

JPC

Octobre 1985

Numéro 28

Le numéro: 35 FF

A PROPOS DU CLUB

MICHEL MARTINET
PIERRE DAVID
JACQUES BAUDIER
PIERRE DAVID
JEAN-PIERRE BONDU
JEAN-PIERRE BONDU
PHILIPPE GUEZ

- | | |
|---|---------------------------------|
| 1 | EDITORIAL |
| 2 | PPC PARIS SE REUNIT |
| 3 | DU NOUVEAU CHEZ HEWLETT-PACKARD |
| 3 | MODULES MEMOIRES POUR HP71B |
| 5 | CHOC EN RETOUR |
| 6 | L'OPTIMISATION |
| 7 | COURRIER DES LECTEURS |
| 8 | PETITES ANNONCES |

HP41

FREDERIC POUPON
HP ET PHILIPPE GUEZ

- | | |
|----|----------------------|
| 10 | TRANSFERT EN XMEMORY |
| 11 | NOUVEAU MODULE HP41 |

HP71

SERGE VAUDENAY
MICHEL WEIL
MICHEL WEIL
JACQUES BAUDIER
JACQUES BAUDIER
JEAN-PIERRE BONDU
SERGE VAUDENAY
ALAIN HERREMAN
ALAIN HERREMAN
LAURENT ISTRIA

- | | |
|----|-----------------------------------|
| 16 | REMORDS |
| 16 | CHOC EN RETOUR |
| 16 | TEST DE KOLMOGOROV ET DE SMIRNOV |
| 17 | TRANSLATER PROGRAM |
| 19 | PASSWORD ! |
| 22 | ECREMEZ VOS PROGRAMMES ! |
| 22 | EQUATE |
| 22 | DEVELOPPEMENTS LIMITEES EN R.P.N. |
| 23 | HP71: TELLEMENT COMPLEXE ! |
| 24 | UN BRIDGE CE SOIR ? |

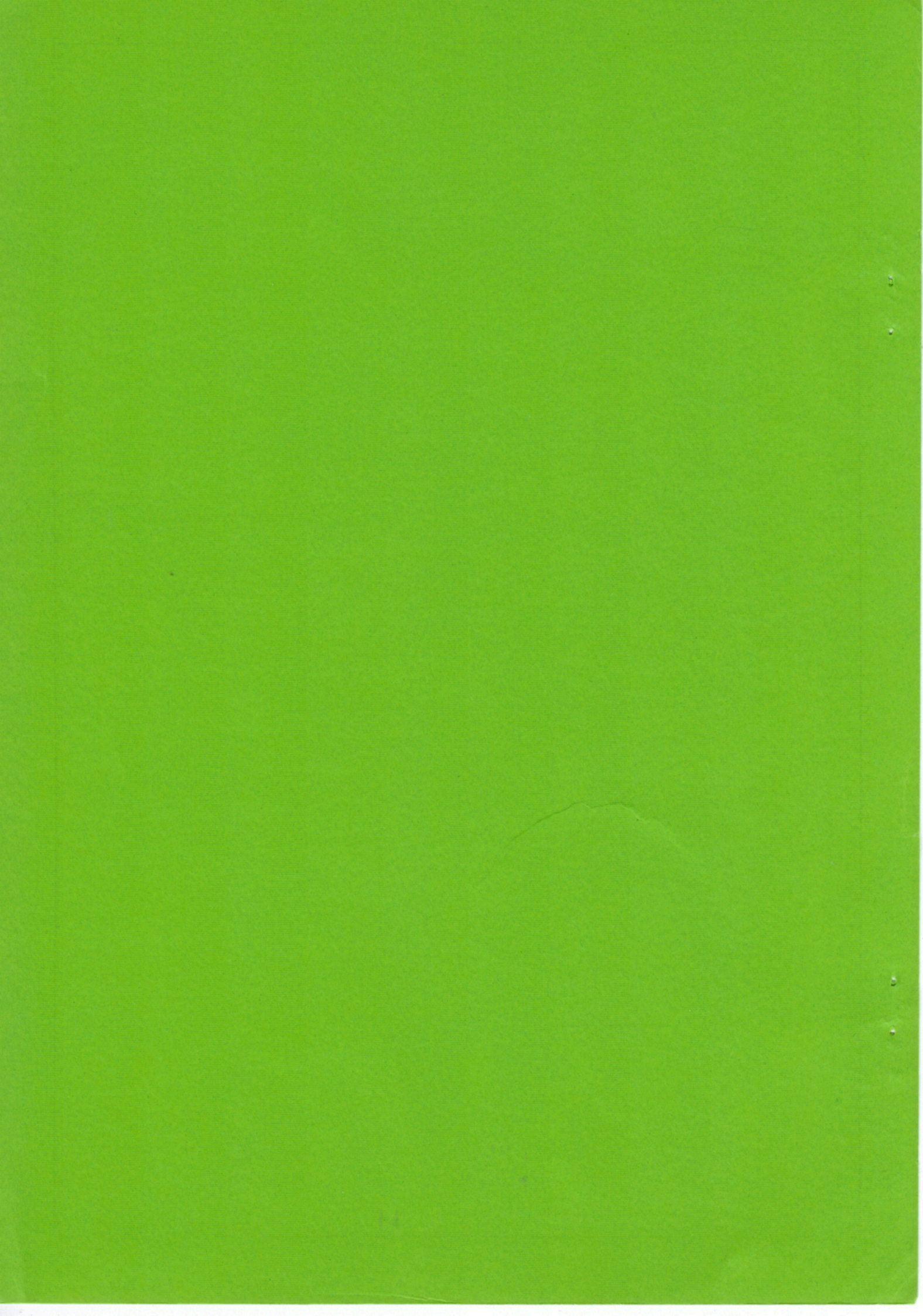
HP71 ASSEMBLEUR

SERGE VAUDENAY
SERGE VAUDENAY
LAURENT ISTRIA
MICHEL MARTINET

- | | |
|----|--------------------------------|
| 36 | ASSEMBLEUR: STOP, ENCORE |
| 37 | GANYMEDE (OTHELLO) |
| 40 | DES DATES, ENCORE DES DATES... |
| 46 | LE COIN DES LHEX |

PPC-PARIS

- | | |
|----|-------------|
| 53 | COOPERATIVE |
|----|-------------|



EDITORIAL

Chers gentils membres,

C'EST LE DRAME ! Nous sommes en retard ! JPC, encore JPC et toujours JPC.

Avec nos manies de sans cesse vouloir faire mieux que précédemment, nous avons décidé de changer d'imprimeur après le retard et la piètre qualité du numéro de septembre.

Mais hélas, la pire des choses nous arrive: nous sommes encore plus en retard. j'écris ces lignes dramatiques le 20 octobre, date à laquelle le numéro de novembre devrait être presque terminé ! De plus, ce changement nous oblige aussi à retrouver l'ancienne formule de 50/55 pages...

Allez ! Je suis sûr que nous sommes déjà tout excusé. C'est promis : d'ici le numéro de décembre/janvier nous serons dans les temps.

Nous sommes toujours de plus en plus nombreux à PPC-PC et surtout vous êtes très nombreux à nous écrire ou à appeler Philippe. Nous allons avoir bientôt une ligne téléphonique équipée d'un répondeur automatique entièrement consacrée à vos appels et à vos questions. Nous vous en dirons plus dans le prochain numéro.

Je vous laisse lire votre journal préféré, il s'est déjà fait assez attendre et il est tout plein de bonnes choses...

Michel MARTINET

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 samedi de chaque mois, 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 19 h.

Pour l'accès en métro, trois possibilités s'offrent à vous à savoir:

Métro Strasbourg St Denis: sortie Porte St Martin / Bld St Denis, coté pairs.

Métro République: sortie Bld St Martin, coté pairs.

Métro J. Bonsergent: Sortie Bld Magenta, coté impairs.

J'oubliais: JPC est souvent distribué en avant-première lors de ces réunions ... A bon entendeur Salut !

Les dates des prochaines réunions sont: (16h-19h)

- Le Samedi 16 novembre 1985

- Le Samedi 7 décembre 1985

A Bientôt...
Pierre DAVID

DU NOUVEAU CHEZ HEWLETT-PACKARD

Le SICOB est toujours le moment où les constructeurs présentent leurs nouveautés. Cette année encore, HP n'a pas manqué à cette règle. Je vais vous parler maintenant de trois nouveaux produits HP, deux étaient au SICOB, le troisième a fait son apparition aux Etats-Unis.

Commençons par le plus gros; le nouvel HP VECTRA est un ordinateur compatible IBM/AT. Il est équipé du microprocesseur Intel 80286 pédalant à 8 Mhz, d'unités à disquette 5 pouces 1/4 de 1,2 Mo et d'unités 3 pouces 1/2. Quand au prix, HP annonce 30% de moins qu'Ibm !!

Deuxième produit, le nouveau PORTABLE PLUS qui n'est autre qu'une version améliorée du Portable HP110. Il comprend un écran à cristaux liquides de 25 lignes de 80 caractères soit 200 par 480 points de résolution. La RAM peut atteindre 896 Ko et la ROM 3 Mo. Il accepte en ROM les logiciels LOTUS 1.2.3, MS-WORD, Memo Marker/Time Management, Executive Card Manager, PC 2622 (programme d'émulation des terminaux HP2622). Sa masse est d'environ 5 kilos et il utilise pour des raisons de fiabilité un disque électronique au lieu d'un disque classique. Le temps d'accès aux données est 10 fois plus court et l'alimentation se fait par batteries. En version de base (192 Ko ROM/128 Ko RAM) il coûte environ 25000 F.HT et dispose d'une interface HPIL lui permettant de se raccorder à la fameuse ThinkJet et au drive HP9114A.

Enfin, le meilleur pour la fin : un nouveau module pour HP41. Ce module vous a déjà été annoncé par Eric GENGOUX PPC #108 dans le numéro de Juillet/Aout. Il est déjà en vente aux Etats-Unis au prix de 49\$! et est donné à tout acheteur d'un calculateur de la série 40 depuis le 1er Août et jusqu'au 15 Novembre. Dépêchez-vous !

J.BAUDIER PPC#192

NDLR: Ceci est valable uniquement aux Etats-Unis mais renseignez-vous. Souvent, avec un petit coup de téléphone plus un numéro de carte VISA,

les achats directs aux Etats-Unis sont moins chers que chez nous (T.T.C.) !

MODULES MEMOIRES POUR HP71

La firme HHP, (Hand-Held Products, ne pas confondre avec HP) aux Etats-Unis, a développé et commercialise des modules mémoire pour le HP71. Ils existent en plusieurs versions:

- 96 Ko Ram	999 \$
- 64 Ko Ram	695 \$
- 32 Ko Ram	399 \$
- 32 Ko Ram + 32 Ko Eprom	495 \$
- 32 Ko Eprom	99 \$

Ils viennent s'enficher dans le logement du lecteur de cartes, d'où incompatibilité si vous possédez déjà ce dernier. La procédure à suivre est classique: éteindre le HP71, enlever le cache du logement de lecteur de cartes, rentrer le module en alignant les broches émergeant du circuit imprimé, appuyer sur le module de manière à le caler dans cette position, remettre le couvercle du logement (pas le cache). La suite est tout aussi classique: rallumer le HP71 avec la fébrilité digne de ce moment, voir le Memory Lost qui peut intervenir, et taper avec anxiété l'ordre MEM. Contempler enfin le chiffre 115244 (Avec le 96 Ko bien entendu) avec toute la satisfaction que procure une affaire rondement menée !

Pour minimiser le risque de Memory Lost inhérent à ce type de situation, il faut éviter d'appuyer sur une des touches du HP71 alors que la manipulation est en cours. Le risque existe toujours, mais est diminué.

Extérieurement, votre HP71 est un HP71 normal, à ceci près qu'il y a à la place du lecteur de cartes un pavé noir, avec écrit en lettres dorées:

71 M/M HAND HELD
96K RAM PRODUCTS

Malgré tous les efforts faits par HHP, la

présentation n'a pas le fini et la qualité HP. Les différences sont minimes, mais perceptibles. Le noir est juste un peu trop noir, et les lettres dorées ont tendance à s'effacer à la longue. Je sais bien que ce n'est pas un critère qui influera sur la décision d'achat ou pas, mais ne frottez quand même pas trop votre module !

Passons à des considérations plus informatiques: les quelques dizaines de milliers d'octets supplémentaires font partie de la Ram au même titre que les 4 Ko des modules HP. Nous parlerons ultérieurement des Eproms. Cette mémoire principale peut aussi être déclarée "mémoire indépendante" (FREE PORT). Ce ne sont plus 4096 octets qui disparaissent alors, mais 32768 ! En effet, les mémoires indépendantes font 32 Ko. Un module de 96 Ko est en fait constitué de 3 modules de 32 Ko, qui sont déclarés indépendants par FREE PORT(5.0), FREE PORT(5.01) et FREE PORT(5.02). Un module de 64 Ko est situé en 5.0 et 5.01. Enfin, un module 32 Ko est en 5.0 uniquement.

Les versions 32 Ko Ram et 64 Ko Ram disposent d'une pile intégrée qui assure, selon le constructeur, 5 ans de bons et loyaux services en "utilisation normale".

Les Eproms sont directement considérées comme des ports indépendants de 32 Ko, ou plus simplement comme un module de mémoire morte logé dans le port 5. Il est intéressant de noter que ce port occupe toujours 32 Ko (ou 64 K-quartets) d'espace adressable, quelle que soit la taille de l'Eprom qui y réside. Le type d'Eprom nécessaire est la 27256, bien que l'on puisse insérer des plus petites capacités. Le temps d'accès doit être inférieur à 200 ns. Voilà pour les considérations techniques.

Mais, me direz-vous, programmer une Eprom demande un matériel coûteux, un logiciel qui ne l'est pas moins, et le tout est hors de portée d'un amateur ? Alors, dois-je me contenter d'Eproms toutes faites, fabriquées et vendues par une quelconque société ?

Non ! PPC-Paris devance vos souhaits. Je suis sûr que vous serez nombreux à acheter un de ces modules. Je peux déjà vous annoncer que vous pourrez acheter une Eprom "standard" (regroupant des programmes parus dans JPC), ou définir vous même ce que vous désirez voir figé. Le coût sera

minime: il sera constitué d'une part du prix de l'Eprom vierge, et d'autre part d'une petite participation aux finances du Club. Il vous sera ainsi possible de définir votre propre module. Les modalités exactes seront exposées le mois prochain. Réfléchissez dès à présent à ce que devrait contenir un module "standard", et faites nous part de vos désirs, sinon nous serons obligés de choisir nous-même...

La version dont je dispose possède 32 Ko de Ram et 32 Ko d'Eprom. Du fait de la présence de l'Eprom, on doit pouvoir ouvrir le boîtier pour la remplacer. Il est retenu par un morceau de ruban adhésif qu'il ne faut surtout pas enlever: il a été, paraît-il, spécialement conçu. Le boîtier s'ouvre de l'autre côté, par une petite encoche. Nous découvrirons alors deux petits circuits imprimés l'un au dessus de l'autre. Le premier porte 4 Ram 64 Kbits Toshiba (8431 HM6264 LFP-15 T0000330), et un circuit plein de pattes que je n'ai pu identifier car le deuxième circuit est posé dessus. Ce dernier contient le support pour l'Eprom, ainsi qu'un circuit d'origine HP dont la référence a été grattée. La confiance règne... En outre, deux "Jumpers" sont prévus, un pour les Eproms 32 Ko, et un pour les 16 Ko et moins. En tous les cas, la place occupée par l'Eprom est fixe à 32 Ko. Seuls, deux composants discrets s'opposent à toutes ces pattes.

Je ne m'étendrai pas sur les avantages que procure un accroissement de mémoire. Pourquoi ais-je choisi un panaché de Ram et d'Eprom, et non pas simplement 32 Ko de Ram ? Pour de multiples raisons. Deux parmi d'autres:

- Beaucoup de programmes sont très utiles, mais hélas encombrants.

- Le message "Memory Lost" apparaît très souvent sur mon HP71.

Ce sont des raisons qui me sont personnelles, mais beaucoup d'entre vous regrettent de ne jamais avoir le programme qu'il faut au moment ou il faut. Dans ce cas, l'Eprom me paraît être une excellente solution.

Pour celui qui est impressionné par l'étonnante

quantité de programmes parus dans JPC, et qui ne peut pas donner beaucoup de temps à son HP71 pour le programmer, l'Eprom seule me semble le meilleur choix. Enfin, pour les avides de mémoire, les modules 32, 64 ou 96 Ko sont là pour déléster votre compte en banque...

A l'heure actuelle, la seule voie pour acheter ces modules consiste à les commander directement aux Etats-Unis. Deux adresses utiles qu'il est bon de connaître: Hand Held Products, Inc. P.O. Box 2388 Charlotte, NC 28211 U.S.A. Tél: (704) 541 1380 et EduCALC Mail Store 27953 Cabot Road Laguna Niguel, CA 92677 U.S.A. Tél: (714) 831 2637. Comptez 20% de taxes et frais de douane divers, et un délai d'au moins une ou deux semaines.

Il est question que ces produits soient importés en France. Nous vous tiendrons bien sûr informés.

Alors, acheter ou ne pas acheter ? La situation est claire: d'une part des modules HP, 4 Ko hors de prix, et de l'autre côté, de 32 à 96 Ko de Ram, et éventuellement 32 Ko d'Eprom. Indépendamment des caractéristiques techniques de ces appareils, le problème est de savoir si on veut (peut) avoir de la mémoire. Il n'y a pas d'équivalent à l'heure actuelle. C'est mon avis, et je le partage, le HP71 avec 16 Ko est trop limité.

Si l'investissement initial peut paraître élevé, je puis vous assurer qu'il en vaut la peine. Passer de 16 à 48 Ko, ça vous change un HP71... et son heureux propriétaire !

A Bientôt,
Pierre DAVID

CHOC EN RETOUR

La lecture des derniers JPC m'a permis de constater que certains des articles dont je suis l'auteur sont bogués. Désagréable ! J'espère que personne ne me lit ... Que les quelques malheureux qui ne sont pas concernés par cette remarque m'excusent et prennent note des rectificatifs suivants. La morale de cette histoire est qu'il vous suffit de venir aux réunions mensuelles pour éviter de pareils déboires... En allant du plus récent au plus ancien:

Prog: EDITEUR (BASIC) JPC No 27 SEPT 85
Remplacer tous les RESTORE#9,INF par des RESTORE#9,1048575

Prog: ADR-AFF (FORTH) JPC No 25 JUIN 85
Ici il s'agit d'un oubli: celui du listing FORTH de la fonction ADR-AFF. Je vous le livre donc:

HEX

: INTERVALLE (INF SUP VAL ...)
DUP ROT 1+ (INF VAL VAL SUP+1 ...)
< (INF VAL b1 ...)

IF

1+ < (b2 ...)

ELSE

2DROP

0

THEN

;

: ADR-AFF (VAL ...)

1- DUP 0 2D ROT INTERVALLE

IF

2* 2E104 +

ELSE

DUP 2E 5D ROT INTERVALLE

IF

2E - 2* 2E200 +

ELSE

DUP 5E 83 ROT INTERVALLE

IF

5E - 2* 2E300 +

ELSE

DROP 0

THEN

THEN

THEN

;

Maintenant vous pouvez comparer la vitesse d'exécution si cela vous chante. Pour saturer l'affichage, il suffit de définir le mot:

DECIMAL : FULL-AFF

255 132 0 DO DUP I ADR-AFF C! LOOP DROP ;

Prog: XERR (ASS) JPC No 25 JUIN 85

La fonction utilisateur correspondant au Lex est:

DEF FNE(X)=X=ERRN OR X=ERRN+128+872*(X<1000))

Vous remplacerez: IF ERRN=57 OR ERRN=1057...

par: IF FNE(57)...

Au plaisir de vous lire
Jean-Pierre BONDU (33)

L'OPTIMISATION

Optimiser un programme, c'est gagner en volume et/ou en vitesse d'exécution. Ce sujet devant intéresser bon nombre de membres, j'ai décidé de vous faire profiter de mon expérience dans ce domaine. Il s'agit soit d'astuces personnelles, soit de 'trucs' piochés dans des documents HP (à commencer par les IDS). Je n'ai pas la prétention de détenir les idées les plus efficaces, et je vous engage vivement à me faire part de votre opinion dans un prochain JPC, ou lors d'une réunion. En avant !...

1) A partir de 3 accès à un même numéro de ligne, remplacez celui-là par un label à une lettre. Ainsi GOTO 150 deviendra GOTO 'S' par exemple. Gain de 2 octets par adressage.

2) Cette méthode (label d'une lettre) peut vous permettre de développer un adressage du type GOTO CHR\$(64+X) qui remplacera avantageusement un ON X GOTO 'A','B','C',... Ici $1 \leq X \leq 26$.

3) Les tests sont grands consommateurs de mémoire et de temps, aussi doivent-ils retenir toute notre attention. Remplacez tous les:

IF cond THEN A=1 ELSE A=4 par
A=4 @ IF cond then A=1, ou encore mieux par
A=(cond)+4*(NOT cond). Le mot 'cond' est l'abréviation de 'condition' et représente l'objet de votre test.

4) En assembleur, et ceci tant que les arguments sont inférieurs à #FF, remplacez LC(5) #1E par: C=0 A et LC(2) #1E

5) Enfin, pensez à déclarer vos variables et à les dimensionner correctement. Rien ne sert de chasser l'octet si par ailleurs vous utilisez 32 caractères (taille par défaut) pour stocker le nom d'un fichier!

6) Ce point m'amène à traiter le cas des fichiers DATA. Vous avez constaté que le 71, dans son infinie bonté, nous gratifie de variables aussi diverses que INTEGER, SHORT ou REAL. La précision étant inversement proportionnelle à l'encombrement mémoire. Mais tous ces avantages sont perdus lorsque l'on écrit ces variables dans un fichier, puisque 8 octets leurs sont réservés, indépendamment de leur type. Pour être tout à fait clair, cela signifie que chacune est transformée en type REAL. Un tableau INTEGER T(8,8) occupe la même place que son collègue REAL T(8,8). La méthode que j'utilise est basée sur l'emploi abusif de POKES.

Admettons que vous n'ayez à stocker que des valeurs comprises entre 0 et 3500. Trois quartets suffiront pour chacune puisque HTD('FFF')=4095. Ce qui représente un gain de $16-3=13$ quartets par valeur. J'avais 1800 valeurs, soit une économie de $1800*13/2=11700$ octets. Je ramenaï ainsi la taille du fichier de 14400 à 2700 octets. Voici le procédé.

Appelons le fichier TOTO. Créons-le en reprenant les données de mon exemple (2700 oct):
CREATE DATA TOTO,1,2700. Pour obtenir l'adresse à laquelle nous rangerons la première donnée, faisons ADDR\$('TOTO')+45. Je rajoute 45 afin de sauter la zone dans laquelle résident les informations inhérentes au fichier: date de création, type, taille, etc... Notons ADR cette adresse. La valeur No1, que nous noterons V1, sera écrite par un POKE DTH\$(ADR),DTH\$(V1). Pour V2, sachant que nous avons réservé 3 quartets par valeur, ce sera POKE DTH\$(ADR+3),DTH\$(V2). Pour la Nième valeur, POKE DTH\$(ADR+(n-1)*3),DTH\$(Vn). La lecture se fait aussi simplement:

Vn=HTD(PEEK\$(DTH\$(ADR+(n-1)*3),3)). Il est très simple alors de se fabriquer les routines suivantes:

```
10 SUB WRITE(n,X)
20 A=HTD(ADDR$('TOTO'))+45
30 POKE DTH$(A+(n-1)*3),DTH$(X)
40 END SUB

60 SUB READ(n,X)
70 A=HTD(ADDR$('TOTO'))+45
80 X=HTD(PEEK$(DTH$(A+(n-1)*3),3))
90 END SUB
```

Bien sûr, le temps d'accès en souffre. Tout est question de choix ! Une dernière remarque: lorsque vous créez un fichier il est initialement rempli d'octets ayant pour valeur 255. Si vous désirez travailler sur un espace 'vierge', il vous faudra POKer une série de zéros.

7) Retour à l'assembleur. Je vous suggère de comparer les vitesses d'exécution des 2 routines suivantes:

I	II
P= 0	P= 0
LC(5) #FFFF	LC(5) #FFFF
bcl C=C-1 A	bcl C=C-1 A
?C#0 A	
GOYES bcl	GONC bcl

Vous constaterez que la première nécessite 41 sec. pour tourner, la deuxième 28. Significatif non ?

