @ STMTD1
        C=DAT0 A
        CD0EX           D0 @ expression
        GOSUB  eval
        GONC   UNTI10
        GOTO   runrt1
UNTI10  D0=(5) (STMTD0) restore address after
*                       REPEAT
        A=DAT0 A
        R2=A
        P=     15
        LC(1)  REPTt
        GOTO   ENDW11   perform trace and return
*                       to Basic

        REL(5) REPTd
        REL(5) REPTp
REPT    AD0EX
        P=     15
        LC(1)  REPTt
        GOSUB  pshgsb
        CSRC
        CD0EX
        GOTO   nxtstm
```

```
        STITLE LEAVE statement
        REL(5) REPTd
        REL(5) REPTp
LEAVE   GOSUB  saveD0   save D0 in STMTD0
        GOSUB  popupd
        GOC    LEAV01
        P=     15
        LC(1)  REPTt
        ?C=D   S
        GOYES  LEAV02
        LC(1)  WHILEt
        ?C=D   S
        GOYES  LEAV02
LEAV01  LC(2)  eLEAVE
        GOTO   myerr

* We use R2[A] as a counter, incremented by
* REPEAT or
* WHILE and decremented by UNTIL or END WHILE.

LEAV02  C=0    A        init. compteur
        C=C+1  A
        R2=C
        ?ST=0  13       from keyboard ?
        GOYES  LEAV05   yes
        D1=(5) PRGMEN   get program end
        C=DAT1 A        .
        GOTO   LEAV06
LEAV05  LC(3)  bSTMT    from kbd, find buffer
        GOSBVL =I/OFND
        CD1EX
        C=C+A  A
LEAV06  D=C    A        store end of scan in D[A]
        D0=(5) (=STMTD0)
        C=DAT0 A
        CD0EX           recover D0
LEAV21  P=     0
        LC(2)  tXWORD
        GOSBVL =TKSCN7
        GOC    LEAV22   if found
        LC(2)  eLEAVE
        GOTO   myerr
LEAV22  D0=D0+ 2
        LC(4)  (tENDW)*#100+(LEXID)
        P=     3
        A=DAT0 WP       read token stream
        ?A=C   WP
        GOYES  LEAV30   @ END WHILE
        P=     2
        LC(2)  tUNTIL
        P=     3
        ?A=C   WP
        GOYES  LEAV30   it is UNTIL <expr>
```

```
       P=     2
       LC(2)  tREPT
       P=     3
       ?A=C   WP
       GOYES  LEAV33   it is REPEAT
       P=     2
       LC(2)  tWHILE
       P=     3
       ?A=C   WP
       GOYES  LEAV33   it is WHILE
LEAV32 GOSUB  eol      go to end of this line DO
*                      @ tEOL or t@
       GOTO   LEAV21
LEAV33 C=R2
       C=C+1  A
       R2=C           update counter
       GOTO   LEAV32
LEAV30 C=R2
       C=C-1  A
       R2=C           update counter
       ?C#0   A
       GOYES  LEAV32   not finished
       GOSUB  eol      DO @ tEOL or t@ before
*                      next stmt
       GOTO   ENDW10   process trace & go to next
*                      stmt
       STITLE Parse/decompile routines

REPTd  GOVLNG =STOPDC
REPTp  RTNCC

ENDWp  GOSBVL =WRDSCN
       CON(2) =tXWORD  try WHILE after END
       CON(2) =LEXID
       CON(2) tWHILE
       REL(3) ENDWp1   if ok
       CON(2) 00
       P=     0        otherwise, try other ENDs
       GOVLNG =REST*
ENDWp1 DO=DO- 6        if it is END WHILE do not
*                      output WHILE
       RTNCC
ENDWd  LCASC  'ELIHW' display WHILE
       P=     9
       GOSBVL =OUTNBC
       GOSBVL =EOLXC*  we should be at EOL

WHILEp GOVLNG =FIXP    thanks HP
WHILEd GOVLNG =FIXDC
       END
```

ASSEMBLER SANS BRUIT

Je comptais intituler cet article:
Assembler sans effort
Mais, pris d'un soudain remords, j'ai pensé qu'il valait mieux changer mon titre, car je voyais déjà des centaines de lettres de protestation arrivées à la rédaction du journal, P.David chancelant, P.Guez exténué, M.Martinet abattu, J-J Dhénin écoeuré, tout cela par la faute d'un imb.... qui avait trouvé un mauvais titre, un titre mensongeur, un individu ignoble, lequel avait voulu faire accroire aux JPCiliens qu'on pouvait assembler sans effort, alors même que ce journal ne cessait de nous prouver le contraire, à chaque page que nous lisions! Non, devant une telle vision je ne pus qu'affirmer: repends-toi, pendant qu'il en est encore temps!

Mais, me direz-vous , et vous auriez raison, pourquoi:
Assembler sans bruit ?
Parce qu'avec le LEX que je vous propose , vous pourrez décider d'assembler avec ou sans bruit, selon que vous validerez la sonnette d'un vigoureux:
BEEP ON
Ou bien que, plein de rage, vous lui ferrez un mauvais sort:
BEEP OFF
J'en entends qui se disent encore que ces id... de programmeurs en assembleur ne sont bons qu'à écrire des co... . Ils ont tort! Non seulement mon LEX vous permet d'assembler avec ou sans bruit, qui plus est il vous permet d'assembler sans effort (Sic).

En effet, il se propose de rémédier au principal défaut de l'assembleur HP, fort utile par ailleurs : entendez-bien, je ne parle pas de sa lenteur, à laquelle, de guerre lasse, on finit par s'habituer, mais de l'impossiblité de créer un fichier-témoin (listing-file) qui ne comportât que des messages d'erreurs. M.Martinet à beau nous dire que:
"[quoiqu'il fasse] il [lui] reste très souvent [...] beaucoup de place"
(Cf JPC 30 Page 4), il n'en est pas toujours de même pour nous. Aussi je disais assembler sans effort, car il est désormais possible

d'assembler un gros fichier (pour la machine), de partir manger pendant que notre 71 travaille puis, après une bonne sieste, de de réveiller Titan qui s'était endomi, et de lui faire dire quelles erreurs il a faites en assemblant. Peut-être que la lecture de l'article coloré qui précède ne vous engage pas à me croire, je puis vous assurer que c'est bien là la tâche de mon LEX!

Je tiens à remercier ici S.Vaudenay, qui m'a montré comment il me fallait procéder (Cf JPC 28 Page 36), et J-J Dhénin, pour l'utilisation abondante que j'ai pu faire de son matériel. L'idée qui est à la base du LEX est fort simple: lorsque l'assembleur veut reporter une erreur, il utilise une routine qui est à peu près celle-ci:

```
" MSG$(xxxxx)" BASIC$
```

L'assembleur reporte donc toujours une erreur, quellequ'elle soit, à l'aide de la fonction BASIC : MSG$ (LEX EDLEX, JPC 30 Page 56). Or MSG$ émet un poll, poll qu'il nous suffit donc d'intercepter pour envoyer le message d'erreur, non plus vers l'afficheur ni vers un hypothétique fichier-témoin, mais vers un fichier que nous aurons crée.

Une fois le poll intercepté, il faut s'assurer que c'est bien l'assembleur qui a utilisé la fonction MSG$. Pour cela, nous commençons par vérifier qu'il existe en début de mémoire centrale un fichier dénommé FORTHRAM, de type FORTH, et qui semble correctement positionné (Cf IMS du FORTH Page 50). Si c'est le cas, nous décrétons que ce fichier a bien été crée par un des ordres:

```
FORTH ou FORTHX
```

Puis nous regardons l'état de la variable FORTH: ACTIVE; si elle se trouve avoir -2 pour valeur , nous pouvons supposer avec raison que l'ordre MSG$ a été appellé depuis le FORTH par le mot FORTH: BASIC$. Reste à savoir si c'est l'assembleur qui a exécuté BASIC$. Les IMS du FORTH nous apprennent que l'assembleur utilise pour ses besoins personnels un tampon (Sic !) de 1332 quartets de long (Cf IMS du FORTH Page 301). Nous considérerons l'existence d'un tel tampon comme suffisante pour prouver que l'assembleur est en pleine marche. Dès lors nous pouvons accomplir notre mission. Une difficulté s'élève: nous voulons manipuler un fichier (Notre fichier-témoin), ce qui nous amènera à faire se mouvoir force de choses dans la mémoire, dont certaines sont utilisées par l'assembleur. Il nous suffirait de réajualiser les pointeurs contenus dans le tampon dont nous avons parlé plus haut; mais nous n'avons pas réussi une telle prouesse. Aussi, pour que les dits pointeurs n'aient pas à être réactualisés, placerons-nous notre fichier en fin de mémoire centrale. L'assembleur créeant lui même 2 fichiers, il nous faut créer le notre après qu'il ait crée les siens. S'il existait déjà nous courrons à la catastrophe (Essayez !).

Publicité: JPC c'est épicé !

Il nous faut donc être sur que lorsque l'assemblage commence, il n'existe aucun fichier qui ait le même nom que le notre, savoir MASERT. La meilleure solution consisterait donc à amener notre LEX à détruire un éventuel fichier MASERT. C'est impossible, rien ne signale, à aucun moment, qu'un assemblage va avoir lieu. Reste donc à redéfinir le mot FORTH: ASSEMBLE comme suit:

```
: ASSEMBLE " MASERT" FFIND SWAP IF DROP " PURGE
'MASERT'" BASICX THEN ;
```

Mais cela oblige à réinitialiser la FORTHRAM après chaque Memory Lost, ce qui ne me semble pas une bonne solution. Devra-t-on détruire le fichier manuellement ? Pour ma part, je vous propose de redéfinir la touche 'f6' selon la technique utilisée par F. Le Grand (Cf JPC 29 Page 22):

```
DIM A$[100]
A$='"  SFLAG -1 @ PURGE MASERT @ CFLAG -1"
BASICX " '
DEF                                    KEY
'f6',CHR$(27)&''&CHR$(27)&'Q'&A$&CHR$(27)&'"
ASSEMBLE'&CHR$(27)&'R'  ;/ \Je ne comprends pas
comment cela peut marcher (Faites maintenant:
FETCH KEY 'f6'
```

le résultât est assez amusant). Dès lors, il vous suffira pour assembler de passer en FORTH, de taper le nom du fichier à assembler, puis d'appuyer sur touches 'f6' et [end/line] (Ex: TOTO [f6] [end/line]).

Il me reste à vous dire que la fin du LEX n'est plus qu'une phase de calcul (Cf LEX MASERLEX ds ce même journal), aussi je vous y renvoie pour de plus amples commentaires. Sachez enfin que si la variable FORTH: LISTING contient le nom d'un fichier, alors mon LEX ne fera rien; qu'il ne fera rien non plus si l'option:

```
LIST OFF
```

Est active; que si il arrive qu'il ait à reporter une erreur, il le fera (Je pense qu'alors l'assemblage devrait prendre fin);sachez encore que ce LEX émet un bruit, pour indiquer qu'une erreur à eu lieu lors de l'assemblage; sachez aussi qu'il ne quitte plus mon 71.

Voici la fin de cet article. J'espère ne pas avoir été trop long, et suffisemment clair.

Au plaisir de vous lire.

Jean-Jacques MOREAU (SIG#? PC#149 CRTF#1)


```
      LEX    'MASERLEX'

      TITLE  Utilitaire pr l'assembleur
*            jj Moreau - <860202-860214>

      ID     #E1      PARIS-ROM
      MSG    0
      POLL   Polhnd
      ENDTXT

=ACTIVE EQU   #2FB0C
=ASRW5  EQU   #0ED0A
=AVMEME EQU   #2F599
=AVMEMS EQU   #2F594
=BSERR  EQU   #0939A
=CHIRP  EQU   #0EC5A
=CRTF   EQU   #116C1
=CSLW5  EQU   #0ED3D
=D0=AVS EQU   #09B2C
=D0ASCI EQU   #09833
=D=AVME EQU   #1A476
=F-R1-2 EQU   #2F8B5
=FILEF  EQU   #09FB0
=FILSK+ EQU   #06F1D
=FthFil EQU   #0E218
=I/OFND EQU   #118BA
=MAINST EQU   #2F558
=MEMERR EQU   #0944D
=MFERR  EQU   #09393
=MGOSUB EQU   #1AF01
=OUTBS  EQU   #2F58F
=POPUPD EQU   #08F3E
=PRGFMF EQU   #0A146
=PSHUPD EQU   #08F0D
=RPLLIN EQU   #013F7
=SWPBYT EQU   #17A24
=TBMSG$ EQU   #099AB
=eEOFIL EQU   #00036
=oFLAGh EQU   #00014
=oFTYPh EQU   #00010
=oLSTNG EQU   #2FC4D
=oVARID EQU   #2FC43
=sEOF   EQU   #00007
=sBADRC EQU   #00008
=pTRANS EQU   #000EF
PgmRun  EQU   #0000D


* Interceptons le poll pTRANS.

Polhnd LC(2)  =pTRANS  Recherche-t-on 1
*                      traducteur?
       ?B=C   B
       GOYES  Rep010
Repend RTNSXM

* N'interceptons le poll pTRANS que si:
*   le fichier FORTHRAM est en place;
*   il est de type FORTH;
*   et correctement placé;
*   si MSG$ à été appellée par le biais de BASIC$;
*   et si l'assembleur tourne.

Rep010 GOSUB  Fth$?    N'a-t-on pas été appellé
*                      par BASIC$?
       GONC   Repend
       GOSUB  Asrun?   L'assembleur n'est-il pas
*                      en action?
       GOC    Repend

* Sauvegardons ds la pile de retour BASIC:
*   D(A);
*   R3(4-0) (D1);
*   R3(9-5) (D0);
*   RSTK(0) (->=FPOLL);
*   RSTK(1) (->=MSG$);
```

```
        C=D    A        Sauvegardons D(A)
        GOSUB  pshup+
        C=RSTK          L'adr de retour à =FPOLL
        GOSUB  pshup+
        C=RSTK          L'adr de retour à =MSG$
        GOSUB  pshup+
        A=R3            Le D1 utilisé par MSG$
        GOSUB  pshupd
        A=R3            Le D0 utilisé par MSG$
        GOSBVL =ASRW5
        GOSUB  pshupd

* Signalons notre passage par un bref message
* sonore

        GOSBVL =CHIRP   BEEP !

* Plaçons notre message ds le tampon de sortie:
*   construisons le message dont le n est ds
*   R0(A);
*   ajoutons-lui ' , in line ';
*   puis le n de ligne courant du fichier source;
*   ajoutons enfin 1 octet supl. si nécessaire
*   (LIF1).

