JPC

AVRIL 1987
NUMERO 43

Le numéro 35 FF.

A PROPOS DU CLUB

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

EDITORIAL

Chers membres,

Je suis heureux de vous présenter ce nouveau JPC du mois d'avril.

Ce numéro est peu plus gros que d'habitude. Il contient en effet un volumineux article que j'ai longtemps hésité à publier.

Plusieurs raisons m'ont amené à le faire. La première est que la raison d'être de notre Club est de faire profiter chacun de la connaissance et des réalisations de l'autre. La seconde est que ce Lex a demandé environ 3 mois à Janick et à moi-même pour sa conception. Voir tant de travail distribué librement est peu courant actuellement. Gardons l'esprit de générosité qui a fait la réputation de PPC. Troisièmement, les amateurs d'assembleur trouveront de quoi étudier. Nous avons particulièrement soigné les commentaires. Enfin, j'espère que tout le monde trouvera là un outil indispensable.

Cette décision nous a amené à augmenter le volume de ce Journal. Vous pourrez ainsi trouver vos rubriques habituelles, et même une nouvelle colonne consacrée au HP-28C.

Bonne lecture, et bon amusement...

Pierre David

PPC PARIS SE REUNIT UNE FOIS PAR MOIS

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

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

Si donc cela vous intéresse, n'hésitez plus un seul instant, venez nous rejoindre tous les premiers samedis de chaque mois (sauf en période de vacances scolaires) au :
Centre de Jeunesse et de Loisirs Jean Verdier
11 rue de Lancry
75010 Paris

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

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

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

Les dates des prochaines réunions sont :

Samedi 4 avril 1987
Samedi 16 mai 1987
Samedi 6 juin 1987

Pierre David (37)

AH ! VOUS ECRIVEZ

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

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

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

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

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

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

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

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

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

Jean-Jacques Dhénin (177)

```
DEF KEY 'fW', CHR$(197);      (é)
DEF KEY 'fE', CHR$(193);      (ê)
DEF KEY 'fR', CHR$(201);      (è)
DEF KEY 'fY', CHR$(203);      (ù)
```

```
DEF KEY 'fU', CHR$(195);      (û)
DEF KEY 'fI', CHR$(209);      (î)
DEF KEY 'fO', CHR$(194);      (ô)
DEF KEY 'f/', CHR$(92);       (\)
DEF KEY 'fA', CHR$(192);      (â)
DEF KEY 'fS', CHR$(200);      (à)
DEF KEY 'fD', CHR$(205);      (ë)
DEF KEY 'fJ', CHR$(207);      (ü)
DEF KEY 'fK', CHR$(221);      (ï)
DEF KEY 'f*', CHR$(124);      (|)
DEF KEY 'fC', CHR$(181);      (ç)
```

COURRIER DU COEUR

PPC-Paris
56 rue J.J. Rousseau
75001 Paris

Vend :
Interfaces vidéo HP82163B neuves (dans leur boîte, avec documentation) : 600 F seulement.

Des lots de 100 cartes magnétiques pour HP-41, HP-67, HP-97 et même HP-65 : 100 F seulement.

Des lecteurs de cartes magnétiques HP82400A pour HP-71 : 500 F pièce.

Shyam JHA
HP Grenoble
Tél: 76 62 54 54

Vend :
HP-75C + module 8 Ko Ram : 2500 F à débattre.

Pierre David
33 Bd St Martin
75003 Paris
Tél: (1) 48 87 68 93

Vend :
Imprimante 80 colonnes HP-IL HP82905B + papier : 2500 F.
Convertisseur HP82166A : 1000 F.

COURRIER DES LECTEURS

Thierry Besançon
9 rue Champ Chapelle
77120 Coulommiers

Devant découvrir à quoi servaient les fonctions du Lex DESLEX (je ne possède pas tous les JPC), je découvris que le comportement de la fonction PAINT était pour le moins bizarre.

Faites donc :

```
10 FOR I=0 TO 120
20 C$=STR$(PAINT(1,I,2))
30 NEXT I
```

Cela trace une ligne sur l'écran de mon HP-71B !

Faites aussi :

```
10 DISP ' ';
20 INVERSE
30 FOR I=0 TO 120
40 C$=STR$(PAINT(0,I,2))
50 NEXT I
```

Cela éteint une partie de l'écran.

Le premier paramètre détermine donc si l'on allume (1) ou si l'on éteint (0) un point. Les deux autres désignent les coordonnées d'un point de la matrice écran.

Alors bogue ou pas bogue ? Quoiqu'il en soit je suggère de ne rien modifier car l'on bénéficie ainsi d'une fonction graphique "bricolée", mais assez efficace.

Thierry Besançon (292)

Réponse du Docteur JPC :

De quelle bogue s'agit-il ?

La fonction PAINT parue dans JPC 24 est destinée à allumer ou éteindre des points sur l'écran graphique du HP-71. En retour, elle donne l'état antérieur du point (0 : éteint, 1 : allumé). A ma connaissance, il n'y a ni bogue, ni "bricolage"...

Pendant que j'y suis, je tiens à préciser que le programme contient une instruction STR$ inutile : il suffit de mettre :
C=PAINT(...

Rappelons que les anciens numéros de JPC sont en vente auprès du Club.

Janick Taillandier (246)

NOUVEAU PRODUIT

Mémoire à bulle

La firme australienne COMSYS vient de lancer sur le marché une unité de stockage HP-IL utilisant des mémoires à bulle.

Compatible avec le lecteur de cassettes numériques HP82161A, elle fournit 128 Ko, et se compose d'un lecteur (8.5x11x19 cm, 2 kg) et de cassettes de mémoire amovibles.

La mémoire est non volatile, l'unité est alimentée par batteries rechargeables, et les temps de transfert sont comparables à ceux de l'unité de disquettes HP9114A/B.

Ce lecteur est conçu pour des utilisation en environnement hostile. Nous ne connaissons malheureusement pas le prix de ce bel appareil.