Pour conclure, car je fatigue, connaissez-vous l'intérêt de RENUMBER 1,1,1,1 ? Ces paramètres d'apparence loufoque permettent de 'compiler' les adressages. Sans modifier la numérotation de votre programme. Rentrez le prog suivant:

```
10 GOTO 20
20 GOTO 30
30 GOTO 40
```

Faites RENUMBER 1,1,1,1. Le message 'ERR:Stmt Not Found' s'affiche. Tapez FETCH, vous êtes positionné sur la ligne 30. Pratique après avoir bidouillé un prog, c'est une vérification à pas cher. Pourquoi s'en priver ?
Au plaisir de vous lire...

Jean-Pierre BONDU (P.P.C. #33,SIG #4)

Nicodème Philippe (PPCP237) 05/08/85
B124 Résidence du Chemin Vert
59131 Rousies

Bonjour,

Heureux possesseur d'un HP71, j'ai entrepris l'apprentissage du Forth. Je recherche des informations sur le fonctionnement de l'éditeur du module Forth/Ass. Question principale, comment sauvegarder sur cassette le résultat d'un week-end de travail?

Pouvez vous répondre à ma question, ou m'indiquer si vous avez déjà publié des articles sur le sujet (et comment se les procurer). Je suis également intéressé par toute référence d'ouvrage consacré au HP71 (en français de préférence).

Avec tous mes remerciements.

Philippe NICODEME.

Cher ami,

Je te comprends quand tu dis "Heureux possesseur", car effectivement, le HP71 est une bien merveilleuse machine. Au Bureau, nous te félicitons d'avoir entrepris l'apprentissage du Forth (c'est vraiment un chouette langage). Si tu as quelques difficultés de programmation, je te conseille un livre qui est très bien fait pour tout débutant en Forth. C'est: "Programmer le Forth" de Robert Van loo, collection Marabout. En ce qui concerne ta question, je pense que la meilleure façon de sauvegarder tes routines, est, soit de sauver la Forthram sur

cassette, soit de mettre tes routines dans un fichier texte, et de sauver ce fichier sur cassette. N'oublie pas que la fonction LOADF te permet par la suite de le recharger dans le fichier forthram. Pour ta seconde question, il n'existe pas d'autre référence d'ouvrage traitant du HP71 que le JPC. Que ceux qui savent en parlent !

Philippe.

Reno Bonzon
BP 119
97230 St Laurent
GUYANNE.

Bonjour,

J'habite la Guyanne, et possède une HP41CV. J'ai deux questions à vous poser:

- 1) Qu'est-ce qu'une HP41CV "accélérée"?
- 2) Les piles de mon HP41CV se sont usées sans qu'apparaisse le signal BAT. J'ai donc perdu tous mes programmes. Y a-t-il une explication?

Merci de me répondre,

Reno.

Cher ami,

Une HP41 accélérée, est une calculatrice ayant subi une petite modification lui permettant de faire tourner les programmes plus vite. Si tu passes en France, nous pouvons nous charger de faire cette transformation. Sinon, envoie nous une reproduction ou une photo du circuit logique de ta machine, et nous t'indiquerons la façon de procéder. Pour ta deuxième question, je ne peux répondre dans l'immédiat, mais le premier test que tu peux faire est d'essayer le programme ci dessous qui te permet d'allumer l'indicateur BAT et donc de vérifier s'il n'est pas défaillant.

01*LBL "IF"

02 ABS	17 X<> d
03 24	18 FC?C IND O
04 +	19 SF IND O
05 STO M	20 X<> d
06 8	21 STO M
07 ST/ M	22 RDN
08 MOD	23 12
09 RCL d	24 -
10 X<> M	25 SCI IND X
11 INT	26 ARCL X
12 SCI IND X	27 X<> O
13 ARCL X	28 STO d
14 X<>Y	29 RDN
15 X<> O	30 CLA
16 X<> N	31 END

Ce programme te permet d'inverser n'importe quel indicateur binaire (même ceux dont la valeur est supérieure à 29). Et, entre autre le drapeau 49 qui allume l'indicateur BAT.

Le mode d'emploi est très simple: Placer en X le numéro du drapeau à inverser (ex: 49), puis faire XEQ "IF". A l'arrêt, le voyant BAT doit être allumé.

Bon courage pour tout,

Philippe.

Alain HERREMAN

2 rue du parc Montsouris 75014 PARIS
Vends HP-41CV + Module MATH à 1000FF, une imprimante 82143A à 1500FF, un port-extender 500FF, un HP-IL 600FF. Tél: 589-09-41.

Michel MARTINET

119 rue de Bagneux 91120 MONTRouGE
Vends HP75C + 8 ko RAM + Visicalc + text formatter + nombreux logiciels. 6000 FF.
Tél: (1) 43 36 12 05 (Département Informatique)

HP41

FREDERIC POUPON

HP ET PHILIPPE GUEZ

TRANSFERT EN XMEMORY

NOUVEAU MODULE HP41

TRANSFERT EN X-MEMORY

Nous allons voir aujourd'hui 3 petits programmes de transfert de fichiers entre la mémoire étendue de la HP41 et l'unité de cassette.

Programme "WRTXM": il vous permet de stocker toute la mémoire étendue dans une cassette. Le mode d'emploi est très simple. Mettre le nom du fichier en ALPHA puis faire XEQ "WRTXM".

Programme "READXM": opération inverse du programme "WRTXM". Permet de stocker un fichier 'XMEM' contenu dans une cassette dans la mémoire étendue du HP41. Pour son utilisation, écrire le nom du fichier en ALPHA, puis faire XEQ "READXM".

Programme "APPXM": Troisième volet de la grande saga de transfert. Ce programme vous permet de faire un 'APPEND' en XMEM. Stocker un premier fichier XMEM dans la mémoire étendue du HP41, puis par un XEQ "APPXM", vous pouvez stocker un deuxième fichier XMEM.

```
01*LBL "WRTXM"
32 PSIZE 598 EMDIR
CLD - CREATE , SEEKR
"" CLFL RDN ENTER^
ENTER^ "i *****"
256 / XTOA X<> L MOD
XTOA RCL M X<> c
RCL M STO 00 X<>Y
STO c R^ SAVEX "" ,
SEEKPTA FLSIZE 32 /
INT SIGN ST+ L ,031
```

```
41*LBL 00
DSE L FS? 30 GTO 01
GETRX WRTRX GTO 00
```

```
48*LBL 01
FLSIZE 32 MOD X=0?
GTO 01 1 - 1 E3 /
GETRX WRTRX
```

```
60*LBL 01
"i " RCL N
X<> c RCL M STO 00
X<>Y STO c CLA CLST
```

END

```
L11= 241 29
L16= 250 1 105 11 224 192 32 0 0 0 0
L31= 241 29
L61= 252 1 105 11 224 192 32 0 0 0 0 0 2
```

```
01*LBL "READXM"
34 PSIZE CLX SEEKR
READRX SEEKR "" "
RCL M X<> c , STO 00
X<>Y STO c RCL 00 ""
1 + CRFLD SF 00 CLA
RCL 00 32 / 1 + INT
STO N
29*LBL 00
DSE N FS? 30 GTO 01
1,032 READRX FC?C 00
GTO 02 1,033 SAVERX
```

```
39*LBL 03
1 ST+ M GTO 00
43*LBL 02
RCL M 32 * 1 -
SEEKPT GETX X<>Y
SEEKPT X<>Y ,033
SAVERX RDN RDN SEEKPT
R^ SAVEX GTO 03
```

```
62*LBL 01
RCL 00 32 MOD X=0?
GTO 02 RCL M 32 * 1
- SEEKPT STO 0 GETX
X<>Y SEEKPT X<>Y RDN
RDN 1 E3 / 1 +
READRX ,999 - SAVERX
RCL 0 SEEKPT R^ SAVEX
```

```
93*LBL 02
RCL 00 SEEKPT
" " RCL M SAVEX
"i " RCL N
X<> c RCL M STO 00
X<>Y STO c CLA CLST
END
```

```
L08= 243 4 0 192
L16= 241 29
L96= 7 espaces
```


L99= 252 1 105 11 224 192 32 0 0 0 0 2

```
01*LBL "APPM"
35 PSIZE 600 EMDIR
CLD - STO 34 CLX
SEEKR READRX SEEKR
RCL 00 "I" 1 + CRFLD
SF 00 CLA RCL 00 32
/ 1 + INT STO N
```

```
27*LBL 00
DSE \ FS? 30 GTO 01
1,032 READRX FC?C 00
GTO 02 1,033 SAVERX
```

```
37*LBL 03
1 ST+ M GTO 00
41*LBL 02
RCL M 32 * 1 -
SEEKPT GETX X<>Y
SEEKPT X<>Y ,033
SAVERX RDN RDN SEEKPT
R^ SAVEX GTO 03
```

```
60*LBL 01
RCL 00 32 MOD X=0?
GTO 02 RCL M 32 * 1
- SEEKPT STO 0 GETX
X<>Y SEEKPT X<>Y RDN
RDN 1 E3 / 1 +
READRX ,999 - SAVERX
RCL 0 SEEKPT R^ SAVEX
```

```
91*LBL 02
RCL 00 SEEKPT
" " RCL M SAVEX
"i " RCL 34
256 / XTOA RCL 34
LASTX MOD XTOA RCL N
X<> c RCL M STO 00
X<>Y X<> c "" FLSIZE
DSE X SEEKPT "*****"
X<> M SAVEX X<>Y
X<> c X<>Y STO 00
X<>Y STO c "I" PURFL
CLST END
```

L014= 241 13
L094= 7 espaces
L097= 250 1 105 11 224 192 32 0 0 0 0

L114= 241 29
L116= 247 32 0 0 0 0 2
L125= 241 13

Frédéric POUPON (P.P.C. #35)

NOUVEAU MODULE HP41/HP12-HP15-HP16

Vous trouverez ci-dessous un article, que j'ai reçu de HP Grenoble, puis traduit, vous présentant la nouvelle ROM pour HP41. Sa sortie est prévue pour novembre 1985.

BE AN EXPERT: HP41 ADVANTAGE QUESTIONS AND ANSWERS
Q: WHAT IS THE HP-41 ADVANTAGE?

A: The HP-41 ADVANTAGE is a powerful new 12k software module for the HP-41. It includes the most comprehensive and popular set of advanced engineering, mathematical and financial functions ever written for the HP-41.

Q: WHAT TYPES OF PROGRAMS ARE INCLUDED IN THE HP-41 ADVANTAGE?

A: The module contains a comprehensive function set that includes

- * Comprehensive advanced matrix math,
- * Real and complex matrix operations,
- * Simultaneous equations on real and complex numbers,
- * Solve (roots of $f(x)=0$) (like the HP15C),
- * Integrate (like the HP15C),
- * Three dimensional vector operations,
- * Complex number operations,
- * Number base conversions and boolean logic (like the HP16C),
- * Curve fitting,

- * Roots and evaluation of polynomial equations,
- * Differential equations,
- * Coordinate transformations,
- * Time value of money (like the HP12C).

Q: WHO SHOULD HAVE THE HP-41 ADVANTAGE?

A: Both students and professionals will benefit from owning an HP-41 ADVANTAGE. The programs for this module were selected to provide a useful function set for the solution of advanced problems in engineering and science.

For technical students, the HP-41 ADVANTAGE provides a set of functions that will be useful in coursework from linear algebra to statics and dynamics and electronics.

The HP-41 ADVANTAGE will help professional engineers and scientists meet challenges from the design and analysis of electrical circuits and complex mechanism to the building of roads.

Q: HOW IS THE HP-41 ADVANTAGE DIFFERENT FROM OTHER HP-41 SOFTWARE?

A: First, the HP-41 ADVANTAGE has considerably more capacity and, therefore, is more powerful than other modules. A full 12k-bytes of ROM is used to contain the most complete and powerful function set ever designed for HP-41.

Second, many important programs in the module have been written in machine-codes. This means that the programs will run faster and be more precise than previous HP-41 software.

Third, the "user-interface" is friendlier, so the HP-41 ADVANTAGE is easier to use.

Q: WHAT IS DIFFERENT ABOUT THE HP-41's USER INTERFACE?

A: Several new functions found in the HP-41 ADVANTAGE are combined into programs that are more sophisticated and easier to use than

previous HP-41 software. Prompts and menus lead the user through the possible choices in data input and output.

Almost all programs redefine keys in the top rows of the calculator to perform, with a single keystroke, operations defined by the program. The "J" Key is used to recall a menu or prompt to the display.

The use of menus allows you to enter data quickly, without the need for overlays or reference to the owner's manual. Cues provided by the menus clearly indicate how the keys are used.

Q: CAN THE PROGRAMS IN THE HP-41 ADVANTAGE BE USED AS PART OF OTHER PROGRAMS?

A: Yes! Most programs, and many subroutines, may be "called" or included as part of user-created programs. Specific instructions on how using the subroutines are included at the end of the owner's manual.

Q: WHAT ARE THE MATRIX CAPABILITIES OF THE HP-41 ADVANTAGE?

A: The matrix programs in this pac are the most powerful matrix programs ever offered on an HP calculator. They provide an easy method of creating, storing and retrieving real or complex matrices using prompts and menus. Matrices may contain numeric or alpha data. Operations such as inversion, transposition, determinant and solving of simultaneous equations may be accomplished easily, even in your own programs. In addition to the major operations provided by the matrix program, individual elements may be manipulated in and between matrices, for example, in moving, swapping, summing and finding the norm, maxima and minima. Over 45 functions are included for maximum flexibility in matrix operations.

Q: CAN A PRINTER BE USED WITH THE HP-41 ADVANTAGE?

A: If you have a printer plugged into the HP-41 when the module is used, inputs and results will be printed automatically.

SOYEZ UN EXPERT: LE HP-41 ADVANTAGE, QUESTIONS ET REPONSES.

Q: QU'EST CE QUE LE HP-41 ADVANTAGE?

R: Le HP-41 ADVANTAGE est un nouveau et puissant module de 12k de logiciel pour HP-41. Il comprend le jeu de fonctions pour HP-41 le plus compréhensible et le plus connu en matière d'ingénierie, de mathématiques et de finances.

Q: QUEL TYPES DE PROGRAMMES SONT INCLUS DANS LE HP-41 ADVANTAGE?

R: Le module comprend le jeu de fonctions suivant.

- * Mathématique matricielle,
- * Opérations sur matrices réelles et complexes
- * Résolution de systèmes réels et complexes
- * Zero de fonctions (Comme sur HP-15C)
- * Fonction d'intégration (comme sur HP-15C),
- * Opérations sur les vecteurs en trois dimensions,
- * Opérations sur les nombres complexes,
- * Changement de base et logique booléenne (comme sur HP-16C),
- * Ajustement de courbes,
- * Racines et évaluation d'un polynôme,
- * Résolution des équations différentielles,
- * Transformations de coordonnées.
- * Calcul financier (comme sur HP-12C).

Q: QUI DOIT POSSEDER LA HP-41 ADVANTAGE?

R: Tous les étudiants et les professionnels tireront bénéfice de leur HP-41 ADVANTAGE. Les programmes de ce module ont été sélectionnés parmi les plus utilisés dans le domaine de ingénierie et des sciences.

Pour les étudiants des sections technique, le

HP-41 ADVANTAGE donne un jeu de fonctions qui sera très utile en algèbre linéaire, statistiques, dynamique et électronique. Le HP-41 ADVANTAGE aidera l'ingénieur et le scientifique.

Q: EN QUOI LE HP-41 ADVANTAGE EST IL DIFFERENT D'UN AUTRE LOGICIEL POUR HP-41?

R: Premièrement, le HP-41 ADVANTAGE a considérablement plus de capacité, et donc, est plus puissant qu'un autre module. Une pleine ROM de 12k-octets est utilisée pour contenir le jeu le plus complet et le plus puissant de fonctions jamais écrits pour la HP-41.

Deuxièmement, la plupart des programmes ont été écrit en codes-machine. Les programmes tournent donc plus vite et sont plus précis que le logiciel normalement prévu pour la HP-41. troisièmement, l'interface d'utilisation a un comportement plus amical, aussi le HP-41 ADVANTAGE est facile d'utilisation

Q: QU'ELLE EST LA DIFFERENCE AVEC UNE AUTRE INTERFACE UTILISATEUR?

R: Plusieurs nouvelles fonctions dans le HP-41 ADVANTAGE combinées aux programmes, les rendent plus complets et plus faciles d'utilisation que le logiciel habituel pour HP-41. Les questions et menus aident l'utilisateur à travers plusieurs choix possibles de données d'entrée et de sortie.

Presque tous les programmes ont une redéfinition des touches de la rangée du haut du calculateur pour les rendre plus performants, avec une simple touche, les opérations définies par le programme. la touche "J" est utilisée pour rappeler un menu ou prompt à l'affichage.

L'utilisation des menus vous permet d'entrer les données rapidement, sans le recours d'overlays ou de référence au manuel d'utilisation. Il est indiqué clairement dans le menu comment les touches sont utilisées.

Q: LES PROGRAMMES DU HP-41 ADVANTAGES, PEUVENT ILS ETRE UTILISES COMME PARTIE D'AUTRE PROGRAMMES?

R: Oui! La plupart des programmes, et des routines, peuvent être "appelées" ou inclus comme partie d'un programme utilisateur. Les spécifications comment utiliser les sous routines, sont incluses à la fin du manuel d'utilisation.

Q: QU'ELLES SONT LES POSSIBILITES MATRICIELLES DE LA HP-41 ADVANTAGE?

R: Les programmes matriciels dans ce pac, sont les plus puissants jamais offerts sur un calculateur HP. Ils autorisent une méthode simple de création, stockage et rappel de matrices complexes utilisant les demandes et les menus. Les matrices peuvent contenir des données numériques ou alphanumériques. les opérations comme inversion, transposition, déterminant et résolution d'équations simultanées peuvent être résolues aisement. En plus, dans la majorité des opérations des programmes sur les matrices, chaque éléments peut être manipulés dans et entre matrices. Par exemple, déplacement, échange, somme et trouver la valeur, maxima et minima. Plus de 45 fonctions sont incluses pour un maximum de flexibilité dans les opérations matricielles.

Q: PEUT ON UTILISER UNE IMPRIMANTE AVEC LE HP-41 ADVANTAGE?

R: Si vous avez une imprimante branchée à votre HP-41 pendant l'utilisation du module, les entrées et les résultats peuvent être imprimés automatiquement.

Philippe

HP71

SERGE VAUDENAY

MICHEL WEIL

MICHEL WEIL

JACQUES BAUDIER

JACQUES BAUDIER

JEAN-PIERRE BONDU

SERGE VAUDENAY

ALAIN HERREMAN

ALAIN HERREMAN

LAURENT ISTRIA

REMORDS

CHOC EN RETOUR

TEST DE KOLMOGOROV ET DE SMIRNOV

TRANSLATER PROGRAM

PASSWORD !

ECREMEZ VOS PROGRAMMES !

EQUATE

DEVELOPPEMENTS LIMITES EN R.P.N.

HP71: TELLEMENT COMPLEXE !

UN BRIDGE CE SOIR ?

REMORDS (suite)

Veillez excuser avoir fait phothe
horthoghaffex article précédant stop bonjour
maman stop

Serge Vaudenay.

CHOC EN RETOUR ET ALLER DANS LE SIG

1. Tout d'abord merci aux lecteurs pour leurs remarques sur les programmes de Statistiques du n°27 JPC (en particulier dans le pgm K12TEST ligne 1770 lire Q et non Q9).
2. Je serai(s) intéressé par un LEX donnant les ordres suivants: SWAP(N,M) et SWAP\$(A\$,B\$). Existent-ils ?
3. Et également par les PRIMITIVES forth suivantes: RND (le truc aléatoire) et RANDOMIZE (surtout le 1er), car le passage forth --> basic --> forth prend trop de temps !
Je signale que TIME existe en PRIMITIVE forth.
4. Et à propos quelle est la période de RND ? (Je crois qu'elle est supérieure à 2^{21} ?)
5. Enfin, last but not the least, les méthodes de triage (Bulle, Shell et (dite) Rapide), sont vraiment trop lentes en basic. Est ce qu' un fameux SIGouilLEXeur pourrait les LEXiser. La syntaxe serait: TRI(X()) et TRI\$(X\$()) (les 2 sont importants). Ici X() (resp. X\$()) est un vecteur de données (resp. alphanumériques), et TRI(X()) (resp. TRI\$(X\$())) serait le vecteur trié dans l' ordre croissant. C'est la méthode SHELL qu'il faut LEXiser. Mais probablement que je rêve...

Bon SIGouilLEXage,
Michel Weil

TEST de KOLMOGOROV et de SMIRNOV

Quand on veut comparer une série expérimentale avec une loi théorique on utilise souvent le test du K12. Lorsque la fonction de répartition de la loi théorique est continue, on montre que le test de KOLMOGOROV est plus stable que celui du K12 (ceci parce que le test de KOLMOGOROV compare des fréquences cumulées, et non des fréquences individuelles, ce qui fait que le KOLMOGOROV-test dépend moins des résultats ponctuels). Bien entendu, plus le nombre d'observations est grand, mieux c'est.

Par exemple lorsque le temps de réponse d'un circuit électrique donne les résultats suivants (en nanosecondes): 21 20 18 15 20 19 20 16 21 14 17 18 18 16 16 17 17 20 20 17 23 16, on désire savoir, ou non, si ces résultats sont dispersés suivant une loi normale $N(m,e)$ (donc KOLMOGOROV-test). Alors il suffit de répondre au menu du programme " KOLMSMIR" (appuyez sur les touches demandées):

```
SMIRMOV(2)KOLMOGOROV(1)? -----> 1
Nres de décimales? 3 -----> N + ENDLINE
Clavier/Fichier(C/F)? -----> C(lavier)
Nom de la loi expé(rimentale)?->CIRCUIT
test de normalité(O/N)? -----> O(ui)
(le programme demande si l'on veut comparer à
une fonction de répartition d'une loi normale,
sinon appuyez sur N(on)).
m , é-t spéciaux? -----> N(on)
(si vous voulez introduire d'autres valeurs
pour la moyenne et l'écart-type, appuyez sur
O(ui), et les introduire lorsque le programme le
demandera).
Nombres de données? -----> 22
Z(1,1)? -----> ttes les données
Ce vecteur ligne s'imprime alors, et sur l'HP71
s'affiche la demande:
Correction (O/N)? ----->
Si -----> O(ui)
alors le programme fait une pause et demande de
changer Z(1,?). Veuillez taper TOUTES les
```


modifications nécessaires: "Z(1,i)=..", et relancez le programme par l'ordre: "f CONT". Le reste suit sont cours et la réponse finale, ici, est:

H0 est accepté à 5% avec $N(m=18.136; e=2.274)$
Vérification du programme: il est bien tapé si à l'ordre: "FOR I=1 TO M @ D(I); @ NEXT I" apparaît la série: .034 .038 .099 .146 .115 .057 .57 .059 .029

Le test de SMIRNOV compare, lui, 2 séries expérimentales, par une syntaxe encore plus simple: au lieu d'entrer une série de val. expérimentale, il faut en entrer deux !

Mathématiques du test de KOLMOGOROV: on calcule, en gros, l'écart:

$$D_n = \sup \text{ en } x \text{ de } |F_n(x) - F(x)|$$

où F est la fonction de répartition continue théorique (ici une loi normale) et F_n est la fonction de répartition empirique des données: $X_1, \dots, X_n$. Quand $n \rightarrow \infty$ la fonction de répartition de $\sqrt{n} D_n$ tend vers quelque chose de tabulé.

Si D_n est "trop grand" on rejette l'hypothèse de comparaison H_0 .

Pour le test de SMIRNOV: lorsque (F_m) et (G_n) sont les fonctions de répartitions empiriques des 2 séries expérimentales, on calcule:

$$D_{m,n} = \sup \text{ en } z \text{ de } |F_m(z) - G_n(z)|$$

et, si l'écart observé $D_{m,n}$ est "trop grand" on rejette l'hypothèse de comparaison H_0 .