        GOSBVL =D0=AVS  D0 @ AVMEMS
        D0=D0+ 4        Saute le futur en-tête de
*                       la ligne
        C=R0            N message -> C(A);
        D1=(5) =F-R1-2
        DAT1=C A                    -> S-R1-2.
        GOSBVL =TBMSG$  Place le msg à partir de
*                       (AVMEMS)+4
        GOSBVL =D=AVME  (AVMEME) -> D(A).
        C=B    A        Pointeur de fin de msg (1
*                       quartet après)
        D0=C              -> D0;
* Ajout de ' , in line '
        D0=D0+ 16       Assurons ns qu'il y a
*              suffisemment de place pr ajouter:
        D0=D0+ 6        ' in , line '.
    *   CD0EX             -> D0.
        ?C<D  A         Y-a-t-il suffisemment de
*                       place?
        GOYES  Rep030
        GOVLNG =MEMERR

Rep030 LCASC  'il ni , ' : msg&' , in li'
        DAT0=C W
        D0=D0+ 16
        LCASC  ' en'    : msg&' , in li'&'ne '
        P=     5
        DAT0=C WP
        D0=D0+ 6
        P=     0
* Ajout du n de ligne
        C=0    A        Trouve le n de ligne
*                       stocké ds VAR (Cf IMS du FORTH)
        LC(2)  35
        GOSUB  assbuf
        B=A    A        N ligne -> B(A).
        B=B-1  A        (En fait l'assembleur est
*                     si pressé qu'il indique déjà
*                      le n de la ligne
*                    suivante; d'où l'instr. B=B-1 A)
        LCHEX  F4       Convertissons le n de
*                     ligne en décimal, puis en ASCII,
        GOSBVL =D0ASCI  supprimons les 0 inutiles,
*                     et ajoutons le tt à la suite
        C=0    A          du message, en prenant
*                    soin d'éliminer tt déchet inutile
        DAT0=C B        (à cause du format LIF1)
        CD0EX           Sauvegardons le nouveau
*                       pointeur de fin de msg
        B=C    A          -> B(A).
        GOSBVL =D0=AVS  D0 @ AVMEMS
        D1=(5) =OUTBS
        DAT1=C A        : LET
* (OUTBS)=(AVMEMS)=Début de la ligne qu'il faudra
*                       mettre ds MASERT.
        D0=C            Pointeur de début de ligne
*                       -> C(A) puis D0.
        D0=D0+ 4        Pointeur de début de msg
*                       -> D0 puis C(A).
        CD0EX
        B=B-C  A          Longueur du msg=(Pt de fin
*                     de msg+1)-(Pt de début de msg)
        BSRB              Longueur en octets
        A=B    A          En fait 1 double
        A=A+1  A          L'arondit au 1er entier
*                       pair, par valeur supérieure
        ASRB              .
        A=A+1  A          (Ajoute 4 quartets pr
*                       l'en-tête)
        A=A+A  A          .
        A=A+A  A          Convertit en quartets
        D1=D1+ (=AVMEMS)-(=OUTBS) LET (AVMEMS)=1
*                     quartet après la fin du msg
        CD0EX             .
        C=C+A  A          .
        DAT1=C A          .
        A=B    A        Récupère le double de la
*                       longueur du  msg
        GOSBVL =SWPBYT  La met ss format LIF1
        D0=D0- 4        D0 @ AVMEMS (c'est à dire
*                       l'en-tête de la ligne)
        DAT0=A 4        Ecrit l'en-tête
```

```
* A ce point on a fabriqué 1 ligne de fichier
*    TEXT contenant:
*    msg&' , in line '&n ligne
* Cette ligne se trouve ds le tampon de sortie,
* c'est à dire entre (OUTBS) et
* (AVMEMS). Il faut maintenant calculer l'adr. de
* la fin du fichier MASERT, et
* déplacer la ligne du tampon de sortie vers le
* fichier.

Rep100 LCASC  '  TRESAM' Nom du fichier
       A=C    W          -> A(W).
       GOSBVL =FILEF     Existe-t-il?
       GONC   Rep500
       D=0    W          Créons-le en mémoire
*                        centrale
       C=0    W          de longueur nulle
       R0=C
       R2=C
       R3=C
       C=C+1  A          et de type TEXT
       R1=C
* Création du fichier TEXT MASERT.
       GOSBVL =CRTF
       GOC    Reperr     Reporte 1 éventuel manque
*                        de place
       GONC   Rep100     (B.E.T.) Cette fois-ci ns
*                        allons trouver le fichier!

* Trouvons la fin du fichier MASERT.

Rep500 A=0    A          Recherchons la dernière
*                    ligne de MASERT, c'est à dire
       A=A-1  A          celle qui n'existe pas
*                        (elle a pr n -1)
* Initialisation des registres pr POSTXT.
       R1=A              -1 -> R1(A).
       AD1EX             Pointeur de début
*                        d'en-tête de fichier
       R3=A                -> R3(A).
       GOSUB  POSTXT     Trouvons la ligne -1
       GONC   Rep510     Tout va bien
       ?C#0   A          Est-ce 1 erreur sérieuse
*                        (Ni EOF ni EOD)?
       GOYES  Reperr
Rep510 ?ST=1  sBADRC     Le fichier MASERT est-il
*                        corrompu?
       GOYES  Repbad
       C=0    A          0 -> R3(A) : on insert 1
*                        ligne à la fin du fichier
       CR3EX             Pt de début de fichier ->
*                        C(A).
       AD1EX             Pt de fin de fichier ->
*                        A(A).

* Transférons la ligne du tampon de sortie à la
* fin du fichier MASERT.

       GOSBVL =MGOSUB    Déplace la ligne,
*                réactualise certains pointeurs...
       CON(5) =RPLLIN
       GONC   Repext     Tout va bien
       GOC    Reperr     (B.E.T.)

Repbad C=0    A          Génère 1 l'erreur : 'Fin
*                        de fichier'
       LC(2)  =eEOFIL
Reperr GOVLNG =BSERR

* Remettons-tt en place:
*   RSTK(1);
*   RSTK(0);
*   R3(9-5);
*   R3(4-0);
*   D(A)   .

Repext GOSUB  popupd    Restaure R3(9-5)
       GOSBVL =CSLW5
       R3=C
       GOSUB  popupd    Restaure R3(4-0)
       C=R3
       C=D    A
       R3=C
       GOSUB  popupd    Restaure RSTK(1)
       RSTK=C
       GOSUB  popupd    Restaure RSTK(0)
       RSTK=C
       GOSUB  popupd    Restaure D(A) (Sic)
      D1=(5) =F-R1-2    Restaure R0 (#msg)
       C=DAT1 A
       R0=C
       RTNSXM           Rends finalement la main


pshup+ A=C    A
pshupd GOVLNG =PSHUPD
popupd GOSBVL =POPUPD
       C=D    A
       RTNCC

* Recherche du buf utilisé par l'ass.:
* CC=>NtFnd;CS=>Ok
* Déplacement de D1 d'après C(A)

assbuf B=C    A          Sauvegarde C(A)
       D1=(5) =oVARID    Recherche du buf
       C=DAT1 A
```

```
       GOSBVL =I/OFND
       RTNNC
       C=0    A
       LCHEX  534       Si ce buf à cette
*      longueur, c'est que l'ass. doit tourner
       ?A=C   A
       GOYES  assbu8
       RTN
assbu8 AD1EX            Déplace D1
       A=A+B  A
       D1=A
       A=DAT1 A         Lit la valeur de la var.
*                       pointée par D1
       RTNSC


* Voyons si c'est BASIC$ qui ns a appelé.

Fth$? D1=(5) =MAINST
       C=DAT1 A
       D1=C
       A=DAT1 W
       LCASC  'MARHTROF'
       ?A=C   W         Le 1er fichier a-t-il pr
*                       nom FORTHRAM?
       GOYES  Fth010
       RTNNC            (B.E.T.)
Fth010 D1=D1+ 16
       A=0    A
       A=DAT1 4
       LC(5)  (=FthFil)+1
       C=C-A  A         Est-il de type:
       C=C-1  A           FORTH?
       GOC    Fth020
       C=C-1  A           FORTH sécurisé?
       RTNNC            Non; désarmons carry

Fth020 D1=D1+ 16
       D1=D1+ 5
       A=DAT1 A
       D1=(5) =MAINST
       C=DAT1 A
       ?A=C   A         Est-il à sa place?
       GOYES  Fth030
       RTNNC            (B.E.T.)
Fth030 D1=(5) =ACTIVE
       C=DAT1 A
       C=C-1  A
       C=C-1  A
       C=C-1  A         Armons carry si la valeur
*                       de ACTIVE est -2
       RTN              Désarmons carry sinon

* Déterminons si l'ass. tourne ou non.

Asrun? C=0    A
       GOSUB  assbuf
       GONC   Asset     Pas de buf. => pas d'ass.
       C=DAT1 A
       C=C-1  A
       C=C-1  A       Ne faisons rien lors de la
*                     1ère passe (Sic)
       RTNC
       C=0    A
       LC(2)  65
       GOSUB  assbuf
       A=A-1  A         Lors de la 2nd, ne faisons
*                       rien non plus si l'option
       RTNC             LIST OFF est active
       D1=(5) (=oLSTNG)+2
       C=DAT1 A
       C=C-1  B         Ne faisons rien non plus
*                       s'il existe 1 listing-file
       GONC   Asset     (Inverse l'état de carry)
       RTNCC            (B.E.T.) (.)
Asset  RTNSC            (.)


* Ceci est (encore) 1 routine HP. Elle trouve 1
* ligne ds 1 fichier TEXT.
* Reportez-vs pr plus de détails aux IDS vol.1.

POSTXT D1=A
       D1=D1+ =oFTYPh
       D1=D1+ (=oFLAGh)-(=oFTYPh)
       GOSBVL =FILSK+
       D=C    A
       D1=D1+ 5
       CD1EX
       A=R1
       GOSUB  FRCRDr
       RTNNC
       C=0    A
       ?ST=0  =sBADRC
       RTNYES
       LC(2)  =eEOFIL
       RTNSC


FRCRDr R1=A
       A=0    W
       A=A-1  A
       R0=A
       ST=0   =sEOF
       ST=0   =sBADRC
FRCR10 GOSUB  PRSREC
       RTNC
```

```
       D0=C
       A=R0
       A=A+1  A
       R0=A
       C=R1
       ?A=C   A
       GOYES  rtncc
       ?ST=1  =sBADRC
       RTNYES
       CD0EX
       GONC   FRCR10     (B.E.T.)


PRSREC B=0    A
       D1=C
       ?C>=D  A
       GOYES  PRSR10
       D1=D1+ 4
       CD1EX
       ?C>D   A
       GOYES  PRSR20
       A=DAT1 4
       GOSBVL =SWPBYT
       P=     3
       B=A    WP
       C=B    A
       B=B+1  WP
       P=     0
       RTNC
       BCEX   A
       CSRB
       C=C+1  A
       C=C+C  A
       C=C+C  A
       AD1EX
       D1=A
       C=A+C  A
       ?C<=D  A
       GOYES  rtncc
       ST=1   =sBADRC
rtncc  RTNCC

PRSR10 ST=1   =sEOF
       RTNSC

PRSR20 B=B-1  A
       ST=1   =sBADRC
       RTNSC
```

GLISSEMENTS ET PERMUTATIONS
Ou DU FAUX Dr JPC

Salut à vous, JPCiliens !

Vous croyiez que l'ignoble S.V. (1) s'était noyé pendant les vacances, ou bien qu'on l'avait pendu, etc... Non seulement il fait encore des articles (Comble de malheur, ils traitent maintenant de l'assembleur), mais voici que la nouvelle année vous le ramène plus omniprésent que jamais !
JPCiliens, unissez vos voix, écrivez au journal, pour qu'on le raye des membres du club !
JPCiliens, la destinée du journal est entre vos mains !

Cet article fait suite à un article de M.Weil (Cf JPC 28 Page 16 Article 2). M. y demandait si les fonctions assembleur suivantes n'existaient pas:
SWAP A,B
SWAP$ A$,B$
SHORT A()
SHORT$ A$()

JJ.Dhénin a résolu le problème du tri numérique (Sic !): voyez à ce sujet son article MATHROM2, qui paraitra bientôt (LEX VECTLEX). M'aidant des commentaires dont il me fit part à propos de VECTLEX, et d'un article qu'il fit paraitre dans JPC (Cf JPC 29 Page 29), j'ai crée pour ma part, la nuit dernière, les ordres SWAP et SWAP$. Il s'agit en fait d'un seul et même ordre:
SWAP
Qui se charge de permuter les valeurs de deux variable, numériques, ou alphanumériques.

Le résultat de mes travaux n'est guère réjouissant: le gain de temps n'est en fait que de 10% pour des variables numériques, de 30% pour des variables alphanumériques de 100 octets de longueur (2); mais l'ordre SWAP a l'avantage de simplifier l'écriture d'un programme BASIC.
Désormais on remplacera les lignes suivantes:

```
10 C=A @ A=B @ B=C
20 C$=A$ @ A$=B$ @ B$=C$
```

Par:

```
10 SWAP A,B
20 SWAP A$,B$
```

Cas des variables numériques:

A : variable entière (INTEGER)
" en simple précision (SHORT)
" en double précision (REAL)
" complexe, de simple précision (COMPLEX SHORT)
" complexe, en double précision (COMPLEX)

Si A est complexe (ou réelle), B doit impérativement être complexe (ou réelle). Lorsque l'on fait SWAP A,B , B est d'abord placée dans A, puis A dans B. Aussi, si la précision de la variable A est inférieure à la précision de la variable B, B sera redimensionnée de manière à avoir la même dimension que A.

Cas des variables alphanumériques:
Ce qui fut dit plus haut à propos des variables numérique s'applique encore aux variables alphanumériques (Redimensionnement, ...).

Remarques:
Pour ne pas surcharger la routine de compilation de l'ordre SWAP, il est possible de taper:
SWAP A,B$
Mais ceci provoquera à l'exécution l'erreur:
Data type

Application:
---- Programme de tri ---

On désire trier le tableau de 100 chaines A$().

```
10 FOR I=1 TO 100 @ FOR J=I+1 TO 100
20 IF A$(I)>A$(J) THEN SWAP A$(I),A$(J)
30 NEXT J @ NEXT I
```

Jean-Jacques Moreau (SIG#? PC#149 CRTF#1)

(1) ndlr: nous avons cherché longtemps pour trouver la signification de "S.V.". Cela pourrait-être, selon certains, "Singe Volant".

(2) note: il semble que l'on puisse améliorer ces résultâts; voir à ce propos le commentaire du LEX.

.....

```
        LEX   'SWAPLEX'

        TITLE  Utilitaire BASIC - J.J. Moreau -
*                                    <860215-860216>
*                                      64 ave de la
*                                           paix
*                                     93150 le
*                                       Blanc-Mesnil
*                                         (1)
*                 48-67-33-04 (SIG#? PC#149 CRTF#1)

        ID     #E1      PARIS-ROM
        MSG    0
        POLL   0

        ENTRY  Swap
        CHAR   #D       Programmable tout-terrain
        KEY    'SWAP'
        TOKEN  71

=ARRYCK EQU   #0366A
=AVE=D1 EQU   #18BB8
=COMCK+ EQU   #032AE
=D1MSTK EQU   #1954E
=DEST   EQU   #0F7B0
=DSTRDC EQU   #05280
=EOLCK  EQU   #02A7E
=EXPEX- EQU   #0F178
=EXPEXC EQU   #0F186
=IVVARe EQU   #02E66
=NTOKEN EQU   #0493B
=NXTSTM EQU   #08A48
=POPMTH EQU   #1B3DB
=POPUPD EQU   #08F3E
=PSHUPD EQU   #08F0D
=RESPTR EQU   #03172
=STORE  EQU   #0F5F8
=SVTRC  EQU   #0FA35
=SYNTXe EQU   #02E2B
=VARP   EQU   #0350E

        ENDTXT
```

```
* Compile l'ordre SWAP:
*   tSWAP tVAR VAR tCOMMA tVAR VAR
*   c'est à dire SWAP var1,var2
*   ex: SWAP Z9$,C0,A$,U5(I)

Swapp  GOSUB  Swap+    Compile 1 var.
       GOSBVL =NTOKEN  Cherche le token suivant
*                      la var. précédente
       GOSBVL =COMCK+  Est-ce 1 virgule?
       GONC   syntxe   Non
       GOSUB  Swap+    Oui: compile 1 2nd var.
       GOSBVL =EOLCK   N'est-elle pas suivie de:
*                      tEOL, tTHEN ou t! ?
       GONC   syntxe
       GOVLNG =RESPTR  Si: alors tout est compilé
*                      comme il faut

Swap+  ST=0   8
       GOSBVL =VARP
       GOC    Swap+5
       ST=1   8

Swap+5 ?ST=0  2
       RTNYES
       GOSBVL =ARRYCK
       ?ST=0  8
       RTNYES
       B=B-1  A
       ?B=0   P
       RTNYES
syntxe GOVLNG =SYNTXe


* Décompile une suite de variable:
*   tVAR VAR1 tCOMMA tVAR VAR2 tCOMMA ... tCOMMA
*   tVAR VARN
*   c'est à dire: VAR1,VAR2,...,VARN
*   ex: [DIM ]A,B,C2$,...,U9$(10)