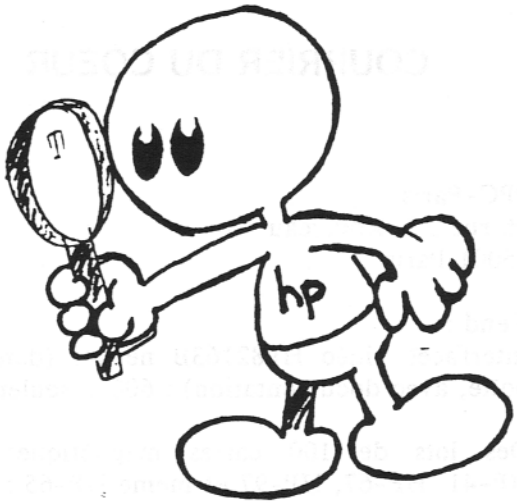
COMSYS Australia Pty Ltd
P. O. Box 153,
Subiaco W.A. 6008
Autralie

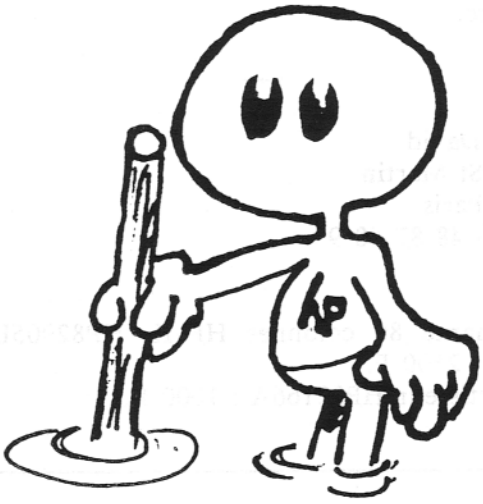
Nouveau module pour le HP-71

Des indiscrétions en provenance des Etats-Unis signaleraient la sortie prochaine d'un module pour HP-71. Ce module aurait pour fonction d'émuler un ordinateur MS-DOS. Logé dans le port 1 du HP-71, il contiendrait, outre un microprocesseur 80C86, une rom BIOS compatible PC. Ce système MS-DOS cohabiterait avec le système d'origine du HP-71, un peu comme Forth et Basic.

Le prix envisagé serait d'environ $ 290, ce qui semble relativement peu élevé compte tenu de la possibilité d'utiliser tous les logiciels standard du marché. Ceux-ci sont déjà souvent disponibles sur disquettes 3 pouces et demi.

Regrettons la juxtaposition d'un système aussi archaïque que MS-DOS et d'un système aussi bien conçu et aussi souple que celui du HP-71. Unix aurait été plus opportun.





HP-28C

P. David — Développez vos expressions 6

DEVELOPPER

Le HP-28C représente un saut extraordinaire dans le domaine du calcul de poche. Les possibilités apportées par le calcul symbolique ouvrent bien des horizons.

Mais si vous avez pratiqué un tant soit peu cette nouvelle caractéristique, vous avez pu constater que développer des expressions n'était pas toujours chose facile.

En effet, il faut passer par les touches [EXPAN] et [COLCT] du menu [ALGEBRA]. Développer une expression du type $(a+b)^3$ ne nécessite pas moins de 5 appuis sur [EXPAN] avant d'exécuter le [COLCT] final qui exprimera le résultat dans une forme compréhensible.

Or, ces 5 appuis sont pénibles. La fonction [EXPAN] est relativement lente. De plus, il faut surveiller attentivement l'affichage pour voir si une expression est «collectable» ou non. Et les formes affichées sont quelquefois difficiles à analyser.

Ces réflexions m'ont conduit à la conception d'un petit programme, qui développe une expression en une seule touche.

Le principe est simple. L'expression est développée par [EXPAN] jusqu'à trouver une forme stable, c'est à dire qui ne soit pas modifiée par les [EXPAN] successifs. Il s'agit en quelque sorte d'un «point fixe»... Arrivé à ce stade, ce petit programme exécute [COLCT] pour ramener l'expression à une forme compréhensible. Voilà. C'est tout !

```
« WHILE          l'exp. est au sommet de la pile
    DUP          dupliquer l'expression
    EXPAN        développer
    DUP ROT      (exp. dév., exp, exp dév.)
    SAME         exp développée = exp précédente ?
    NOT REPEAT   Non : sortir
  END            recommencer avec exp. développée
  COLCT          simplifier
»
```

Pour l'utiliser, rentrez ce programme (sans les commentaires, bien sûr), sur une seule ligne dans votre HP-28C.

Mettez-le alors dans une variable, par exemple DEV. Faites :

```
'DEV [STO]
```

Il est à présent stocké dans le HP-28C. Utilisons-le sur $(a+b)^3$:

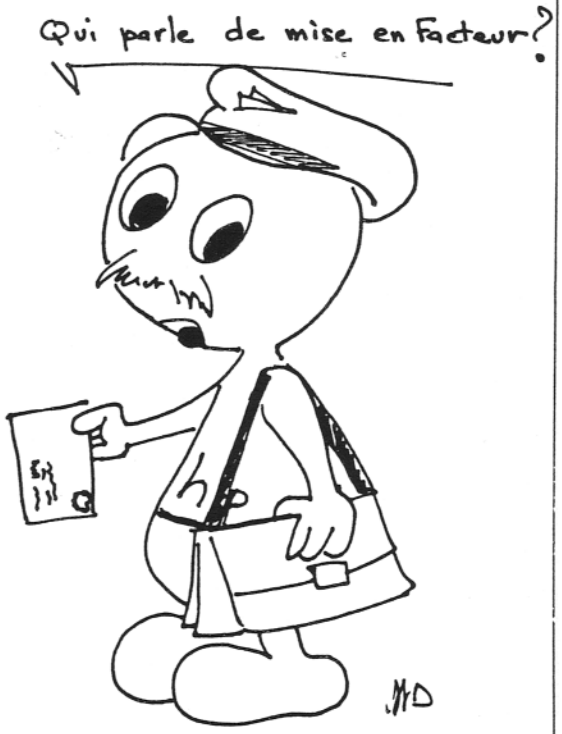
```
'A+B [ENTER]
3 [^]
[USER] [DEV]
```

Après environ 40 secondes, votre HP-28C affiche :

```
'3*A*B^2+A^3+3*A^2*B+B^3'
```

Voilà, à vous de faire le reste...

Pierre David (37)



HP-41

G. Toublanc

Fractionnez aussi *8*

FRACTIONNEZ AUSSI

Une fois n'est pas coutume : transposer un fichier Lex (assembleur) HP71 en fichier pour HP41. Et pourquoi pas ?

Les possesseurs de HP41 profiteront de l'étude faite pour HP71 : approximation d'un nombre décimal par une fraction. Les tout débutants y trouveront aussi leur compte : le programme est commenté pas à pas et la programmation est des plus simples (pas de programmation synthétique).

Pour l'algorithme employé, voyez l'article réfractaire dans JPC 42.