REMARQUES:

*1. Dans ce programme un vecteur X a les trois représentations suivantes: $X()$ $X(1,)$ et $X(,1)$. Ceci parce que $X(1,)$ est plus utile pour l'ordre MAT PRINT X , $X(,1)$ est mieux pour supprimer un un des éléments du vecteur (le sous-programme LIGNMINU est plus rapide (et prend moins de place) qu'un sous-programme de suppression de colonnes).

*2. Le sous-programme LIGNMINU, précité, est un programme de suppression de lignes dans une matrice.

*3. Le sous-programme QUINON est un programme de réponse aux trois questions: (O/N)? (F/C)? (2/1)?

*4. Un FICHER de données n'est ici, qu'une simple suite de nombres.

*5. Le module de MATH est utilisé, mais n'est vraiment, vraiment pas indispensable (cf. *1.).

*6. Si vous voulez une copie du programme, envoyez une CASSETTE.

*7. Avec les remarques le programme KOLMSMIR fait 4853 octets, et 3429 sans celles-ci.

Michel Weil

2a, rue des jardins
25000 BESANCON

TRANSLATER PROGRAM

Cher amis de PPC, j'ai le plaisir de vous annoncer la venue au monde du programme HP41T071. Comme son nom l'indique, ce programme traduit un programme de 41 sur un 71. En fait, ce programme décode une chaîne de caractères représentant un programme, et range le tout dans un fichier TEXT que l'on peut ensuite utiliser. L'avantage du programme n'est pas dans son utilisation mais plutôt dans son étude, en effet, on peut y découvrir de quelle manière sont codés les programmes dans la mémoire de la 41.

Que fait exactement ce programme ?

Tout d'abord, je vous dois quelques explications au sujet de la fonction OUTP du module EXTENDED IO; cette fonction est un peu spéciale et on peut se demander à quoi elle peut bien servir. Comme on peut le lire dans le manuel (en anglais of course) de grand-mère HP, une utilisation possible de OUTP est l'envoi d'un programme d'une 41 à une autre via deux interfaces du même type, par exemple deux modems acoustiques HP82198A qui à ma connaissance ne sont pas distribués en France, ou deux interfaces HPIL/RS232 HP82164A ou encore deux interfaces HPIL/HPIB HP82169A etc... Une autre utilisation possible est l'envoi d'un programme 41 à un ordinateur via une de ces interfaces

pour effectuer un stockage sur disque. On pourra écrire alors sur cet ordinateur un programme équivalent à HP41T071. Dans le cas de deux 41, il faudra exécuter l'ordre OUTP sur la 41 émettrice et l'ordre frère INP sur l'autre 41.

La particularité de la fonction OUTP est la suivante: cet ordre envoie sur la boucle une séquence d'octets de données représentant le programme mais en compliquant les choses. Tout d'abord la 41 place devant la chaîne quatre octets qui contiennent la longueur du programme, ensuite elle ajoute en queue de chaîne deux autres octets représentant la somme de contrôle des autres octets (en anglais checksum) soit la somme modulo 256 de ces fameux octets (qu'est qu'y a comme octets !). Enfin, pour chaque octet sont envoyés deux caractères (0 à 9 ou A à F) qui forment en hexadécimal le code ASCII du caractère. Par exemple, l'ordre COS de la 41 ne sera pas envoyé sous forme d'un 'Z' mais sera représenté par la séquence des deux caractères '5' et 'A'. Pourquoi? , tout simplement pour éviter l'envoi indésiré de séquences d'échappement aux appareils HPIL placés sur la boucle.

J'ai essayé de rendre ce programme le plus lisible possible. Il est composé de 6 parties: le programme principal qui va de la ligne 10 à la ligne 160 se charge de préparer la boucle en faisant CONTROL OFF car la 41 est obligatoirement contrôleur sur une boucle, de dimensionner les variables et de recevoir la chaîne venant de la 41. Il appelle ensuite le SUB CHECKSUM en lui passant la dite chaîne, le checksum reçu et il récupère R=1 ou R=0 suivant que ce checksum est bon ou mauvais. Ensuite, le programme principal appelle le sous-programme TEXT41 qui fait tout le boulot.

Voyons maintenant ce que fait ce SUB TEXT41 placé lignes 210 à 770:

Il exécute la boucle DESAS autant de fois qu'il y a d'instructions à décoder. Pour chaque instruction, il regarde si elle est d'un type spécial par exemple un LBL, une chaîne alphanumérique, un nombre, un XROM etc...(lignes 260 à 300). Sinon, il appelle le SUB DESAS41 qui va lire dans le fichier FONCTS41 la fonction

désirées. Pour chaque type de fonction spéciale il y a un label dans ce SUB. Par exemple, XEQA pour les XEQ alphanumériques. Toute boucle se termine par un passage par "PRINT" qui se charge d'affecter un numéro à la ligne en cours d'étude, de la mettre dans le fichier TEXT et de l'afficher.

TEXT41 peut utiliser les SUB ALPHA, DESAS41 et SXROM. Pour DESAS, on envoie la valeur hexadécimale de la fonction que l'on cherche, et l'on récupère cette fonction sous forme d'une chaîne de caractères. Vous pouvez par exemple essayer CALL DESAS41('5A',A\$) et vous retrouverez A\$="COS". Pour ALPHA on envoie la chaîne P\$ que l'on est en train de traiter et l'on récupère cette chaîne moins ce dont le SUB s'est servi pour créer la chaîne qu'il renvoie. Pour être plus clair prenons un exemple; si dans un programme 41 on a la chaîne "POPOL", celle-ci sera codée F5.50.4F.50.4F.4C, le F indique que ce qui suit est du texte, le 5 indique le nombre de caractères qui composent la chaîne et les autres caractères représentent la chaîne. Lors de l'exécution du programme HP41T071 le test ligne 260 dirigera la chaîne P\$ vers 'TEXT' (ligne 420) qui en appelant le SUB ALPHA retrouvera C\$="POPOL" et P\$ auquel on aura retiré "F5504F504F4C". Enfin, le SUB SXROM se contente de traduire s'il le peut (c'est à dire si le 71 a en mémoire le fichier correspondant au module dont on veut traduire une des fonctions) une chaîne de la forme "XROM 25,26" en une autre chaîne (ici se serait "PASN").

Voilà, j'espère avoir été assez clair et je vous conseille de bien étudier ce programme !

Voyons voir maintenant comment l'utiliser :

Après avoir lancé le programme par un RUN il faut donner comme réponse à la question "Name ?" : le nom du fichier qui va recevoir le programme 41 traduit. Ensuite on a l'affichage "Ready", le programme indique qu'il est prêt, il attend alors la pression d'une touche pour continuer (ligne 80). Si l'on appuie sur une touche apparaît alors le message "Reading loop ...", le 71 attend alors la chaîne qui doit lui parvenir. Reste alors à envoyer le programme désiré à

partir de la 41 en exécutant le programme SENDP.

```
01 LBL "SENDP"  
02 OUTP  
03 10  
04 OUTXB  
05 END
```

Il faudra préalablement avoir fait GTO .. (très important) et placer dans alpha le nom du programme à envoyer. Ensuite, le programme fait le reste et affiche, au fur et à mesure de sa progression, les lignes traduites. Lorsqu'il a fini, il BEEP pour vous l'indiquer et affiche pour vous le rappeler le nom du fichier créer. On voit bien que l'utilisation du programme est fort simple et c'est tant mieux.

Quelques remarques maintenant:

Pour fonctionner correctement le programme nécessite d'avoir dans le 71 au minimum le fichier FONCTS41. De plus, si vous voulez traduire des fonctions de modules d'extension en clair (c'est à dire les avoir sous la forme exacte et nom sous la forme XROM XX,YY) il vous faudra avoir en mémoire le fichier TEXT correspondant c'est-à-dire XFUNCT pour le module X-FUNCTIONS, TIME41 pour le module TIME, HPIL141 et HPIL241 pour l'HPIL etc... Ensuite, il faut signaler que le programme met automatiquement en tête du fichier TEXT l'heure et la date si le fichier TIME41 est présent (comme le fait la 41 si elle a le module TIME). Enfin, pour gagner du temps on pourra supprimer la ligne 180 et le SUB CKECKSUM qui ne sont pas indispensables. De plus on pourra modifier la ligne 230 M\$="... si l'on préfère avoir des STO M, RCL Q etc... à la place de STO [, RCL ^ .

Pour finir, je dirais que l'on peut remplacer le programme HP41T071 par le programme suivant qui lit ce qu'envoie une 41 en mode MANIO après l'exécution d'un PRP.

```
10 INPUT "Name : ?";B$ @ CREATE TEXT B$  
20 ASSIGN #1 TO B$ @ CONTROL OFF  
30 FOR I=1 TO INF  
40 ENTER :LOOP ;A$ @ PRINT #1;A$ @ DISP I  
50 NEXT I
```

Vous aller me dire alors que ce programme est inutile, pour l'instant, son seul avantage est dans son étude mais peut-être que plus tard ...

Au niveau rapidité ce n'est pas trop mal : il a fallu 1 min et 55 sec à HP41T071 pour traduire un programme de 310 octets contre environ 30 sec pour son équivalent cité plus haut.

Voilà, j'espère vous avoir intéressé par ce programme et vous proposer bientôt un programme qui fera le contraire à savoir convertir un fichier TEXT en une chaîne compréhensible par INP. Pour tout renseignement ou remarque écrire au club ou me téléphoner au 39.68.33.93 le soir, le Jeudi après-midi ou le week-end.

Jacques BAUDIER PPC #192

A PROPOS DE PASSWORD

On apprend dans le volume I des I.D.S. que le mot de passe du 71 est stocké à partir de 2F7B2 et sur 16 nibbles (cf. page 3.7). Il ne reste plus qu'à lire ces nibbles. MAIS il faut savoir que ces demi-octets sont inversés deux à deux .

Par exemple le mot de passe POPOL sera codé 05F405F4C4000000 au lieu de 504F504F4C000000. Il s'agit donc de remettre tout ce petit monde dans le bon sens.

C'est ce que font les programmes PWORDB et PWORDF qui font la même chose. Mais PWORDB est écrit en BASIC ,utilise 190 octets et est lent alors que PWORDF utilisant des fonctions du modules FORTH/ASSEMBLER est plus rapide , n'occupe que 82 octets de mémoire mais a besoin de FORTHRAM soit au bas mot 2957 octets.Je vous laisse donc choisir celui que vous voudrez et réfléchir à l'utilité de tels programmes ...

J. Baudier PPC 192

ECREMEZ VOS PROGRAMMES !

Le programme ECREM, pour ECrase REMarques, est la transcription du programme SUPREM de Laurent ISTRIA (JPC No 26 juillet/août), adapté aux fonctions du fichier Lex EDLEX. Son avantage: le fichier travail n'a pas besoin d'être recopié, d'où une économie significative en mémoire. D'autre part, en cas de pépin, le message d'erreur adéquat est affiché. Si tout s'est correctement déroulé, vous verrez apparaître 'End of File'. Cela indique que le fichier a bien été traité jusqu'à la dernière ligne. C'est une amélioration notable par rapport à SUPREM qui se gardait bien de vous informer de son mal-fonctionnement éventuel. Par contre j'ai conservé dans son intégralité la partie analyse qui donnait toute satisfaction. La mise en oeuvre est immédiate et identique à celle de son grand-frère, je me dispense donc de vous la décrire. Au plaisir de vous lire...

Jean-Pierre BONDU (P33)

*** EQUATE ***

Rebonjour,
Voici un programme BASIC (ça existe encore), qui peut être très utile pour récupérer les équivalences d'un fichier assemblé. Une fois lancé, il vous demande le fichier objet (qui est le fichier listing du programme que vous avez déjà assemblé, et dans lequel se trouve la table des symboles). Il vous demande ensuite le nom du

fichier objet qui est le source que vous êtes en train d'écrire. Il se renferme ensuite dans son mutisme profond et inquiétant, et si tout va bien (pas de memory lost ou autres erreurs), le curseur réapparaît.

Lorsque vous repassez en éditeur de texte pour retrouver votre fichier, vous voyez sous vos yeux ébahis toutes les équivalences du listing dont le symbole NE SE TERMINE PAS PAR UN CHIFFRE. Vous avez ainsi récupéré toutes les adresses importantes de vos programmes. Amusez-vous bien, faites plein de memory lost, et si vous faites encore des branchements de programme où il ne faut pas, ce ne sera plus une erreur de frappe, ce sera votre programme qui aura changé de place.

Note: il y a un bug dans l'assembleur. Quand la table des symboles est à cheval sur plusieurs pages, seule la première page est correcte, mais les autres ont les valeurs numériques écrites en décimal.. Mon programme ne récupère que la première page.

Serge Vaudenay

DEVELOPPEMENTS LIMITES EN R.P.N.

Possesseurs du 71, il m'arrive en mode CALC de regretter la R.P.N. de la bonne vieille 41. Excusez ma familiarité(!). En ajoutant à ce sentiment le désir de posséder un programme qui donne le DL(0) (lire développement limité en 0) de n'importe quelle fonction, on arrive à l'origine de 'DL'.

Le calcul d'un DL peut se décomposer en un nombre fini d'opérations (additions, soustractions, multiplications, divisions, compositions) sur des polynômes. Nous pouvons "apprendre" au 71 quelques DL classiques
COS,SIN,ASIN,TAN,ATAN,CH,SH,TH
LN(1+X),EXP,(1+X)^a

Nous allons donc créer sur le 71 une pile

identique à celle de la 41, et alors tout sera possible ... Il faut savoir que les DL classiques se trouvent à l'emplacement de la fonction correspondante (par exemple pour avoir le DL de COS appuyez sur f5). Pour avoir le DL de $(1+X)^a$ appuyez sur '^'

Pour entrer le DL des fonctions hyperboliques (qui ne sont pas accessibles au clavier) appuyez sur 'f7' pour SH, sur 'f8' pour CH, sur 'f9' pour TH

Pour entrer un polynôme quelconque appuyez sur '?'. Pour additionner appuyez sur '+', pour multiplier sur '*' etc ... Pour composer appuyez sur 'C', et grâce au programme de Pierre DAVID en appuyant sur 'F' vous avez la fraction du coefficient désiré du polynôme qui est en X. Pour connaître le DL qui est en X appuyez sur END-LINE.

Opération dans la pile : $X \leftrightarrow Y$ se trouve en 'X', RDN en '#51', RUP en '#50', CLX en '#49', CHS sur '='. Ceux qui ne connaissent pas la 41, n'ont qu'à se renseigner!

Exemple : DL de $\exp(\text{ASIN}(X))$ à l'ordre 4 : faites RUN DL, entrez 4 (attention il vaut parfois mieux rentrer un degré supérieur à celui désiré), tapez ensuite sur 'f1' (donne le DL de ASIN) puis sur 'f*' et enfin sur 'C' pour avoir les coefficients appuyez à chaque fois sur END-LINE, vous obtenez $a(0)=1$, $a(1)=1$, $a(2)=.5$ etc...

Alain HERREMAN

HP71: TELLEMENT COMPLEXE !

Les programmes proposés par HP concernant les opérations sur les nombres complexes étaient déjà sur la 41 d'un emploi peu commode. A la réunion de juin, J.J. Moreau nous a présenté le module MATH, qui, semble-t-il est très bien. Mais, le programme concernant les complexes m'est apparu encore une fois peu pratique

d'emploi (je ne l'ai jamais eu entre les mains, alors si quelqu'un peut m'en envoyer un pour démentir ce fait, qu'il n'hésite pas.) Par conséquent, je vous propose un programme recréant une pile identique à celle de la 41, mais pour des nombres complexes!

Vous y trouverez les fonctions suivantes : $/, *, +, -, \text{ENTER}, \text{RUP}, \text{RDN}, \text{CLX}, \text{CHS}, \text{CLST}, X \leftrightarrow Y, \text{RCL } X \ Y \ Z \ T, \text{STO } X \ Y \ Z \ T$ plus : BAR (donne le conjugué) et MOD (module et argument).

Le programme a été créé de manière à ce qu'il soit facile d'ajouter d'autres fonctions. En plus du programme, il faut un fichier KEY appelé 'CX' avec les assignations suivantes :

```
KEY 'P', 'CONT BAR': KEY '^', 'CONT DIV':
KEY ':', 'CONT MUL': KEY '@', 'CONT SOUS':
KEY '?', 'CONT ADD': KEY '#38', 'CONT ENTER':
KEY '#50', 'CONT RUP': KEY '#51', 'CONT RDN':
KEY '#49', 'CONT CLX': KEY '=', 'CONT CHS':
KEY 'C', 'CONT CLST': KEY 'X', 'CONT ECH':
KEY 'I', 'i'; KEY 'M', 'CONT MOD':
KEY 'R', 'CONT RCL': KEY 'S', 'CONT STO':
KEY 'F', 'CONT FIN':
```

Vous avez ainsi, en regardant les codes, l'emplacement des fonctions sur le clavier. Passons à l'exemple : RUN COMPLEXE

Il est possible que vous ayez un peu n'importe quoi à l'affichage, ceci est dû au fait qu'on ne fait pas de DESTROY, afin d'avoir une 'pile complexe 41 constant memory', n'hésitez pas à rajouter cet ordre si la 'constant memory' ne vous semble pas indispensable.

important : $i^2 = -1$

Faites i (appuyez sur 'I') ENTER g* vous obtenez $-1+i0$; ça marche ! Vous obtenez autre chose, alors relisez tout et/ou listez votre programme, et 'CX' ! Plus difficile maintenant : $i(1+i2)(1-i3)+(1+i2)(3-i5)5+1+i2$ Faites $1+i2$ STO Y (appuyez sur 'S', puis 'Y' END-LINE) $2i$ g* $1-i3$ g* RCL Y (appuyez sur 'R', puis 'Y' END-LINE) $3-i5$ g* 5 g* g* $1+i2$ g* et vous obtenez : $68+i21$

N.B. : Pour les pointilleux, il y a un petit BUG : faites CLST puis MOD vous obtenez $\text{mod}(z)=0$ et $=0 \text{ Deg}$; horreur ! Un argument pour un complexe nul !

Alain HERREMAN

UN BRIDGE CE SOIR ?

Il arrive parfois que les bridgeurs de haut niveau fassent de savants calculs pour essayer de connaître la probabilité pour que leur adversaire (ou leur partenaire) ait tel ou tel jeu. Un des moyens de calcul utilisés est le savoir du nombre de point de l'autre. Je commencerais les explications par la définition du terme 'main'. On appelle 'main' tout ensemble de 13 cartes qu'un joueur est susceptible de retourner après la donne. A chaque main, on peut attribuer une valeur qui diffère suivant le type de cartes qu'elle contient. Si une main contient A as, R rois, D dames et V valets, alors sa valeur est affectée du coefficient $4*A+3*R+2*D+V$. C'est à l'aide de cette valeur que les joueurs vont pouvoir agir dans la première partie d'un tour, les 'annonces'. En effet, à tour de rôle, chaque joueur va essayer de faire comprendre à son partenaire, situé en face de lui, quelle est la valeur de sa main. Je m'arrêterai ici pour les explications relatives au bridge, et vous parlerai maintenant du petit programme que je sou mets à votre approbation ce mois-ci

Tout d'abord, quel peut bien être le nombre de mains existantes au bridge; autrement dit, soit un bridgeur qui se distribuerait des cartes sans arrêt, au bout de combien de coups pourrait-il supposer avoir eu toutes les mains ?

Il a 52 façons de se distribuer la première carte; pour la deuxième, il n'a plus que 51 solutions, puis 50, 49, ..., jusqu'à ce qu'il ait ses 13 cartes en main. Ce nombre s'appelle $C(52,13)$ et est égal à 635013559600 très exactement. Bon, maintenant, fort de ce fait, notre joueur voudrait bien savoir quelle est sa probabilité de relever une main de 10 points

Pour simplifier le problème, nous allons raisonner sur une main de 0 point. Comment peut-elle être formée ? La réponse, dans ce cas précis, est assez simple. Il lui suffit de piocher ses 13 cartes parmi celles qui n'ont pas

de valeur, soit 52-4 as-4 rois-4 dames-4 valets, c'est -à-dire, 36 cartes. Le nombre total de mains valant 0 point est donc égal à $C(36,13)=231078960$. Sa probabilité est alors: $2310789600/635013559600=3.64 \text{ E-}3$, soit une main sur 274 environ.

Poussons un peu le raisonnement pour 1 point par main. Il lui faut au plus un valet, et douze cartes basses. Il a donc 4 possibilités pour le valet ($C(4,1)$ soit un valet parmi 4) et $C(36,12)$ possibilités pour les cartes basses, soit 1251677700 possibilités. Le nombre de mains est donc: $4*125167770=5006710800$, la probabilité: $5006710800/635013559600=7.88 \text{ E-}3$, soit une main sur 126). Pour les mains de 10 points le problème qui se pose est de chercher toutes les mains qui donnent 10 points. Nous verrons donc comment faire par la suite. On pourrait alors continuer jusqu'au nombre maximum de points (et nous allons du reste le faire) mais, en utilisant cette méthode, il nous faudrait un bon bout de temps

L'art du pseudo-informaticien en herbe que je suis consiste donc à faire un programme qui me simplifiera le travail. Je vous signale tout de suite que le résultat de mes dernières cogitations suppose la présence en mémoire du LEX que j'avais fait paraître dans le numéro de juin: COMBARR.

Le programme est d'une facilité de compréhension déconcertante, et on peut être étonné de sa simplicité et de sa brièveté, mais cela est dû au fait que l'analyse logique du problème est très simple. Passons donc aux explications.

Déterminons tout d'abord le nombre maximum de points que peut contenir une main. Cette main, la plus forte qui soit au bridge, contiendra nécessairement un maximum d'as, de rois, de dames et de valets. Mais, la possession de toutes ces cartes implique une main de 16 cartes. Il faut donc en éliminer 3. Ce seront bien entendu les 3 valets, ce qui nous donnera une main de 37 points. La panoplie des valeurs s'étend donc de 0 à 37 points.

Dimensionnons une variable en OPTION BASE 0 à 38 indices (de 0 à 37). Il nous suffit de faire varier le nombre d'as, de rois, de dames et de valets respectivement de 0 à 4 pour obtenir toutes les mains possibles (les autres cartes

valent 0, et seul le nombre de point nous intéresse ici). A chacun de ces 'types de mains' ainsi obtenu, on peut faire correspondre le nombre de mains qu'il génère. Il suffit alors d'ajouter dans la variable indiquée adéquate ce nombre. Mais, un exemple concret vaut parfois mieux qu'une longue suite d'explications, alors, allons-y.

Soit la quadruplet $(A,R,D,V)=(2,1,0,4)$. On lui affecte la valeur de 15 points. Mais combien de mains sont-elles de ce type ? Il y a $C(4,2)$ façons de prendre le as, $C(4,1)$ façons de prendre les rois, on ne prend pas de dame, et tous les valets. La main contient alors $2+1+4=7$ cartes. Il en faut donc encore 6 pour arriver à 13. Ces dernières sont piochées parmi les cartes basses au nombre de 36. Il y a donc $C(36,6)$ possibilités différentes. Le quadruplet $(2,1,0,4)$ génère donc : $C(4,2)*C(4,1)*C(4,0)*C(4,4)*C(36,6)=6*4*1*1*194779=46747008$ mains. Mais, il existe bien d'autres quadruplets générant 15 points : $(3,0,1,1)$, $(0,3,1,4)$, ... C'est la raison pour laquelle on fait varier chaque carte haute de 0 à 4.

Mais il faut prendre garde à 2 choses : tout d'abord, on ne doit pas dépasser 37 points, ensuite, il faut éliminer radicalement tout ensemble de cartes hautes comprenant à lui seul plus de 13 cartes (ex : $(4,4,3,4)$). Une fois exclues ces 2 causes de falsification des résultats, il suffit de lancer le programme. Au bout de 177 secondes, il s'arrête et, de vos doigts fébriles, vous pouvez extraire de chaque mémoire le nombre de mains ayant tel ou tel nombre de points. Un excellent moyen de vérification est de faire la somme de toutes ces mains. Si vous ne vous êtes pas trompé, (mon programme tourne, je l'ai essayé) vous devez retrouver $C(52,13)$, le nombre total de mains accessibles.

Voilà, peut-être ces quelques lignes vous auront-elles intéressés, je le souhaite tout au moins, et je pense déjà à vous soumettre un autre article, mais cette fois sur les probabilités de réussite de culture des topinambours dans le haut Tibet.