Swapd  GOVLNG =DSTRDC  Décompile une liste de
*                      variables



       REL(5) Swapd
       REL(5) Swapp    --- SWAP Var1,Var2 ---
Swap   GOSUB  Swapt    Initialise le mode TRACE
*                      VARS
       GOSBVL =EXPEX-  Place la valeur de la 1ère
*                      var. sur la pile
*                      Réactualise AVMEME
       GOSBVL =DEST    Sauvegarde d'adrs pr le
*                      stackage de la 2nd var.
       D0=D0+ 2        Saute la virgule (,)
       ADOEX           Sauvegade le pointeur du
*                      tampon d'entrée...
       R0=A            ... dans R0 ...
       GOSBVL =PSHUPD  ... et dans la pile de
*                      retour BASIC
       A=R0            Restaure ce pointeur
       D0=A            .
       GOSBVL =EXPEXC  Evalue la 2nd variable
*
* Il est à noter que EXPEX- ne préserve pas ce
* qui se trouve sous
* l'AVMEME, mais considère que AVMEME=FORSTK, ce
* qui n'est pas
* le cas de EXPEXC qui ne touche en aucun à ce
* qui se trouve entre
* AVMEME et FORSTK, ce qui est très utile si l'on
* veut preserver qqch sur la pile.
* Notons encore que les 2 routines réactualise
* AVMEME avant de rendre la main, et préserve ains
* les informations qu'elles viennent de déposer
* sur la pile.
*
       GOSBVL =STORE   Stocke la 2nd var. ds la
*                      1ère (Cf =RECALL, IDS vol.3
       GOSBVL =POPUPD  Récupérons maintenant le
*                      pointeur du tampon
       C=D    A        de sortie
       D0=C           .
*
* Rappellons que EXPEX- à laissé sur la pile les
* informations relatives
* à la 1ère var.; EXPEXC ceux relative à la 2nd,
* et au-dessus.
* Ces dernières informations ne nous intéressent
* plus aussi allons-nous
* les éliminer, en incrémentant AVMEME.
*
       GOSUB  Swaps    Détruit les info.
*       inutiles, c'est à dire saute par-dessus...
       GOSBVL =AVE=D1  ... puis réactualise AVMEME
       GOSUB  Swapt    Initialisons le mode TRACE
*                      VARS
       GOSBVL =EXPEXC  Rappellons que D0 @ tampon
*                      de sortie
*
* La ligne précédente peut vs paraître inutile,
* car elle
* sert à mettre sur la pile la valeur de la 2nd
* var., alors qu'on
* vs à expliquer plus haut qu'on enlevait cette
* info. qui ne servait
* à rien. En fait c'est vrai; mais EXPEXC
* initialise les reg. du CPU pr =DEST.
* Pr rendre SWAP plus rapide, il faudra sans
* doute remplacer ce dernier appel de EXPEXC
```

```
* par un appel a RECALL ou ADDRSS.
*
       GOSBVL =DEST    Prépare le stockage ds la
*                      2nd var.
       GOSUB  Swaps    Enlève une fois pr tte les
*                      info. inutiles
       A=DAT1 W        Initialise A(W)
       GOSBVL =STORE   Stocke var1 ds var2
       GOVLNG =NXTSTM  C'est tout

Swaps  GOSBVL =D1MSTK  D1 @ AVMEME
       GOVLNG =POPMTH  Saute le 1er truc sur la
*                      pile

Swapt  ?ST=0  15       Mode TRACE?
       RTNYES          Non
       GOVLNG =SVTRC   Oui; l'initialise

* Comme vous avez pu le constater, il existe en
* assembleur un petit nombre de
* de routines extrêmement puissantes qui
* permettent de travailler sur les var.
* C'est pourquoi ce LEX ne fait qu'une centaine
* d'octets.
```

LA PILE D'INSTRUCTIONS

Tous les programmes, BASIC ou BIN qui nous ont été proposés à ce sujet, pèchent par un grave défaut : ils "virginisent" la pile d'instructions.

Certain qu'il est bien utile de conserver les dernières instructions dont on avait l'usage, j'ai écrit ce programme LEX qui ne détruit rien lorsqu'on ajoute des niveaux et qui ne supprime que les instructions les plus anciennes lorsqu'on réduit la taille de la pile.

Bien qu'en possession d'un assembleur, je préfère assemble "à la main" ce qui permet, entre autre, de tester la validité du programme en cours d'élaboration. Aussi ne vous étonnez pas de la présentation du LEX qui est naturelle et diffère en cela de celle dont vous avez l'habitude.

Le lex s'appelle TOTO, mais vous pouvez lui donner un autre nom. J'ai utilisé l'ID #93 qui correspond à un SCRATH LEX, mais n'importe quel autre numéro fera aussi bien l'affaire.

147 octets avec l'entête et les tables (129 au cat)
114.5 octets en l'insérant dans un LEX incomplet.

Je signale, par honnêteté, que l'entrée retenue dans la routine "CMDFND" n'est pas supportée. J'attaque, en effet, cette routine à l'adresse #016A1, quatrième ligne après le début de la routine.

Je me suis permis cette licence dans le but de commodité (raccourcissement du PGM). en faisant l'hypothèse que la routine, étant supportée à la première ligne, a fort peu de probalité d'évoluer dans les 4 premières lignes : il suffit pour s'en convaincre de l'étudier.

Si, cependant, quelqu'esprit scupuleux ne se contentait pas de cette mise au point et veuille, par rigueur, s'en tenir au supporté, j'indique 2 façons de procéder :

1 - réécrire la routine à partir de la 4ème ligne ! 36 digits !

2 - plus commodémment, attaquer la routine au début, en plaçant préalablement à CMDPTR (#2F6D4) le nombre de niveaux-1 désiré.

C. MARCOIN

```
45F445F402020202      LEX    'TOTO'
802E        File type : LEX
00          flags et copy code
7421        heure
612158      date
60100       offset
39          ID# 93
10          from 01
10          to  01
00000       Next LEX 00000
F           CON(1) F
7100        CON(4) 0017
0000        MSG 0
00000       POLL 0
         *** MAIN TABLE ***
000         CON(3) 000
32000       REL(5) e
```

```
D          CHAR D
       *** TEXT TABLE ***
5          CON(1) 5
3545B4     NIBASC STK
10         Token 01
1FF        NIBHEX 1FF

A> 8D91FB0 GOVLNG =ARGERR

EB000      REL(5) A*       offset decompile
0C000      REL(5) A**      offset parse
8F681F0    GOSBVL =EXPEXC  évalue l'argument
8F322B1    GOSBVL =FLTDH   le converti en HEX
CC         A=A-1 A
814        ASRC            teste
8AC        ?A#0 A          la validité
9D         GOYES  A>       0<arg<16
810        ASLC
136        CD0EX           PC dans C
06         RSTK=C          sauvé dans STK
D2         C=0 A
1F679F2    D1=(5) =MAXCMD
15F0       C=DAT1 1        C= actuel niveaux-1
1590       DAT1=A 1        stoore niveaux désirés
1B675F2    D0=(5) =CLCBFR
EE         C=A-C A         différ. niveaux
404        GOC    A>>      contraction
DA         A=C A           multiplicande
D7         B=C A           differ. --> D
306        LC(1) 6
8F943B1    GOSBVL =A-MULT  multiplication
D8         B=A A           bloc à ajouter
146        C=DAT0 A        @ CLCBFR dans C
169        D0=D0+ 10       D0 @ RAWBFR
142        A=DAT0 A        @ RAWBFR dans A
C0         A=A+B A         nouvel RAWBFR
131        D1=A            dans D1
E0         A=A-B A         actuel RAWBFR
8F401B1    GOSBVL =MOVED2  ajoute bloc
           *               à CLCBFR
AF2        C=0 W           prépare 000300
23         P=  3
303        LC(1) 3
A<    CF   D=D-1 A         et le compteur
492        GOC    A>>>     à l'underflow
           *               ajuste pointeurs
1C5        D1=D1- 6        inscription
15D5       DAT1=C 6
53F        GONC   A<
A>>    814 ASRC            contraction
           *               A(S)= niveaux-1
8F1A610    GOSBVL =CMDFND  A @ D1 (adr basse)
DA         A=C A
146        C=DAT0 A        C @ CLCBFR
135        D1=C
E2         C=C-A A         bloc à supprimer
D5         B=C A           pour PRYAD2
8F861B1    GOSBVL =MOVEUA  contraction
A>>>  24   P=  4
1B085F2    D0=(5) =RAWBFR
8FA8F80    GOSBVL =PTRAD2
07         C=RSTK          PC est récupéré
136        CD0EX
8F84A80    GOSBVL =NXTSTM  retour à BASIC

A* 8D39450  GOVLNG =FIXDC
A** 8DE6A20 GOVLNG =FIXP
```

EDIT ET COPY (v.o.)

LE BUT

Partant de la constatation qu'il était pénible de taper COPY TOTO:TAPE suivi de EDIT TOTO, j'ai réalisé ce Lex. Croyant la chose difficile j'ai longtemps hésité... à tort, vous en jugerez par vous-même.
D'abord la routine la plus importante: COPYu. Dans un espace réservé de 50 quartets (appelé Save Stack), mettez le spécificateur du fichier source (0 à 25) suivi de celui du fichier objet (26 à 50). Je vous rappelle qu'un spécificateur est composé d'un nom de fichier associé à un périphérique. Il peut être partiellement ou même totalement omis. Par exemple COPY JPC:TAPE sous entend TO JPC:MAIN. Donc une fois la Save Stack remplie appelez COPYu. Cette routine se charge de compléter de manière univoque les spécificateurs et d'effectuer la recopie (hpil inclus). Mieux: si le fichier est recopié en mémoire vive R1 le pointe en sortie. Or la deuxième routine importante est EDIT80 qui permet d'éditer (c'est original je sais!) le fichier pointé par D1. Il suffira donc de transférer R1 en D1 pour conclure cette 'kolozal' commande. Vraiment pas de quoi épater la galerie, domage pour moi ! Enfin le principal est que ça fonctionne bien.

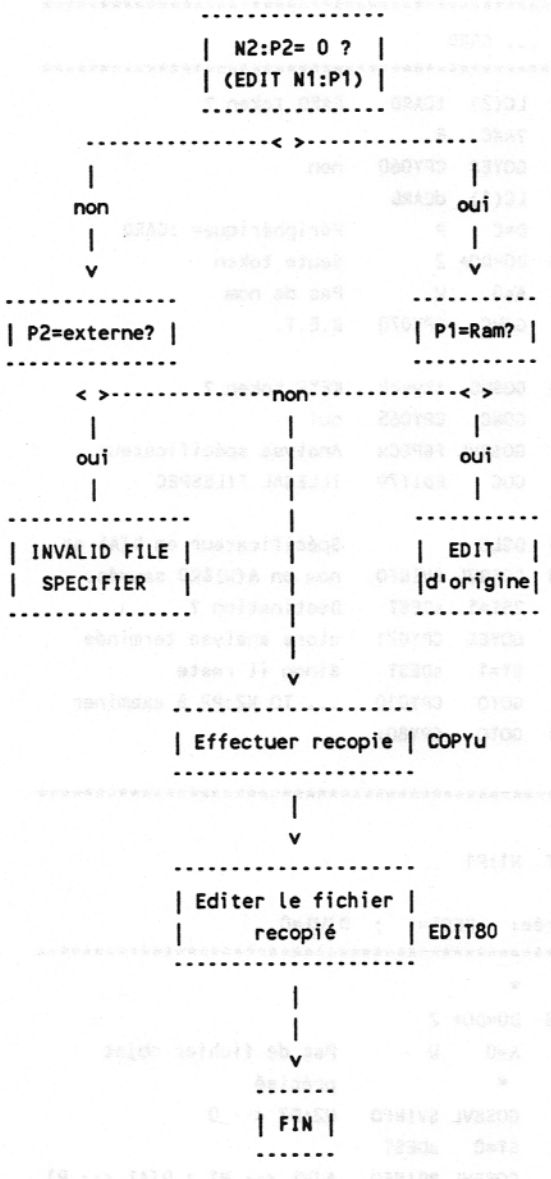
LA SYNTAXE

De la forme EDIT [N1:P1 [TO N2:P2]]

Avec: N1 = nom du fichier source.
P1 = périphérique source.
N2 = nom du fichier objet.
P2 = périphérique objet.

Ce qui est entre crochets peut être omis, et P2 doit toujours représenter un périphérique interne de mémoire (vive évidement!). Par exemple EDIT TOTO:TAPE(2) TO TOTOS, EDIT STAR:PORT(0.01) sont valables mais EDIT TOTO TO : APE(2) est à banir puisque TAPE(2) est un périphérique externe. Les messages d'erreur sont ceux de COPY + 'Illegal Filespec' pour P2 valide mais externe.

L'ALGORITHME



J'espère que ces explications vous seront utiles et que le Lex EDIT comptera parmi ceux qui résident constament dans votre 71. En attendant la prochaine réunion je retourne à mes IDS. Salutations...

Jean-Pierre BONDU (PC 33, SIG 4)

PS: selon une source bien informée, il apparaitrait que SIG signifie "Sertainement Impossible mais G'y arriverai". L'orthographe ne serait là que pour brouiller les pistes. Affaire à suivre...

```
        LEX     'EDIT'
        ID      1
        MSG     0
        POLL    0
        ENTRY   EDITe
        CHAR    5       Non programmable
        KEY     'EDIT'  Ecrase la fonction
        TOKEN   184     originelle.
        ENDTXT

BSERR   EQU     #0939A
COPYu   EQU     #08269
CRETF+  EQU     #084C4
CURDVC  EQU     #0A60B
CURRST  EQU     #2F55D
EDIT80  EQU     #0A5A5
EDITWF  EQU     #0A533
EOLCK   EQU     #02A7E
FINDF+  EQU     #09F63
FLDEVX  EQU     #01154
FSPECe  EQU     #02F02  * INVALID FILESPEC
FSPECp  EQU     #03CC5
FSPECx  EQU     #09F2D
NXTSTM  EQU     #08A48
RDINFO  EQU     #0846B
RESPTR  EQU     #03172
SALLOC  EQU     #0153B
SVINFO  EQU     #0845A
WRDSCN  EQU     #02C2A
dCARD   EQU     #00007
eFSPEC  EQU     #0003A  * ILLEGAL FILESPEC
fBASIC  EQU     #E214   * Fichier BASIC
lDATEh  EQU     #00006
```

```
lFILSV  EQU    #00032
lFLAGh  EQU    #00002
lFLENh  EQU    #00005
lFNAMh  EQU    #00010
lFTYPh  EQU    #00004
lTIMEh  EQU    #00004
oBSsod  EQU    #00011
oFLENh  EQU    #00020
sDEST   EQU    #00003
tCARD   EQU    #000D0
tEOL    EQU    #000F0
tKEYS   EQU    #000CF
tTO     EQU    #000F3

        REL(5) EDITP

EDITe   LC(5)  lFILSV   Réserve 50 quartets
        P=     1        pour la SAVSTK
        GOSBVL SALLOC   (= zone de sauvegarde)
        GOC    jp       INSUFFICENT MEMORY
CPY010  D=0    W        Périphérique initialisé
        LCASC  '  '     Caractères 9 et 10
        R0=C            du nom du fichier éffacés
        A=DAT0 B        Lit token suivant

************************************************
* COPY TO  /  COPY <spec> TO ...
************************************************
        LC(2)  tTO
        ?A#C   B        Token <> TO ?
        GOYES  CPY030
        D0=D0+ 2        Saute TO token
        ?ST=1  sDEST    Destination ?
        GOYES  CPY040   Analyse N2:P2

************************************************
*
* COPY  TO  ...
* Pas de fichier source spécifié
* Prend le fichier et spécificateur courant
*
************************************************
CPY020  GOSUB  deffil   N1:P1 <-- courant
        GONC   CPY065   B.E.T.

************************************************
*
* EDIT avec/sans paramètres  /  EDIT <spec> EOL
*
*    Si Source
*      EDIT workfile
*    Sinon       (CPY045)
*      N2:P2 <-- 0
*      Si N1 pas en mémoire
*         Recopier N1  (CPY080)
*      EDIT N1   (edit80)
*
************************************************
        *
CPY030  LCHEX  F0       Token <> EOL ?
        ?A<C   B
        GOYES  CPY040   oui
        ?ST=1  sDEST    Destination ?
        GOYES  CPY045

        *
        * EDIT workfile
        *
        GOVLNG EDITWF

************************************************
* COPY ... CARD
************************************************
CPY040  LC(2)  tCARD    CARD token ?
        ?A#C   B
        GOYES  CPY060   non
        LC(1)  dCARD
        D=C    P        Périphérique= :CARD
CPY050  D0=D0+ 2        Saute token
        A=0    W        Pas de nom
        GONC   CPY070   B.E.T.