MODE D'EMPLOI

Soit n le nombre décimal approximé par la fraction N/D

FRAC
précision implicite 10^{-8} + exposant de n

entrée : n en X
sorties : chaîne alpha, N en Z, D en X

```
PI XEQ"FRAC" -> 104348/33215
```

PFRC
précision optionnelle 10^{-p}
entrées : p en X, n en Y
sorties : dito FRAC

```
PI ENTER 2 XEQ"PFRC" -> 22/7
```

AFRC
ordre d'approximation optionnel : a
entrées : a en X, n en Y
sorties : dito FRAC

```
PI ENTER 1 XEQ"AFRC" -> 3
PI ENTER 2 XEQ"AFRC" -> 22/7
PI ENTER 3 XEQ"AFRC" -> 333/106
```

VALF
après FRAC, AFRC ou PFRC renvoie la valeur décimale de N/D en FIX 9 puis delta = N/D - n en SCI 2

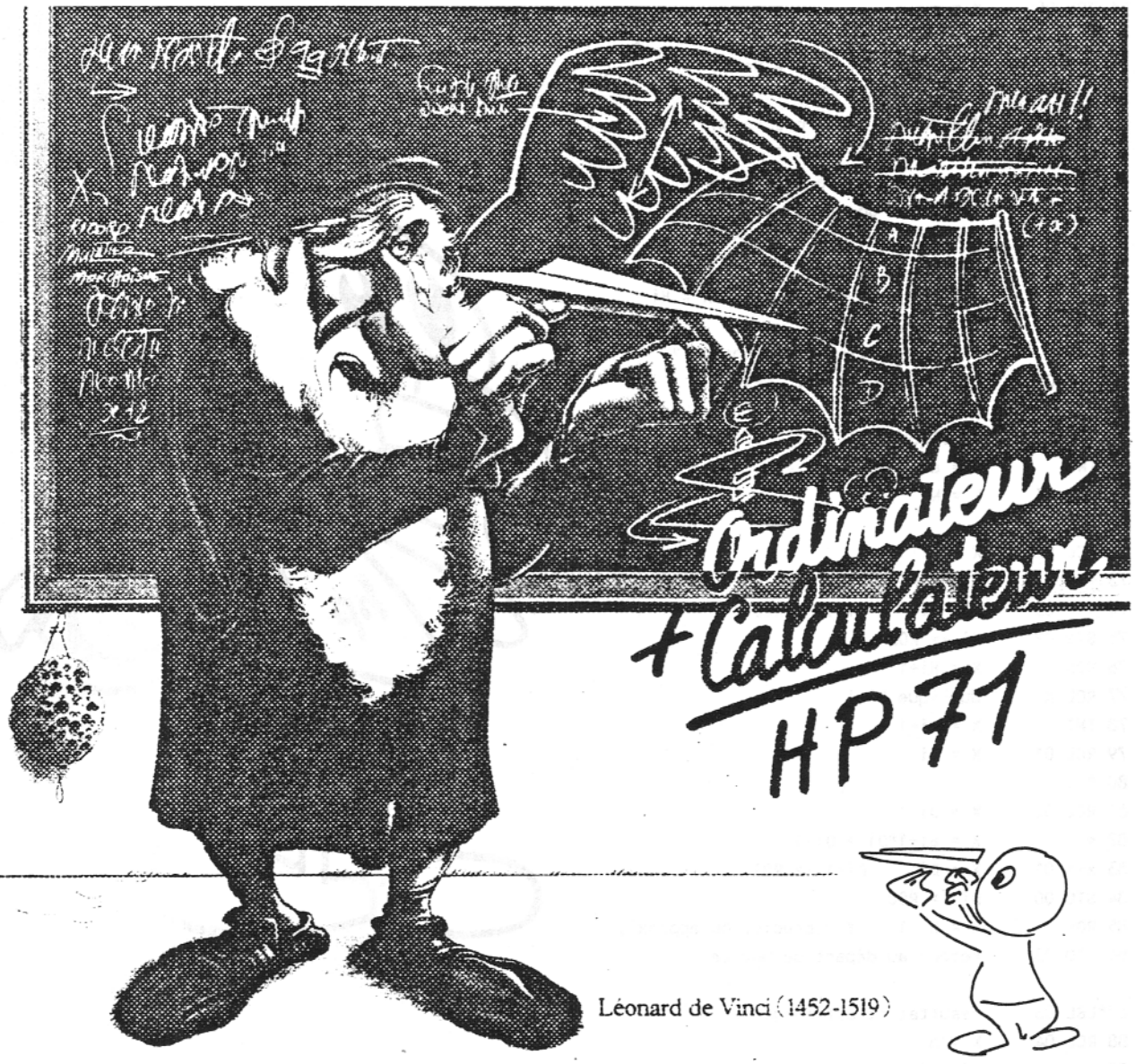
```
PI XEQ"FRAC" -> 104348/33215

XEQ"VALF" -> 3,141592654 R/S -> 0,00      00
```