L. ISTRIA (P.P.C. #3, S.I.G #3)

FICHER TEXT "HP41" DU TRANSLATER PROGRAM

NULL	STO 02	X>0?	ISG	GLOBAL	TEXT 10	CRFLD
LBL 00	STO 03	LN1+X	DSE	GLOBAL	TEXT 11	DELCHR
LBL 01	STO 04	X<0?	VIEW	GLOBAL	TEXT 12	DELREC
LBL 02	STO 05	X=0?	_REG	GLOBAL	TEXT 13	EMDIR
LBL 03	STO 06	INT	ASTO	GLOBAL	TEXT 14	FLSIZE
LBL 04	STO 07	FRC	ARCL	GLOBAL		GETAS
LBL 05	STO 08	D-R	FIX	X<>		GETKEY
LBL 06	STO 09	R-D	SCI	LBL	-TIME- C	GETP
LBL 07	STO 10	HMS	ENG	GTO	ADATE	GETR
LBL 08	STO 11	HR	ZONE	GTO	ALMCAT	GETREC
LBL 09	STO 12	RND	XROM	GTO	ALMNOW	GETRX
LBL 10	STO 13	OCT	XROM	GTO	ATIME	GETSUB
LBL 11	STO 14	CL_	XROM	GTO	ATIME24	GETX
LBL 12	STO 15	X<>Y	XROM	GTO	CLK12	INSCHR
LBL 13	+	PI	XROM	GTO	CLK24	INSREC
LBL 14	-	CLST	XROM	GTO	CLKT	PASN
0	*	R^	XROM	GTO	CLKTD	PCLPS
1	/	RDN	XROM	GTO	CLOCK	POSA
2	X<Y?	LASTX	SF	GTO	CORRECT	POSFL
3	X>Y?	CLX	CF	GTO	DATE	PSIZE
4	X<=Y?	X=Y?	FS?C	GTO	DATE+	PURFL
5	_+	X#Y?	FC?C	GTO	DDAYS	RCLFLAG
6	_-	SIGN	FS?	GTO	DMY	RCLPT
7	HMS+	X<=0?	FC?	GTO	DOW	RCLPTA
8	HMS-	MEAN	GTO.EXQ IND	XEQ	MDY	REGMOVE
9	MOD	SDEV	SPARE	XEQ	RCLAF	REGSWAP
.	%	AVIEW	SPARE	XEQ	RCLSW	SAVEAS
EEX	%CH	CLD	GTO 00	XEQ	RUNSW	SAVEP
NEG	P-R	DEG	GTO 01	XEQ	SETAF	SAVER
GTO	R-P	RAD	GTO 02	XEQ	SETDATE	SAVERX
XEQ	LN	GRAD	GTO 03	XEQ	SETIME	SAVEX
W	X^2	ENTER^	GTO 04	XEQ	SETSW	SEEKPT
RCL 00	SQRT	STOP	GTO 05	XEQ	STOPSW	SEEKPTA
RCL 01	Y^X	RTN	GTO 06	XEQ	SW	SIZE?
RCL 02	CHS	BEEP	GTO 07	XEQ	T+X	STOFLAG
RCL 03	E^X	CLA	GTO 08	XEQ	TIME	X<>F
RCL 04	LOG	ASHF	GTO 09	XEQ	XYZALM	XTOA
RCL 05	10^X	PSE	GTO 10	XEQ		
RCL 06	E^X-1	CLRG	GTO 11	XEQ		
RCL 07	SIN	AOFF	GTO 12	XEQ	-EXT FCN 1C	-PRINTER 2E
RCL 08	COS	AON	GTO 13	TEXT 0	ALENG	ACA
RCL 09	TAN	OFF	GTO 14	TEXT 1	ANUM	ACCHR
RCL 10	ASIN	PROMPT	GLOBAL	TEXT 2	APPCHR	ACCOL
RCL 11	ACOS	ADV	GLOBAL	TEXT 3	APPREC	ACSPEC
RCL 12	ATAN	RCL	GLOBAL	TEXT 4	ARCLREC	ACX
RCL 13	DEC	STO	GLOBAL	TEXT 5	AROT	BLDSPEC
RCL 14	1/X	ST+	GLOBAL	TEXT 6	ATOX	LIST
RCL 15	ABS	ST-	GLOBAL	TEXT 7	CLFL	PRA
STO 00	FACT	ST*	GLOBAL	TEXT 8	CLKEYS	XROM "PRAXIS"
STO 01	X#0?	ST/	GLOBAL	TEXT 9	CRFLAS	PRBUF

PRFLAGS	LOCAL	OUTXB	PENUP	CARD RDR 1F	BUF-XA	X<>FLAG
PRKEYS	MANIO	OUTP	PINIT	MRG	BUF-XB	
PRP	OUTA	POLL	PLOT	RDTA	CF33	
XROM "PRPLOT"	PWRDN	POLLD	PLREGX	RDTAX	CMD	---
XROM "PRPLOT"	PWRUP	POLLE	RATIO	RSUB	DDL	AND
PRREG	REMOTE	POLLUNC	RPLOT	VER	DDT	ASIZE?
PRREGX	SELECT	RCLSEL	SCALE	WALL	FRAV?	A-XL
PR	STOPIO	SRQ?	SETGU	WDTA	FRNS?	A-XR
PRSTK	TRIGGER	STAT	SETUU	WDTAX	GET	A-XX
PRX		XFER	TICLEN	WPRV	GTL	BININ
REGPLOT		XFERC	UNCLIP	WSTS	IDY	BINVIEW
SKPCHR	-X MASS 1A	XFERCL	WHERE	7CLREG	IFCR?	BIT?
SKPCOL	COPYFL	XFERE	XAXIS	7DSP0	IFC	HEXIN
STKPLOT	DIRX	XFERN	XAXISO	7DSP1	INBIN	HEXVIEW
FMT	FLENG	-ADV CTL FN	YAXIS	7DSP2	INBUF	NOT
	FLTYPE	ADROFF	YAXISO	7DSP3	LAD	OR
	MCOPY	ADRON		7DSP4	LPD	OCTIN
-MASS ST 1H	MCOPYPV	DDL		7DSP5	MIPT	OCTVIEW
CREATE	MVERIFY	DDT	PLOTTER 2A-	7DSP6	MONITOR	ROMCHKX
DIR	-X EXT FCN	LAD	PCLBUF	7DSP7	NRD	ROTX
NEWM	ALENGIO	SEND	PDIR	7DSP8	NRE	XOR
PURGE	ANUMDEL	TAD	PRCL	7DSP9	ORAV?	X-AL
READA	ATOXL	UNL	XROM "NEWPLOT"	7DSP1	OUTBIN	X-AR
READK	ATOXR	UNT	XROM "REPLOT"	7DSZ	OUTBINY	Y-AX
READP	ATOXX		XROM "PLINIT	7DSZ1	OUTBUF	
READR	XTOAL		XROM "PLTUXY"	7ENG	PT=	
READRX	XTOAR	PLOTTER 1A-	XROM "PLANOT"	7FIX	PT?	
READS	X<>FIO	CLIPUU	XROM "Y?"	7GSBI	PRBYTES	
READSUB	YTOAX	Csize	XROM "X?"	7GTOI	PRFRMS	
RENAME	-X CTL FNS	CZIZEO	BC	7ISZ	REN	
SED	AID	DGTIZE	BCA	7ISZ1	RFRM	
SEEKR	CLRDEV	DRAW	BCAA	7P<>S	RG-BUFX	
UNSEC	CLRLOOP	FRAME	BCKSM	7PRREG	RG=BUF?	
VERIFY	DEVL	GCLEAR	BCO	7PRSTK	RREG	
WRTA	DEVT	IDRAW	BCP	7PRTX	SAI	
WRTK	FINDAID	IMOVE	BCREGX	7RLC_	SCOPE	
WRTP	ID	IPLOT	BCSIZE	7SCI	SDA	
WRTPV	INAC	LABEL	BCX		SDC	
WRTR	INACL	LDIR	BCXS		SDI	
WRTRX	INAE	LIMIT		-HP-IL DEV	SF33	
WRTS	INAN	LOCATD		A-BUF	SRQR?	
ZERO	INXB	LOCATE	- WAND 1F -	A=BUF?	SST	
--	INP	LORG	WNDDTA	A=BUFX?	TAD	
-CTL FNS	LOCK	LTYPE	WNDDTX	AAD	TCT	
AUTOIO	NLOOP	LTYPEO	WNDLNK	AAU	UNL	
FINDID	NOTREM	LXAXIS	WNDSUB	AIPT	UNT	
INA	OUTAC	LYAXIS	WNDSCN	BSIZE?	WRFRM	
IND	OUTACL	MOVE	XROM "WNDTST"	BSIZE	WREG	
INSTAT	OUTAE	PEN		BUF-AX	X-BUF	
LISTEN	OUTAN	PENDN		BUF-RGX	X=BUF?	

TEST DE KOLMOGOROV ET DE SMIRNOV

```

- test de KOLMOGOROV pour comparer une loi expé. à une fonction de répartition théorique
test de SMIRNOV pour comparer 2 lois expérimentales
sauvegarde automatique des lois
LEX KEYWAIT utilisé
Michel WEIL, 2a rue des jardins, 25000 BESANÇON
flag 1 armé: test de SMIRNOV
flag 1 desarmé: test de KOLMOGOROV
flag 3 armé: saisie au clavier
flag 3 desarmé: saisie par fichier
flag 4 armé: test de normalité (KOLMOGOROV)
flag 0 armé: moyenne et é-t. spéciaux (dans le cas de test de normalité)
40 DESTROY ALL @ CFLAG 0,1,3,4 @ OPTION BASE 1 @ DELAY 0,0 @ STD
50 INTEGER I,J,M,M1,N,N1 @ DIM M$(8),N$(8),X(100,1),Y(100,1)
60 CALL MENU(M$,N$) @ IF FLAG(1) THEN REAL C,C1,C5,A,B ELSE REAL M5,E,T
  - entrée des données:
70 IF FLAG(3) THEN GOSUB 'CLAV' @ GOTO 90
80 CALL FICHIER(M$,M,X(,)) @ IF FLAG(1) THEN CALL FICHIER(N$,N,Y(,))
90 DIM X(M) @ M1=M @ IF FLAG(1) THEN DIM Y(N) @ N1=N
  - moyenne et é-t.:
100 IF FLAG(0) THEN INPUT "m ,é-t = ?";M5,E ELSE CALL MOYET1(M,X(,),M5,E)
  - trie les entrées:
110 CALL SHELL(M,X(,)) @ IF FLAG(1) THEN CALL SHELL(N,Y(,))
120 DIM X(M,1),P(M,1) @ MAT P=(1/M) @ IF FLAG(1) THEN DIM Y(N,1),Q(N,1) @ MAT Q=(1/N)
  - écrémage et poids du 1er facteur...
140 I=M @ J=N
150 CALL ECREPOID(I,M,X(,),P(,))
160 DIM X(M,1),P(M,1) @ I=I-1 @ DISP I @ IF I>1 THEN 150
170 IF NOT FLAG(1) THEN 200
  - ...puis du 2er facteur:
180 CALL ECREPOID(J,N,Y(,),Q(,))
190 DIM Y(N,1),Q(N,1) @ J=J-1 @ DISP J @ IF J>1 THEN 180
  - Choix du test:
200 IF FLAG(1) THEN 'SMIRNOV' ELSE 'KOLMOGOR'
  - test de KOLMOGOROV

=====
210 'KOLMOGOR': DIM X(M),P(M),R(M),X1(M),P1(M),D(M) ! re-dimensionnement général
220 R(1)=0 @ FOR I=2 TO M @ R(I)=R(I-1)+P(I-1) @ NEXT I ! poids des répétitions
230 IF NOT FLAG(4) THEN 290 ! pas de test de normalité
240 FOR I=1 TO M @ X1(I)=(X(I)-M5)/E @ NEXT I @ T=SQR(2*PI) ! réduction
  - Calcul des probabilités à l'aide de la loi normale(m,e) :
260 P1(1)=FNI(-5,X1(1),T) @ FOR I=2 TO M
270 DISP I @ P1(I)=P1(I-1)+FNI(X1(I-1),X1(I),T)
280 NEXT I @ GOTO 310
290 DISP "fct.de répart.théo.?" @ DISP M; @ MAT INPUT P1 ! Cas de non normalité
  - Calcul l'écart de Kolmogorof...
310 FOR I=1 TO M @ D(I)=MAX(ABS(R(I)-P1(I)),ABS(R(I)+P(I)-P1(I)))
  - ...et tests finaux:
330 IF D(I)>=.989/SQR(M1) THEN 'R1'
340 NEXT I @ FOR J=1 TO M @ IF D(J)>=.878/SQR(M1) THEN 'R5'
350 NEXT J @ PRINT "H0 est accepté à 5% ";
360 IF FLAG(4) THEN PRINT "pour N(m=";M5;" ;é-t=";E;")"
370 GOTO 'FIN'

=====
380 'R1': PRINT 'H0 est refusé à 1%' @ GOTO 'FIN'

```

```
=====
390 'R5': PRINT 'H0 est refusé à 5%'
=====
```

```
400 'FIN': BEEP @ END
- test de SMIRNOV
```

```
=====
410 'SMIRNOV': DIM X(M),P(M),R(M),Y(N),Q(N),S(N),Z1(M+N),D(M+N)
- fréquences cumulées de X et Y:
420 R(1)=0 @ FOR I=2 TO M @ R(I)=R(I-1)+P(I-1) @ NEXT I
430 S(1)=0 @ FOR I=2 TO N @ S(I)=S(I-1)+Q(I-1) @ NEXT I
- les 2 données sont réunies... et retriées:
440 FOR I=1 TO M @ Z1(I)=X(I) @ NEXT I
450 FOR I=M+1 TO M+N @ Z1(I)=Y(I-M) @ NEXT I
460 CALL SHELL(N+M,Z1())
- Voici le test de SMIRNOV:
470 C=SQR(1/M+1/N) @ C5=1.3581*C @ C1=1.6276*C @ FOR J=1 TO M+N @ DISP J
480 CALL FCTREPAR(Z1(J),X(),R(),M,A) @ CALL FCTREPAR(Z1(J),Y(),S(),N,B)
490 D(J)=ABS(A-B) @ IF D(J)>C1 THEN 'R1'
500 NEXT J
510 FOR J=1 TO M+N @ IF D(J)>C5 THEN 'R5'
520 NEXT J
530 PRINT "H0 accepté à 5%"
540 BEEP @ END
- Saisie au clavier
```

```
=====
560 'CLAV': IF FLAG(1) THEN 'LOIS2' ELSE INPUT "Nbres de données? ";M
570 DIM X(1,M) @ CALL ENTRE(M,X(),M$) @ RETURN
```

```
=====
580 'LOIS2': INPUT "Nbre données :1°loi?";M @ DIM X(1,M) @ CALL ENTRE(M,X(),M$)
590 INPUT "Nbre données :2°loi?";N @ DIM Y(1,N) @ CALL ENTRE(N,Y(),N$) @ RETURN
```

```
=====
600 DEF FNI(A,B,T)=INTEGRAL(A,B,.001,EXP(-IVAR*IVAR/2))/T ! proba entre A et B
```

```
=====
- entrée des données au clavier et sauvegarde:
```

```
=====
1000 SUB ENTRE(M,Z(),M$)
1010 MAT INPUT Z @ MAT PRINT Z;
1020 Q$="Correction (O/N) ?" @ CALL OUINON(Q$,K) @ ON K+1 GOTO 1040,1030
1030 DISP "Changer z(1,?) !" @ PAUSE
1040 CREATE DATA M$ @ ASSIGN #1 TO M$ @ PRINT #1;Z @ ASSIGN #1 TO * @ END SUB
- calcule moyenne et é-t:
```

```
=====
1110 SUB MOYET1(N,X(),M,E) @ STAT H(1)
1120 FOR I=1 TO N @ ADD X(I) @ NEXT I @ M=MEAN @ E=SDEV @ CLSTAT @ END SUB
- Lecture de fichier
```

```
=====
1210 SUB FICHIER(M$,L,Z(),)
1220 ASSIGN #1 TO M$ @ L=0 @ ON ERROR GOTO 1240
1230 L=L+1 @ READ #1;Z(L,1) @ GOTO 1230
1240 OFF ERROR @ ASSIGN #1 TO * @ L=L-1
1250 END SUB
```

=====

- fct réponse

=====

```
1812 SUB OUI NON(Q$,Q)
1814 DISP Q$ @ I=POS("NOFC12",UPRC$(KEYWAIT$(1,1)))-1 @ IF I<0 THEN 1814
1816 Q=MOD(I,2) @ END SUB
- Choix des options
```

=====

```
1852 SUB MENU(M$,N$)
1854 Q$="Smirnov(2)Kolmogrof(1)" @ CALL OUI NON(Q$,Q) @ DISP FLAG(1,Q)
1856 IF FLAG(1) THEN 1860
1858 LINPUT "Nbres de décimales? ", "3";I$ @ FIX VAL(I$)
1860 Q$="Clavier/Fichier (C/F)?" @ CALL OUI NON(Q$,Q) @ DISP FLAG(3,Q)
1862 IF FLAG(1) THEN 1874
1864 INPUT "Nom de la loi expér.?" ;M$
1866 Q$="test de normalité(O/N)" @ CALL OUI NON(Q$,Q) @ DISP FLAG(4,Q)
1868 IF NOT FLAG(4) THEN 'FIN'
1870 Q$="m ; é-t:spéciaux?(O/N)" @ CALL OUI NON(Q$,Q) @ DISP FLAG(0,Q)
1872 GOTO 'FIN'
1874 INPUT "Nom de la 1e loi ?" ;M$
1876 INPUT "Nom de la 2e loi ?" ;N$ @ 'FIN': END SUB
- suppression de la ligne n° N (matrice RxC) :
```

=====

```
2700 SUB LIGNMINU(A(,),R,C,N)
2710 IF N<1 OR N>R THEN DISP "la ligne n'existe pas" @ PAUSE
2720 IF R=1 THEN DISP "Erreur:1 seule ligne" @ STOP
2730 DISP "du calme.." @ IF N=R THEN 2760
2740 FOR I=N TO R-1 @ FOR J=1 TO C
2750 A(I,J)=A(I+1,J) @ NEXT J @ NEXT I
2760 END SUB
```

=====

- tri Shell:

=====

```
3000 SUB SHELL(N,X()) @ DISP "ça trie Shell.." @ K=N
3010 K=INT(K/2)
3020 IF K=0 THEN 3070
3030 L=N-K @ FOR J=1 TO L @ I=J
3040 M=I+K @ IF X(I)<=X(M) THEN 3060
3050 H=X(I) @ X(I)=X(M) @ X(M)=H @ I=I-K @ IF I>0 THEN 3040
3060 NEXT J @ GOTO 3010
3070 END SUB
```

=====

- suppression des répétitions ds X et poids ds P:

=====

```
5000 SUB ECREPOID(I,M,X(,),P(,))
5010 IF X(I,1)=X(I-1,1) THEN P(I-1,1)=P(I-1,1)+P(I,1) @ CALL MOINS(I,M,X(,),P(,))
5020 END SUB
```

=====

- suppression de lignes maudites:

=====


```

5100 SUB MOINS(I,M,X(,),P(,))
5110 CALL LIGNMINU(X(,),M,1,I) @ CALL LIGNMINU(P(,),M,1,I) @ M=M-1 @ END SUB
- A: est la fct de répartition de X()

```

```

=====
6010 SUB FCTREPAR(Z,X(,),R(,),M,A)
6020 IF Z<X(1) THEN A=R(1)
6030 FOR I=1 TO M-1 @ IF X(I)<=Z AND Z<X(I+1) THEN A=R(I+1)
6040 NEXT I
6050 IF X(M)<=Z THEN A=1
6060 END SUB

```

MOT DE PASSE

```

10 DIM A$(16) @ A$=PEEK$('2F7B2',16)
20 A$=FNAS(A$)
30 IF NUM(A$) THEN DISP 'Password : ';A$ ELSE DISP 'Pas de password'
40 END

```

=====

```

50 DEF FNAS(A$)
60 FOR P=1 TO 8
70 A$=A$[0,P-1]&CHR$(HTD(A$[P+1,P+1]&A$[P,P]))&A$[P+2]
80 NEXT P
90 FNAS=A$
100 END DEF

```

MOT DE PASSE

```

10 FORTHX ' HEX 2F7B2 9 ' @ A$=FORTH$
20 IF NUM(A$) THEN DISP 'Password : ';A$ ELSE DISP 'Pas de password'

```

SUPPRIME LES REMARQUES D'UN PROGRAMME

```

- ECREM 530 26/08/85
10 INPUT 'Fichier: ';A$ @ CALL ECREM(A$) @ BEEP @ END

```

```

=====
1000 SUB ECREM(A$)
1010 DIM C$(140)
1020 TRANSFORM A$ INTO TEXT @ ASSIGN #1 TO A$
1030 ON ERROR GOTO 'ERR' @ FOR P=0 TO INF
1040 READ #1,P;C$ @ DISP C$[1,4]
1050 IF NOT (POS(C$,'!') OR POS(C$,' REM ')) THEN 'NXT'
1060 IF C$[6,6]='!' OR C$[6,8]='REM' THEN GOSUB 'DEL' @ GOTO 'NXT'

```

```

1070 FOR J=6 TO LEN(C$)
1080 IF C$(J,J)="" THEN J=POS(C$,"",J+1) @ GOTO 1110
1090 IF C$(J,J)="" THEN J=POS(C$,"",J+1) @ GOTO 1110
1100 IF C$(J,J)='!' OR C$(J,J+2)='REM' THEN C$=C$[1,J-2] @ GOTO 1120
1110 NEXT J
1120 IF C$(LEN(C$))='@' THEN C$(LEN(C$))=""
1130 REPLACE #1,P;C$

=====
1140 'NXT': NEXT P

=====
1150 'DEL': DELETE #1,P @ P=P-1 @ RETURN

=====
1160 'ERR': OFF ERROR @ DISP ERRMS @ ASSIGN #1 TO *
1170 TRANSFORM A$ INTO BASIC
1180 END SUB

```

EQUIVALENCES D'UN LISTAGE SOURCE

```

10 INPUT "Fichier objet ? ";F$
20 INPUT "Fichier source ? ";S$
30 CALL EQUATE(F$,S$)

=====
40 SUB EQUATE(F$,S$)
50 ASSIGN #1 TO S$
60 ASSIGN #2 TO F$
  - on met le pointeur au début de la table.
70 RESTORE #1,IP(SEARCH("SYMBOL TABLE",1,1,9999,1))+2
  - BouCLE de transfert.

=====
80 'BCL': READ #1;A$
  - Si on a un enregistrement vide, c'est fini.
90 IF A$="" THEN END
  - Si c'est le label FileNd, ou un label se terminant par un chiffre, on l'ignore.
100 IF A$[1,6]="FileNd" OR FNL(A$,8)>47 AND FNL(A$,8)<58 THEN 'BCL'
  - Transfert de l'équivalence.
110 PRINT #2;A$[1,7]&" EQU #"&A$[8]
120 GOTO 'BCL'
  - Un peut de récursivité pour avoir le dernier caractère qui n'est pas un espace dans une
    chaîne...

=====
130 DEF FNL(A$,N) @ IF A$[N,N]=" " THEN FNL=FNL(A$,N-1) ELSE FNL=NUM(A$[N])

=====
140 END DEF

=====
150 END SUB