CPY060  GOSUB  tkysck   KEYS token ?
        GONC   CPY065   oui
        GOSBVL FSPECx   Analyse spécificateur
jp      GOC    EDIT79   ILLEGAL FILESPEC

CPY065  DSLC            Spécificateur en D[A] et
CPY070  GOSBVL SVINFO   nom en A[W]&R0 sauvés.
        ?ST=1  sDEST    Destination ?
        GOYES  CPY071   alors analyse terminée
        ST=1   sDEST    Sinon il reste
        GOTO   CPY010   ... TO N2:P2 à examiner
CPY071  GOTO   CPY80+

************************************************
*
* EDIT  N1:P1
*
* Entrée:  sDEST=1  ;  D[W]=0
************************************************
        *
CPY045  D0=D0+ 2
        A=0    W        Pas de fichier objet
         *              précisé
        GOSBVL SVINFO   N2:P2 <-- 0
        ST=0   sDEST
        GOSBVL RDINFO   A[W] <-- N1 ; D[A] <-- P1
```

```
        DSRC            D[S]= spécificateur
        C=D     W
        R3=C
        GOSBVL FINDF+   N1 en mémoire ?
        GONC    edit80  oui
        ?ST=0   6       N1= 0 ou P1= externe ?
        GOYES   CPY080
        C=R3            Crée le fichier N1
        D=C     W       en respectant P1
        R2=A
        C=0     A
        LC(2)   (oFLENh)+(oBSsod)
        GOSBVL CRETF+
EDIT79  GOC     CPYERX
        A=R1
        D1=A
        C=R2
        DAT1=C lFNAMh
        D1=D1+ lFNAMh
        LC(4)   fBASIC
        DAT1=C lFTYPh
        D1=D1+ (lFTYPh)+(lTIMEh)+(lDATEh)+(lFLAGh)
        D1=D1+ lFLENh
        C=0     W
        DAT1=C 10
        D1=D1+ 10
        LC(2)   tEOL
        DAT1=C B
        D1=A
        *
        * Edite le fichier pointé par D1
        *
edit80  GOSBVL EDIT80
        GOVLNG NXTSTM   FIN

************************************************
*
* COPY N1:P1 TO N2:P2  et  EDITE N2:P2
*
* 1) Recopie N1:P1 dans N2:P2
* 2) Puis édite N2
*
* Entrée: suppose P2= périph. interne
************************************************
        *
CPY080  GOSBVL COPYu
        GOC     CPYERX  Erreur
        C=R1            R1 ^ fichier copié
        D1=C            D1 <-- R1
        GONC    edit80  B.E.T.

        * Teste si P2= Périphérique interne
        *       oui -> CPY080
        *       non -> ILLEGAL FILESPEC
        * Entrée: D[A] = :P2
        *
CPY80+  DSRC            D[S]= type de P2
        D=D+1   S       P2 indéfini ?
        GOC     CPY080  MAIN par défaut
        D=D+D   S       externe ?
        GONC    CPY080  non
        LC(4)   eFSPEC  ILLEGAL FILE SPECIFIER
CPYERX  GOVLNG BSERR    Basic Error

deffil  GOSBVL CURDVC   Trouve périphérique
        GOSBVL FLDEVX   courant
        D1=(5) CURRST
        C=DAT1 A
        D1=C            D1 ^ sommet fichier
         *              courant
        A=DAT1 W        A[W] <-- nom du fichier
        LCASC  '  '
        R0=C            cf. CPY010
        RTNCC

tkysck  A=DAT0 B
        LC(2)   tKEYS
        ?A#C    B
        RTNYES
        D0=D0+ 2        Saute token
        LCASC  '    syek'
        D=0     S
        A=C     W
        RTN

*
******* ROUTINE DE PARSE *******
*
EDITP   GOSUB   eolck+
        GOC     NAMEP7
        GONC    EDP10   B.E.T.
RNMP05  ST=1    8
EDP10   GOSBVL FSPECp
        GONC    RNMP25
        GOVLNG FSPECe

RNMP25  ?ST=1   8
        GOYES   LSTPDN
        GOSBVL WRDSCN
        CON(2) tTO
        REL(5) RNMP05
        CON(2) 0
NAMEP7  GOVLNG RESPTR
LSTPDN  RTNCC
eolck+  GOSBVL EOLCK
        RTNC
        GONC    NAMEP7  B.E.T.
```

```
END
```

.....

CONSRUCTION DE LA TABLE DES EQUIVALENCES

La partie la plus ennuyeuse de la réalisation d'un Lex est l'élaboration de la table d'équivalence (TOTO EQU #08B24 ...). Je dédicace donc ce programme à tous les Sigouilleurs, en espérant que cela les poussera à n'utiliser que des points d'entrée supportés par HP. Son rôle est le suivant: partant d'un source 'vierge', il repère tous les appels faits à des étiquettes externes, en cherche les valeurs, et insère finalement la table d'équivalence ainsi constituée.
Dans ce but vous devrez avoir soit en mémoire centrale, soit sur mémoire de masse les différents fichiers PENA à PENX plus PENXXX. PEN signifie "Points d'Entrée Noms", la dernière lette indiquant la première lettre de toutes les étiquettes contenues dans ce fichier. Par exemple FNRTN1 est rangé dans PENF, NXTSTM dans PENN, RNDAHX dans PENR ... Les labels ne commencant pas par une lettre de l'alphabet sont rangés dans PENXXX.
Seule limitation importante de ce programme: le source doit résider en mémoire. Je sais que cela peut poser problème, mais l'emploi de la si pratique fonction SEARCH m'y a contraint. Le seul moyen de l'éviter aurait été de concevoir ce programme autrement. Plutôt que de chercher une instruction particulière dans l'ensemble du source, il fallait lire séquentiellement le fichier et tester systématiquement l'appartenance de la mnémonique à un ensemble prédéfini. Ce système a deux inconvénients: sa lenteur (10 à 15 tests par ligne...) et le fait que pour insérer la table d'équivalence il faudra bien, de toute manière, recopier le source en mémoire un jour. Voila les raisons de mon choix.

L'utilisation du programme est évidente. Les mnémoniques 'intéréssantes' étant dans des DATA, vous pouvez en modifier la liste à votre convenance. Quelques explications sur les fichiers PEN. Chaque enregistrement contient une adresse, sur les 5 premiers caractères, suivie du label à partir de la colonne 6. Par exemple la routine A-MULT débute en 1B349, ADHEAD en 181B7, ce qui donne:

```
1B349A-MULT
   .
   .
181B7ADHEAD
   .
   .
```

Afin de respecter la tabulation le zéro figurera comme tout autre chiffre. Le plus simple est encore que vous vous procuriez ces fichiers auprès du club, à l'occasion d'une réunion par exemple (je vous rappelle qu'il y a près de 800 points référencés!). Afin d'accélérer sensiblement ce programme j'ai conçu deux fonctions en assembleur. MNEMO renvoie la position de la mnémonique dans un enregistrement donné, LABEL$ vérifie qu'une étiquette est autorisée. Voyez les sources pour plus de renseignements. A titre d'exemple l'analyse du fichier MMLEX, d'un volume de 8K, requiert 2mn40, auxquels il convient d'ajouter le temps d'accès aux fichiers PEN. Je vous dois quelques explications à ce sujet.

Le programme fonctionne de la manière suivante:
1- Chaque mnémonique pouvant admettre un label en paramètre est recherchée (lignes 1120->1140). Dans l'enregistrement ainsi localisé on recherche le label (1150->1170). Sa crédibilité est vérifiée dans 'VERLAB'. Si le test est positif il est enregistré, par ordre alphabétique, dans le fichier LKBUF. C'est le rôle assigné à 'RECLAB'. Il s'agit donc d'établir, dans le fichier LKBUF, la liste de tous les labels externes utilisés.
2- Suit l'identification et l'enregistrement de chacun de ces labels (1440->1540).

Me sentant gagné par la fatigue du Juste, dans la quiétude du devoir accompli, je me permets de vous quitter ici et de vous souhaiter bonne nuit, 12 coups ayant déja retentis.

Jean-Pierre BONDU (PC 33, SIG 4)

96	11	89	68
88	69	91	16
61	86	18	99
19	98	66	81

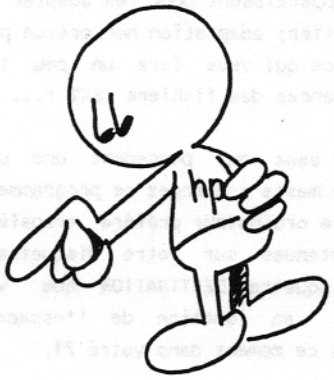
Programme "LINK" (construit la table des équivalences)

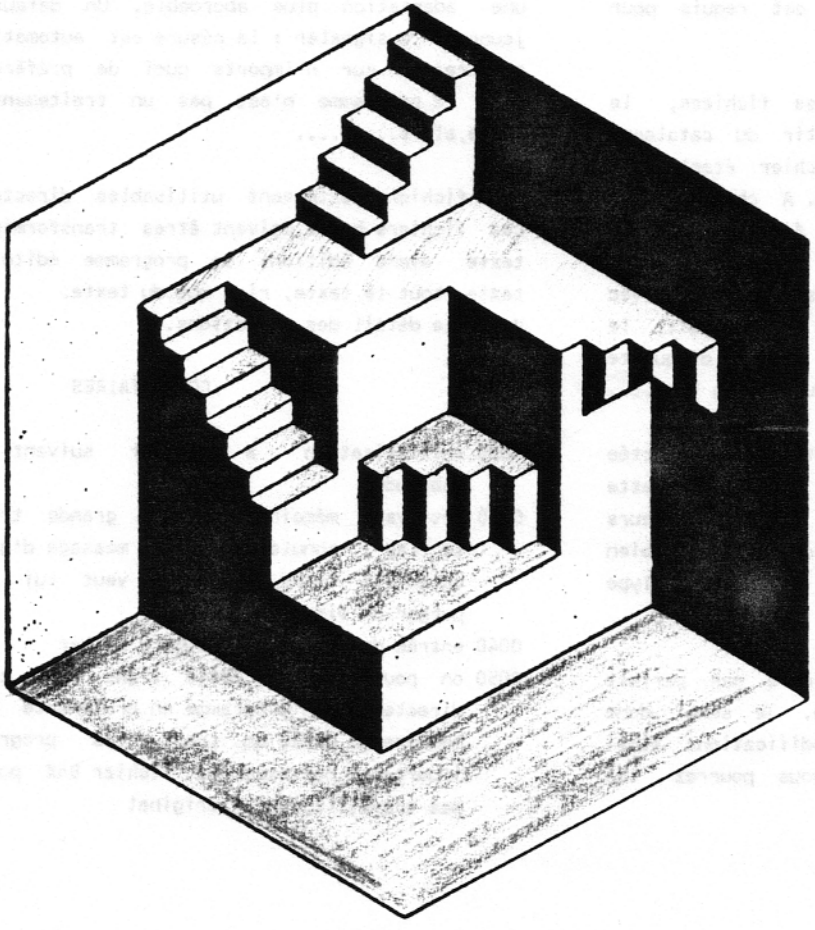
```
   - Etablit la table d'équivalence d'un fichier source quelconque (LEX,FORTH...)
     Nécéssite: REDUCE$ (cf FORMALEX), LABEL$/MNEMO (cf LINKLX), (EDLEX)
  10 INPUT "Source File ";F$ @ CALL LINK(F$)
  20 BEEP @ END

================================================================
1000 SUB LINK(F$)

================================================================
1010 DATA GOSBVL,GOVLNG,D0=D0,D1=D1,DAT,D0=(,D1=(,LC(,CON(,ST=,ST#,P=,P#,C=P,CPEX,*

================================================================
   - CON(5)... Attention aux conflits entre les routines hpil, via JUMPER,
     et d'autres routines, de même nom, de la ROM du 71.
1020 INTEGER P0,P1,P2,P3,F,F2 @ DIM H,T,M$[8],S$[8],G$[13],E$[96]
1030 F=FILESZR(F$) @ IF F<0 THEN DISP MSG$(-F) @ END
1040 SFLAG -1 @ ON ERROR GOTO 1050 @ UNSECURE LKBUF @ PURGE LKBUF
1050 OFF ERROR @ CFLAG -1 @ ASSIGN #1 TO F$
1060 CALL SRCH(#1,"ENDTXT",2,1,9999,H,E$)
1070 IF NOT H THEN DISP "ENDTXT introuvable" @ END
1080 DELAY 0 @ P1=H+1 @ F2=FLAG(2,0) ! P1= ^ ENDTXT +1
1090 CALL SRCH(#1,"END",2,P1,9999,H,E$)
1100 P3=H-1 @ IF NOT H THEN P3=F ! P3= dernière ligne recherche
1110 H=P1 @ CREATE TEXT LKBUF @ ASSIGN #2 TO LKBUF

================================================================
1120 'NXMNEMO': READ M$ @ DISP @ DISP M$;TAB(8); @ IF M$='*' THEN 'SUITE' ! M$= mnémonique

================================================================
1130 'NXLINE': CALL SRCH(#1,M$,2,H+1,P3,H,E$) ! H= position de M$ dans #1
1140 IF NOT H THEN 'NXMNEMO' ! Pas trouvée => mnémo suivante
1150 T=POS(E$," ",IP(FP(H)*1000)) ! T= ^ dernier caractère mnémo +1
1160 IF NOT T THEN 'NXLINE' ! Toute mnémo recherchée est suivie d'un 'modifier', donc d'un
   - espace. Dans le cas contraire il y a erreur, que l'on ne traite pas -> NXLINE.
     L'assembleur se chargera de vous sanctionner !...
1170 E$=REDUCE$(E$[T]) @ T=POS(E$," ") @ E$=E$[1,T-1+NOT T*90] ! E$= label

================================================================
1180 'VERLAB': DIM X$[96] @ CFLAG 2
1190 P0=POS(E$,"(") @ P2=POS(E$,")",P0)
1200 IF NOT P0 THEN 1230
1210 IF NOT P2 THEN DISP MSG$(76)&" in line";H @ GOTO 1240 ! '(' trouvée mais pas ')': ERR
1220 X$=E$[P2+1] @ E$=E$[P0+1,P2-1] @ GOSUB 'RECLAB' @ E$=X$ @ SFLAG 2 @ GOTO 1190
1230 IF NOT FLAG(2,0) THEN GOSUB 'RECLAB'
1240 DESTROY X$ @ GOTO 'NXLINE'

================================================================
1250 'RECLAB': DISP '.'; @ E$=LABEL$(E$) @ IF NOT LEN(E$) THEN RETURN
   - Teste la crédibilité du label E$, l'enregistre si OK, sinon retour immédiat.
1260 T=SEARCH(E$,1,0,999,2) ! Cherche label dans LKBUF
1270 IF NOT FILESZR('LKBUF') THEN 'P'
1280 READ #2,T;S$ @ IF T AND S$=E$ THEN 1320 ! Déja présent
   - Insertion du label E$ dans LKBUF
1290 FOR T=0 TO FILESZR('LKBUF')-1 @ READ #2,T;S$ @ S$=UPRC$(S$)
1300 IF UPRC$(E$)>S$ THEN INSERT #2,T;E$ @ GOTO 1320
1310 NEXT T @ 'P': PRINT #2;E$
1320 RETURN
```

```
   - *** FIN  DE  LA PARTIE  I: recherche  des  labels ***
     *** DEBUT DE LA PARTIE II: étiquettage/insertion  ***

============================================================
1330 'COPYF': ! E$= label  ;  M$= fichier de E$  ;  S$= fichier précédent
1340 M$=FNN$(E$) @ IF M$=S$ THEN 1385 ! Fichier présent: inutile de le recopier
1350 IF FLAG(2,0) AND LEN(S$) THEN PURGE S$
   - LEN(S$)<>0 : 'COPYF' a déja été appelé
     FLAG(2)= 1 : S$ n'était pas en mémoire avant utilisation, il faut l'effacer
1360 S$=M$ @ IF FILESZR(M$)<0 THEN COPY :TAPE TO M$ @ SFLAG 2
   - FLAG(2)= 1 : ce fichier PEN n'était pas en mémoire, il faudra l'effacer après emploi
1370 ASSIGN #3 TO M$
1385 RETURN

============================================================
1400 DEF FNN$(E$)
1410 F$='PENXXX' @ IF NUM(E$)<123 AND NUM(E$)>64 THEN F$[4]=UPRC$(E$[1,1])
1420 FNN$=F$ ! Définit le nom du fichier PEN auquel
1430 END DEF ! appartient le label E$.