LE PROGRAMME

```
01*LBL "VALF" valeur de N/D et delta
02 RCL Z      X = n
03 X<>Y       X = D
04 /          X = N/D
05 FIX 9
06 STOP       affichage valeur décimale de la
              fraction en FIX 9
07 ABS        X = ABS(N/D)
08 RCL 05     X = ABS(n)
09 -          X = ABS(N/D) - ABS(n) = delta
10 SCI 2
11 RTN        affichage delta en FIX 2


12*LBL "AFRC" option approximations  X = a   Y = n
13 CF 00      repère approximations
14 GTO 02     saut routine commune à FRAC et PFRC


15*LBL "FRAC" précision implicite   X = n
16 ENTER^     sauve n en Y
17 ABS        X = ABS(n)
18 LOG
19 INT        X = exposant de n
20 8
21 -          exposant de n  - 8
22 GTO 01     saut routine commune à PFRC

23*LBL "PFRC" option précision    X = p    Y = n
24 CHS        X = -p

25*LBL 01     départ commun à FRAC et PFRC
26 10^X       X = 10^(-p)  : précision ou
              10^(-8 + exposant de n) : précision
              flottante
27 SF 00      repère précision

28*LBL 02     départ commun à FRAC , AFRC et PFRC
29 X<>Y       X = n   Y = précis. ou approxima.
30 STO 04     n -> R04
31 ABS
32 STO 05     ABS(n) -> R05    à partir d'ici
              calculs avec valeurs absolues
33 INT
34 STO 03     No = INT(n) -> R03
35 RDN
36 0
37 STO 00     D-1 = 0 -> R00
38 RDN
39 1
40 STO 01     Do = 1 -> R01
41 STO 02     N-1 = 1 -> R02
42 RDN
43 LASTX      X = ABS(n) = Ho    Y = p ou a
```

```
44*LBL 03     départ de boucle de tests et calcul
45 FS? 00     test précision ?
46 GTO 04     oui saut test précision
47 DSE Y      non alors compteur approxi. décrém.
48 GTO 05     valeur compteur non nulle alors on
              continue dans la boucle
49 GTO 06     dernière approximation -> sortie

50*LBL 04     test précision
51 RCL 03     X = Ni
52 RCL 01     X = Di
53 /          X = Ni / Di
54 RCL 05     X = n
55 -
56 ABS        X = ABS(Ni/Di - n) = delta
57 R^         X = précision
58 X<>Y       X = delta    Y = précision   Z = Hi
59 X<=Y?      delta <= précision ?
60 GTO 06     oui alors sortie pour résultat
61 RDN        non alors on réordonne la pile
62 RDN        X = Hi     Y = précision

63*LBL 05     calcul de Ni+1 et Di+1    X = Hi
64 FRC
65 X=0?       Hi sans partie fractionnaire ?
66 GTO 06     oui alors sortie et résultat
67 1/X        X = Hi+1
68 RCL X      duplique Hi+1
69 INT        X = ai+1
70 RCL 03     X = Ni
71 *
72 RCL 02     X = Ni-1
73 +          X = ai+1*Ni + Ni-1 = Ni+1
74 X<> 03     X = Ni     Ni+1 -> R03
75 STO 02     Ni -> R02
76 RDN        X = Hi+1
77 RCL X      duplique Hi+1
78 INT        X = ai+1
79 RCL 01     X = Di
80 *
81 RCL 00     X = Di-1
82 +          X = ai+1*Di + Di-1
83 X<> 01     X = Di     Di+1 -> R01
84 STO 00     Di -> R00
85 RDN        X = Hi+1    Y = précis. ou approxi.
86 GTO 03     retour au départ de boucle

87*LBL 06     résultat et affichage
88 RCL 04     X = n
89 SIGN       récupération du signe de n
90 RCL 03     X = ABS(N)
91 *          X = N et son signe
92 CF 28      suppression des séparateurs pour
93 CF 29      FIX 0
94 FIX 0
95 CLA
96 ARCL X     N -> alpha
97 1
 98 RCL 01    X = D
 99 X=Y?      D = 1 ?
100 GTO 07    oui alors on ne l'affiche pas
101 "/"
102 ARCL X    D -> alpha

103*LBL 07
104 AVIEW     affichage de N/D  ou  N  si D = 1
105 END
```

Guy Toublanc (276)



Léonard de Vinci (1452-1519)

HP-75

LE P.M.S. DU HP-75
(Port Module Simulator)

Ce mois-ci, nous vous présentons un périphérique très, très peu connu, et très spécialisé : c'est une mémoire particulière, qui simule un module d'application externe branché dans l'un des trois ports du HP-75. Il peut se charger, mais ne constitue pas une extension de la mémoire puisqu'on ne peut ni l'éditer, ni y lire ou y écrire des données, mais simplement y copier des programmes complets à partir de la mémoire normale, et les exécuter comme s'ils venaient d'un module d'application.

L'appareil est indispensable pour développer des modules d'application : Rom masquées à peu près hors d'atteinte, ou Eprom à présent disponibles et qui sont la seule solution économique pour des toutes petites séries. En effet, contrairement au 71 pour lequel c'est tout simple (il suffit d'un port indépendant et de l'un des deux Lex ROMCOPY ou EPRMCOPY), pour le 75, il faut passer par un accessoire spécial.

L'appareil coûte assez cher (HP82713A, $420.00 aux USA, 4795 Francs HT au tarif HP). Il est livré avec un Lex nommé "PMSCMDS", et est vu du 75 comme deux Rom distinctes de 8 Ko chacune. On peut donc connecter autant de PMS qu'il y a de ports disponibles, soit de quoi créer ou simuler un module de 8, 16, 32 ou 48 Ko. Il est alimenté par le 75, mais contient une pile au lithium qui assure la sauvegarde du contenu quand le PMS est débranché du 75. Cette pile dure assez longtemps, mais elle est soudée sur la carte mère : il faut donc ramener l'appareil au service après vente HP pour la changer, eh oui !

Le principe d'utilisation est le suivant: on crée un programme Basic (moins de 8 Ko) en mémoire, puis on le copie dans une page du PMS (chacune peut en contenir plusieurs, dans la limite de 8 Ko). Dès lors, après avoir purgé le programme de la mémoire (pour éviter tout conflit...), il peut être exécuté et n'encombre plus votre précieuse mémoire... Mais il ne peut plus être ni listé, ni édité : pour ce faire, il faut à nouveau le copier en mémoire. C'est donc très simple... pour du Basic ; pour les Lex, il y a quelques restrictions (voir plus bas).

LES MOTS-CLES DU LEX "PMS"

L'exécution des programmes dans le PMS ne nécessite pas la présence du Lex. Par contre, il est absolument nécessaire pour pouvoir charger l'appareil ou modifier son contenu. Il comprend 9 mots-clés, dont cinq extensions de mots existants :

```
BUILD ":ROMx",<romid> [,"<lexname>"]
CAT "[<filename>] :ROMx"
CHECKSUM (":ROMx")
COPY "<filename>" TO "<filename>"
```
de PMS vers Ram
```
COPY "<filename>" TO "[<filename>]:ROMx"
```
de Ram vers PMS
```
COPY "ALL:ROMx" TO "<filename>"
```
page complète 8 Ko PMS vers Ram, sous un nom unique ("image de Rom")
```
PRIVATE "<filename>:ROMx"
PURGE "<filename>:ROMx"
ROMAVAIL (":ROMx")
ROMID(":ROMx")
ROMSIZE ("<RAM file>")
```

Les pages de 8 Ko sont identifiées par un nom d'unité en fonction du port dans lequel elles sont branchées et, s'il y a plusieurs PMS, de leur ordre en partant de la gauche. Dans le cas le plus courant, on a seulement ":ROMA" et ":ROMB".

Avant de pouvoir être utilisé, le PMS doit être initialisé. Il faut pour cela donner un numéro de "romid" (-32767 à -257 ou +3 à +32767), en prenant garde à ne surtout pas utiliser un numéro de module existant (par exemple, VisiCalc, Rom Math, Rom I/O ou Text-Formatter...), sous peine de plantage ! HP recommande soit de se faire allouer un numéro de Rom par leurs soins (c'est long...), soit d'utiliser la plage [100,199].