```

DEVELOPPEMENTS LIMITES EN R.P.N.

```

10 DESTROY ALL
20 INPUT 'Degre des poly. : ' ; N @ OPTION BASE 0 @ DIM X(N),Y(N),Z(N),T(N),E(N),C(N)
30 STD @ DELAY 0,.3 @ DIM X$(96),Y$(96),Z$(96),T$(96),C$(96)
40 DISP X$
50 AS=KEY$ @ IF AS='' THEN 50
60 IF AS='/' THEN 470
70 IF AS='*' THEN 330
80 IF AS='+' THEN 360
90 IF AS='-' THEN 380
100 IF AS='C' THEN 400
110 IF AS='#50' THEN 710
120 IF AS='#51' THEN 690
130 IF AS='X' THEN 730
140 IF AS='f4' THEN 610
150 IF AS='f1' THEN 820
160 IF AS='f5' THEN 630
170 IF AS='f6' THEN 860
180 IF AS='f3' THEN 780
190 IF AS='f7' THEN 650
200 IF AS='g7' THEN 840
210 IF AS='f8' THEN 670
220 IF AS='f9' THEN 890
230 IF AS='g9' THEN 800
240 IF AS='#38' THEN 520
250 IF AS='?' THEN 530
260 IF AS='f-' THEN 590
270 IF AS='f*' THEN 600
280 IF AS='F' THEN 570
290 IF AS='g/' THEN 740
300 IF AS='' THEN 760
310 IF AS='=' THEN 560
320 GOTO 50
330 DISP '*' @ ON ERROR GOTO 340 @ FOR K=0 TO N @ S=0 @ FOR I=0 TO K @ S=S+X(I)*Y(K-I)
340 NEXT I @ E(K)=S @ NEXT K @ FOR I=0 TO N @ X(I)=E(I) @ Y(I)=Z(I) @ Z(I)=Y(I) @ NEXT I
350 X$='('&Y$&')*( '&X$&')' @ Y$=Z$ @ Z$=T$ @ GOTO 40
360 DISP '+' @ FOR I=0 TO N @ X(I)=X(I)+Y(I) @ Y(I)=Z(I) @ Z(I)=T(I) @ NEXT I
370 X$='('&X$&')+( '&Y$&')' @ Y$=Z$ @ Z$=T$ @ GOTO 40
380 DISP '-' @ FOR I=0 TO N @ X(I)=Y(I)-X(I) @ Y(I)=Z(I) @ Z(I)=T(I) @ NEXT I
390 X$='('&Y$&')-( '&X$&')' @ Y$=Z$ @ Z$=T$ @ GOTO 40
400 DISP 'Comp.' @ IF Y(0) THEN BEEP @ DISP 'WRN: Y(0)#0' @ WAIT 2 @ GOTO 40
410 DESTROY D @ DIM D(N) @ FOR I=0 TO N @ C(I)=Y(I) @ NEXT I
420 D(0)=X(0) @ FOR I=0 TO N @ D(I)=D(I)+X(I)*Y(I) @ NEXT I @ FOR L=2 TO N
430 ON ERROR GOTO 440 @ FOR K=0 TO N @ S=0 @ FOR I=0 TO K @ S=S+Y(I)*C(K-I)
440 NEXT I @ E(K)=S @ NEXT K @ FOR I=0 TO N @ Y(I)=E(I) @ NEXT I
450 FOR I=0 TO N @ D(I)=D(I)+X(L)*Y(I) @ NEXT I @ NEXT L @ FOR I=0 TO N @ X(I)=D(I)
460 OFF ERROR @ Y(I)=Z(I) @ Z(I)=T(I) @ NEXT I @ X$=X$&'['&Y$&']' @ Y$=Z$ @ Z$=T$ @ GOTO 40
470 DISP '/' @ DESTROY E @ DIM E(N) @ IF X(J) THEN 480 ELSE J=J+1 @ GOTO 470
480 FOR K=J TO N @ E(K-J)=Y(K)/X(J)
490 ON ERROR GOTO 500 @ FOR I=N-K TO 0 STEP -1 @ Y(K+I)=Y(K+I)-X(I+J)*Y(K)/X(J) @ NEXT I
500 NEXT K @ FOR I=0 TO N @ X(I)=E(I) @ Y(I)=Z(I) @ Z(I)=T(I) @ NEXT I @ J=0
510 X$='('&Y$&')/( '&X$&')' @ Y$=Z$ @ Z$=T$ @ GOTO 40
520 FOR I=0 TO N @ DISP 'a(' ; STR$(I) ; ')=' ; X(I) ; @ INPUT '' ; C$ @ NEXT I @ GOTO 40
530 ON ERROR GOTO 40 @ FOR I=0 TO N @ T(I)=Z(I) @ Z(I)=Y(I) @ Y(I)=X(I) @ NEXT I
540 T$=Z$ @ Z$=Y$ @ Y$=X$ @ DESTROY X @ DIM X(N) @ INPUT 'Nom du poly. : ' ; X$ @ FOR I=0 TO N

```

```

550 DISP 'a(';STR$(I);')='; @ INPUT '';X(I) @ NEXT I @ GOTO 40
560 DISP 'CHS' @ FOR I=0 TO N @ X(I)=-X(I) @ NEXT I @ X$='-'&X$ @ GOTO 40
570 INTEGER A,B @ INPUT 'I=';I @ IF I>N THEN 570 ELSE CALL FR(X(I),A,B,10^(-8))
580 DISP USING 'K,A,K';A,'/',B @ WAIT 2 @ GOTO 40
590 GOSUB 920 @ FOR I=1 TO N @ X(I)=(-1)^(I+1)/I @ NEXT I @ X(0)=0 @ X$='LOG' @ GOTO 40
600 GOSUB 920 @ FOR I=0 TO N @ X(I)=1/FACT(I) @ NEXT I @ X$='EXP' @ GOTO 40
610 GOSUB 920 @ DESTROY X @ DIM X(N) @ FOR I=1 TO N STEP 2 @ X(I)=(-1)^IP(I/2)/FACT(I)
620 NEXT I @ X$='SIN' @ GOTO 40
630 GOSUB 920 @ DESTROY X @ DIM X(N) @ FOR I=0 TO N STEP 2 @ X(I)=(-1)^IP(I/2)/FACT(I)
640 NEXT I @ X$='COS' @ GOTO 40
650 GOSUB 920 @ DESTROY X @ DIM X(N) @ FOR I=1 TO N STEP 2 @ X(I)=1/FACT(I) @ NEXT I
660 X$='SH' @ GOTO 40
670 GOSUB 920 @ DESTROY X @ DIM X(N) @ FOR I=0 TO N STEP 2 @ X(I)=1/FACT(I) @ NEXT I
680 X$='CH' @ GOTO 40
690 C$=X$ @ X$=Y$ @ Y$=Z$ @ Z$=T$ @ T$=C$ @ FOR I=0 TO N @ E(I)=X(I) @ X(I)=Y(I)
700 Y(I)=Z(I) @ Z(I)=T(I) @ T(I)=E(I) @ NEXT I @ GOTO 40
710 C$=T$ @ T$=Z$ @ Z$=Y$ @ Y$=X$ @ X$=C$ @ FOR I=0 TO N @ E(I)=T(I) @ T(I)=Z(I)
720 Z(I)=X(I) @ X(I)=E(I) @ NEXT I @ GOTO 40
730 C$=X$ @ X$=Y$ @ Y$=C$ @ FOR I=0 TO N @ E(I)=X(I) @ X(I)=Y(I) @ Y(I)=E(I) @ NEXT I @ GOTO 40
740 GOSUB 920 @ INPUT 'a=';A$ @ S=1 @ X(0)=1 @ FOR I=1 TO N
750 S=S*(VAL(A$)-I+1) @ X(I)=S/FACT(I) @ NEXT I @ X$="(1+X)^(\"&A$&\"" @ GOTO 40
760 DISP 'C|x' @ FOR I=0 TO N @ X(I)=Y(I) @ Y(I)=Z(I) @ Z(I)=T(I)
770 NEXT I @ X$=Y$ @ Y$=Z$ @ Z$=T$ @ GOTO 40
780 GOSUB 920 @ DESTROY X @ DIM X(N) @ FOR I=1 TO N STEP 2 @ X(I)=(-1)^IP(I/2)/I @ NEXT I
790 X$='ATAN' @ GOTO 40
800 GOSUB 920 @ DESTROY X @ DIM X(N) @ FOR I=1 TO N STEP 2 @ X(I)=1/I @ NEXT I
810 X$='ATH' @ GOTO 40
820 GOSUB 920 @ DESTROY X @ DIM X(N) @ S=1 @ X(1)=1 @ FOR I=3 TO N STEP 2
830 S=S*(I-2)/(I-1) @ X(I)=S/I @ NEXT I @ X$='ASIN' @ GOTO 40
840 GOSUB 920 @ DESTROY X @ DIM X(N) @ S=1 @ X(1)=1 @ FOR I=3 TO N STEP 2
850 S=S*(I-2)*(I-1) @ X(I)=(-1)^IP(I/2)*S/I @ NEXT I @ X$='ASH' @ GOTO 40
860 GOSUB 920 @ RESTORE 880 @ IF N>10 THEN DISP 'WRN: N>10' @ BEEP @ WAIT 2 @ GOTO 40
870 FOR I=0 TO N @ READ X(I) @ NEXT I @ X$='TAN' @ GOTO 40

```

=====

```
880 DATA 0,1,0,1/3,0,2/15,0,17/315,0,62/2835,0
```

=====

```

890 GOSUB 920 @ RESTORE 910 @ IF N>10 THEN DISP 'WRN: N>10' @ BEEP @ WAIT 2 @ GOTO 40
900 FOR I=0 TO N @ READ X(I) @ NEXT I @ X$='TH' @ GOTO 40

```

=====

```
910 DATA 0,1,0,-1/3,0,2/15,0,-17/315,0,62/2835,0
```

=====

```

920 T$=Z$ @ Z$=Y$ @ Y$=X$ @ FOR I=0 TO N @ T(I)=Z(I) @ Z(I)=Y(I) @ Y(I)=X(I) @ NEXT I
930 RETURN

```

=====

```

940 SUB FR(G,B,O,P)
950 H=G @ M=INT(H) @ B=M @ O=1 @ A=1 @ IF NOT FP(ABS(G)) THEN 970
960 H=1/(H-M) @ M=INT(H) @ U=B*M+A @ A=B @ B=U @ F=O*M+D @ D=O @ O=F @ IF ABS(G-U/F)>P THEN 960
970 END SUB

```

HP71: TELLEMENT COMPLEXE


```

10 ON ERROR GOTO 20 @ PURGE KEYS
20 COPY CX TO KEYS @ OFF ERROR
30 GDISP CHR$(62)&CHR$(65)&CHR$(127)&CHR$(65)&CHR$(34) @ WINDOW 3 @ CFLAG 5,6 @ USER ON
40 DELAY 0 @ DEGREES @ STD @ LC OFF

```

```

=====
50 'SHOW': DISP X$ @ STOP

```

```

=====
60 'SOUS': SFLAG 6 @ 'ADD': GOSUB 'PILE'
70 X=VAL(Y$(J-1,J-1)&Y$(J+1,J+1))+(-1*FLAG(6)+NOT FLAG(6))*VAL(X$(I-1,I-1)&X$(I+1,I+1))
80 A=(SGN(X)+2)*(X#0)+(X=0)*3
90 X$=STR$(VAL(Y$(1,J-2))+(-1*FLAG(6)+NOT FLAG(6))*VAL(X$(1,I-2)))&'-' + '[A,A]&'i'&STR$(ABS(X))
100 CFLAG 6 @ GOTO 160

```

```

=====
110 'DIV': SFLAG 6 @ 'MUL': GOSUB 'PILE'
120 A=VAL(X$(1,I-2)) @ C=VAL(Y$(1,J-2)) @ B=VAL(X$(I-1,I-1)&X$(I+1,I+1))
130 D=VAL(Y$(J-1,J-1)&Y$(J+1,J+1)) @ IF FLAG(6) THEN 440
140 X=B*C+A*D @ E=(SGN(X)+2)*(X#0)+(X=0)*3
150 X$=STR$(A*C-B*D)&'-' + '[E,E]&'i'&STR$(ABS(X))
160 Y$=Z$ @ Z$=T$ @ GOTO 'SHOW'

```

```

=====
170 'ENTER': GOSUB 'PILE' @ T$=Z$ @ Z$=Y$ @ Y$=X$ @ SFLAG 5 @ GOTO 'SHOW'

```

```

=====
180 'ECH': GOSUB 'PILE' @ A$=X$ @ X$=Y$ @ Y$=A$ @ GOTO 'SHOW'

```

```

=====
190 'RDN': GOSUB 'PILE' @ A$=X$ @ X$=Y$ @ Y$=Z$ @ Z$=T$ @ T$=A$ @ GOTO 'SHOW'

```

```

=====
200 'RUP': GOSUB 'PILE' @ A$=T$ @ T$=Z$ @ Z$=Y$ @ Y$=X$ @ X$=A$ @ GOTO 'SHOW'

```

```

=====
210 'STO': GOSUB 'PILE'
220 USER @ INPUT 'STO ' ;A$ @ USER
230 ON POS('YZT',A$)+1 GOTO 220,240,250,260
240 Y$=X$ @ GOTO 'SHOW'
250 Z$=X$ @ GOTO 'SHOW'
260 T$=X$ @ GOTO 'SHOW'

```

```

=====
270 'RCL': GOSUB 'PILE'
280 USER @ INPUT 'RCL ' ;A$ @ USER
290 ON POS('XYZT',A$)+1 GOTO 280,300,310,320,330
300 A$=X$ @ GOTO 340
310 A$=Y$ @ GOTO 340
320 A$=Z$ @ GOTO 340
330 A$=T$
340 T$=Z$ @ Z$=Y$ @ Y$=X$ @ X$=A$ @ GOTO 'SHOW'

```

```

=====
350 'MOD': GOSUB 'PILE'
360 A=VAL(X$(1,I-2)) @ B=VAL(X$(I-1,I-1)&X$(I+1,I+1))
370 DISP CHR$(124)&'z'&CHR$(124)&'=';SQR(A^2+B^2) @ WAIT 3 @ DISP CHR$(16)&'=';ANGLE(A,B);'Deg'
380 WAIT 3 @ GOTO 'SHOW'

```

```

=====
390 'CLST': DESTROY ALL @ STOP

=====

400 'BAR': SFLAG 6 @ 'CHS': GOSUB 'PILE'
410 E=(SGN(VAL(X$[I-1,I-1]&'1'))+2)*(VAL(X$[I+1])#0)+(VAL(X$[I+1])=0)*3
420 X$=STR$((-1*NOT FLAG(6)+FLAG(6))*VAL(X$[1,I-2]))&' - +' [E,E]&X$[I] @ CFLAG 6 @ GOTO 'SHOW'

=====

430 'CLX': GOSUB 'PILE' @ X$=Y$ @ Y$=Z$ @ Z$=T$ @ GOTO 'SHOW'
440 SHORT I,J @ I=(C*A+D*B)/(A^2+B^2) @ J=(D*A-C*B)/(A^2+B^2) @ E=(SGN(J)+2)*(J#0)+(J=0)*3
450 X$=STR$(I)&' - +' [E,E]&'i'&STR$(ABS(J)) @ Y$=Z$ @ Z$=T$ @ CFLAG 6 @ GOTO 'SHOW'

=====

460 'PILE': IF DISP$='' THEN CFLAG 5 @ GOTO 490 ELSE IF FLAG(5) THEN CFLAG 5 @ GOTO 480
470 T$=Z$ @ Z$=Y$ @ Y$=X$
480 X$=DISP$
490 IF X$='' THEN X$='0+i0'
500 IF Y$='' THEN Y$='0+i0'
510 I=POS(X$,'i') @ J=POS(Y$,'i')
520 IF I<3 THEN ON I+1 GOTO 530,540,550 ELSE 560
530 X$=X$&'i0' @ GOTO 560
540 X$='0+&X$&'1'[LEN(X$),LEN(X$)] @ GOTO 560
550 X$='0'&X$
560 IF J<3 THEN ON J+1 GOTO 570,580,590 ELSE 600
570 Y$=Y$&'i0' @ GOTO 600
580 Y$='0+&Y$ @ GOTO 600
590 Y$='0'&Y$
600 I=POS(X$,'i') @ J=POS(Y$,'i') @ IF X$[I+1]='' THEN X$=X$&'1'
610 IF Y$[J+1]='' THEN Y$=Y$&'1'
620 RETURN

=====

630 'FIN': USER @ INPUT 'FIN: Oui ou Non :';A$ @ USER @ IF A$# 'O' THEN 'SHOW'
640 PURGE KEYS @ WINDOW 1 @ END

*****

UN BRIDGE CE SOIR ?

=====

5 SUB MAIN
10 DESTROY R1 @ OPTION BASE 0 @ DIM R1(38)
20 FOR A=0 TO 4 @ FOR R=0 TO 4 @ FOR D=0 TO 4 @ FOR V=0 TO 4
40 IF A+R+D+V>13 THEN 70
50 S=4*A+3*R+2*D+V

```

HP-71 ASSEMBLEUR

SERGE VAUDENAY 36
 SERGE VAUDENAY 37
 LAURENT ISTRIA 40
 MICHEL MARTINET 46

ASSEMBLEUR: STOP, ENCORE
 GANYMEDE (OTHELLO)
 DES DATES, ENCORE DES DATES...
 LE COIN DES LHEX

ASSEMBLEUR: STOP, ENCORE

Et si on passait aux choses sérieuses.

Coucou, c'est encore moi.

Vous vous demandiez ce que j'étais devenu pendant tout ce temps, avais-je enfin été renvoyé du club, avais-je enfin été censuré ? Et bien non. J'étais tout simplement en vacances, retranché chez moi pour vous pondre de nouveaux programmes.

Tenez vous bien ! Je me suis lancé dans l'assembleur. Et oui, j'ai moi aussi eu le plaisir d'avoir des memory lost (au fait, quelqu'un a-t-il réussi à traduire ce message d'erreur, ou est-il condamné à rester anglais???) et aux anomalies d'un 71 planté... Je dirai à ceux qui n'osent toujours pas assembler en coeur que mis à part les problèmes de plantage, l'assemblage sur 71 est incomparablement plus souple que celui sur 41. J'ai tout d'abord eu la surprise de voir mes premiers programmes tourner du premier coup. Mais cela n'a pas toujours été le cas, par la suite.

Je me suis attaqué à un programme d'OTHELLO. Pour cela, il m'a fallu faire l'utilitaire dont je vais vous parler aujourd'hui. Le problème que j'ai rencontré était de dépister les erreurs d'assemblage d'un programme. Une méthode bien connue consiste à faire un fichier listing et de lancer la commande S/\^* avec l'éditeur de texte. Mais j'ai eu un problème de mémoire, car le fichier listing est beaucoup plus grand que le fichier source, ce qui n'est pas peu dire... J'ai donc écrit un lex qui permet d'arrêter l'assemblage sur un message d'erreur en faisant bip, et d'attendre une pression de touche pour continuer. Je tiens à vous avertir cependant que ce lex est à utiliser sous toute réserve, car il m'est arrivé, plusieurs fois, de voir un assemblage s'arrêter sur un memory lost. Je ne sais pas si c'est à cause de ce lex, mais je vous conseille tout de même de garder une copie de vos fichiers.

Le principe du programme est simple. Il "fait semblant" d'intercepter le poll de traduction pTRANS, qu'utilise l'assembleur pour éventuellement demander la traduction d'un

message d'erreur, et il affiche le message avant l'assembleur en respectant le DELAY et en envoyant un bip grâce à la routine MFWRN. Il suffit alors de mettre un DELAY infini pour que le programme s'arrête et attende une pression de touche. L'assembleur affichera alors son message suivi du numéro de ligne où il y a eu erreur, et l'assemblage reprend. Oui, me direz-vous, mais comment reconnaître que c'est bien l'assembleur qui a envoyé le poll ? Avant de lancer un assemblage, on armera le flag 63 qui servira à mettre le lex en action, et on le désarmera ensuite. Pour ne pas avoir tous ces ennuis de configuration, nous allons définir un mot forth qui remplacera ASSEMBLE.

```
DECIMAL 20 STRING SCT " SFLAG 63 @ DELAY INF"
SCT SI ( crée la chaine de configuration.)
0 STRING NULL ( fait une chaine NULL$ qui ne
touche pas au PAD.)
: ASS ( str--- ) NULL LISTING SI ( ne fait pas
de fichier LISTING.)
SCT BASICX ( configure.)
ASSEMBLE ( assemble le fichier dont la chaine
est dans la pile.)
" CFLAG 63 @ DELAY 0,0 @ BEEP" BASICX (
déconfigure.)
;
```

Vous devrez désormais utiliser ASS au lieu de ASSEMBLE (remerciez-moi je vous épargne 5 pressions de touches). Entrez maintenant le petit LEX listé ci-après, n'oubliez pas d'éteindre et de rallumer votre 71 pour rendre le lex actif, si vous n'avez fait aucune erreur. Sinon, n'hésitez pas à tout détruire. J'espère qu'il vous rendra plein de services, et plein de memory lost, car si je suis le seul à en profiter, he ben c'est pas juste. N'oubliez pas d'adresser vos réclamations au journal ou d'envoyer des insultes anonymes. A bientôt, (peut-être).

Serge Vaudenay


```

LEX 'ASSERR' * Nom du LEX
ID #E1 * à vous de choisir l'id
MSG 0 * aucun message
POLL POLHND * poll actif
ENDTXT * fin du "préambule"
pTRANS EQU #EF * valeur du poll pTRANS
FLGREG EQU #2F6E9 * adresse de stockage
* * des drapeaux (cf IDS 1)
MFWRN EQU #093BC * routine d'affichage
* * d'avertissement
HERE EQU #2FB93 * adresse de la variable
* * forth HERE
POLHND LC(2) pTRANS * traitement si poll
* * pTRANS
?B=C B
GOYES ITRANS
RTNSXM
ITRANS LC(5) (FLGREG)+15 * va voir le flag 63
CDOEX
C=DATO S
CDOEX
C=C+C S
GOC WARM * flag levé?
RTNSXM * non, fin
WARN P= 14 * faire bip, respecter
* * le DELAY, ne pas
* * toucher ERRN, pas de
* * préfixe
* * (WRN: ou WRN:) (cf doc
* * sur MFWRN, IDS 2 ou 3)
C=R0
D0=(5) HERE * met l'adresse de fin
* * du dictionnaire forth
* * dans D0
A=DATO A
D0=A
DATO=C W * sauvegarde de R0 et R2
D0=D0+ 16
A=R2
DATO=A W
GOSBVL MFWRN * affichage de l'erreur
D0=(5) HERE * remet HERE dans D0
C=DATO A
D0=C
C=DATO W * recupere R0 et R2
R0=C
D0=D0+ 16

```

C=DATO W

R2=C

RTNSXM

*

* fin, on fait comme si

* rien ne s'était passé

GANYMEDE (partie I)

C'est encore moi!

Cet article est le premier d'une série qui décrira mon programme d'othello. Ce programme (écrit en assembleur, forth et basic) s'appelle Ganyède. Son intérêt est qu'il montre les avantages du Forth et des primitives Forth. Il pourra servir de modèle ou d'exemple au projet pop-corn pour l'année prochaine.(NDLR: Si tout va bien). Nous allons, aujourd'hui, étudier la manière dont Ganyède stocke le damier. Un damier, en Forth, sera crée par:

CREATE nom-du-damier 16 ALLOT

C'est donc une zone de 16 octets.

Les huit premiers contiennent une configuration de bits indiquant les cases vides (bit=0) et les cases pleines (bit=1). Toutes les cases des bords sont vides, car seules les 6x6 cases du centre sont disponibles.

Les 8 octets suivant contiennent une configuration de bits indiquant les pions 1 (bit=1) et les pions 0 (bit=0) des cases occupées. Le bit de poids faible de chaque configuration, c'est à dire le bit de poids faible du premier quartet dans la mémoire correspond à la case 00 de l'échiquier. Le bit de poids faible du deuxième quartet correspond à la case 10 (en hexadécimal).

Je vous donne donc rendez-vous au prochain article pour décrire les primitives de manipulation de case. Je conseille aux débutants, et même aux autres d'essayer d'y réfléchir, et de faire une primitive pour aller chercher le contenu d'une case, et pour mettre

quelque chose (ou rien) dedans.
A bientot.

```

-----
7 I I I I I I I I
-----
6 I I I I I I I I
-----
5 I I I I I I I I
-----
4 I I I I oI*I I I I
-----
3 I I I I*I oI I I I
-----
2 I I I I I I I I
-----
1 I=I I I I I I I I
-----
0 I I I I I I I I
-----
0 1 2 3 4 5 6 7

```

= case 10

Serge Vaudenay.

GANYMEDE (suite).

Bonjour, me revoila.

Je vous ai parlé, le mois dernier, de mon programme d'othello, et nous avons défini un tableau de pions. Avant de poursuivre, je dois dire que mon programme a fini bon dernier au concours de l'OI à cause d'une erreur dans la fonction d'évaluation. Mais nous n'en sommes pas encore là. Je suppose que vous avez tous réfléchis sur le problème de la semaine dernière: faire les primitives CJ@ et CJ!. Voici ma solution. Ce n'est pas forcément la meilleure, mais pour la suite du programme, il

faudra prendre cette version pour avoir les mêmes adresses dans le dictionnaire. Prenez donc un dictionnaire vide, et assemblez les primitives suivantes.

```

FORTH
WORD 'CJ@' * (adr c --- n)
* * n=contenu de la case
* * (-1:pion1 0:vide
* * 1:pion0)
C=DAT1 A * dépilage de c
D1=D1+ 5
A=DAT1 A * dépilage de adr
* * (adresse du tableau)
GOSUB CJ@ * calcul de n
GOSUB GTC
DAT1=C A * empilage de n
RTNCC * fin.