============================================================
1440 'SUITE': DISP ! Etiquette les labels qui sont dans LKBUF
1450 DESTROY S$ @ FOR T=1 TO FILESZR('LKBUF')
1460 READ #2,0;E$ @ GOSUB 'COPYF' @ H=-1
1470 H=H+1 @ H=SEARCH(E$,6,H,999,3) @ READ #3,H;G$
1480 IF NOT H THEN E$[16]='??' @ GOTO 1500
1490 IF G$[6]=E$ THEN E$[16]='#'&G$[1,5] ELSE 1470
1500 E$[9,11]='EQU' @ INSERT #1,P1;E$ @ DELETE #2,0
1510 DISP E$ @ NEXT T
1520 PURGE LKBUF @ IF FLAG(2,F2) THEN PURGE M$
1530 DISP MSG$(240011) @ INSERT #1,P1;""
1540 END SUB

============================================================
2000 SUB SRCH(#1,C$,T,D1,D2,H,E$) @ GOTO 'DEBUT'
   - Entrée:
     #1 = canal du fichier source
     C$ = chaîne recherchée
     T  = No du champ (1=label, 2=mnémo)
     D1 = Ligne début recherche
     D2 = ligne  fin  recherche
     Sortie:
     H  = réponse au format eee,ppplll
     eee= No enregistrement, ppp= position 1er caractère, lll= longueur chaîne
     E$ = enregistrement eee


============================================================
2020 'LABEL': ! Sortie: H1=0 si C$ n'est pas label dans A$
2030 H1=REDUCE$(E$[1,P-1])=C$
2040 RETURN

============================================================
2050 'MNEMO': ! Sortie: H1=0 si C$ n'est pas dans le champ mnémonique
2060 H1=P AND POS(E$[P,P+5],C$) AND E$[P,P]#'*'
2070 RETURN

============================================================
2080 'DEBUT': H=D1-1
```

```
================================================================
2090 'NXT': L=H+1 @ H=SEARCH(C$,1,L,D2,1)
2100 IF NOT H THEN 'FIN'
2110 READ #1,H;E$ @ P=MNEMO(E$)
2120 ON T GOSUB 'LABEL','MNEMO'
2130 IF NOT H1 THEN 'NXT'

================================================================
2140 'FIN': END SUB
```

****





COPYDISK

SIMON (P160, T516)

Le nom de ce programme est assez éloquent pour que je ne vous fasse pas un long discours à son sujet.

Il s'adresse néanmoins aux heureux possesseurs d'un lecteur de disquettes HP-9114 (ou éventuellement d'un lecteur de casettes), d'un HP-71 et de 2 cordons HP-IL (indispensables...). Le module EDTEXT est également nécessaire pour ce qui concerne les instructions INSERT, DELETE, REPLACE et FILESZR. Néanmoins il est toujours possible de modifier les lignes dans lesquelles celles-ci apparaissent pour les adapter à votre cas particulier; adaptation non prévue par votre serviteur, ce qui vous fera un peu travailler vos connaissances des fichiers TEXT !...

Donc pour ceux qui possèdent une unité de stockage de masse : chargez ce programme, faites RUN et votre ordinateur préféré transfèrera les données contenues sur votre disquette ORIGINE sur la disquette DESTINATION que vous lui présenterez, en fonction de l'espace mémoire disponible à ce moment dans votre 71.

A noter qu'un minimum de 3 Ko est requis pour que le programme démarre.

Afin d'optimiser la copie des fichiers, le programme crée un fichier à partir du catalogue de la disquette ORIGINE, ce fichier étant trié par ordre décroissant de volume. A chaque passe il recherche en premier le fichier le plus important qu'il puisse charger en mémoire puis essaye de remplir ce qui lui reste de libre avec des fichiers plus petits. Cela étant fait, le transfert s'effectue vers la disquette DESTINATION. Et ainsi de suite jusqu'à la fin...

FACULTATIF : si une imprimante est connectée sur la boucle, le catalogue de la disquette DESTINATION sera listé avec les indicateurs permettant de savoir si la recopie s'est bien déroulée (* Sauvé, @ Type inconnu = non recopié).

Ce programme n'est certes peut-être pas parfait mais remplit bien son office. Je serai donc intéressé de connaitre les modifications ainsi que les améliorations que vous pourrez lui apporter.

---

programme FORMLIST
Listing de programmes sur 3 colonnes par page

P. DAVASE (P67, T404)

Ce programme permet d'éditer un listing ou un texte sur 3 colonnes par page, sur imprimante 82905B + boucle HPIL + HP71B de 17k RAM.

Le programme permet un gain de place important sur le papier listing, et surtout une diminution très sensible du nombre de pages par rapport à un listing courant.

Le programme peut être facilement adapté à toute imprimante. Il n'est pas optimisé dans son écriture ,et ne comporte pas de fonction provenant de fichier LEX, ceci pour permettre une adaptation plus abordable. Un défaut de jeunesse à signaler : la césure est automatique, arbitraire, sur n'importe quoi de préférence, mais ce programme n'est pas un traitement de texte,alors.....

Les fichiers texte sont utilisables directement Les fichiers basic doivent êtres transformés en texte avant édition. Ce programme édite du texte, tout le texte, rien que du texte.
Voici le détail des opérations.

LIGNE	COMMENTAIRES
0020	initialisation à modeler suivant les habitudes
0030	nettoyage mémoire avant les grands travaux le flag-1 annule l'éventuel message d'erreur qu'envoie TITAN quand on veut lui faire purger du vide ...
0040	entrée du nom de programme a éditer
0050	on pourra adapter cette ligne pour copier directement un programme en provenance d'une mémoire de masse qq. (pour les programmes importants par exemple) fichier BAK pour ne pas travailler sur l'original

0060 transposition de BAK en fichier texte
0070 initialisation des variables
A$ en lecture sur le fichier a lister et dupliquer en BAK. B1-B2-B3$ en écriture sur fichier temporaire FTEXT
0080 FTEXT= fichier temporaire de stockage des lignes formattées
0090 ouverture des canaux respectifs
0110 boucle de traitement des lignes une par une soit:
en lecture une ligne maxi de 96 caractères en écriture 3 lignes de 40 caractères maxi
0120 si fin de fichier --- on édite
0130 lecture du premier enregistrement du fichier BAK
0150 on va analyser le problème.....
0160 pour ne pas se retrouver rapidement avec un beau memory full il faut pour chaque ligne traitée:
écrire les lignes formattées sur FTEXT et supprimer la ligne étudiée de BAK
encore le principe des vases communiquants avec en plus un robinet qui fuit (l'interposition du caractère séparatif chr$(0) et des blancs de tabulation) attention donc car le fichier final FTEXT sera légèrement plus important que le fichier origine BAK
0170 programme d'édition et de remise en état de la mémoire
0180 on purge le fichier origine (BAK) et son canal (#1)
0190 préparation imprimante 82905B et boucle HPIL
0200 passage en mode 132 colonnes par ligne
0210 N1=1ère ligne a éditer en 1ère colonne
N2=59ème ligne a éditer en première colonne
0220 de la première à la dernière ligne de la 1ère colonne
0230 si fin du fichier ,arrêt de l'édition
0240 édition sur première colonne
avec tabulation des numéros de ligne
0260 séparatif des colonnes
0270 on cherche du travail pour la deuxième colonne si pas de boulot alors suite première colonne ligne suivante
0310 idem 3ème colonne
0340 impression de la ligne complète, retour chariot et incrément de ligne
0350 saut de page
0360 initialisation des bornes de recherche pour la nouvelle page
0370 s'il reste des lignes a éditer on recommence la boucle
0380 petit nettoyage en fin d'édition
0390 sous programme dé decomposition et formattage des lignes a traiter
0400 on décompose la ligne originelle en 1-2 ou 3 morceaux
0410 écriture premier tronçon dans FTEXT et incrément du compteur de lignes validées
0420 si 2ème ligne vide ... retour a la boucle sinon écriture et incrément
0430 idem
0440 retour a la boucle principale

Tous les avis, critiques, remarques, et surtout OPTIMISATIONS de ce programme sont non seulement justifiées mais surtout souhaitables et bien venues. Peut-être un de nos saturniens de genie aura la générosité de concocter un petit Lex ... LT3COLEX

JE RECHERCHE UN PROGRAMME POUR FAIRE UNE COPIE BIT PAR BIT D'UNE ZONE DONNEE DE MEMOIRE VIVE , SUR IMPRIMANTE GRAPHIQUE. IL S'AGIT EN FAIT DE SIMULER LA ZONE MEMOIRE D'UN ECRAN GRAPHIQUE DONC DE DESSINER POINT PAR POINT DANS CETTE ZONE (BIT PAR BIT) PUIS DE FAIRE UN REPORT DE LA ZONE AINSI TRAITEE SUR L'IMPRIMANTE 82905B UNE SORTE DE HARD COPIE DE LA PAGE ECRAN TOUTE PROPOSITION, RENSEIGNEMENT, ASTUCE EST BIEN VENUE LANGUAGES POSSIBLES : BASIC, FORTH, FICHIERS LEX.

Philippe DAVASE

PORTFOLIO

Si, comme moi, vous possédez un certain nombre de cartes du PORTFOLIO, ce programme est susceptible de vous interesser. En effet, à partir d'un certain nombre... de ces cartes, vous vous apercevez que vous devez calculer la totalité des différences des 44 valeurs du tableau proposé quotidiennement.

Mais, au fait, je pars du principe que vous connaissez ce nouveau jeu dorénavant célèbre ! Ce jeu grâce auquel vous (Hé oui, vous !) pouvez gagner 30.000 Frs par jour du mardi au samedi et

jusqu'à 100.000 Frs avec le Super tirage du lundi (que de babasses(*) cela représente...).

Néanmoins, pour ceux qui ne connaissent pas encore ce nouvel engouement ludique, il faut savoir que tous les jours, le journal Le Figaro édite une grille différente de 44 valeurs (numérotées de 1 à 44) côtées en Bourse dans laquelle figurent le cours du jour ainsi que le cours de la veille pour chacune de ces valeurs. D'autre part, sur chaque carte personnelle que le joueur possède, sont indiqués 8 numéros différents, compris également entre 1 et 44, qui permettent de se reporter sur la grille pour calculer les écarts correspondants entre cours de la veille et cours du jour.

De plus, quotidiennement ce journal indique le PORTFOLIO du jour qui est un montant à atteindre (par ex. + 80) en additionnant les 8 différences concernées. Le Super PORTFOLIO du lundi est obtenu en additionnant tous les résultats de la semaine précédente pour une carte.

Le programme commence donc par demander de rentrer les différences entre cours de la veille et cours du jour pour les 44 valeurs de la grille. TRUC : il s'agit d'un INPUT, vous pouvez donc directement écrire par ex. 896 - 883 au lieu de vous creuser la tête à calculer mentalement la différence (Toujours moins fort!). Une fois ces différences entrées, le programme effectue la sommation pour chaque carte et imprime les résultats obtenus sur l'imprimante que vous aurez préalablement connectée sur la boucle. Puis s'imprimeront les différences que vous avez entrées ainsi que les numéros des valeurs correspondantes, ceci pour contrôle.

Le programme sauve les résultats obtenus pour chaque carte dans un fichier et vous demande si vous désirez effectuer le calcul du Super PORTFOLIO, c'est-à-dire la somme des résultats du mardi au samedi. Lorsque vous répondrez oui en fin de semaine (après avoir calculé les résultats du PORTFOLIO du samedi par ex...), vous pourrez obtenir 2 types d'impression, selon l'imprimante connectée sur la boucle.

Si vous avez l'imprimante thermique HP-82162A, seuls les résultats du Super-PORTFOLIO apparaîtront face aux numéros des cartes. A ce sujet, je dois préciser que pour reconnaître facilement mes cartes, j'ai pris les 4 premiers chiffres du numéro qui figure en bas à gauche de chacune d'elles. Toute autre convention est laissée à l'appréciation de chacun.

Si vous possédez une Thinkjet, l'impression sera plus complète : le programme trace alors un tableau qui reprend l'ensemble des résultats obtenus durant la semaine, avec bien entendu le Super PORTFOLIO qu'il vient de calculer ! Ceci est réalisé grâce aux possibilités de cette imprimante. C'est la deuxième série de DATA qui permet le tracé du cadre du tableau, la première série correspondant aux cartes du PORTFOLIO avec le numéro de la carte (4 chiffres) suivi des 8 numéros figurants sur chaque carte. Le tout premier DATA est le nombre total de cartes en votre possession.

A la suite de cela, le programme propose une sauvegarde du fichier de la semaine. En répondant par l'affirmative, vous autoriserez le programme à recréer un fichier vierge pour la semaine suivante et à renommer le fichier de la semaine sous la forme:
FOL + Millésime de l'année + Numéro de la semaine (FOL8546).

Mais avant de lancer pour la première fois le programme, n'oubliez pas de créer un fichier TEXT appelé SFOLIO ayant la structure suivante : 2243 000 000 000 000 000 000 2243 est le numéro d'une de vos cartes, suivi de 2 espaces et de 6 séries de 000 correspondants aux jours de la semaine, du mardi au lundi (Pas de tirage le dimanche...). Ce fichier comportera autant de lignes que vous possédez de cartes.

Il ne me reste plus qu'à vous souhaiter bonne chance en espérant que ma prose ne vous aura pas trop ennuyé et que ce programme vous permettra peut-être, si ce n'est de gagner de l'argent, tout du moins de gagner du temps...

(*)BABASSE = Machine à clavier...
PS : Je tiens à préciser que je ne travaille pas au FIGARO, que je n'en possède aucune action et que je n'ai touché aucune commission pour lui faire de la publicité. Qu'on se le dise !... Ceci dit et pour conclure : si vous n'avez pas encore de cartes, écrivez donc au FIGARO pour leur en demander...

jusqu'à 100.000 Frs avec le Super tirage du lundi (que de babasses(*) cela représente...).

Néanmoins, pour ceux qui ne connaissent pas encore ce nouvel engouement ludique, il faut savoir que tous les jours, le journal Le Figaro édite une grille différente de 44 valeurs (numérotées de 1 à 44) côtées en Bourse dans laquelle figurent le cours du jour ainsi que le cours de la veille pour chacune de ces valeurs. D'autre part, sur chaque carte personnelle que le joueur possède, sont indiqués 8 numéros différents, compris également entre 1 et 44, qui permettent de se reporter sur la grille pour calculer les écarts correspondants entre cours de la veille et cours du jour.

De plus, quotidiennement ce journal indique le PORTFOLIO du jour qui est un montant à atteindre (par ex. + 80) en additionnant les 8 différences concernées. Le Super PORTFOLIO du lundi est obtenu en additionnant tous les résultats de la semaine précédente pour une carte.

Le programme commence donc par demander de rentrer les différences entre cours de la veille et cours du jour pour les 44 valeurs de la grille. TRUC : il s'agit d'un INPUT, vous pouvez donc directement écrire par ex. 896 - 883 au lieu de vous creuser la tête à calculer mentalement la différence (Toujours moins fort!). Une fois ces différences entrées, le programme effectue la sommation pour chaque carte et imprime les résultats obtenus sur l'imprimante que vous aurez préalablement connectée sur la boucle. Puis s'imprimeront les différences que vous avez entrées ainsi que les numéros des valeurs correspondantes, ceci pour contrôle.

Le programme sauve les résultats obtenus pour chaque carte dans un fichier et vous demande si vous désirez effectuer le calcul du Super PORTFOLIO, c'est-à-dire la somme des résultats du mardi au samedi. Lorsque vous répondrez oui en fin de semaine (après avoir calculé les résultats du PORTFOLIO du samedi par ex...), vous pourrez obtenir 2 types d'impression, selon l'imprimante connectée sur la boucle.

Si vous avez l'imprimante thermique HP-82162A, seuls les résultats du Super-PORTFOLIO apparaîtront face aux numéros des cartes. A ce sujet, je dois préciser que pour reconnaître facilement mes cartes, j'ai pris les 4 premiers chiffres du numéro qui figure en bas à gauche de chacune d'elles. Toute autre convention est laissée à l'appréciation de chacun.

Si vous possédez une Thinkjet, l'impression sera plus complète : le programme trace alors un tableau qui reprend l'ensemble des résultats obtenus durant la semaine, avec bien entendu le Super PORTFOLIO qu'il vient de calculer ! Ceci est réalisé grâce aux possibilités de cette imprimante. C'est la deuxième série de DATA qui permet le tracé du cadre du tableau, la première série correspondant aux cartes du PORTFOLIO avec le numéro de la carte (4 chiffres) suivi des 8 numéros figurants sur chaque carte. Le tout premier DATA est le nombre total de cartes en votre possession.

A la suite de cela, le programme propose une sauvegarde du fichier de la semaine. En répondant par l'affirmative, vous autoriserez le programme à recréer un fichier vierge pour la semaine suivante et à renommer le fichier de la semaine sous la forme:
FOL + Millésime de l'année + Numéro de la semaine (FOL8546).

Mais avant de lancer pour la première fois le programme, n'oubliez pas de créer un fichier TEXT appelé SFOLIO ayant la structure suivante : 2243 000 000 000 000 000 000 2243 est le numéro d'une de vos cartes, suivi de 2 espaces et de 6 séries de 000 correspondants aux jours de la semaine, du mardi au lundi (Pas de tirage le dimanche...). Ce fichier comportera autant de lignes que vous possédez de cartes.

Il ne me reste plus qu'à vous souhaiter bonne chance en espérant que ma prose ne vous aura pas trop ennuyé et que ce programme vous permettra peut-être, si ce n'est de gagner de l'argent, tout du moins de gagner du temps...

(*)BABASSE = Machine à clavier...
PS : Je tiens à préciser que je ne travaille pas au FIGARO, que je n'en possède aucune action et que je n'ai touché aucune commission pour lui faire de la publicité. Qu'on se le dise !... Ceci dit et pour conclure : si vous n'avez pas encore de cartes, écrivez donc au FIGARO pour leur en demander...

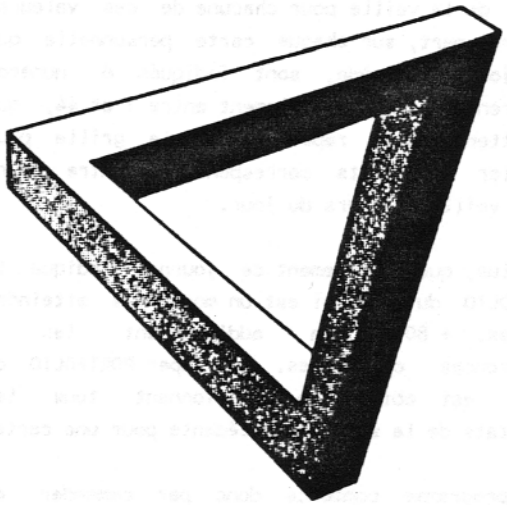
A bientôt

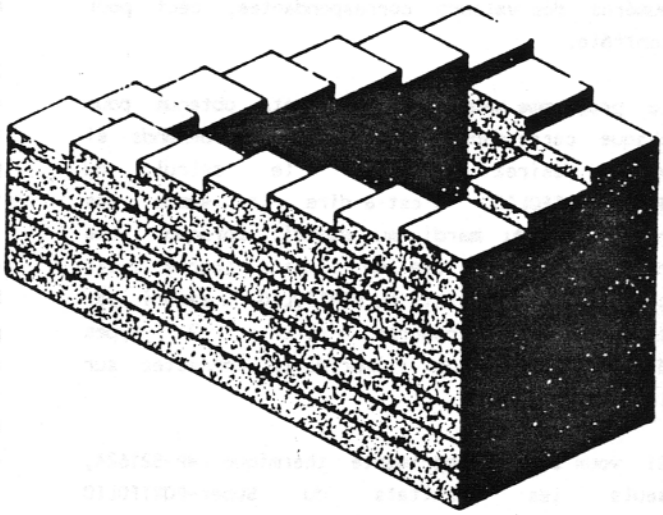
Jean-Marie SIMON (P160, T516)

46, rue de Paradis, 75010 PARIS

Je profite de cet article pour demander à certains d'entre vous, d'une part s'ils utilisent mon programme de navigation aérienne et d'autre part s'ils y ont apporté des modifications ou des améliorations.
En fait ma question est plus générale : y a-t-il des pilotes dans la salle ?...

Merci par avance de bien vouloir satisfaire à ma curiosité.





Programme "FOLIO" (Nécessite EDLEX)