Ensuite, mettre en mémoire du 75 le fichier Lex qui sera implanté au cours de l'initialisation. C'est ici que ça se corse : vous n'avez droit qu'à un seul Lex par page de 8 Ko, et en plus, certains Lex, ou bien ne seront pas chargés, ou bien bloqueront le 75 ! C'est le problème des Lex "non-ROMables", avec diverses finesses liées à l'architecture interne : on m'a dit que ça avait gêné même les développeurs de chez HP (ceux de VisiCalc...), et on voit bien alors qu'il est impératif de tout bien tester avant de faire brûler une Eprom !

Une fois les images de Rom constituées et testées, avant de les copier sur cassette pour les faire graver en Eprom, il faut déterminer le "checksum" de chacune des pages, au moyen de l'ordre approprié. ROMAVAIL vous permet de savoir à tout moment combien il vous reste de place, et ROMSIZE combien un fichier Basic ou Lex demandera de place dans le PMS, c'est un peu plus qu'en mémoire.

Un sujet que le cas des Lex a permis d'entrevoir, ce sont les limitations et les bugs de l'appareil. Certains sont bien indiqués dans le manuel (commandes VER$ et MARGIN interdites dans un fichier PMS), mais il y en a bien d'autres, en particulier avec la Rom I/O. Voici donc une première liste, que l'expérience a permis de découvrir :

- les tableaux de chaînes (Rom I/O) perturbent les allocations de variables du 75 (erreur 127 : "bad parameter value" et plantage à peu près garanti !)
- EXIT (toujours la Rom I/O !) provoque un blocage : la "pile de retour" ne suit pas ! Il vaut donc mieux sortir "salement" de la boucle, sans EXIT...

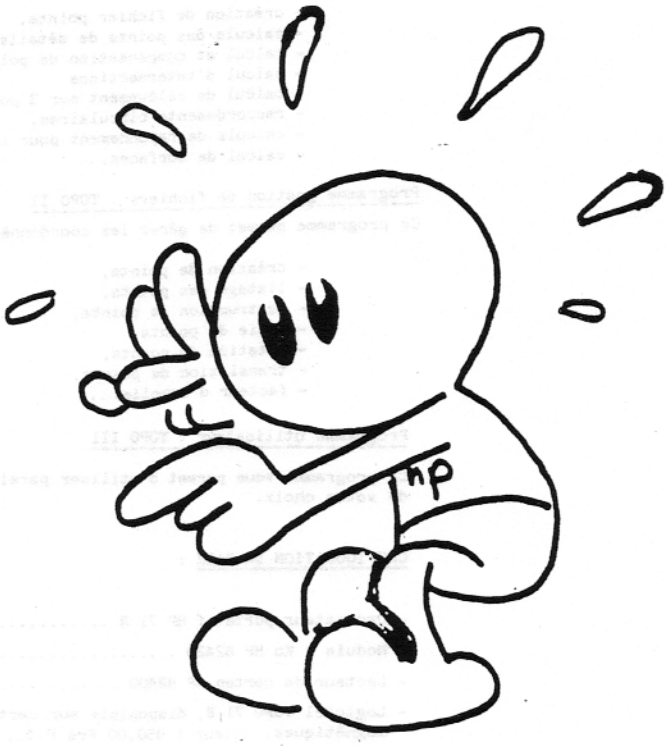
Ceci est sans doute à rapprocher de certaines incompatibilités signalées entre modules VisiCalc et I/O (ce dernier doit toujours être mis dans un port plus à gauche que VisiCalc), et entre Pod et module I/O. Les géniales fonctions avancées de gestion de fichiers ne peuvent pas être utilisées avec le Pod...

Parmi les Lex exploitables sans piège, signalons DISPLEX (VSoft) et HMSLEX (J.Y. Hervé), plus la Plotter-Rom de Corvallis (#03371-75-0) ; dans les "booby trap Lex", il y a MASSLEX et MADLEX (dommage, l'accès direct à la disquette en Rom, c'était tentant !), plus DATELEX (cas bizarre : une seule fonction marche bien DCONV), les autres bloquent le 75.... N'espérez pas charger PEEKPOKE ou HELP, c'est impossible !

J'ajoute que, ni la version "b" des Rom système du 75, ni celle de "PMSCMDS" n'apportent de solution... Mais je ne voudrais pas vous décourager, et je vous propose de vous dire un mot pour finir de ces fameuses Eprom 32 Ko de chez CMT.

Elles sont disponibles chez EduCALC (#75-669, prix $229.95, auquel il faut rajouter $60.00 pour la programmation chez CMT), ou en France chez COSERM (réf. CMT75-32KE, prix public 3719 F HT, plus programmation 600 F). Compte tenu à la fois du faible marché du 75 et des particularités de brochage de la bête, seul CMT assure la programmation. Il faut donc lui envoyer (via votre fournisseur) vos logiciels, sous forme des n sauvegardes des pages de PMS (COPY ALL:ROMx TO..., fichiers de type "R" come "ROM"), chacun acompagné de son catalogue (sur papier) et de son checksum. Vous postez cassette ou disquette et lettre d'accompagnement, et... je vous en dirai plus lorsque j'aurai reçu mon Eprom. A bientôt!

Eric Gengoux (108)



LE PONT EQUIPEMENTS

Rue Copernic (acces autoroute Lyon - Marseille)
☎ 78 73 02 88 - Telex LEPONT 380034 F
F 38670 CHASSE SUR RHONE - B.P. N 11

mesure topographie calcul dessin
importation exportation gros

UNE EXCLUSIVITE LE PONT EQUIPEMENTS !

TOPO 71 B

Un logiciel TOPOGRAPHIE
développé sur l'ordinateur portable HP 71 B

La solution HP 71 B

L'ordinateur HP 71 B, muni d'un langage BASIC résident, associe la puissance informatique aux dimensions d'un calculateur de poche.

Un système de traitement sophistiqué et évolutif

Grâce à la boucle d'interface HP.IL, le HP 71 B devient un véritable système capable d'imprimer, de tracer, de stocker et de récupérer des données, d'afficher des informations et de gérer des périphériques (mémoire de masse, imprimante, table à dessiner).

LE LOGICIEL : TOPO HP 71 B

Ce logiciel résout les problèmes classiques de topométrie :

Programme calculs topographiques : TOPO I

- création de fichier points,
- calculs des points de détails en X, Y, Z,
- calcul et compensation de polygonales,
- calcul d'intersections
- calcul de relèvement sur 3 points,
- raccordements circulaires,
- calculs de rayonnement pour implantation,
- calcul de surfaces...