```

données A[A] adresse du tableau
C[B] numéro de case
sortie A[A] inchangé
C[A] valeur de la case
(0:vide 1:pion0 3:pion1)
D[W] masque

utilise A[14:15] C[W] D[W]

0 niveau de sous programme

Donne le contenu d'une case

```

CJ@ AD1EX * fabrique un "masque"
* * dans D

```

ASLC

ASLC

A=C B

ASRC

C=0 W

C=C+1 W

CJ@1 A=A-1 S

GOC CJ@2

C=C+C B

GOTO CJ@1

CJ@2 ASRC

CJ@3 A=A-1 S

GOC CJ@4

CSL W

CSL W

GOTO CJ@3

CJ@4 D=C W

* masque construit;

C=DAT1 W

* recherche du contenu

*

* de la case

*

* (vide-occupée)

	C=C&D W				* * imaginaire), fin.
	A=0 S			GONC CJ12	
	?C=0 W	* case vide?		RTNCC	
	GOYES CJ05	* oui, n=0, CJ05 et fin	CJ12	C=C-1 B	
	A=A+1 S	* non, n>0 (1 au minimum)		GOTO CJ1C	
	D1=D1+ 16	* recherche du contenu		*****	
	*	* de la case (pion0-pion1)		données A[A] adresse du tableau	
	C=DAT1 W			C[B] numéro de case	
	D1=D1- 16			ST0 bit de contenu (pion0-pion1)	
	C=C&D W			sortie A[A] inchangé	
	?C=0 W	* pion0?		utilise A[14:15] C[W] D[W] ST1	
	GOYES CJ05	* oui, n=1, CJ05 et fin		0 niveau de sous-programme	
	A=A+1 S	* non, n=3		*****	
	A=A+1 S			CJ! ST=0 1	* force un remplissage
CJ05	C=0 W	* mise en place du	CJ!C	AD1EX	* construit un "masque"
	*	* résultat		ASLC	
	C=A S			ASLC	
	CSLC			A=C B	
	AD1EX			ASRC	
	RTNCC	* fin.		C=0 W	
GTC	A=0 B	* transforme 0 en 0, 1		C=C+1 W	
	*	* en -1, et 3 en 1	CJ14	A=A-1 S	
	A=A+1 B			GOC CJ15	
	A=A+1 B			C=C+C B	
	A=A+1 B			GOTO CJ14	
	?A#C B			CJ15 ASRC	
	RTNYES			CJ16 A=A-1 S	
	C=0 A			GOC CJ17	
	C=C-1 A			CSL W	
	RTNCC			CSL W	
	WORD 'CJ!'	* (adr c n ---) remplit		GOTO CJ16	
	*	* la case c du tableau	CJ17	?ST=0 1	* inverse le masque si
	*	* adr avec n		*	* on doit vider une case
	C=DAT1 A	* dépile n en ST0 et ST1		GOYES CJ18	
	*	* (ST0=bit de contenu,		C=-C-1 W	
	*	* ST1=bit de remplissage)	CJ18	D=C W	* masque terminé
	ST=1 1			C=DAT1 W	
	?C=0 A			?ST=0 1	* remplissage de case?
	GOYES CJ11			GOYES CJ19	* oui, CJ19
	ST=0 1			C=C&D W	* non, met le bit de
	ST=0 0			*	* remplissage de la case
	C=C+1 A			*	* à 0
	GONC CJ11			DAT1=C W	
	ST=1 0			AD1EX	
CJ11	D1=D1+ 5	* dépile adr en A[A]		RTNCC	* case vidée, fin.
	C=DAT1 A		CJ19	C=C!D W	* met le bit de
	D1=D1+ 5			*	* remplissage de la case
	A=DAT1 A	* dépile c en C[B]		*	* à 1
	D1=D1+ 5			DAT1=C W	
	C=C+1 B	* si c=FF (case		D1=D1+ 16	

```

C=DAT1 W
?ST=1 0      * pion1? Fait un "masque
*            * de contenu" (inverse
*            * le masque pour un pion0)
GOYES CJ!3    * oui, CJ!3 et fin
CDEX W        * non,
C=-C-1 W      * inverse le masque
C=C&D W       * met un pion0
GOTO CJ!10    * CJ!10 et fin
CJ!3 C=C!D W   * met un pion1
CJ!10 DAT1=C W
D1=D1- 16
AD1EX
RTNCC          * fin.
WORD 'RESET'   * (adr ---) initialise
*             * un tableau adr
A=DAT1 A       * dépile adr
D1=D1+ 5
AD1EX
C=0 W          * met tous les bits de
*             * remplissage à 0
DAT1=C W
AD1EX
RTNCC          * fin.

```

Voilà, si tout va bien, vous devez avoir les équivalences suivantes:

CJ@ EQU #2FCD9

CJ! EQU #2FDAB

CJ!C EQU #2FDAE

GTC EQU #2FD4B

La primitive RESET a pour effet de vider un tableau.

-Pour créer un tableau, faites: DECIMAL CREATE (nom) 16 ALLOT (nom) RESET

-Pour voir le contenu d'une case: (nom) (case) CJ@ .

-Pour remplir une case: (nom) (case) (contenu) CJ!

Ceux qui ont eu le courage de désosser le listing se sont sûrement heurtés aux termes "remplissage" et "contenu". J'ai appelé bits de remplissage les 64 premiers bits d'un tableau disant si une case est vide, et bits de contenu les 64 bits suivant du tableau, définissant le contenu des cases non vides (voir épisode précédent).

Quand vous aurez bien compris les principes de ce programme, et la philosophie du FORTH, vous pourrez faire une primitive de test de validité d'un coup (un coup nul, ou "passe", se note FF).

Je vous proposerai ma solution la prochaine fois. Je vous propose également, ce mois-ci, la partie BASIC du programme. Bien sur, vous ne pourrez pas l'utiliser tant que vous n'aurez pas le mot FORTH "GO" qui est le coeur du programme. Pour finir, voici la liste des primitives que vous aurez la prochaine fois:

-?CANI (adr c j --- b) teste la validité d'un coup

-PUTC (adr c j ---) joue un coup

-LIST (adr j --- liste) donne une liste de coups jouables

Tous à vos claviers.

Serge Vaudenay

DES DATES, ENCORE DES DATES, TOUJOURS DES DATES.

"Bis repetita placent" dit le proverbe.

Et bien, puisque l'entendement général semble prétendre que les choses répétées plaisent, répétons-nous. Dans le numero 27 de JPC, Monsieur Xavier Bille nous faisait part de son programme de calcul sur les dates. Celui-ci étant écrit en BASIC, il prenait une place relativement importante dans la mémoire de titan. Or, de longue date, j'avais envie d'écrire le LEX capable d'en faire autant. Il n'en fallait pas plus pour me décider; "doublé" sur un terrain qui m'était cher, je ne pouvais l'accepter.

C'est ainsi que je me mis en quête des routines assembleur susceptibles de me faciliter le travail. Le HP71 acceptant des entrées de dates, il était forcé qu'il les puisse traiter. Un instant de recherche me permit de découvrir YMDDAY, routine permettant la conversion d'une date dont le quantième, le mois et l'année sont stockés respectivement dans D, B et A en un entier dans C qui représente le nombre de jours écoulés entre la date et le premier janvier de l'an 1. Nantis de cette routine éminement

importante, je me mis à concevoir ce LEX.

Le mode d'utilisation de ses fonctions est fort simple; si l'on considère D1 et D2 comme étant deux dates exprimées soit sous la forme jj.mmaaaa soit sous la forme mm.jjaaaa, (nous verrons plus tard comment passer d'un format à l'autre), alors, les opérations se font ainsi:

-DDAYS(D1,D2) donne le nombre de jours écoulés entre les dates D1 et D2

-DOW(D1) rend un nombre correspondant au jour de la semaine de D1 (0 pour Dimanche, etc)

-DOW\$(D1) donne le jour dans la semaine de D1

-DMY permet le passage dans le format européen: jj.mmaaaa

-MDY permet le passage dans le format américain: mm.jjaaaa

Je n'ai pas pu écrire une routine DATE+ qui aurait permis d'ajouter un certain nombre de jours à une date. En effet, il semble que la routine DAYYMD (devinez ce qu'elle fait) soit "bugger".

Les résultats obtenus en appliquant les formules utilisées dans la ROM du 71 ne sont valables que pour des dates comprises entre le 15 octobre 1582 et le 31 décembre 9999. Pourquoi une date de départ si précise, me demandez-vous? La réponse est fort simple

Or donc, en ces temps reculés, où les grands se battaient à coups d'arsenic et de duels, et où les faibles n'avaient droit qu'à la massue, vivait en sa ROM (excusez-moi, ROME, déformation professionnelle) merveilleuse, un bon pape. Et ce pape, non content d'avoir une mule, avait une passion: L'astrologie-nomie(à l'époque, on ne faisait pas la différence)

Et ce pape avait remarqué que des années de 365 jours avaient pour conséquence de nous faire perdre du temps par rapport au soleil. Il décidât donc, bon gré, mal gré, d'ajouter un jour supplémentaire, disons, tous les 4 ans.

Mais, en poussant plus avant ses calculs, il s'aperçut qu'au bout de quelques temps nous nous mettrions à gagner du temps sur la course de Râ. La décision fût alors prise de compléter la loi énoncée plus haut: elle ne s'appliquerait que si ces mêmes dates divisibles par 4 ne l'étaient pas par 100.

Mais, le mal était fait. Nous avions dix jour de retard sur le soleil. C'est ainsi que les gens

s'endormir le 4 octobre 1582, et se réveillèrent le 15 octobre, ayant ainsi rattrapés les dix jours. Les seules personnes à se plaindre furent les femmes qui décrèterent qu'elles avaient vieilli de 10 jours en une nuit. C'est ainsi que nous vivons maintenant sous le calendrier Grégorien, du nom de ce ci-gentil pape Grégoire VII. Aux alentours de la fin du 19ème siècle, une nouvelle clause fût par ailleurs ajoutée, décrétant que si une date était divisible par 400, elle serait alors aussi bissextile (de bis- et sextus, car sous le calendrier Julien, le sixième jour avant les calendes de mars était doublé tous les 4 ans.)

Pour les explications relatives au fonctionnement du programme, reportez-vous au listing source.

Voilà, j'espère que ces quelques instants d'histoire du calendrier vous auront intéressés

L. ISTRIA (S.I.G.#3,P.P.C.#3)

P.S. je tiens à remercier ici un vieil ami à moi, monsieur A. Daudet pour m'avoir prêté quelques passages de ses "Lettres de mon moulin". Qu'il lui soit témoigné toute ma gratitude.

LEX	'DATELEX'
ID	#E1
MSG	0
POLL	0
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

ENTRY DELTA		*	* la semaine
CHAR #F		P= 0	
ENTRY EUR		C=0 A	
CHAR #D		LCHEX 7	
ENTRY WEEK\$		GOSBVL IDIVA	* MOD(Nombre de jour
CHAR #F		*	* depuis le 1er janvier
ENTRY WEEK		*	* 0001,7)
CHAR #F		?ST=1 2	
ENTRY AME		GOYES DOWTXT	* Si flag(2)=1 alors il
CHAR #D		*	* faut convertir en une
KEY 'DDAYS'		*	* chaîne de caractères
TOKEN 52		CSRC	
KEY 'DMY'		C=0 M	
TOKEN 53		C=0 XS	
KEY 'DOW\$'		CSRC	* Le nombre tient dans
TOKEN 54		*	* un seul digit (de 1 a
KEY 'DOW'		*	* 7)
TOKEN 55		GOTO FIN2	
KEY 'MDY'		DOWTXT D1=D1+ 16	
TOKEN 56		AD1EX	
ENDTXT		R1=A	* R1(A)= début de la
NIBHEX 811		*	* pile de données (hi mem)
WEEK\$ ST=1 2		AD1EX	
GOTO WEEKE		GOSBVL TBLJMC	* Su-per-be !!! un
NIBHEX 8822		*	* modèle de CASE OF de
DELTA GOSUB FLTDB0		*	* poche
D1=D1+ 16	* Paramètre suivant	REL(3) DIM	
R0=C	* Sauvegarde de la	REL(3) LUN	
*	* conversion de la	REL(3) MAR	
*	* première date	REL(3) MER	
GOSUB FLTDB0		REL(3) JEU	
C=R0		REL(3) VEN	
?C>A W	* Attention, si C est	REL(3) SAM	
*	* supérieur à A	LUN LCASC 'LUNDI'	
GOYES POS		P= 9	
A=A-C W		GOTO M10	
GOSBVL HDFLT		MAR LCASC 'MARDI'	
A=-A-1 S	* Il faut rétablir le	P= 9	
*	* signe	GOTO M10	
GOTO FIN		MER LCASC 'MERCREDI'	
POS C=C-A W		P= 15	
A=C W		GOTO M16	
GOSBVL HDFLT		JEU LCASC 'JEUDI'	
FIN C=A W		P= 9	
FIN2 GOVLNG FNRTN4		GOTO M10	
NIBHEX 811		VEN LCASC 'VENDREDI'	
WEEK ST=0 2		P= 15	
WEEKE GOSUB FLTDB0		GOTO M16	
A=A-1 A	* Nécessaire pour	SAM LCASC 'SAMEDI'	
*	* obtenir le bon jour de	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 * D1 à été restauré, on
* * écrit la valeur de C
* * sur la MATHSTACK
GOSBVL D=AVMS * Sans commentaires
ST=0 0
P= 0
GOVLNG ADHEAD * Et hop, on ajoute l'en
* * tête et on sort par
* * EXPR (ST=0 0,
* * précédemment)

```

```

* FLTDBO
* But: C'est la routine principale, elle permet
* de traduire un nombre en decimal flottant situé
* sur la MATH-STACK en une série de 3 nombres BCD
* situés dans A,B et D
* A contient l'année
* B contient le mois
* D contient le jour

```

```

FLTDBO GOSBVL POP1N
B=A W
P= 0
LC(2) fLDATE
SETHex
ST=0 0
GOSBVL SFLAG? * Routine machine de
* * test d'un drapeau
GONC S0
ST=1 0
S0 C=B W * On replace la date à
* * analyser dans C
?C=0 P * Le jour du mois est-il
* * inferieur a 10 ?
GOYES S1
CSLC
S1 CSLC
CSLC
D=C W * Nombre de jour doit
* * être écrit dans D, en
* * BCD (décimal codé
* * binaire)
P= 1

```

```

C=0 WP
D=D-C W
CSLC
CSLC
B=C W
C=0 WP
B=B-C W * Même chose pour le
* * mois dans B
GOSBVL CSLC4
A=C W
P= 3
C=0 WP
A=A-C W * Et pour l'année dans A
?ST=0 0 * On procède à
* * l'actualisation du
* * mois et du jour

```

GOYES NON

BCEX A

CDEX A

BCEX A

NON C=0 A

```

* C'est ici que commence la routine qui teste la
* validité de la date. Elle doit vérifier 3 faits
* différents:
* La date est-elle située après le 15 octobre 1582
* Le nombre de jour dans le mois est-il correct ?
* Si 29 février, l'année est-elle bissextile ?

```

```

P= 0
LCHEX 1582
?A>C A * Année correcte ?
GOYES SUITE
?A=C A
GOYES TRAITE
arg1 GOTO argerr
SUITE LCHEX 12
?B>C B * Mois correct ?
GOYES arg1
LCHEX 07 * Problème de traitement
* * du nombre de jour dans
* * le mois
?B>C B
GOYES >JUIL
LCHEX 2
?C=B B * Traitement du mois de
* * février
GOYES FEVR
C=C-1 P
C=C&B P

```

```

P= 1
LCHEX 3 * On trouve ici le
* * nombre maximum de jour
* * dans les mois avant août
GOTO TEST
>JUIL R1=A
LCHEX 1
A=C P
C=C&B P
C=A-C P
A=R1
P= 1
LCHEX 3 * Même chose, pour les
* * mois à partir d'août
TEST ?D>C B
GOYES argerr
GONC TERM
FEVR LCHEX 29
?D>=C B
GOYES PROB
GONC TERM
argerr GOVLNG ARGERR
TRAITE LCHEX 0010 * Traite l'année 1582,
* * où commence le
* * calendrier grégorien
?B>C B
GOYES >JUIL
?B#C B
GOYES argerr
LCHEX 15
?D<C B
GOYES argerr
GOTO >JUIL * Le premier jour est le
* * 15 octobre 1582
* * Jour impossible
PROB ?D>C B
GOYES argerr
C=B B * Ici, la date est
* * forcément un 29 février
R1=C
R2=A
C=0 A
LCHEX 4
SETDEC
GOSBVL IDIVA
?B#0 A * Si l'année n'est pas
* * multiple de quatre, il
* * y a erreur
GOYES argerr
A=R2
?A#0 B

```

```

GOYES REST
LCHEX 400
GOSBVL IDIVA
?B=0 A * Si l'année est
* * divisible par 400, la
* * date est valide
GOYES REST
GOTO argerr
REST C=R1
B=C B
A=R2
TERM GOVLNG YMDDAY * Routine de
* * transformation d'une
* * date en un nombre de
* * jour
REL(5) WDC
REL(5) WP
EUR LC(2) fIDATE
GOTO SORTIE
REL(5) WDC
REL(5) WP
AME LC(2) fIDATE
P= 1
SORTIE GOSBVL SFLAG* * On arme ou l'on
* * désarme le flag -27
* * suivant le cas
GOVLNG NXTSTM
WDC GOVLNG OUTELA
WP RTNCC

```

PROGRAMME D'OTHELLO

10 CALL GANYMEDE

```
=====
20 SUB GANYMEDE ! ce programme nécessite la fonction KEYWAIT$ et un fichier FORTH adéquat
30 WINDOW 3,17 @ GDISP '' @ DIM A$[132]
40 FORTHX 'HEX ECH1 RESET' @ LC OFF @ OPTION ROUND NEAR
50 N=4 @ T0=0 @ T1=0 @ F=0
60 DISP 'Tu commences(?)'
70 ON POS('NO',KEYWAIT$)+1 GOTO 70,80,90
80 F=1-F
90 IF F THEN DISP 'Je commence.' ELSE DISP 'Tu commences.'
100 FORTHX 'ECH1 SWAP INIT',F
110 T=TIME @ ON 2-F GOTO 120,130
120 FORTHX 'ECH1 32 1 PUTC' @ DISP 'Je joue C5.' @ N=N+1
130 T1=T1+TIME-T @ T=TIME @ D1=T1 DIV 60
140 T0=T0+TIME-T @ T=TIME @ D0=T0 DIV 60
150 GOSUB 360
160 K$=KEYWAIT$ @ T0=T0+TIME-T @ T=TIME @ IF T0 DIV 60>D0 THEN D0=T0 DIV 60 @ GOSUB 360
170 I=POS('P/+ABCDEF##46#99',K$)+1
180 ON I GOTO 160,430,350,340,380,200,200,200,200,200,160,190,160,160,420,160,160
190 FORTHX 'ECH1 VIEW-ECH' @ GOTO 160
200 X=(NUM(K$)-64)*16 @ DISP K$; @ INPUT '';Y
210 CFLAG 5 @ IF Y<1 OR Y>?6 OR FP(Y) THEN 160
    - coeur du programme: mot FORTH "GO"
220 T0=T0+TIME-T @ T=TIME @ DISP 'Heum! Heum!'; @ FORTHX 'GO',X+7-Y @ R=FORTH1 @ DISP
230 IF R=0 THEN DISP K$&STR$(Y)&' ???' @ GOTO 140
240 N=N-NOT FLAG(5) @ IF N=36 THEN 280
250 IF R=255 AND FLAG(5) THEN 280 ELSE IF R=255 THEN DISP 'Je passe.' @ GOTO 130
260 X=R DIV 16 @ Y=7-MOD(R,16) @ DISP 'Je joue '&CHR$(64+X)&&CHR$(48+Y)&'.'
270 N=N+1 @ IF N=36 THEN 150 ELSE 130
280 T1=T1+TIME-T @ GOSUB 360 @ FORTHX 'ECH1 SCORE' @ P0=FORTH1 @ P1=FORTH1 @ WINDOW 1,17
290 V=IP(MAX(P0,P1)*36/(P0+P1)+.5) @ P=36-V
300 IF P1=P0 THEN DISP 'Egalite: 36-36'
310 IF P1<P0 THEN DISP 'Gn! gn!: '&STR$(V)&'-'&STR$(P)
320 IF P1>P0 THEN DISP 'Bravo: '&STR$(V)&'-'&STR$(P)
330 K$=KEYWAIT$
340 WINDOW 1 @ DISP 'Au revoir.' @ END
350 SFLAG 5 @ X=255 @ Y=7 @ GOTO 220
360 D0=MOD(D0,100) @ D1=MOD(D1,100) @ A$=GDISP$ @ WINDOW 3 @ DISP USING '15X2D':"2D';D0,D1
370 WINDOW 3,17 @ GDISP A$[1,102]&GDISP$[103] @ RETURN
380 FORTHX 'ECH1 0 LIST ECH1 0 LVAL 0 ORDER' @ N=FORTH1
390 IF N=0 THEN DISP 'Tu passes.' @ WAIT 1 @ GOTO 350
400 C=FORTH1 @ FORTHX 'O SWAP LCOL',N @ DISP 'Conseil:'&CHR$(64+C DIV 16)&&CHR$(55-MOD(C,16))
410 GOTO 160
420 OFF @ GOTO 140
430 I=FORTHF
440 IF I>9 THEN DISP 'Archi nul!!!' @ GOTO 160
450 IF I>6 THEN DISP 'Mauvais!' @ GOTO 160
460 IF I>3 THEN DISP 'Pas mal!' @ GOTO 160
470 DISP 'Excellent!' @ GOTO 160
```

LE COIN DES LHEX

Au mois de juillet nous vous proposons cette nouvelle rubrique assembleur, tout content de pouvoir permettre à tous d'utiliser les nouvelles fonctions basic que nous avons créées. Nous nous étions accordé sans prendre garde l'ID 113 (#71 en hexa) pour Hexiser tous nos Chefs-d'oeuvre. Mais voilà ! Dès la mi-septembre, HP Corvallis nous attribuait le numéro 225 (#E1 en hexa).

Si vous êtes possesseur de ces anciennes versions, il vous suffira de changer l'ID 113 contre un ID 225 comme suit:

```
POKE DTH$(HTD(ADDR$("TOTOLEX"))+37),"1E"
```

Autrement vous trouverez dans les prochaines pages un récapitulatif de tous les Lex parus dans le journal depuis le mois de février 1985.

Heureuse programmation à tous
Michel MARTINET

PROGRAMME MAKELEX

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


DESAL ID#E1 718 octets

0123456789ABCDEF sm

000: 44543514C4020202 34B
001: 802E009371700158 6A3
002: 1A5001E10506E200 A03
003: FB30000000000000 D3E
004: 012100FF0019100F 09F
005: A10BD100F5200500 40A
006: 0F03025200FB1444 771
007: 2455644210714353 AB9
008: 4422071445844230 DFF
009: 7844514424072554 151
00A: 4442501FF4118F83 4D0
00B: DB0D60A871A51360 861
00C: 6137C2134135AD2D BE3
00D: 681E81ED7C6BF2BF FBB
00E: 2A0E109CF4811831 340
00F: 5A32172201C11495 69F
010: 7E071341CF1198D8 A33
011: 32F08D91FB080CF2 DDF
012: 08F670B15F18FAA2 18B
013: 513314648FC70B14 505
014: 4D309A0ABF1A88BF 8CD
015: 4BF4A4E59CAF4034 C8A
016: 118F13DB0D23068A 01C
017: 63A15B5175137062 383
018: 2769FD68F1C81158 724
019: 11C613324AC280FF AB6
01A: 8F84171071355901 E18
01B: C915991098508F06 197
01C: 4A18F7B1818FE83B 54E
01D: 18DC32F041113713 8BE
01E: 5068F83DB0AD1D88 C72
01F: 1D31E20A3302E7D7 002
020: CD4C114BDB8FC70B 3D8
021: 15700914D17153E0 73D
022: 71358DC32F04118F ACB
023: 83DB013606137135 E30
024: 1348F064A11CF137 1B7
025: E28B37413506D6C6 548
026: BF2BF230F155717F 8F5
027: AF2D681ED7A4ECF4 CD8
028: 3214ABCA8F841711 06D
029: 59316117353E8DD4 3F2
02A: 49007135071348DC 75D
02B: 32F04117B007D207 ACB
02C: 3004248F13DB0AD1 E56
02D: D881D3102CD48214 1DF
02E: B966001715FED4E4 580
02F: C4BF0BF0A0C1CF15 94B
030: 17010772EF8DC32F CE9
031: 010F304BF100F020 058
032: 0000000000000151 36F
033: 00FD00B2000D9055 6E4
034: 454B442F3705F4B4 A72
035: 54041FFF70001800 DE2
036: 08F681F070901F17 164
037: 8F2D41411618F871 4E8

038: F08F83DB025A8020 87F
039: 81C1B178F2D6D714 C20
03A: 21311BE95F214213 F90
03B: 0CF4B1181AC214A7 32D
03C: E70D4159017054E8 6AE
03D: D84A808D394508FA A57
03E: B630JFEA2304908D DF4
03F: B2E208DD97308F13 194
040: DB0D230A886F2D68 54E
041: 0D00D442133C2131 8B1
042: AF01531135D1207F C2F
043: 009680056F8D91FB FC5
044: 08F670B1401BF1A8 35A
045: 8BF4BF4018FAA251 713
046: 3314648FC70B14CC AAD
047: 3173B6A53D84228F E3E
048: DF8E017F1361BB98 1F3
049: F21448F322B15D9D 592
04A: 6C64691007F4F118 916
04B: D7DAC9109D9108C4 CC9
04C: D68FA90F0110130C 05A
04D: F4D1AC215A08F841 401
04E: 7114917116063EF1 76C
04F: 19135208FE83B11B AF5
050: B98F21461348DC32 E87
051: F025101086000F71 1E2
052: 00000000000000E1 508
053: 000FFB4549575149 88B
054: 44542101FF001361 BE7
055: 081371098F2C6004 F51
056: 511191351181348D 2A9
057: 8ACA18F127006BDF 661
058: 25C0C000000F7100 9BC
059: 00000000000007100 CD4
05A: 0F725546542C01FF 062
05B: 4118FE83B18DC32F 413
05C: 0 444

REPLEX ID#E1 320 octets

0123456789ABCDEF sm

000: 255405C454850202 356
001: 802E001471700158 6A7
002: 582001E606000000 9E8
003: F710000000000000 D16
004: 0D5000FF255405C4 097
005: 14345442601FF8A9 417
006: 001811C114B11892 772
007: E70962A014E96600 AEF
008: CDCD58D11C8DB14B EC8
009: 18D91FB04443480 246
00A: DF884468F83DB08A 606
00B: 88534200008B64D8 974
00C: 0C214F1081715541 CD5
00D: 1B8AEB08A8D06110 075
00E: 8A8B0017C9F1351C 40F
00F: F076AA11371F495F 7AF
010: 2143135010C80C21 B04
011: 088F83DB01371358 E87

012: F534B110C137C213 204
013: 58F83DB0137C2135 591
014: 1038F83DB0718F13 91E
015: 710AC2D7D2109118 CA1
016: ABA136BF2BF2BF2A 07E
017: B61081BB98F2DB14 428
018: 411311AD5DBE98B6 7D1
019: 65DB13411A1351CF B60
01A: 11BD574BE8ADA375 F1C
01B: DE1351CD1471C113 2AB
01C: 7111E2137DA78BEC 655
01D: 2136D87D1F704011 9CC
01E: BE35E27D0FDB1341 D82
01F: 19D57F8EE91351CF 140
020: 3420000D57110136 485
021: D711A8A3E3666F13 823
022: 7E98B692C91358A9 BCA
023: 001811C114E14D11 F2E
024: 9E6E6109CDCD50E8 2F7
025: DD44901FB98F2147 69E
026: 135702EDA3401000 A02
027: EA1198F601B11CF1 DA6
028: 18BF68F68F613411 153
029: 9BF2BF230F15D68D 515
02A: C32F0 638

FILELEX ID#E1 64 octets

0123456789ABCDEF sm

000: 649C454C4548502 376
001: 802E001471700158 6C7
002: 580001E707000000 A08
003: F710000000000000 D36
004: 091000F96494C454 0A6
005: F3701FF411136068 421
006: F59B9057097C908D 7C4
007: A9390137068F77F9 B56
008: 0AF0480B44BF4071 EE7
009: 351CF1517071348D 262
00A: C32F0 385

ATTNLEX ID#E1 68 octets

0123456789ABCDEF sm

000: 144545E4C4548502 36F
001: 802E002471700158 6C1
002: D80001E808000000 A13
003: F710000000000000 D41
004: 0E1000D7144545E4 0B2
005: 801FF05000F20001 415
006: 37061F144F214A31 787
007: 1E96240CE15D0071 B0C
008: 358D84A808FA2C20 EAB
009: 0E1101EC00008DB2 22B
00A: E20038DCF250F 524

DRIVELEX ID#E1 277 octets

0123456789ABCDEF sm

000: 4425946554C45485 36D
001: 802E002471700158 68F
002: E22001E90A000000 A18
003: F0200E3008600000 D66
004: 0BA000D110EC000D 0EB
005: D4494351424C4549 460
006: 0B54E41424C454A0 7E0
007: 1FF10200120402C4 B40
008: 568702C811054427 E9F
009: 96675627D20EAEFC 259
00A: F31D29614000D2A6 5DA
00B: E8A2400076100005 931
00C: 044D2C454851FF00 C89
00D: FF07135AC0B44821 04E
00E: 01E5100321007260 38F
00F: D2A6E8A2C434802E 739
010: 08A674D2A6E4A223 AD1
011: 1007F00076303480 E1E
012: 2E08A2E1D5D2A6E8 1D9
013: A691D915D38408F2 570
014: 12018D84A8031F34 8E6
015: 36048F67B90D3CF8 C91
016: F77F90484AC2B469 03F
017: C3551371088FD979 3CE
018: 0137135C2D7351E9 749
019: 0A02515B59122317 AA7
01A: A1378BF0513758E3 E38
01B: 1938D393033101E 1A6
01C: 8DA939031C347E17 53A
01D: 51432034E3000EA1 89F
01E: 33D61537970FC137 C23
01F: 174679F201181351 F81
020: 7FD015B3018FC9E3 330
021: 050287722314C8F8 6A2
022: EC20AEA80DF8D624 A6D
023: 508D20F208D27130 DE1
024: 8F957508D00350F 132

FKEYLEX ID#E1 81 octets

0123456789ABCDEF sm

000: 64B45495C4548502 376
001: 802E002471700158 6C8
002: 6A0001EB0B000000 A29
003: F710000000000000 D57
004: 0E1000D764B45495 0CF
005: B01FFA6000E50008 457
006: F29D80D910834C10 7E5
007: 00D534444F2DAE6E B8E
008: 6808F8F803101101 EF7
009: B444F21481801524 25F
00A: B44550A4C1504808 5D5
00B: 08D84A808DD97308 971
00C: D39450F B07

DESLEX ID#E1 281 octets

0123456789ABCDEF sm

000: 445435C454850202 35A
001: 802E003471700158 6AD
002: 732001EC0F000000 A08
003: F230000000000000 D33
004: 0CE000F31016000D 0A0
005: 4200B000FF20DF00 420
006: 0FF34F4E44525143 7AE
007: 545C0D94E4655425 B34
008: 3554D0794E46542E EBC
009: 09051494E445F01F 23B
00A: FF30001400034C50 59E
00B: 001F401E27C20310 901
00C: 61E002E7E1031C41 C7D
00D: E003E70108D84A80 003
00E: 8D3035003148BEC1 38F
00F: 49171CECE8AEDE01 75D
010: 4111371351088F83 AB9
011: DB0D68AA607BCF12 E7E
012: 81358DC32F000136 1F3
013: 1081BEF3E2D015A0 58F
014: 8FB13B11201188D9 918
015: 12F0888331361081 C73
016: 371091358FC8CB01 FFF
017: 0A8F322B11228F32 380
018: 2B112210317F8FC1 6FE
019: DB08F322B1123313 A79
01A: 89E65112231709E6 DEF
01B: 901226700608034D 143
01C: 50001B442E28B680 4B4
01D: 1A4A1E31D28B6801 844
01E: A401E136C2C21361 BBE
01F: 4A11AAE5AF230196 F5D
020: 9C0A66A6D64FFAE7 334
021: 0E62123AE811310A 6AC
022: AEB969A00E6E6A00 A64
023: BEE0E661481128FB E12
024: 13B1120119137118 15C
025: 8D912F0 2F1

ENDUPLEX ID#E1 252 octets

0123456789ABCDEF sm

000: 54E4445505C45485 376
001: 802E003471700158 6C9
002: DF1001E012100000 A1E
003: F920000002500000 D56
004: 0A8100FF00AC000D 0E4
005: C10E6100FB54E444 476
006: 55054201954E4445 7D4
007: 50511F3545142545 B2C
008: 550542211FF31BF9 EB6
009: 613131CF961C1531 22B
00A: 118D73200A8FFF81 5C9
00B: 1D0E4003200A8FAB 95E
00C: 8115DEDB10831FC8 D0A

00D: FC463159C31DC8FC 0CE
00E: 46314BB31DC8FAF5 489
00F: 311378D626208DD9 80D
010: 7308D394509FFFFD BCD
011: EFFF8F681F08FE44 FB2
012: A131D021840137FA 32D
013: 809FA8B354137155 6B1
014: 1208F7B181D23021 A21
015: 083200A10A8F13DB DA9
016: 01371288B6A111A8 11C
017: F14A118D84A808DD 4CB
018: 4490D8D2310C8BA9 869
019: 08DA114110111A8F BE0
01A: D791157D11013011 F3E
01B: 98F771B18D84A800 2D8
01C: 032808590003200A 621
01D: 1331018FAB811137 990
01E: 136D58F064A11111 D01
01F: 315E114A31D09622 071
020: 1AE68F4058116156 3F1
021: ED91348D7B181 6E7

CMD16LEX ID#E1 77 octets

0123456789ABCDEF sm

000: 34D4441363C45485 371
001: 802E004471700158 6C5
002: E90001E313100000 A11
003: F710000000000000 D3F
004: 002000D934D44413 099
005: 63311FF95000B500 408
006: 01F675F2143131AF 78D
007: 2A4E2380C315D517 B13
008: 5A4E55F1371F085F EB7
009: 20D1451740D57F1F 240
00A: 679F215548D84A80 5CE
00B: 8D303508D2BF30F 933

COMBARR ID#E1 168 octets

0123456789ABCDEF sm

000: 34F4D42414252502 364
001: 802E004471700158 6B8
002: 551001E415100000 9F4
003: F020000000000000 D1C
004: 0A2000F90073000F 07E
005: 514252541734F4D4 3EB
006: 24511FF882275508 761
007: F499C08D612F0882 AFD
008: 27F308F004D01118 E70
009: FFB6C08F757E0AF9 24D
00A: AF7AF6113AF81128 5F8
00B: FCA4C08F499C08D6 9C2
00C: 12F08FC8B017F7C D81
00D: 40110119720001BC 0D1
00E: E8F439C08F363C08 47C
00F: F757E08F004D0110 802
010: 8FFB6C08F757E08F BD5

011: C14D08FCA4C00110 F6C
 012: 01097D1011876101 2BB
 013: 10119268F674D040 626
 014: 06C2080DF89922AF 9C8
 015: A20322009B6318FF D4B
 016: B6C08F4F6C04008D 0F7
 017: 91FB0 21E

HMSLEX ID#E1 279 octets

0123456789ABCDEF sm

000: 84D435C454850202 36D
 001: 802E005471700158 6C2
 002: 332001E619100000 A01
 003: F230000000000000 D2C
 004: 0E4000FB00E6000F 0AE
 005: 61058000FF10FB00 427
 006: 0F784D435B261784 7AE
 007: D435D271584D4358 B35
 008: 138425911FF88228 EAF
 009: 507880799170807B 21D
 00A: 618F363C074E0840 59D
 00B: 6E2088228507F507 914
 00C: 071775072418FA91 C86
 00D: E056D81184074A07 003
 00E: B117EE0AF22E306A 382
 00F: F7CE8FA34C08FC14 786
 010: D08F363C077B08FC B38
 011: A4C06A4081184071 EA9
 012: 607E907BA0AF22D3 250
 013: 163AF7E68FCA4C08 614
 014: FE04D074B08FCA4C 9DD
 015: 08FC14D08F363C08 D7C
 016: F5F3D08F363C0870 120
 017: 008218F499C08D83 4A9
 018: 2F08FDF8E08FFB6C 895
 019: 08F9D3D08F4F6C08 C50
 01A: F5F3D08F9D3D0948 00F
 01B: 80AC0550A4C8F363 3A7
 01C: C003AF22E301AF7E 754
 01D: 6E6018FA34C08F00 AED
 01E: 4D08F4F6C08FC14D EBO
 01F: 08F004D071BF01AF 24E
 020: 22E306AF7E601137 5D5
 021: 1081FBA8F21577AF 983
 022: 71CF157712813511 CEE
 023: 8011371FB98F2151 06B
 024: 717FAF4151713517 3EA
 025: F01 494

XERRLEX ID#E1 90 octets

0123456789ABCDEF sm

000: 85542525C4548502 363
 001: 802E005471700158 688
 002: 8B0001EA1A100000 A1C
 003: F710000000000000 D4A
 004: 071000F785542525 0A3

005: A11FF8118FBC631D 45D
 006: 68FC2530346E7F21 7F0
 007: 37AF0AF2148328E3 B96
 008: D78F943B1D21C114 F32
 009: FCAD88F74030DAAC 30D
 00A: 08A0E18B7B034863 697
 00B: 00CAAF23108CA8A4 A4
 00C: 50B44AF2AC6BF617 DF9
 00D: F8D612F0F 019

STKLEX ID#E1 107 octets

0123456789ABCDEF sm

000: 3545B4C454850202 368
 001: 802E006471700158 68E
 002: BD0001EB1B100000 A30
 003: F710000000000000 D5E
 004: 002000D935451434 0AA
 005: B4B11FFD90001900 437
 006: 0048F681F08FBC63 7DA
 007: 1440D08AC40E4D23 B69
 008: 1018BA40DA1001F6 EED
 009: 75F2147D7D23068F 285
 00A: 943B1DBC21742217 60B
 00B: 41450D57FDB13511 990
 00C: 0CCD823AF230315D D30
 00D: 5175CC56F1F679F2 0DE
 00E: D415908D84A808DE 487
 00F: 6A208D39450 6EC

REPEALEX ID#E1 266 octets

0123456789ABCDEF sm

000: 2554055414C45485 360
 001: 802E006471700158 686
 002: 912001EC1C100000 A10
 003: F710000000200000 D40
 004: 03B100DB25540554 0AA
 005: 1445C11FF31C1961 431
 006: 4000248FDD410716 7A4
 007: 04828F08B108F898 B32
 008: 108F851318F2C600 EAC
 009: 4C18FEE010D91082 245
 00A: 48F8A41082103731 5AE
 00B: 08FD7521863EC313 93F
 00C: 65AD8F08B108D127 CDE
 00D: 00315C8FC4631350 051
 00E: 00007492108AFOAF 3CE
 00F: 21B989F21560B8EA 771
 010: 0C8FBBCE0118AE67 B3B
 011: 9B0315C8FAF53128 EDA
 012: 8F7FC001F064F2D0 285
 013: 14BCCD231E08FE6C 64D
 014: E02034F64F2E9135 9DA
 015: 15723030E06A0C49 D4B
 016: 0A2666FFA2647131 0D5
 017: 5C8F10631AF2B266 46F
 018: C401BD3E21F344F 825

019: 215203079020278D B79
 01A: F1B064F2153292CF F13
 01B: B14AD1AE80215329 2A4
 01C: 2CCA7CFE655F188F 689
 01D: 3E22515C515A5912 A00
 01E: 6015C52003140003 D41
 01F: 40008F681F08FBC6 0E1
 020: 315C0D2A0E8BA908 485
 021: D029E08C1B989F2 83B
 022: 15008D84A808D394 BBE
 023: 508DE6A20 DC6

MARGELEX ID#E1 92 octets

0123456789ABCDEF sm

000: D414257454C45485 375
 001: 802E007471700158 6CC
 002: DB0001ED1D100000 A42
 003: F710000000200000 D72
 004: 085000DBD4142574 0E4
 005: 94E4D11FF31C1961 484
 006: 4000DB10B1F789F2 818
 007: 14B1EE74F14F9669 BC4
 008: 08FA5CE011BD7007 F66
 009: 4000B30008F871F0 2D3
 00A: 8FBC6315708AC40D 680
 00B: 031069EA40DA1F78 A18
 00C: 9F2CC1498D84A808 DC6
 00D: DE6A208D39450 0B6

PRIMLEX ID#E1 148 octets

0123456789ABCDEF sm

000: 052594D4C4548502 370
 001: 802E007471700158 6C7
 002: C21001EE1E100000 A2F
 003: F710000000000000 D5D
 004: 071000F7052594D4 0C3
 005: E11FF8118FCD801 479
 006: 00AF210922719021 7D4
 007: 7B80227580227F70 B45
 008: 1118F322B1AF2D6A EE4
 009: FA048FBBCE08F4BC 2D0
 00A: E08F223B11189722 64B
 00B: 4248F674D04A3247 9CD
 00C: C302276302470302 D19
 00D: 27A20247420267E1 07D
 00E: 0227810267210629 3C4
 00F: F60501188D612F0A 74B
 010: F280C010A1118F32 AC6
 011: 2B1D6AF0DA11ACAB E90
 012: FB13B11011101198 1F6
 013: F8A4C08F4F6C0500 5A2
 014: 071198D612F0F 882

ONKEYLEX ID#E1 195 octets

0123456789ABCDEF sm

000: F4E4B45495C45485 39D
001: 802E008471700158 6F5
002: B81001EF10200000 A4F
003: F020000000000000 D77
004: 0A3000F3108C000F 0E6
005: F3455253544F75E 46F
006: 4F1F34552535B454 7EF
007: 9535021FF00A82B0 B71
008: 6808F800802F0C4C EFF
009: 4AF38AC606D40D82 2AB
00A: 77E505802E693028 623
00B: 7F405802E67202A7 9A1
00C: 0405802E65102B71 D01
00D: 305412E6300B07B0 069
00E: 7B07B0720A82A0E8 3FD
00F: 008080AFB1328D91 782
010: 2F0D4C40C5BFC401 B31
011: 0013610A1371088F E8F
012: D4D008FEE010AF34 242
013: C2AE48F3E320F294 5F1
014: 003A4023B4033C40 951
015: 2AD403AE4000AFB1 CF3
016: 101311128D912F0B 05D
017: 07B07B07B07B07B0 3EA
018: 72068DF2E64FF2E6 7A5
019: BEF2E62EF2E69DF2 B8E
01A: E60DF2E67CF E3B

MENULEX ID#E1 318 octets

0123456789ABCDEF sm

000: D454E455C4548502 383
001: 802E008471700158 6DB
002: 182001E121200000 A12
003: F710000000000000 D40
004: 0F1100F7D454E455 0C9
005: 121FF8FBC6315C0D 482
006: 2A6E8BA008D029E0 827
007: 1B765F2146D71A29 BB7
008: 6F1468AEB01A265F F63
009: 14613414A31C2962 2C6
00A: A218390C50163316 62A
00B: C8FB6A804117F803 9CA
00C: 1028D39390161136 D23
00D: 13410A31C2AE531D 0A9
00E: 014A960A01619662 40E
00F: F118A6E10896E991 7AA
010: 011FB98F21B045F2 B41
011: 7E108FFD15111111 EC2
012: A1358F327908D572 249
013: 511577154717F16F 5BF
014: 271571154120018F 912
015: E92208DEE0108812 C9E
016: 71FE17F103D0E4A4 045
017: EA4E4C07ADE17F12 40D

018: 311BF2F2AE61328F 7B4
019: B1DE01338FB1DE01 B62
01A: 0410810B740002FF ECB
01B: 071358FE0C101BB9 260
01C: 8F21F045F2746F8F 60D
01D: 88B70779E787F8FE 9D1
01E: 21208F3E320B2990 D47
01F: 33A30237102AD103 0A9
020: AC3062640005DC11 420
021: BA6E96EB011BAE2B 7F2
022: 6610810B65AF11BD B84
023: AF4F4B669E67E11B F4B
024: DAF4F4AE658D7EFE 348
025: 8FE9220D011BAEA8 6FF
026: FB13B104AF61148F AA3
027: A0DE01318FA0DE08 E52
028: D912F01B976F2142 1E0
029: 1308F7AA808D7E47 586
02A: 0 587

FORMALEX ID#E1 454 octets

0123456789ABCDEF sm

000: 64F425D414C45485 385
001: 802E009471700158 6DE
002: 093001E226200000 A1C
003: FB30000000000000 D57
004: 066100F110EA200F 0CA
005: 020EC100F130E110 436
006: 0F2407D000FD3454 7B7
007: E44554254222B345 B21
008: 43555255432D64F4 E94
009: 25D414454242D255 1FD
00A: 4445534544252B35 55A
00B: 0514345442621FF0 8BF
00C: 48F83DB01368F534 C56
00D: B1137C2109134135 FB4
00E: D38508418423102A 31A
00F: E5CC4E2851CC1811 6C4
010: 4E965F08708EE785 A6C
011: 05808408521C114D DD2
012: 51D86011861C0862 13E
013: 70171CF018118FBC 4D5
014: 631440D017F13713 838
015: 51098F064A173004 B9F
016: B231025908F40581 F07
017: CC56F01411714F11 289
018: 98FB14B11368408F 61E
019: 064A18D7B1818D91 9AF
01A: FB0842274508F13D D3D
01B: B0137135C2109F0 0B0
01C: 64A1118C642D8A88 442
01D: 18BE31E2AD0DA81C 803
01E: 81C7B7F8508F7B18 BB2
01F: 18FE83B18DC32F08 F66
020: FBC6315598A80917 300
021: F100018DD4490842 677
022: 27ADF769E119110E A18
023: 244EE24FD10ADA13 DC7

024: 416F41D1361B495F 156
025: 21328BEFB119AF0D 50B
026: A137135EA81C1188 894
027: B6348A2A3E48AB33 C3C
028: EAF8DB10B8FE6CE0 02A
029: 20108D8E57320110 389
02A: 11BE2CD751011A13 70F
02B: 56CCE31528D39390 AA4
02C: D73102662014B171 E03
02D: 1481619661FD4580 179
02E: 14C161CC57FCF8AF 549
02F: AD018422730FD88F 8E9
030: 83DB0D6AD0DA81C1 CAB
031: 37135C2C2D78B497 041
032: E9E91353F02F312E 3E2
033: 2C2A3B3920A14B96 774
034: 261850CD4801715C AEE
035: E1104241108A0D12 E51
036: 71C114BBF6BF6962 201
037: 710D52F4B18709CC 59F
038: C10051C1188A160E 91B
039: 5E5D4CCDB1351CF8 CE9
03A: FB13B1AF68D832F0 0A4
03B: F 0EB

PRINTLEX ID#E1 565 octets

0123456789ABCDEF sm

000: 052594E445C45485 378
001: 802E000571700158 6C9
002: E64001E721300000 A1B
003: F170000000000000 D49
004: 08C100D800D100D 0C4
005: 6106E100DD10DF10 44B
006: 0F82042200DF2003 7B2
007: 200D63034200D140 B03
008: B8200DC4093300D3 E80
009: 50CA300D660EC300 207
00A: D72454C4C472724F 59B
00B: 4C44482334259275 900
00C: 4353442A236464B2 C67
00D: 3C464C27D4F44454 FFA
00E: D2705542564E2305 365
00F: C4F2F55E4445425C 708
010: 494E403775251405 A69
011: 131FF84466008542 DD9
012: 0048F73321F0831D 151
013: 031048D393900713 4A8
014: 5E6D7700007D534D 835
015: 1000C906D606D020 B9E
016: 153101874008D84A F05
017: 8014A310E9620003 260
018: 8D30350335402181 5B2
019: 8FDFC201488DCF25 982
01A: 08F8BF303154966F D1B
01B: 11718FA2C200E1CF 0BA
01C: 1ECBF008D530308D 45E
01D: B2E208FE7A204B17 804
01E: 5108D6CA208DE6A2 BAC

01F: 08D074508D271301 F12
020: 8FFFA7FFF780F170 2F3
021: A8FFF9FFF745F49 6F2
022: 070FE1F079EE1E01 A9C
023: 5FFFA4FFF74006D2 E71
024: F1F659F2AE214D7A 233
025: CE1D001001371091 597
026: 351C12031B114D8F 914
027: 064A18408D7B1811 C8C
028: 0FFFAFEFF748F748 08C
029: E1C0CEEFF5EEFF7F 4A6
02A: 9F7F6E1A08DC2B90 86B
02B: 14FFF53FFF7460AF C44
02C: 2A96976D7043103A FC3
02D: 6A8FE1DE08F12DE0 391
02E: 3A5B162B60035A72 70F
02F: 75BF15CA13606671 A9B
030: E99EFFEAEFF736ED EB3
031: 031C4540E47D206C 236
032: 4E048F871F08F2EA 5ED
033: 21044F0AF2328219 960
034: F2008D91FB0AE7D8 D1F
035: 755F367B162C615C 0B3
036: 6166815815310322 3FD
037: D4D1A6A148161811 781
038: 0D5DEAE14C16131 B2F
039: 1014C18E700007DA EA9
03A: 349BEFFC20613606 247
03B: 686D6F8F93EFF81E 62E
03C: FF14A8F504505D17 9CD
03D: 24DBB162C6732307 D53
03E: 63636613C4743F17 0C8
03F: F1331011BE95F214 44A
040: 68BE121331647A3 7C2
041: F1111318FDF8E079 B65
042: 0F13313331057B1F ED2
043: 6A3D35DFF86DFF70 2AD
044: 3D4F07CCC4B16246 657
045: 047FBC4B1624644B 9F0
046: 3DFF05DFF750D411 DAE
047: 71AC5B1623713347 126
048: 29C5B162370334F 46D

INDATLEX ID#E1 147 octets

0123456789ABCDEF sm

000: 94E4441445C45485 379
001: 802E001571700158 6CB
002: A21001E232300000 A0F
003: F710000000000000 D3D
004: 0C1000FD94E44414 0BE
005: 455442231FF00137 423
006: 10913610A7040B1C 78A
007: 34414455402A3022 AD6
008: 2B1E3A4A4B1C3F2B E90
009: 1E3D4D4B1C3F2B1E 252
00A: 31414B1C322FF071 5D0
00B: 358FE0C108FFD151 97E
00C: 8F99C418FA3A8111 D24

00D: A1361191358508FB 0A0
00E: D58117F17115FB17 42F
00F: B1331011331CF812 796
010: 81231F28F234B1DA B2C
011: 8F834B129B92B92B ED3
012: 92B92B92B92D6243 268
013: 1F22015578F064A1 5DF
014: 8408D7B181F 85B

KBDLEX ID#E1 309 octets

0123456789ABCDEF sm

000: B42444C454850202 365
001: 802E001571700158 687
002: E62001E333300000 A06
003: F71000000B800000 D4E
004: 0C1000D5B4244433 0B5
005: 1FFA6000E5000048 42F
006: F871F08FBC6315A0 7DE
007: 3048B240D0814A44 B4F
008: 317F948C08FAF531 EF6
009: 5908F106311F9E6F 28E
00A: 215141EA89F15148 60C
00B: D84A808DE6A208D3 9BE
00C: 945031B19614000D D1F
00D: BD5317F8FC4631D9 0D7
00E: D75AE1B9E6F21521 485
00F: 1AA89F15619160D3 813
010: 060E1291A4C3090E B94
011: 1291E8B1108FA0DE F3A
012: 015248F3E320E2B0 2BA
013: 123511331216209F 608
014: F2C21A8831003048 97A
015: 160E4294A4370000 CD8
016: 7D53432000C17470 03A
017: B0D091B172925363 3B2
018: FF30481694254700 723
019: 007D534F3000C172 A90
01A: 4010A0F071D132A2 E01
01B: A213137373A6A617 176
01C: B7F788D8398989FF 542
01D: 1F769F23200415D2 8BC
01E: 60500713614E9E22 C24
01F: 116114E1619E6CE0 FAB
020: 0D9060107134D21F 31B
021: 769F2156114D1713 68B
022: 021551170136E614 9DB
023: 5821850017ACF5B5 D66
024: 2514A5D57B8F6B51 110
025: 4944454D57AAF8B5 4B9
026: 54252554552D5AF 83A
027: 2307253176108003 B7A
028: 048169425F654FF ED1

DATELEX ID#E1 427 octets

0123456789ABCDEF sm

000: 44144554C4548502 35E

001: 802E002571700158 6B1
002: B53001E438300000 A03
003: FB30000000000000 D3E
004: 0E6000FD0020300D 0AC
005: 61015000F1204900 3EF
006: 0FA209F200D94444 776
007: 14953543544D4955 AE0
008: 3744F4754263544F E5A
009: 475735D44495831F 1D9
00A: F811852684088227 54F
00B: 34117F1087931118 8AC
00C: 9F23187A8FB1381B C64
00D: CC6010B72AFA8FB1 025
00E: 3B1AF68D832F0811 3C1
00F: 8427FF0CC20D2307 75A
010: 8FE6CE087221816A AFF
011: D2AA281660CF17F1 EBA
012: 331011338F624202 207
013: 9021012003054045 53B
014: 0960399444E455C4 8B5
015: 2969803994442514 C14
016: D42967703F944454 F92
017: 25342554D42F6850 303
018: 3994445554A4296D 681
019: 403F9444542544E4 9F4
01A: 54652F6E203B9444 D77
01B: 54D414352B1CB612 0F8
01C: 03F548434E414D49 47C
01D: 4442F1CF66001C91 804
01E: 5518F064A1840208 B6F
01F: D7B1818FC1DB0AF8 F3D
020: 2031BE048408FC46 2C5
021: 31550850AF990A50 63F
022: 812812812AF721A9 9BB
023: 2B73812812AF5A92 D47
024: B718F834B1AFA23A OFF
025: 92B7A86080DDDFDD 4D4
026: D220332851886808 84C
027: A25685031219E15 BC2
028: F31709E1A1302961 F39
029: 33A0E0E05213036B 2B0
02A: 10101301A8A0E05B 61D
02B: 0E111213039E3316 979
02C: B7031929EBA26E60 D15
02D: 8D91FB03301009E1 099
02E: 2C9658E31519E72E 43F
02F: 61BF9E39DAE91091 7F6
030: 02D2304058FE6CE0 8BB
031: 8ADD811296C71320 F21
032: 048FE6CE08A96061 2C5
033: AF119AE51128D403 659
034: 31030002300031BE 9A0
035: 6310E10000200031 CD6
036: BE218F3F5318D84A 089
037: 808D3035003 2D6

KBEEP ID#E1 38 octets

0123456789ABCDEF sm

00D: 182184001

FBC

0123456789ABCDEF sm

000: 345525C454850202 359

CALCLEX ID#E1 41 octets

001: 802E003571700158 6AD

0123456789ABCDEF sm

000: B424545405020202 349

002: 9B00017000000000 9E0

000: 3414C434C4548502 369

001: 802E003571700158 69D

003: FE0000000800001F D3A

001: 802E003571700158 6BD

002: 150001E000000000 9C9

004: F31B196140001103 090

002: 750001E000000000 9EF

003: FE0000000800001F D23

005: 1B09664FDBD5317F 442

003: FE0000000800001F D49

004: F31B19614000DB10 09B

006: 8FC4631D9D75ED1F 80E

004: F31B196140001B4E 0C6

005: B0420348E300D732 40A

007: 7B8F2D231C28FFA0 BCB

005: 6F2142C451F110AF 457

006: 0208F04BE011BD70 794

008: B11FE74F2AF214F8 F89

006: 23088A63EA6E2531 7E2

007: 0 7C5

009: F4BCE0B64041F7B8 345

007: 7610800 94F

CURLEX ID#71 90 octets

00A: F23103814908E0A6 6BE

00B: A149171AE0810A6A A4D

00C: 1491F7B8F28F7415 DEA

NOUS EN AVONS

Sont en vente au Club:

- Cartes magnétiques pour HP41 au prix de 150FF + 3,10FF de port les 50 cartes;
- Cartes magnétiques pour HP71B au prix de 91FF + 3,10FF de port les 10 cartes;
- Cassettes digitales pour HP82161A: 100 FF pièce, plus 3,10 FF de port par cassette.
- Eproms (2716 & 2732) vierges au prix de 60FF + 3,10FF de port pièce. Pour les Eproms programmées, nous consulter;
- Module TIME (82182A) au prix de 400FF + 6,50FF de port. Livré avec manuel en français;
- Module QUADRAM (82170A) au prix de 400FF + 6,50FF de port;
- Module HPIL (82160A) au prix de 970FF + 6,50FF de port. Livré avec manuel en français;
- Pour les commandes des ZENROM, CCDROM et ROM PANAME nous envoyer 500FF minimum d'arrhes;
- VASM (listing en anglais des 3 premières pages de la 41) au prix de 170FF + 13,50FF de port;
- Le manuel de service de la 41 (en anglais) au prix de 50FF + 6,50FF de port;
- Le manuel technique du convertisseur (en anglais) au prix de 20FF + 3,20FF de port;
- HPIL Interface Specifications (en anglais) au prix de 80 FF + 9,50 FF de port
- The HPIL Integrated Circuit User's manual (en anglais): 40 FF + 6,50 FF de port
- Interface Kit Technical Guide (en anglais): 15 FF + 3,50 FF de port
- Manuel technique du Minitel R (Ref. 250 Alcatel): 40 FF + 6,50 FF de port
- Les anciens numéros au prix de:
 - No 1 à 3 au prix de 15FF + 3,20FF de port par numéro,
 - No 4 à 10 au prix de 20FF + 3,20FF de port par numéro,
 - No 11 à 17 au prix de 25FF + 6,50FF de port par numéro,
 - à partir du No 18 au prix de 35FF + 6,50FF de port par numéro.

Les ports sont les tarifs PTT actuels en non urgent.

ACHATS GROUPES

Nous assurons des commandes groupées pour ceux qui désirent profiter de notre réduction "spécial Club", soit 25% par rapport au prix HP. Pour ceux qui le désirent, nous assurons l'envoi postal en urgent ou en non urgent. Ces achats concernent aussi bien des modules que des calculatrices et autres.

NOTA: ces deux rubriques "nous en avons" et "achats groupés", sont des services "Club" que nous rendons exclusivement aux membres effectivement inscrits au club et possédant leur carte de membre PPC-PARIS.

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 ____/____/____

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 ____/____/____ 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 ____/____/____

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)

TRANSLATER PROGRAM (ADDENDUM JPC28)

- dessassembleur HP-41 version 4.1

le 12.10.85

J.Baudier PPC-PC 192