```
  - N()   S()   B0   B1   B2   B3   B4   B5   B6   C0   C1   I   J   T   X   Y
    A$   A1$   A2$   A3$   A4$   B$   D$   NO$
 140 RESTORE IO @ PRINTER IS :PRINTER @ DESTROY ALL @ STD
 160 DIM N(50)
 180 J=MOD(DATE-85315,7) !  85315 correspond à un LUNDI, changer cette valeur tous les ans
 200 IMAGE X,ZZZZ,10X,SDDD
 220 IMAGE ZZZZ,4X,SDDD
 240 IMAGE DD,2X,"=",3X,SDD
 260 FOR I=1 TO 44 @ DISP I;' =  '; @ INPUT ' ';N(I) @ NEXT I
  - IMPRESSION RESULTATS PORTFOLIO
 300 D$=DATE$
 320 PRINT 'PORTFOLIO  DU  '&D$[7]&'/'&D$[4,6]&D$[1,2]
 340 PRINT '======================' @ PRINT
 360 PRINT 'Cartes';TAB(14);'Resultats'
 380 PRINT '------';TAB(14);'---------'
 400 ASSIGN #1 TO SFOLIO
 420 RESTORE @ READ C0
 440 FOR X=0 TO C0-1
 460 READ C1
 480 READ #1,X;A$
 500 T=0 @ FOR Y=1 TO 8 @ READ S(Y) @ T=T+N(S(Y)) @ NEXT Y
 520 B$='000'&STR$(T) @ B$=' '&B$[LEN(B$)-2]
 540 REPLACE #1,X;A$[1,1+4*J]&B$&A$[2+4*(J+1)]

 580 PRINT USING 200;C1,T
 600 NEXT X @ PRINT @ PRINT '======================' @ PRINT
 620 FOR I=1 TO 44 @ PRINT USING 240;I,N(I) @ NEXT I
 640 PRINT @ PRINT @ PRINT @ PRINT @ PRINT

 680 ASSIGN #1 TO SFOLIO
 700 DISP 'Calcul SUPERFOLIO ?'
 720 A$=KEYWAIT$ @ A$=UPRC$(A$)
 740 IF A$<>'O' AND A$<>'N' THEN 720
 760 IF A$='O' THEN 780 ELSE 1960
 780 DISP 'Une petite minute...'
 800 FOR X=0 TO C0-1
 820 READ #1,X;A$
 840 T=0 @ FOR I=1 TO 5 @ T=T+VAL(A$[2+4*I,1+4*(I+1)]) @ NEXT I
 860 B$='000'&STR$(T) @ B$=' '&B$[LEN(B$)-2]
 880 REPLACE #1,X;A$[1,25]&B$
 900 NEXT X @ DISP 'Impression'
  - SUPER-PORTFOLIO SUR THINKJET
 940 T=DEVADDR("HP2225B") @ IF T<0 THEN 'THERMIC'
 960 PWIDTH INF @ PRINT CHR$(12)
 980 DIM A$[90],A1$[90],A2$[90],A3$[90],A4$[90],B$[80],NO$[4]
1000 RESTORE 2240
1020 FOR I=1 TO 80 @ READ K @ A1$=A1$&CHR$(K) @ NEXT I
1040 FOR I=1 TO 80 @ READ K @ A2$=A2$&CHR$(K) @ NEXT I
1060 FOR I=1 TO 6 @ READ K @ A3$=A3$&CHR$(K) @ NEXT I
1080 IMAGE #,7X,B,"&k3S",ZZZZ,B,"&k0S",5X,S3D,4(5X,S3D),10X,B,"&k3S",S3D,B,"&k0S"

1120 PRINT CHR$(27)&'&k1S   PORTFOLIO  DU  FIGARO'
1140 PRINT '   ====================' @ PRINT CHR$(27)&'&k0S' @ PRINT @ PRINT @ PRINT
1160 PRINT TAB(17);'MARDI   MERCREDI  JEUDI   VENDREDI  SAMEDI        LUNDI'
1180 CALL CADRE(1,A1$,A2$,A3$)
1200 RESTORE #1 @ FOR I=1 TO C0-1 @ READ #1;B$
```

```
1220 CALL CADRE(2,A1$,A2$,A3$)
1240 B0=VAL(B$[1,4]) @ B1=VAL(B$[7,9]) @ B2=VAL(B$[11,13]) @ B3=VAL(B$[15,17])
1260 B4=VAL(B$[19,21]) @ B5=VAL(B$[23,25]) @ B6=VAL(B$[27,29])
1280 PRINT USING 1080;27,B0,27,B1,B2,B3,B4,B5,27,B6,27;
1300 CALL CADRE(2,A1$,A2$,A3$) @ CALL CADRE(3,A1$,A2$,A3$)
1320 NEXT I
1340 READ #1;B$
1360 B0=VAL(B$[1,4]) @ B1=VAL(B$[7,9]) @ B2=VAL(B$[11,13]) @ B3=VAL(B$[15,17])
1380 B4=VAL(B$[19,21]) @ B5=VAL(B$[23,25]) @ B6=VAL(B$[27,29])
1400 CALL CADRE(2,A1$,A2$,A3$)
1420 PRINT USING 1080;27,B0,27,B1,B2,B3,B4,B5,27,B6,27
1440 CALL CADRE(2,A1$,A2$,A3$) @ CALL CADRE(4,A1$,A2$,A3$)
1460 PRINT CHR$(12) @ GOTO 1680
   - SUPER-PORTFOLIO SUR HP-82162A

============================================================
1500 'THERMIC':
1520 T=DEVAID("PRINTER") @ IF T<0 THEN 'MESSAGE'
1540 PRINT @ PRINT @ PRINT
1560 PRINT CHR$(27)&'&k1SSUPER PFOLIO' @ PRINT '************'
1580 PRINT @ PRINT
1600 RESTORE #1 @ FOR I=1 TO C0 @ READ #1;A$
1620 C1=VAL(A$[1,4]) @ T=VAL(A$[26])
1640 PRINT USING 220;C1,T
1660 NEXT I @ PRINT CHR$(27)&'&k0S' @ PRINT @ PRINT
1680 ASSIGN #1 TO *

1720 INPUT 'Sauvegarde FOLIO ? ','O';A$
1740 A$=UPRC$(A$) @ IF A$<>'O' AND A$<>'N' THEN 1720
1760 IF A$='N' THEN 1980
1780 B$='FOL'&DATE$[1,2]&STR$(1+(DATE-85000) DIV 7)
1800 RENAME SFOLIO TO B$
1820 CREATE TEXT SFOLIO
1840 ASSIGN #1 TO SFOLIO @ RESTORE @ READ C0
1860 FOR I=1 TO C0 @ READ A
1880 FOR J=1 TO 8 @ READ X @ NEXT J
1900 B$='0000'&STR$(A) @ B$=B$[LEN(B$)-3]
1920 PRINT #1;B$&'  000 000 000 000 000 000'
1940 NEXT I
1960 ASSIGN #1 TO * @ DISP 'Fin...' @ WAIT 1 @ PUT "#43"
1980 END

============================================================
2000 'MESSAGE': DISP "Pas d'imprimante" @ BEEP @ BEEP @ BEEP @ WAIT 1 @ GOTO 1960
   - CARTES  PORTFOLIO
     1er DATA = Nombre de cartes

============================================================
2060 DATA 5
2080 DATA 1654,5,8,10,11,23,31,42,44
2100 DATA 5144,1,3,13,20,27,30,36,42
2120 DATA 3728,3,4,18,22,24,32,40,43
2140 DATA 18,2,6,9,15,21,33,34,39
2160 DATA 299,7,12,14,16,28,29,35,41

============================================================
   - Et ainsi de suite pour les autres cartes...

     DATA POUR CREATION TABLEAU SUPER-PORTFOLIO
```

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

Programme "FORMLIST" (Pour lister en 3 colonnes)


  - programme FORMLIST le 15.12.85  listing/3 colonnes
 20 DESTROY ALL @ DELAY 1,1 @ STD
 30 SFLAG -1 @ PURGE FTEXT @ PURGE BAK @ CFLAG -1
 40 INPUT 'Fichier: ';F$
 50 DISP 'COPIE DE : ';F$ @ COPY F$ TO BAK
 60 DISP 'CODAGE ASCII :';F$ @ TRANSFORM BAK INTO TEXT
 70 DIM A$[256],B1$[50],B2$[55],B3$[45]
 80 CREATE TEXT FTEXT
 90 ASSIGN #1 TO BAK @ ASSIGN #2 TO FTEXT
100 DISP 'FORMATTAGE DE:';F$ @ N=0

============================================================
110 'BOUCLE':
120 ON ERROR GOTO 'FIN'
130 READ #1,0;A$
140 OFF ERROR
150 GOSUB 'FORM1'
160 DELETE #1,0 @ GOTO 'BOUCLE'

============================================================
170 'FIN':
180 OFF ERROR @ PURGE BAK @ ASSIGN #1 TO *
190 PWIDTH 132 @ RESET HPIL @ RESTORE IO
200 PRINTER IS :1 @ PRINT CHR$(27)&"&k2S"
210 DISP 'EDITION DE:';F$ @ N=N-1 @ N1=0 @ N2=59
220 FOR I=N1 TO N2
230 IF I>N THEN 350
240 READ #2,I;A$ @ IF A$[1,1]=CHR$(0) THEN A$="     "&A$[2]
250 PRINT A$;
260 PRINT TAB(43);"I";
270 I1=I+60 @ IF I1>N THEN 340
280 READ #2,I1;A$ @ IF A$[1,1]=CHR$(0) THEN A$="     "&A$[2]
290 PRINT TAB(46);A$;
300 PRINT TAB(89);"I";
310 I2=I+120 @ IF I2>N THEN 340
320 READ #2,I2;A$ @ IF A$[1,1]=CHR$(0) THEN A$="     "&A$[2]
330 PRINT TAB(92);A$;
340 PRINT @ NEXT I
350 PRINT CHR$(12)
360 N1=N1+180 @ N2=N2+180
370 IF N1<N THEN 220 ELSE DISP "FIN DU TRAITEMENT"
380 DESTROY ALL @ ASSIGN #2 TO * @ PURGE FTEXT @ OFF IO @ BEEP @ END

============================================================
390 'FORM1':
400 B1$=A$[1,40] @ B2$=CHR$(0)&A$[41,75] @ B3$=CHR$(0)&A$[76]
410 PRINT #2;B1$ @ N=N+1
420 IF B2$=CHR$(0) THEN 440 ELSE PRINT #2;B2$ @ N=N+1
430 IF B3$=CHR$(0) THEN 440 ELSE PRINT #2;B3$ @ N=N+1
440 RETURN


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

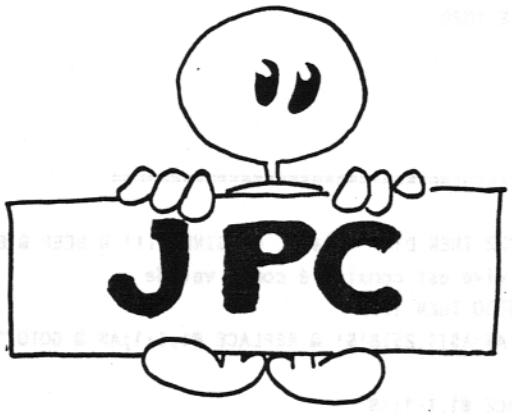
Programme "COPYDISK" (Nécessite les LEX : DESAL EDLEX et KEYWAIT)

```
100 IF MEM<3000 THEN DISP MSG$(24) @ END ! pas assez de mémoire
  - A0   A1   A2   A3   I   J   L   X
     A$   B$   C$   M$   N$   Z$
    initialisation
180 DESTROY ALL @ CFLAG 0,1
200 DIM A$[41],B$[26],C$[4],N$[8],M$[8]

============================================================
220 DEF FNL(X) @ READ #1,X-1;B$ @ FNL=VAL(B$[20,24]) @ END DEF

============================================================
240 DEF FNE(X) @ READ #1,X-1;B$ @ B$=B$[26,26]

============================================================
260 FNE=(B$='*')+2*(B$='#')+3*(B$='@')+4*(B$='&') @ END DEF
  - *=sauvé, #=lu et non sauvé, @=type inconnu, &=déjà en mémoire (ne doit pas être purgé)
300 C$='CATA' @ M$=':MASSMEM'
320 INPUT 'Disqu. Sauvegarde : ';Z$
340 IF Z$='' THEN 320
360 Z$=RED$(UPRC$(Z$)) @ IF Z$[1,1]#'.' THEN Z$='.'&Z$

400 DISP 'Insérer disqu. ORIGINE'
420 A$=KEYWAIT$
  - LECTURE ET TRI CATALOGUE DISQUETTE
460 CREATE TEXT C$
480 ASSIGN #1 TO C$
500 I=1
520 PRINT #1;'NNNNNNNNNN S TTTTT 99999  '
540 PRINT #1;'NNNNNNNNNN S TTTTT 00000  '