Programme gestion de fichiers : TOPO II

Ce programme permet de gérer les coordonnées des points calculés.

- création de points,
- listage des points,
- destruction de points,
- copie de points,
- rotation de points,
- translation de points,
- facteur d'échelle...

Programme utilisateur : TOPO III

Ce programme vous permet d'utiliser parallèlement au logiciel TOPO 71 un programme de votre choix.

CONFIGURATION DE BASE :

- Ordinateur portatif HP 71 B	4 988,00 Frs HT
- Module 4 Ko HP 82420	704,50 Frs HT
- Lecteur de cartes HP 82400	1 604,00 Frs HT
- Logiciel TOPO 71 B, disponible sur cartes magnétiques, valeur 1 950,00 Frs H.T., actuellement promotion jusqu'au 30.11.86, offert gracieusement	Gracieusement
	7 296,50 Frs HT

Option cassette :

- Ordinateur portatif HP 71 B	4 988,00 Frs HT
- Module 4 Ko HP 82420	704,50 Frs HT
- Interface HP.IL HP 82401	1 184,00 Frs HT
- Unité de cassette HP 82161	5 351,00 Frs HT
- Logiciel TOPO 71 B, disponible sur cassette valeur 1 950,00 Frs HT, actuellement promotion jusqu'au 30.11.86, offert gracieusement	Gracieusement
	12 227,50 Frs HT

ASSEMBLEUR

BASIC

LE COIN DES LHEX

FINPUT

Le HP-71 est un ordinateur de poche dont la puissance n'en finira pas de nous étonner. Malheureusement, certaines de ces possibilités sont peu documentées ou quelquefois lourdes à utiliser.

Il en est ainsi des champs protégés. Le manuel d'utilisation les confine dans un coin de la page 237, dans une région que beaucoup d'entre nous considèrent comme ténébreuse.

En quelques mots, les champs protégés servent lors d'un INPUT ou LINPUT à isoler des caractères et empêcher l'utilisateur du programme de les effacer ou d'écrire autre chose à la place.

A titre d'exemple, supposons que nous ayons à entrer une date. L'affichage devra avoir l'aspect suivant :

```
Date: Dy/Mo/Yr
```

L'utilisateur devra remplacer les seuls caractères Dy (Day), Mo (Month) et Yr (Year) par leur valeur. Les autres ne doivent pas changer. Si vous êtes courageux, voici le programme pour atteindre ce but :

```
100 E$=ESC$("<") ! Eteint le curseur
110 A$=ESC$(">") ! Allume le curseur
120 D$=A$&"Jr"&E$"/"&A$&"Mo"&E$&"/"&A$&"Yr"
130 DISP E$&"Date: "&D$&E$; ! Affichage
140 INPUT "";I$ ! Introduction de la date
```

Si vous n'avez pas PRINTLEX dans votre machine, il faut remplacer les deux lignes 100 et 110 par :

```
100 E$=CHR$(27)&"<"
110 A$=CHR$(27)&">"
```

Première observation : ce programme est illisible. Malgré les commentaires, on a du mal à comprendre ce qu'il fait.

Deuxième observation : si vous essayez ces quelques lignes, vous verrez la lenteur affligeante de l'affichage du masque de saisie. Ca ne fait pas très sérieux...

Troisième observation : rentrez une date, puis appuyez sur [ATTN]. Le mois s'efface, et le curseur revient au début. Appuyez une deuxième fois sur [ATTN]. Rien ne se passe. La touche [ATTN] est inactive, il n'y a aucun moyen d'arrêter le programme. La seule solution est d'appuyer sur [ENDLINE]. Et votre date est validée telle quelle, malgré votre volonté d'arrêter. Fâcheux !

Quatrième observation : rentrez la date, et continuez à rentrer des caractères supplémentaires. Vous voyez que rien ne vous en empêche. Comment y remédier ?

Le problème vient de ce qu'aucun caractère n'est plus protégé après la date. Le HP-71 n'a donc plus aucune raison de bloquer le reste de l'affichage. Pour y arriver, il faut calculer : nous devons afficher un masque de 14 caractères, et les autres doivent être tous protégés. Les autres, cela fait 96-14, c'est à dire 82 espaces à protéger. Eh bien, allons-y, ajoutons quelques lignes :

```
121 DIM S$[82]
122 S$=""
123 FOR I=1 TO 82
124 S$=S$&" "
125 NEXT I
```

Si vous avez STRINGLX, vous pouvez simplifier en ne tapant que :

```
121 DIM S$[82]
122 S$=RPT$(" ",82)
```

Puis, remplaçons la ligne 130 par :

```
129 WIDTH INF
130 DISP E$&"Date: "&D$&E$&S$;
```

A l'exécution, il y a un clignotement indésirable de l'affichage, puis tout s'affiche. C'est merveilleux, on ne peut rentrer de caractère au-delà de la date !

Mais il y a un nouveau problème. Appuyez sur la touche [->] après la deuxième année : vous voyez l'affichage se dérober sur la gauche. Mieux encore : appuyez sur [g][->]. Après un temps relativement long, l'affichage se vide. Curieux, n'est ce pas ?

Bref, on le voit, l'utilisation des champs protégés est une possibilité fantastique, mais mal servie par :

- un manuel confus et incomplet,
- une lourdeur d'utilisation sans égale,
- et un comportement "sale".

L'idée nous vint donc logiquement de créer un ordre permettant d'exploiter à fond cette caractéristique du HP-71, en simplifiant la tâche du programmeur. Voici donc le résultat de cette cogitation, nous avons nommé : FINPUT.

UTILISATION DE FINPUT

Voici comment utiliser l'ordre FINPUT (Formatted INPUT). Notre principale préoccupation lors de la conception de ce Lex a été la facilité d'utilisation pour le programmeur. Toutefois, FINPUT permet de faire simplement des choses extrêmement complexes.

FINPUT mono-ligne

Dans sa forme la plus simple, FINPUT est une extension de l'ordre LINPUT pour manipuler facilement les champs protégés.

Le programme Basic qui nous a servi d'exemple se résume alors à la forme suivante :

```
100 DIM I$[6]
110 FINPUT I$,"Date: Dy/Mo/Yr","6P2UP2UP2UP",A
```

Dans cet exemple, nous pouvons voir que :

- I$ est la chaîne de destination. Elle doit être créée lorsque l'exécution de FINPUT débute.

- Ensuite, vient la «chaîne de message». C'est ce qui apparaîtra sur l'écran. Nous remarquons que tout y est, que ce soit protégé ou non.