```
40 DESTROY ALL @ INTEGER R,L,C @ R=4474 @ DIM N$(8),R$(MEM>=R)*R+(MEM<R)*MEM/2-50]
50 DISP "41 translator program" @ WAIT 2
60 ON ERROR GOTO 'ERR1' @ 'INP': INPUT 'Name ? : ',N$;N$ @ SFLAG -1 @ PURGE N$ @ CFLAG -1
```

```
=====
70 'PROG': CONTROL OFF @ RESTORE IO @ DISP 'Ready ...'
80 IF NOT LEN(KEY$) THEN 80
90 DISP 'Reading loop ...' @ ENTER :LOOP ;R$ @ L=LEN(R$) @ DIM P$(L) @ P$=R$ @ DESTROY R$
100 C=HTD(P$(L-1)) @ CALL CHECKSUM(P$,C,R) @ IF NOT R THEN DISP 'Checksum error' @ GOTO 'ERR2'
110 DISP 'Prog 41 :';HTD(P$(1,4));'bytes'
120 P$=P$(5,L-2) @ CALL TEXT41(N$,P$) @ BEEP 400 @ BEEP 800
130 DISP 'File is now translated' @ WAIT 2 @ DISP 'into : ';N$;' (text)' @ WAIT 2
140 DESTROY ALL @ PUT '#43' @ END
```

```
=====
150 'ERR1': BEEP 100 @ DISP 'ERR: ';ERRM$ @ WAIT 2 @ GOTO 'INP'
```

```
=====
160 'ERR2': BEEP 100 @ DISP 'Retry ...' @ WAIT 2 @ GOTO 'PROG'
```

```
=====
170 SUB CHECKSUM(P$,C,R)
180 DISP 'Checksum test ...' @ FOR I=1 TO LEN(P$)-2 STEP 2 @ R=R+HTD(P$(I,I+1)) @ NEXT I
190 R=MOD(R,256)=C
200 END SUB
```

```
=====
210 SUB TEXT41(A$,P$)
220 DIM C$(24),F$(29),H$(2),I$(1),J$(1),M$(26),X$(2),Y$(2) @ INTEGER P,R,X,Y,T @ CREATE TEXT AS
230 M$='ABCDEFGHIJKTZYL'&CHR$(91)&CHR$(92)&CHR$(93)&CHR$(94)&'Q'&CHR$(127)&'abcde'
240 ASSIGN #1 TO A$ @ P=1
250 ON ERROR GOTO 270 @ C$=ADDR$(TIME41)
260 PRINT #1;' ' &TIME$(1,2)&' ' &TIME$(4,5)&' ' &DATE$(5,6)&' ' &DATE$(3,4) @ GOTO 280
270 P=1 @ PRINT #1;' '
```

```
=====
280 'DESAS': H$=P$(1,2) @ L=LEN(P$) @ I$=H$(1,1) @ J$=H$(2)
290 IF I$='F' THEN 'TEXT'
300 P$=P$(3) @ IF L=6 THEN 'END'
310 IF H$='00' OR H$='D2' THEN 'DESAS'
320 IF I$='1' AND J$='D' THEN 'NBRE'
330 IF I$='9' OR I$='A' AND J$='7' AND J$='E' OR H$='CE' THEN 'OCT2'
340 IF I$='C' AND J$='E' THEN 'LBLA'
350 IF H$='CF' THEN 'LBL'
360 IF I$='A' AND J$='B' THEN 'XROM'
370 IF H$='1D' THEN 'GTOA'
380 IF H$='1E' THEN 'XEQA'
390 IF H$='AE' THEN 'GTOXEQID'
400 IF H$='D0' THEN 'GTO'
410 IF I$='E' THEN 'XEQ'
420 CALL DESAS41(H$,F$)
```

```
=====
430 'PRINT': R=F$(1,3)#'LBL' @ F$=' '[1+(P>99)]&'0'[1+(P>9)]&STR$(P)&(CHR$(0)&' ')[R+1][1,1]&F$
```

```

440 PRINT #1;FS @ DISP FS @ P=P+1 @ IF LEN(PS) THEN 'DESAS' ELSE END

=====
450 'TEXT': CALL ALPHA(PS,FS) @ GOTO 'PRINT'

=====
460 'NBRE': FS=''

=====
470 'BOUCLE':
480 IF JS<'A' THEN FS=FS&STR$(HTD(JS)) ELSE FS=FS&' E-'[HTD(JS)+(JS>'B')-9][1,1+(JS='B')]
490 IF PS[1,1]='1' AND PS[2,2]<'D' THEN HS=PS[1,2] @ PS=PS[3] @ JS=HS[2] @ GOTO 'BOUCLE'
500 IF PS[1,2]='00' THEN PS=PS[3]
510 GOTO 'PRINT'

=====
520 'LBLA': PS=PS[3] @ CALL ALPHA(PS,CS) @ IF NUM(CS[2])<32 THEN CS=CS[1,1]&CS[3]
530 FS='LBL ' & CS @ GOTO 'PRINT'

=====
540 'LBL': X=HTD(PS[1,2]) @ FS='LBL '
550 IF X>100 THEN FS=FS&CHR$(X-37+11*(X>112)) ELSE FS=FS&STR$(X)
560 PS=PS[3] @ GOTO 'PRINT'

=====
570 'END': FS='END' @ DESTROY PS @ GOTO 'PRINT'

=====
580 'OCT2': CALL DESAS41(HS,FS) @ R=HTD(PS[1,2]) @ PS=PS[3] @ T=IS='9' AND JS>'B'
590 IF R<102 THEN FS=FS&' 0'[1,1+(R<10 AND NOT T)]&STR$(R)[1+(R>99)+T*(R>10)] @ GOTO 'PRINT'
600 IF R>127 AND R<230 THEN FS=FS&' IND 0'[1,5+(R<138)]&STR$(R-128)[1+(R>227)] @ GOTO 'PRINT'
610 FS=FS&' IND'[1,4*(R>127)]&' &MS[R-101-(R>127)*128][1,1] @ GOTO 'PRINT'

=====
620 'GTOA': SFLAG 0

=====
630 'XEQA': CALL ALPHA(PS,CS) @ FS='XEQ GTO '[5*FLAG(0)][1,4]&CS @ CFLAG 0 @ GOTO 'PRINT'

=====
640 'GTOXEQID': R=HTD(PS[1,2]) @ PS=PS[3]
650 IF R<102 THEN FS='GTO IND 0'[1,8+(R<10)]&STR$(R)[1+(R>99)] @ GOTO 'PRINT'
660 IF R>127 AND R<230 THEN FS='XEQ IND 0'[1,8+(R<138)]&STR$(R-128)[1+(R>227)] @ GOTO 'PRINT'
670 FS='XEQGTO'[4*(R<128)][1,3]&' IND '&MS[R-101-(R>127)*128][1,1] @ GOTO 'PRINT'

=====
680 'XEQ': SFLAG 0

=====
690 'GTO': R=MOD(HTD(PS[3,4]),128) @ PS=PS[5] @ FS='GTO OXEQ 0'[6*FLAG(0)][1,4+(R<10)]
700 IF R<100 THEN FS=FS&STR$(R) ELSE FS=FS&CHR$(R-37+11*(R>112))
710 CFLAG 0 @ GOTO 'PRINT'

=====
720 'XROM': R=HTD(JS&PS[1,2]) @ X=R DIV 64 @ Y=MOD(R,64)
730 XS='0'[1+(X>9)]&STR$(X) @ YS='0'[1+(Y>9)]&STR$(Y)
740 FS='XROM ' & XS&' ' & YS @ CALL SXROM(FS) @ PS=PS[3] @ GOTO 'PRINT'
750 END SUB

=====

```



```

760 SUB ALPHA(P$,A$)
770 A=HTD(P$[2,2]) @ A$=' ' @ P$=P$[3,LEN(P$)]
780 FOR P=1 TO A @ R=HTD(P$[1,2]) @ IF R#13 THEN A$=A$&CHR$(R) ELSE A$=A$&' '
790 P$=P$[3,LEN(P$)] @ NEXT P @ A$=A$&' '
800 END SUB

```

```

=====
810 SUB DESAS41(H$,F$)
820 ON ERROR GOTO 'ERR'
830 I$=ADDR$('FONCTS41') @ GOTO 850

```

```

=====
840 'ERR': BEEP 100 @ DISP 'File FONCTS41 needed' @ END ALL
850 ASSIGN #2 TO FONCTS41 @ READ #2,HTD(H$);F$ @ END
860 END SUB

```

```

=====
870 SUB SXROM(F$)
880 X=VAL(F$[6,7]) @ Y=VAL(F$[9,10]) @ N$=' '
890 ON ERROR GOTO 'NOTFOUND'
900 IF X=17 THEN N$='PLOT141'
910 IF X=18 THEN N$='PLOT241'
920 IF X=22 THEN N$='ILDEV141'
930 IF X=23 THEN N$='EXTIO41'
940 IF X=24 THEN N$='ILDEV241'
950 IF X=25 THEN N$='XFUNCT41'
960 IF X=26 THEN N$='TIME41'
970 IF X=27 THEN N$='WAND41'
980 IF X=28 THEN N$='HPIL241'
990 IF X=29 THEN N$='HPIL141'
1000 IF LEN(N$) THEN I$=ADDR$(N$) @ ASSIGN #3 TO N$ ELSE END
1010 READ #3,Y;F$

```

```

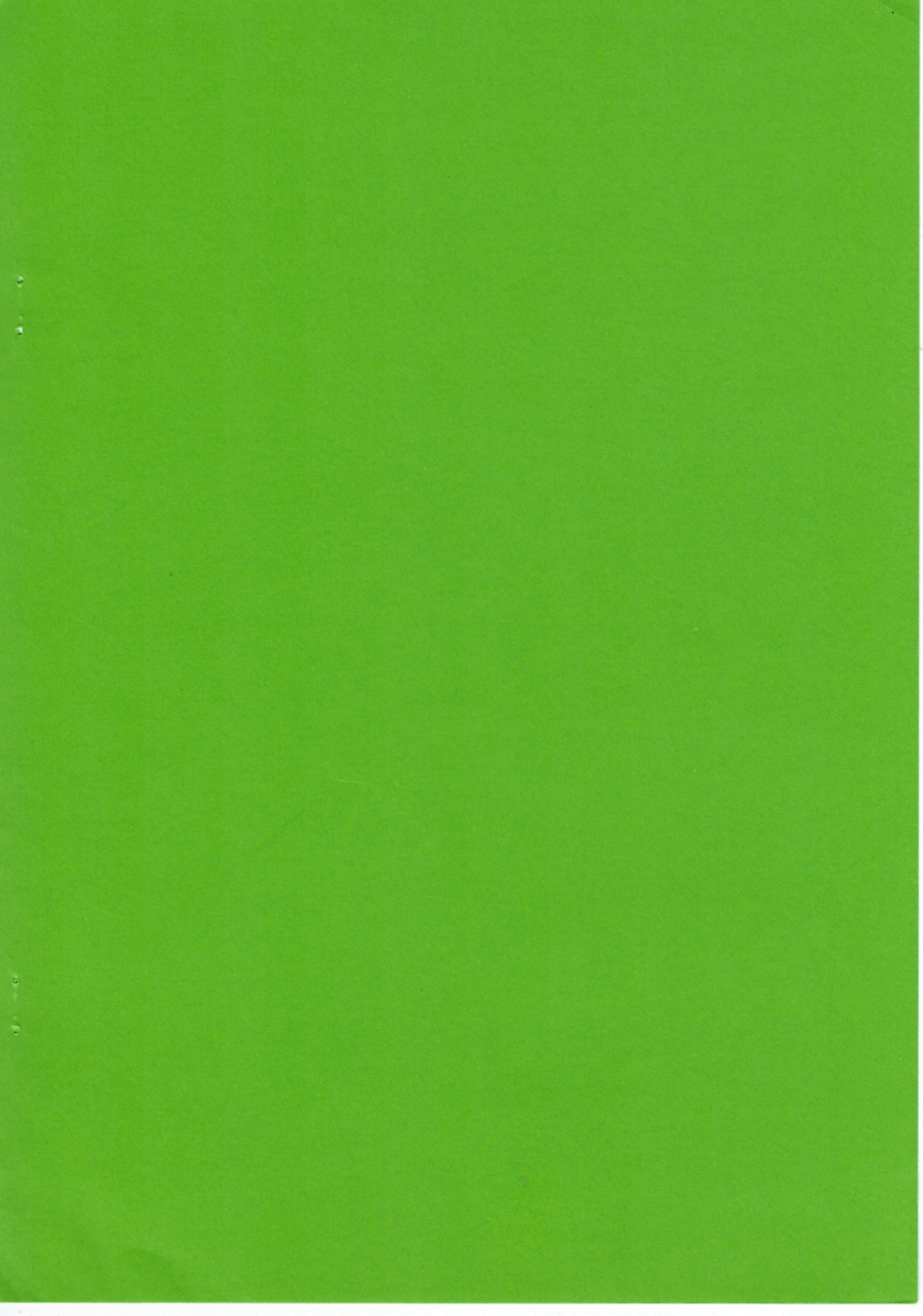
=====
1020 'NOTFOUND':
1030 END SUB

```

```

*****

```

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 du journal a été préparée par Laurent Istria, Michel Martinet, Jacques Baudier, Jean-Jacques Dhénin et Philippe Guez. Elle a été réalisée par un système comprenant un HP71B, un lecteur de disques HP9114A et une imprimante "LaserJet".

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