============================================================
580 'TRICATA':
600 A$=CAT$(I,M$) @ A$=A$[1,25]&' '
620 IF A$=' ' THEN 'FINCAT'
640 DISP A$[1,10]
660 L=VAL(A$[20,24])
680 FOR J=1 TO I+2
700 IF L>FNL(J) THEN INSERT #1,J-1;A$ @ J=I+2
720 NEXT J
740 I=I+1 @ GOTO 'TRICATA'


============================================================
780 'FINCAT':
800 DELETE #1,0 @ DELETE #1,I-1
820 ASSIGN #1 TO *
840 DISP 'Fin lecture CATALOGUE'
  - LECTURE ET ECRITURE CAT. EN MEMOIRE
880 A0=MEM @ 'BOUCLE':
900 SFLAG 1 ! permet de savoir quand on ne peut plus rien copier
  - recherche du plus gros fichier copiable
940 ASSIGN #1 TO C$

============================================================
960 'LECTURE':
980 FOR I=1 TO FILESZR(C$)
```

```
1000 IF FNE(I)#0 OR FNL(I)>MEM-200 THEN 1180
1020 CFLAG 1 ! ce n'est pas la dernière passe
1040 READ #1,I-1;A$
1060 B$=RED$(A$[1,10])&M$ @ N$=A$[1,8]
1080 ON ERROR GOTO 'ERREUR'
1100 DISP A$[1,10]
1120 COPY B$ TO N$ ! Attention aux noms de plus de 8 caractères
1140 OFF ERROR
1160 A$=A$[1,25]&'#' @ REPLACE #1,I-1;A$
1180 X=FLAG(0,FLAG(1))
1200 NEXT I
1220 ASSIGN #1 TO *
   - E C R I T U R E

================================================================
1260 'DISKOBJ':
1280 DISP 'Insérer disqu. ';Z$[2]
1300 A$=KEYWAIT$
1320 A2=A2+1 @ ASSIGN #1 TO C$
1340 FOR I=1 TO FILESZR(C$)
1360 IF FNE(I)#2 AND FNE(I)#4 THEN 1540
1380 READ #1,I-1;A$ @ B$=RED$(A$[1,10])&Z$ @ N$=A$[1,8]
1400 DISP A$[1,10]
1420 COPY N$ TO B$
1440 IF FNE(I)#4 THEN PURGE N$
1460 A$=A$[1,25]&'*' @ REPLACE #1,I-1;A$
1480 IF A$[12,12]='S' THEN SECURE RED$(A$[1,10])&Z$
1500 IF A$[12,12]='P' THEN PRIVATE RED$(A$[1,10])&Z$
1520 A1=A1+FNL(I)
1540 NEXT I @ ASSIGN #1 TO *

1580 IF FLAG(0) THEN 'FIN'
1600 DISP "Insérer disquette ORG"
1620 A$=KEYWAIT$
1640 GOTO 'BOUCLE'
   - F I N

================================================================
1680 'FIN':
1700 DISP "Fini" @ BEEP @ BEEP @ BEEP
1720 CFLAG 0,1
1740 A3=A1/(A2*A0) @ DISP 'Tx de transfert =';IP(A3*1000)/10;'%' @ WAIT 3
1760 DISP 'Impression CATALOGUE ?' @ A$=KEYWAIT$
1780 IF A$='O' THEN 1800 ELSE 1820
1800 PWIDTH 80 @ PLIST C$
1820 PURGE C$ @ END
   - TRAITEMENT D'ERREUR

================================================================
1860 'ERREUR':
1880 IF ERRN=57 OR ERRN=255022 THEN DISP 'Disqu. ORIGINE !!!' @ BEEP @ E$=KEYWAIT$ @ GOTO 1120
   -  Le fichier en mémoire vive est considéré comme valide
1920 IF ERRN#59 AND ERRN#255030 THEN 1960
1940 DISP MSG$(59) @ BEEP @ A$=A$[1,25]&'&' @ REPLACE #1,I-1;A$ @ GOTO 1180
1960 IF ERRN#63 THEN 2020
1980 A$=A$[1,25]&'@' @ REPLACE #1,I-1;A$
2000 DISP MSG$(63) @ BEEP @ GOTO 'LECTURE'
2020 DISP MSG$(ERRN) @ BEEP @ STOP !  Cas non prévu...
```

```
============================================================
2240 DATA 27,42,98,55,52,87
2260 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,31,255,255,255,255,255,255,255,255,255,255
2280 DATA 255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
2300 DATA 255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
2320 DATA 240,0,0,0,0,31,255,255,255,255,255,255,255,255,255,240
2340 DATA 27,42,98,55,52,87
2360 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,16,0,0,0,0,0,0,0,0,16
2380 DATA 0,0,0,0,0,0,0,0,0,16,0,0,0,0,0,0,0,0,16
2400 DATA 0,0,0,0,0,0,0,0,0,16,0,0,0,0,0,0,0,0,16
2420 DATA 0,0,0,0,16,0,0,0,0,0,0,0,0,16
2440 DATA 27,42,98,49,87,0

============================================================


============================================================
2480 SUB CADRE(A,A1$,A2$,A3$)
2500 ON A GOSUB 'A1','A2','A3','A4'
2520 GOTO 2680

============================================================
2540 'A1': FOR I=1 TO 8 @ PRINT A3$ @ NEXT I @ PRINT A1$ @ FOR I=10 TO 12 @ PRINT A2$ @ NEXT I
2560 RETURN

============================================================
2580 'A2': FOR I=1 TO 12 @ PRINT A2$ @ NEXT I @ RETURN

============================================================
2600 'A3': FOR I=1 TO 8 @ PRINT A2$ @ NEXT I @ PRINT A1$ @ FOR I=10 TO 12 @ PRINT A2$ @ NEXT I
2620 RETURN

============================================================
2640 'A4': FOR I=1 TO 8 @ PRINT A2$ @ NEXT I @ PRINT A1$ @ FOR I=10 TO 12 @ PRINT A3$ @ NEXT I
2660 RETURN
2680 END SUB
```

****



LE COIN DES LHEX

Toute la partie du journal qui précède est écrite quand je commence le coin des Lhex. Et, mauvaise surprise, en préparant cette rubrique je constate un oubli : LINKLEX n'a pas été copié

Conséquence : mes excuses auprès de J.Pierre BONDU je passe quand même la version pour MAKELEX, et le mois prochain, nous difuserons le source.

Ce mois-ci donc :

DECOMLEX	CSWAP	092255
CONTRAST	CONTRAST	001023
SCANLEX	ENTRY$	240072
	TOKEN	240073
	TYPE	240074
CONFIGLX		
FINDLEX	FIND	240075
	MNEMO	240077
ENDUPLEX	ENDUP$	240016
	ENDUP	240017
	EXECUTE	240019
	STARTUP$	240018
SECURELX	SECURE	001079
DATELEX	DATE+	240051
	DDAYS	240052
	DMY	240053
	DOW$	240054
	DOW	240055
	MDY	240056
STRUC1	END	240066
	LEAVE	240070
	REPEAT	240068
	UNTIL	240069
	WHILE	240067
MASERLEX		
SWAPLEX	SWAP	240071
TOTO	STK	093001
EDITLEX	EDIT	001184
LINKLEX	LABEL$	240076

PROGAMME MAKELEX

```
10 CALL MLEX @ SUB MLEX @ SFLAG -1 @ PURGE AH @ INPUT "Nb. d'octets: ";N @LCOFF
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
0060 DISP DTH$(X)[3]; @ INPUT ": ",P$[1,MOD(N,16)];C$ @ GOSUB 90
0070 POKE DTH$(A),C$ @ POKE DTH$(B),A$ @ CFLAG -1 @ END
0080 DISP DTH$(X)[3]; @ INPUT ": ",P$;C$
0090 DISP DTH$(X)[3]; @ INPUT " sm ","---";D$
0100 M=S @ FOR Z=1 TO LEN(C$) @ M=NUM(C$[Z])+M+1 @ NEXT Z
0110 IF D$=DTH$(MOD(M,4096))[3] THEN GOSUB 130 @ S=M @ RETURN
0120 DISP "Erreur de somme" @ BEEP @ P$=C$ @ POP @ ON Q GOTO 30,40,50,60
0130 P$="----------------" @ RETURN
```

DECOMLEX ID#5C 80 octets

0123456789ABCDEF sm

000: 445434F4D4C45485 387
001: 802E002001304068 6CE
002: 5A000C5FFFF00000 A64
003: F710000000000000 D92
004: 091000F934357514 0EB
005: 05FF1FF8118FC1DB 4BB
006: 0480AF2108101492 81A
007: 137D7DAD22031218 B9C
008: F7C210DF1355B0DB F4F
009: 8DA9390111120101 2A9
00A: 11115171CF110151 5FD
00B: 71C131E014D8218D 982
00C: C32F0 AA5

CONTRLEX ID#01 130 octets

0123456789ABCDEF sm

000: 34F4E44525C45485 387
001: 802E003001304068 6CF
002: 8010010717100000 9F9
003: F710000000000000 D27
004: 0F2000DF34F4E445 0BC
005: 25143545711FF8D3 43D
006: 0350037FFFF9FFFF 818
007: 7E2035564702 36F6 B8B
008: E64727163747A3B5 F11
009: B202D202B45495D5 295
00A: FF071358FE0C108F 646
00B: E92208FEE0108FAB A02
00C: 251D44908FE21208 D7F
00D: F3E32083420A2730 0F0
00E: 008FA8C4186D908F 498
00F: EE0108D84A801BEF 852
010: 3E21524B444AA150 BCE
011: 453A1BEF3E21524A F70
012: 4C42957EF 186

SCANLEX ID#E1 232 octets

0123456789ABCDEF sm

000: 353414E4C4548502 36C
001: 802E003001304068 6B4
002: 5D1001E84A400000 A15
003: F920000000000000 D46
004: 094100FF00DB000F 0CC
005: C1009100FB54E445 44D
006: 25954284945F4B45 7CB
007: 4E494745950554A4 B46
008: 1FF80DFD0891418F EFE
009: BC63154417FCC4C3 2A8
00A: 101208F13DB01371 610
00B: 35C2D78FD3DE0136 9C0
00C: 13410B8508FB3940 D38
00D: 87B708AC908D91FB 0F3
00E: 0121CC1214008F27 45C
00F: 130850DA8FC4A406 7F2
010: 0DF8412787F84431 B7B
011: FE96202313B96271 EFB
012: B66962C0F0AD0F05 2A3
013: A0854BF4F4100AD0 641
014: AA0D2328E38F943B 9EE
015: 1118F6AD2F6CA8FB DBC
016: 13B1AF88F74030AF 165
017: 4AFE864802980FF8 516
018: D912F0841275FE8F 8C0
019: 74030134D4AF2248 C35
01A: 0FFAF78F84171B47 FEF
01B: A47813DB8FA90F08 3A3
01C: 0D00DAF415111CF2 73A
01D: 08FE83B18DC32F08 AED
01E: 4127C9ED009A8A84 E8F
01F: 46B6F FBC

CONFGLEX ID#E1 69 octets

0123456789ABCDEF sm

000: 34F4E46474C45485 38C
001: 802E004001304068 605
002: E80001E000000000 A18
003: FE0000000800001F D72
004: F31FF9614000DBDA 127
005: 8FD0F8007DA8FD0F 4FB
006: 8007DA8FD0F8085D 8B0
007: 8FC8D81419F4C34F C6E
008: 4E46494748F0F8FE 024
009: 3F80DB068FE3F80D 3E9
00A: B068FE3F80FE00F 775

FINDLEX ID#E1 310 octets

0123456789ABCDEF sm

000: 6494E444C4548502 377
001: 802E004001304068 6C0
002: 072001EB4B400000 A1B
003: F710002000000000 D4B
004: 0E3000776494E444 0B9
005: B41FF101081108E4 439
006: F6470264F657E646 7C9
007: CFF8DD97309FFFF8 BC4
008: F681F08F13DB08A8 F78
009: E41BB88F21401371 2FF
00A: 35C2CECE1641448F 6AE
00B: 8BB818FEEB604018 A60
00C: F627705D031F38D3 DF1
00D: 93908F9997051120 15F
00E: 3310E58DA93901F8 4F1
00F: E7F2D215F38F4EFF 8CD
010: 05CD137D51378F13 C5A
011: 0018B370D4131DB8 FDA
012: FD3DE01371351088 36F
013: FF6F4020D4C41B1C 723
014: 6F2146C210A1351B A9F
015: 098F214610918414 DFF
016: 6EA4E5AD1D881DCE 1E0
017: 80D0F6D75D2D9061 582
018: 37061371611711DBD 8F8
019: 58FFE1B10713507D C95
01A: 5562CD1191344511 FFC
01B: 4A1C114F9620CCD5 3A1
01C: 2F11A13511AAF0DA 744
01D: 137EA81C2011B135 AC0
01E: 8A8E68F96F401BF8 E84
01F: 5F21461341F1C6F2 20D
020: AD0143EA81C100D9 5AA
021: 067C00B1C3E3B1E3 946
022: FF071358FE0C1007 CE0
023: D58F617901188F1C 07A
024: 6908F898108DE730 40C
025: 08FC2DE0D78F1300 7B5
026: 12088360657E11B1 B27
027: 3710B8FDA67011B1 EB9
028: 354908D84A806F9E 259
029: F 2A0

ENDUPLEX ID#E1 286 octets

0123456789ABCDEF sm

000: 54E4445505C45485 376
001: 802E004001304068 68F
002: 042001E013100000 9F0
003: F230000000000000 D1B
004: 0DC100FF00BE000D 0BA
005: D201B100FC103010 422
006: 0DB54E4445505420 797
007: 1954E444550511D5 B05
008: 485543455455431F E6C
009: 3545142545550542 1B7
00A: 211FF31BF9611131 539
00B: CF9614200D9D7320 8C5
00C: 0A8FFF8113210A8F C6F
00D: FF811FC003200A8F 012
00E: AB8115FEDBD531FC 3EB
00F: 8FC463151C31DC8F 79C
010: C463143B31DC8FAF B56
011: 5311378D626208DD ED6
012: 97308D394509FFFF 288
013: DEFFF8F681F0D231 663
014: 0C10B3200A7B308D 9E7
015: 84A80FCFFF3CFFF8 DE7

```
016: F681F0D2CE10B321 189
017: 0A731057D3210A8F 50A
018: AB8116D8F10A8FE4 8C8
019: 4A131D01C114D850 C41
01A: 75B08F13DB013710 FC3
01B: 8D2E6E68B6E011A8 377
01C: F14A1103D811B8BA 713
01D: 908DA114110111A8 A7D
01E: FD79114908DD4490 E17
01F: 1101301198F771B1 178
020: 0200328085900032 4B2
021: 00A1331018FAB811 827
022: 137136D58F064A11 BA0
023: 111315E114A31D09 F09
024: 6221AE68F4058116 287
025: 156ED91348D7B181 61C
026: F                663

SECURLEX  ID#01  403 octets

     0123456789ABCDEF sm

000: 3554345525C45485 366
001: 802E005001304068 6B0
002: A230010F4F44D200 A25
003: F710000000000000 D53
004: 066000DB35543455 0B7
005: 2554F41FF1FF8FE7 485
006: A205908D2713075F 801
007: F8F5CC30500318F9 BA3
008: 62B031FC96640038 F1E
009: D20F208D547509FF 2C2
00A: FFABFFF84B14A8F5 6BA
00B: 04505711FD55F214 A32
00C: 7135708169C02F30 D9F
00D: E201F088F2155431 118
00E: 8F96273AC215548F 4B4
00F: D2F905908DA93908 851
010: F36F905DB876CE94 C14
011: A7EACBA4E56067A0 FDA
012: A4E4E28F73321B04 373
013: 8C831D5B47A47470 707
014: 97C60694FD231C35 A9E
015: 8AB675E11F088F21 E41
016: 5547E11514D23104 1A5
017: 678F716141F1376F 531
018: 401F088F2157494E 8B7
019: 908D84A80B465B18 C4D
01A: FF20114BE14B9680 FE6
01B: F1376910155475C0 355
01C: 65EF1F855F21471F 6FF
01D: 678F21456C00DA8F AA2
01E: D1F6013514B96CEE E4F
01F: 1FB78F21451F678F 1FB
020: 214313014E96A731 55E
021: C41411317C201F17 8CB
022: 8F2143D23102CA13 C47
023: 0142136CA13014E9 FB3
024: 6E0D6D3F1378F116 356
025: A08F7F5A0460653F 6F6
026: 17F17314B30186BC A7D
027: 030E0E065600E0E1 DF6
028: 510012031EF8FCA9 189
029: 0193800170AB815F 4FD
02A: 7AF717215F217215 87A
02B: 74B46AC7173F2F23 C1A
02C: 08133135BF0BF0BF FBA
02D: 0AB410103A6FDB10 34A
02E: B70AF4005C08FF20 6EC
02F: 1140011B9670F031 A46
030: 0A5A500000F71000 DA0
031: 0000000000062000 0B8
032: DF55E43554345525 43E
033: 54A51FFF5DFF02DF 821
034: F85B8C46DFF       AD6