- Après, nous avons la «chaîne de protection». Examinons de près ce qu'elle contient : le 6P signifie que les six premiers caractères sont protégés (Protected). Le 2U précise que les deux caractères suivants sont «non protégés» (Unprotected). Vous avez compris, le P indique ensuite que le caractère suivant est protégé, et ainsi de suite... Le P final signifie que le reste de l'affichage est protégé. Il n'y a donc pas besoin de mettre 82P.

- Enfin, nous avons la «variable attn». En sortie, elle vaudra 0 si la touche [ATTN] a été utilisée pour sortir de FINPUT.

Comme vous pouvez le constater, le problème de l'utilisation des champs protégés est simplifié à l'extrême. Utiliser FINPUT est maintenant devenu un réflexe. Mais, ce n'est pas tout ce qu'apporte FINPUT. Parmi ses nombreux avantages, nous pouvons citer :

Simplification de la spécification de la protection :
Comme nous l'avons vu, la spécification des champs protégés est extrêmement simple.

Gestion de la touche [ATTN] :
Lors d'un FINPUT, la touche [ATTN] sert une première fois pour revenir à l'affichage par défaut (la «chaîne de message»). La deuxième fois, [ATTN] sort de FINPUT en mettant la «variable attn» à 0. Le programme n'est pas interrompu, et il est facile de gérer la touche [ATTN]. Un simple test :

```
IF NOT A THEN ...
```

règle le problème de cette touche.

Gestion des touches [->] *et* [g][->] *:*
Essayez, vous verrez ! Les touches [->] et [g][->] n'ont plus l'effet désagréable qu'elles avaient lors du premier exemple.

Gestion des «variables courtes» :
Dans l'exemple ci-dessus, si vous aviez déclaré la variable I$ à moins de 6 caractères, mettons 3, vous n'auriez pas pu en rentrer plus de 3. FINPUT met cette sécurité supplémentaire. Si vous écrivez votre programme avec FINPUT, vous réduisez les chances de "String Overflow" !

FINPUT sans chaîne de protection

Dans bien des cas, il n'y a pas besoin d'une gestion sophistiquée de l'affichage. Par exemple, pour rentrer le nom d'un fichier. Avec INPUT, vous feriez :

```
100 INPUT "Fichier: ";F$
```

Mais, un nom de fichier en mémoire ne peut faire plus de 8 caractère. Avec FINPUT, vous ferez :

```
100 DIM F$[8]
110 FINPUT F$,"Fichier: ",A
120 IF NOT A THEN END
```

Et voilà, le tour est joué ! Quoi de plus facile ? Vous ne pouvez pas rentrer plus de 8 caractères, et si vous désirez vous rétracter en appuyant sur [ATTN], le programme l'autorise de la manière la plus simple.

Cette «variable de protection» est donc facultative.

Si elle n'est pas présente, FINPUT la crée. Par défaut, cette chaîne de protection est : STR$(LEN(message))&"PU". C'est à dire que tous les caractères de la «chaîne de message» sont protégés. Les suivants sont tous non protégés, jusqu'à la dimension maximum de la chaîne résultat.

FINPUT sur plusieurs lignes

Mais la principale caractéristique de FINPUT est la gestion de plusieurs lignes de saisie. C'est, en quelque sorte, un «masque d'écran» complet.

Supposons que nous voulions rentrer la date et l'heure. Ceci pourrait être fait de la manière suivante :

```
100 DIM D$[6],H$[6]
110 FINPUT D$,"Date: Dy/Mo/Yr","6P2UP2UP2UP",A
120 IF NOT A THEN END
130  traitement de la date
 :
 :
200 FINPUT H$,"Heure: Hr:Mn:Sc","7P2UP2UP2UP",A
210 IF NOT A THEN END
220  traitement de l'heure
```

Mais il y a une autre solution :

```
100 OPTION BASE 1 ! Pour démarrer les tableaux à 1
110 DIM I$(2)[6],M$(2),P$(2)
120 DATA Date: Dy/Mo/Yr,Heure: Hr:Mn:Sc
130 DATA 6P2UP2UP2UP,7P2UP2UP2UP
160 READ M$ ! Lit les 2 messages
170 READ P$ ! Lit les 2 chaînes de protection
180 FINPUT I$,M$,P$,A
190 IF NOT A THEN END
200  traitement de la date (I$(1))
210  traitement de l'heure (I$(2))
```

Cette dernière solution est beaucoup plus pratique que la première. L'introduction des données se fait en une seule fois.

Les touches de curseur vertical servent à passer d'une ligne à l'autre. La touche [ENDLINE] sert à valider une ligne. **Important** : la touche [RUN] sert à valider la ligne courante et sortir de FINPUT. La «variable attn» contient alors le numéro de la ligne sur laquelle on est sorti. La valeur 0 indique toujours une sortie par [ATTN].

Cette forme d'utilisation de FINPUT permet de saisir des fiches complètes en une seule opération. Le programmeur n'a donc plus à gérer tous les déplacements à l'intérieur de cette fiche, FINPUT le fait à sa place !

FINPUT remplace MENU

Vous vous rappelez sans doute de la fonction MENU, écrite par Jean-Jacques Dhénin. FINPUT remplace en quelque sorte cette fonction. En mode "tableaux", si tous vos caractères sont protégés, FINPUT offre des fonctionnalités similaires à celles de MENU. Une pression sur [RUN] indique quelle ligne vous choisissez, ligne dont vous avez le numéro dans la variable *attn*.

RESUME

La syntaxe :

FINPUT *v1, exp, exp, attn*
FINPUT *v1, exp, v2*

ou bien

FINPUT *tab, tab, tab, attn*
FINPUT *tab, tab, v2*

où :
- *v1* est un nom de variable alphanumérique,
- *exp* sont des expressions alphanumériques,
- *attn* est un nom de variable numérique,
- *tab* sont des noms de tableaux alphanumériques de même dimension

La «chaîne de message» ne doit contenir que des caractères affichables, c'est à dire des caractères dont le code n'est ni 0 (NULL), ni 27 (ESC), ni 13 (CR) ni 10 (LF), ni 8 (BS).

La spécification de format doit contenir les lettres "U" et "P" (en majuscules ou minuscules) précédées ou non d'un facteur de répétition pour indiquer les caractères protégés ou non.

Les variables simples peuvent être assimilées à des tableaux à un seul élément.

La variable de destination doit être déjà créée.

L'utilisation

Sous FINPUT, la signification des touches est la suivante :

[ATTN]
Si des caractères ont été entrés, restauration de l'affichage par défaut, c'est à dire de la «chaîne de message» telle quelle.
Une deuxième fois : sortie de FINPUT.