DATELEX   ID#E1  545 octets

     0123456789ABCDEF sm

000: 44144554C4548502 35E
001: 802E006001304068 6A9
002: 744001E338300000 9EF
003: F440028000000000 D27
004: 0FA000FD00D4100F 0B7
005: A105E300D3200310 413
006: 0FE2037100F730CD 7A2
007: 300D944144554B23 B08
008: 3944441495354354 E5F
009: 4D49553744F47542 1DC
00A: 63544F475735D444 557
00B: 95831FF101001004 8B4
00C: 4414455402CF110B C22
00D: BF4574702F666022 FA3
00E: 516E67656CFF8822 345
00F: 8FC1DB08F322B184 6F1
010: 1401831606F92851 A4E
011: F810017F7DD17FB1 DFB
012: D9110861D08B6CDE 1A9
013: 26500C273C28F533 523
014: 3186080DDDFDD840 8D3
015: AFB2190A50850812 C56
016: 812A798F834B1A72 FEC
017: 8FF24B187080812A 37C
018: 3EB3668408118526 6F0
019: B4088227E5117F10 A6B
01A: 874511109F661B72 DDD
01B: AFA8FB13B1BCC6D0 1BE
01C: 0B7A8FB13B1AF68D 588
01D: 832F08118427A11A 8FE
01E: FACCD23078FE6CE0 CDC
01F: 87221816AD2AA281 063
020: 66CCF17F13310113 3E6
021: 38F6242029021012 736
022: 0030540450960399 A7F
023: 646E657C42969803 E02
024: 996462716D429677 17B
025: 03F9646562736275 4E8
026: 6D42F68503996465 86B
027: 756A4296D403F964 BF7
028: 6562746E656652F6 F7A
029: E203B96465D6163 2FE
02A: 52B1CB61203F5686 688
02B: 36E6160696442F1C A20
02C: F66001C915518F06 D9F
02D: 4A1840208D7B1812 114
02E: 0315E048408FC463 490
02F: 1550850018FC1DB0 811
030: AF878DFAF990A508 BE0
031: 12812812AF721A92 F56
032: B73812812AF5A92B 2F2
033: 718F834B1AFA23A9 6A1
034: 2B7A86080DDDFDDD A81
035: 22031219E1F43170 DE0
036: 9E1A130296123A0E 161
037: 0E05213036B10101 4AF
038: 301A8A0E05B0E111 831
039: 213039E3115C5319 B97
03A: 29EBC05058D91FB0 F44
03B: 9E36FAE9109102D2 2E3
03C: 304058FE6CE08ADA 69E
03D: D11296C71320048F A17
03E: E6CE08A9606EBF11 DD7
03F: 9AE51128F40331AF 16F
040: EAF22034532D89F2 50D
041: F0AF2CEAFE9FE000 8F4
042: 733101E8DA939003 C6D
043: 00023000315E6310 FAA
044: E100002000315E21 2F3
045: 8F3F5318D84A808D 6A0
046: 3035003          805

STRUCLEX  ID#E1  637 octets

     0123456789ABCDEF sm

000: 3545255534C45485 366
001: 802E007001304068 6B2
002: EF4001E246400000 A17
003: FB30087000000000 D61
004: 07B000D2306D100D 0D2
005: 610D7300D520AE20 44A
006: 0D900B8300D554E4 7CF
007: 44249C4541465546 B35
008: 4B25540554144544 E8F
009: 955E44594C454975 216
00A: 8494C454341FF305 598
00B: 001304758494C454 8F8
```

```
00C: C81408E6F6022554 C79
00D: 05541445C3150ECE FFE
00E: 4C454146554CFF60 390
00F: 4001E3001361B698 6F1
010: F21441367A124E02 A62
011: F3092094351313O2 DB5
012: 2311E208DA939087 134
013: DF1321088FAB811D 4DC
014: 81378B74DC98BBDC 8A7
015: DB1B198F21441367 C34
016: 2814311B698F2146 FA4
017: 1366E4013610A1B1 300
018: 98F21422F3097F81 697
019: 11A13686F927A204 A11
01A: 2213610A8F95EF01 D99
01B: 1A1348FB7EF01121 12A
01C: 308D7E47084A86F0 4C1
01D: 086D0087A001F0B7 843
01E: F22030214B0E06A0 BB5
01F: C1FD55F214713517 F42
020: E143F434412E08A2 2C7
021: 400203103005F200 601
022: 79011367AB058213 969
023: 610A1B198F21422F CED
024: 30978D011A1368D8 06D
025: 4A80D2E610A86D01 400
026: 1F765F2147641032 76D
027: 1088FAB811137C2D B05
028: 72031FE8F99A804D EAE
029: 031718D393901613 209
02A: 31E34231521916E0 56C
02B: 11AE610A6D102231 8DF
02C: 2423916F011ACE8A C74
02D: AD010A713066AF16 000
02E: 3659E8F681F08F19 3A7
02F: DB0042E5A091CC01 73F
030: 1091C40030218300 A96
031: 14AD8132C0130018 DFE
032: D31F808DE3F80136 1A0
033: 1B198F214401BD10 51E
034: 0FC1001361B698F2 8A4
035: 1447FCF4E02F30A2 C4A
036: 0943A0314064BD87 FC2
037: DF1321088FAB811D 36A
038: 81378B7FDC98B88D 73C
039: DB1B198F21441641 AC4
03A: 461367C3F560651E E47
03B: 1B198F21421022F3 1B9
03C: 0A69CD5010070100 51E
03D: 1322F30A734F8161 894
03E: 366A6E5E0007E000 C15
03F: 793F7E2F4412F30A FB7
040: 94321309943A0315 310
041: 06D0DD2E610A86D0 683
042: 11F765F214764103 A1F
043: 21088FAB811137C2 DA5
044: D71B198F21461362 124
045: 031FE8F99A804A03 4C4
046: 1506BBC161331E24 841
047: 2315219120322315 B8F
048: 4239120322314423 ECC
049: 9127122313423912 210
04A: A0756E68AF11AE61 5BC
04B: 0A6FEF11ACE10A8A 98A
04C: E1E744E6FAC8D303 D4B
04D: 50038FA2C20FE1E3 0ED
04E: 4E0000208D530301 440
04F: 850339758494C454 7B1
050: 298F324508FCE250 B45
051: 8DE6A208D39450F  EB5

MASERLEX  ID#E1  468 octets

      0123456789ABCDEF sm

000: D414355425C45485 372
001: 802E008001304068 6BF
002: CA3001E000000000 A0C
003: FE0000000800001F D66
004: F31FE96140007512 0DE
005: 59F728242FDB70C1 482
006: 077AB10774B11137 7F5
007: FA11138FA0DE071A BA3
008: 18FA5CE08FC2B901 F5F
009: 631181F5B8F21458 2E2
00A: FBA9908F674A1D91 69C
00B: 3416F1651368B390 A0C
00C: 8DD44903F02C2029 D98
00D: 6E602C696154716F 121
00E: 35E6560225154116 47A
00F: 520D231327731D8C 7F1
010: D314F8F33890D214 B83
011: C136058FC2B90D91 F35
012: 361FF85F21451341 2B3
013: 63136E181DD4E481 643
014: CE4C4C4174136C21 9D9
015: 45D48F42A7118315 D59
016: 833FD41435542545 0CB
017: 0202AFA8F0BF9053 46E
018: 2AF3AF210810A10B 801
019: E61098F1C6114345 B80
01A: 2CD0CC1011331037 EF3
01B: C515708AE72878C1 28A
01C: D212B1338F10FA17 61A
01D: F310521480D23163 97B
01E: 8DA93907E308FD3D D35
01F: E010B703011BDB10 0B2
020: B7420067E1006781 41A
021: 01F5B8F214710800 78D
022: DA8DD0F808FE3F80 B64
023: DB03D51F34CF2147 F0B
024: 8FAB811500D23243 28D
025: 58A24001133C0131 5E1
026: 143021F855F21471 949
027: 3515373F64F42545 CBE
028: 8425140497250500 01A
029: 17FD015B334912E0 39F
02A: E2CE470CE50017F1 745
02B: 741431F855F21478 ABD
02C: A2505001FC0BF214 E43
02D: 7CECECE01D2755F5 215
02E: D2147CECE400D231 5B5
02F: 147E3FCC4001FF4C 960
030: F2147A6E54003021 CDC
031: 3117F1738FD1F60D 07C
032: 7174137111701050 3BA
033: 0D28680031630210 706
034: 1AF0CC1008478487 A93
035: 120400134110E410 DCE
036: 01198A2758780013 12B
037: 65DDD11358BF4417 4CC
038: 31378B7E315B38F4 860
039: 2A7123A98D9B1520 BE9
03A: 400DD81EE6C6C613 F94
03B: 3131C28BB5085803 30A
03C: 85702CD85802F    5F1

SWAPLEX   ID#5E  149 octets

      0123456789ABCDEF sm

000: 35751405C4548502 35D
001: 802E009001304068 6AB
002: E2100E5010100000 9EF
003: F710000000000000 D1D
004: 0E7000D735751405 082
005: 011FF76208FB3940 40F
006: 8FEA2305F371108F 7AD
007: E7A205138D271308 B26
008: 488FE05304508588 EA3
009: 62008FA663086800 20F
00A: CD909008DB2E208D 5BB
00B: 082509FFFFB9FFF7 99F
00C: F708F871F08F0B7F D5D
00D: 01611321008FD0F8 0CC
00E: 01101308F681F08F 443
00F: 8F5F08FE3F80DB13 80A
010: 47F208F8BB817230 B9C
011: 8F681F08F0B7F072 F45
012: 1015378F8F5F08D8 2E1
013: 4A808FE45918DBD3 699
014: B186F008D53AF0F  A10

TOTOLEX   ID#93  129 octets

      0123456789ABCDEF sm

000: 45F445F4C4548502 385
```

```
001: 802E009001304068 6D3
002: 6010039101000000 9F8
003: F710000000000000 D26
004: 032000D53545B410 07C
005: 1FF8D91FB0EB0000 42E
006: C0008F681F08F322 7B9
007: B1CC8148AC9D8101 B62
008: 3606D21F679F215F EF8
009: 015901B675F2EE40 282
00A: 4DAD73068F943B1D 634
00B: 8146169142C0131E 99B
00C: 08F401B1AF223303 D15
00D: CF4921C515D553F8 0BB
00E: 148F1A610DA14613 43B
00F: 5E2D58F861B1241B 7D9
010: 085F28FA8F800713 B6E
011: 68F84A808D394508 F00
012: DE6A20F          0AF

EDITLEX   ID#01  270 octets

     0123456789ABCDEF sm

000: 54449445C4548502 366
001: 802E000101304068 6AC
002: 12200108B8B00000 9F6
003: F710000000000000 D24
004: 0910005754449445 071
005: 8B1FF3A100342300 3E9
006: 0218FB3510446AF3 76D
007: 33020210814A313F AC3
008: 96611161873E1741 E26
009: 1504310F9E2E0873 1A1
00A: C48D335A0310D966 52D
00B: 11307A87161AF056 8A3
00C: 17A015C08FD2F904 C3C
00D: 168138FA54808739 FBA
00E: 0853688F60A0161A 336
00F: F08FA54808438FB6 6E1
010: 480817AFB10B8F36 A80
011: F90545866D511BAF E25
012: 7102D231138F4C48 19E
013: 04C611113111A15D 500
014: F17F33412E15D317 88B
015: F174AF215D917931 C1E
016: 0F14D1318F5A5A08 FB4
017: D84A808F96280402 33A
018: 11913551E817B474 6AA
019: 6EA4750E33A3008D A41
01A: A93908FB06A08F45 DE5
01B: 1101FD55F2147135 159
01C: 1537330202108031 490
01D: 4A31FC966001613F 818
01E: B656973702020202 B6D
01F: AC3AFA0178304E25 F0F
020: 508588F5CC305908 29B
021: D20F20878918FA2C 63E
022: 203FEDFFF008D271 9FA
023: 30038FE7A204005C D79
024: E                DBF

LINKLEX   ID#E1  208 octets

     0123456789ABCDEF sm

000: C494E4B4C4548502 392
001: 802E000101304068 6D8
002: 5A1001EC4D400000 A44
003: F020000000000000 D6C
004: 09E000FF0082000F 0E6
005: BC4142454C442C49 470
006: D4E454D4F4D41FF4 835
007: 118F13DB0137135C BB6
008: 20681C1018507270 F09
009: 444AE531A2961831 283
00A: 19E2D5D231818B16 611
00B: 231308B522071351 95E
00C: CFD48FB13B1AF68D D3F
00D: 832F0D142E840712 0B8
00E: 05C0111E4D85EC85 452
00F: 07D004EE119E2D55 7EB
010: AB3102AE5CC40014 B7E
011: F17187090965DE01 F03
012: 9616E014118F13DB 28D
013: 0AF1D881DD2E68BD 65D
014: 0514331D3966A0CD 9E2
015: 171143312088B5333 D31
016: 068B1B2313296222 094
017: 8F670B1581312096 405
018: 5B133853523916E0 772
019: 137C9C9137D120A6 AFE
01A: 58F064A1C9109850 E80
01B: 8F7B1818FE83B18D 23E
01C: C32F0            361
```

PPC PARIS CHAPTER

ASSOCIATION REGIE PAR LA LOI DE 1901, ENREGISTREE A PARIS LE 2 DECEMBRE 1982 SOUS LE NUMERO 82/3240

BULLETIN D'ADHESION

NOM |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
PRENOM |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| DATE DE NAISSANCE |_|_|/|_|_|/|_|_|
ADRESSE |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
COMMUNE |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
CODE POSTAL |_|_|_|_|_| PAYS |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
TELEPHONE DOMICILE |_|_|/|_|_|_|_|_|_|_|_|_|_| BUREAU |_|_|/|_|_|_|_|_|_|_|_|_|

PROFESSION ____________________
INTERETS ____________________
____

MATERIEL HP EN VOTRE POSSESSION ____________________
____
____

AUTRE MATERIEL MICRO-INFORMATIQUE ____________________
____
____

COMMENT AVEZ-VOUS CONNU PPC PARIS CHAPTER ?
PUBLICITE ____________ MAGAZINE ____________
AUTRE CLUB ____________ HP ____________
RELATIONS, MEMBRES DU CLUB, AUTRES ____________

QUE RECHERCHEZ-VOUS AU SEIN DU PPC PARIS CHAPTER ? ____________
____
____
____

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

DATE |_|_|/|_|_|/19|_|_|
SIGNATURE, PRECEDEE DE LA MENTION "LU ET APPROUVE" ____________

LE MONTANT DE LA COTISATION AU PPC PARIS CHAPTER S'ELEVE A 300.00 FF.
ETUDIANTS: 250.00 FF. (JUSTIFICATIF INDISPENSABLE)
PAIEMENT EFFECTUE LE |_|_|/|_|_|/19|_|_| A L'ORDRE DE PPC PARIS CHAPTER.
PAR [ ] CHEQUE BANCAIRE N° ____________ BANQUE ____________
[ ] CHEQUE POSTAL 3 VOLETS N° ____________
[ ] MANDAT LETTRE
EVENTUELLEMENT: JE M'ABONNE A COMPTER DU |_|_|/|_|_|/19|_|_|
JOINDRE A VOTRE INSCRIPTION UNE PHOTO D'IDENTITE ET UNE ENVELOPPE TIMBREE A VOTRE ADRESSE.

VEUILLEZ ENVOYER TOUTE CORRESPONDANCE A:
MR PHILIPPE GUEZ, 56 RUE J.J. ROUSSEAU, 75001 PARIS (FRANCE)

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

La maquette de ce numéro a été préparée par Jean-Jacques DHENIN et Philippe GUEZ. Elle a été réalisée par Jean-Jacques DHENIN et Pierre DAVID sur un système comprenant un HP-71B, un lecteur de disque HP9114A et une imprimante "Laserjet".

Directeur de la publication Philippe GUEZ.
Numéro ISSN : 0762 - 381X.