[f] [OFF]
Sortie directe de FINPUT.

[ENDLINE]
Validation de la ligne courante et passage à la ligne suivante. Si il n'y a qu'une seule ligne, sort de FINPUT.

[RUN]
Validation de la ligne courante et sort de FINPUT. Si il n'y a qu'une seule ligne, [RUN] est identique à [ENDLINE].

[^], [v], [g] [^] et [g] [v]
Change de ligne sans valider la ligne courante. Si une seule ligne, remet l'affichage par défaut.

Contenu des variables en sortie

Les variables «message» et «format» ne sont jamais modifiées.

En cas de sortie normale (par [ENDLINE] ou [RUN]), la variable *attn* contient le numéro de la ligne sur laquelle on est sorti. Ce numéro varie entre 1 et la dimension du tableau.

La variable destination contient les valeurs introduites.

En cas de sortie prématurée (par [ATTN] ou [f] [OFF]), la variable *attn* contient 0. La variable destination n'est pas changée.

CONCEPTION DE FINPUT

Le but de ce chapitre est de décrire brièvement quelques points importants de FINPUT. Ces quelques 2 Ko d'assembleur nous semblent, a posteriori, très intéressants et bourrés de trucs et d'exemples de résolution de problèmes.

Historique

Pour les curieux, un peu d'histoire...

Tout a commencé par une envie de créer un masque d'écran permettant d'utiliser des champs protégés, pour se constituer un outil général de saisie.

Janick étant un fervent (voire farouche) défenseur du HP-75, nous avons ouvert le manuel de la Rom I/O, complément naturel (et indispensable) du HP-75. La fonction TEMPLATE$ nous a servi de base de travail.

Cependant, la syntaxe de la chaîne de protection était trop liée aux spécificités de la machine. Nous avons donc défini une syntaxe basée uniquement sur les caractères "P" et "U". Elle est vite apparue lourde à manipuler. Nous avons donc introduit les facteurs de répétition dans le projet.

La fonction TEMPLATE$ est limitée à une ligne. Il faut préciser que le HP-75 ne permet pas d'utiliser, de base, les champs protégés. Et TEMPLATE$ est le seul moyen d'y arriver. Nous avons étendu le projet aux tableaux entiers.

Dans le projet initial, il fallait une variable indiquant sur quelle ligne s'était terminée la saisie. Peu à peu, cette variable s'est muée en *variable attn*. Aujourd'hui, elle est plus utilisée comme indicateur booléen, sauf bien sûr en utilisation de type "menu".

Le projet ainsi défini, nous avons commencé à coder. Nous sommes rapidement arrivés devant le problème posé par les touches [->] et [g] [->].

Nous avons alors envoyé une lettre au Support Technique aux Laboratoires de Corvallis. Nous nous sommes fait gentiment «envoyer promener». Il est vrai que la question dépassait quelque peu leurs attributions.

Nous nous sommes alors remis à l'ouvrage. Et FINPUT est sorti des ateliers vers la mi-novembre 1986. Depuis cette première version, quelques bogues mineures ont été corrigées.

Au cours de la conception, le nom de l'orde a changé deux fois. Aux tous débuts, nous parlions de SAISY, pour mettre l'accent sur l'aspect «saisie d'écran», «masque de saisie». Ensuite, nous l'avons nommé SINPUT (Screen INPUT), par opposition à LINPUT (Line INPUT). Et les capacités de formattage de la ligne de saisie nous ont conduits à renommer en FINPUT, à défaut de INPUT USING...

La parse

La première difficulté de FINPUT était la parse de l'ordre. En effet, le paramètre de protection est optionnel. A l'exécution, nous devons savoir si ce paramètre est présent sans avoir à évaluer le paramètre suivant. Nous avons donc mis en place un «quartet de reconnaissance», dont la valeur est 0 si il n'y a pas de chaîne de protection.

Autrement dit, la tokenisation de FINPUT I$,M$,P$,A est :

```
tXWORD id tFINPUT
tokenisation de I$
tCOMMMA
tokenisation de M$
tCOMMA
quartet à #F
tokenisation de P$
tCOMMA
tokenisation de A
```

alors que celle de FINPUT I$,M$,A est :

```
tXWORD id tFINPUT
tokenisation de I$
tCOMMMA
tokenisation de M$
tCOMMA
quartet à #0
tokenisation de A
```

D'autre part, il faut faire attention à ce que I$ soit un nom de variable ou de tableau, alors que M$ et P$ sont des expressions.

Principe général

Accrochez-vous !

Tout est représenté de manière interne par des tableaux. Chaque tableau est manipulé par l'intermédiaire d'un «dope vecteuuuuur» bien à lui. Le format nous est propre, et n'est pas similaire à ce qui existe actuellement, sur la Math Stack par exemple. Si vous voulez vous rafaraichir la mémoire sur les dope vecteurs normaux, revoyez vos IDS I, § 13.3.2.2. La raison de ce format particulier est le manque de mémoire. Il aurait suffit d'un quartet supplémentaire dans le dope vecteuuuuur pour que FINPUT ne soit pas réalisable !

Une chaîne est assimilée à un tableau d'un seul élément. Tous les acteurs sont ainsi banalisés. Le seul objet qui fasse exception à la règle est la variable *attn*, stockée sous une forme un peu spéciale. Mais ne nous attardons pas.

Toute l'astuce de FINPUT, et ce qui fait sa supériorité sur les séquences d'échappement, est l'exploitation de la «bit map d'affichage», ou DSPMSK. Chaque bit de cette zone représente la protection d'un caractère. Une des étapes du processus consiste donc à compiler la chaîne de protection (caractères "U" et "P", avec facteurs multiplicatifs) en un champ de bits compréhensible par le HP-71.

Pour afficher une ligne, il suffit donc d'afficher le message, puis de mettre dans DSPMSK le champ de bit compilé précédemment.

On voit donc que les données nécessaires lors de l'exécution de FINPUT sont :

- Le message par défaut (M$),
- Le champ de bits compilé,
- La saisie de l'utilisateur, qui ira ensuite dans la variable I$,
- L'image du display buffer, qui sera réinjectée à l'affichage lors du prochain retour sur la ligne.

Nous voyons donc trois nouveaux tableaux apparaître. Ils existent entre AVMEMS et la Math-Stack. Bien qu'ils n'aient aucune relation avec des tableaux Basic, nous les avons nommés respectivement B$, U$ et D$. Après l'exécution de FINPUT, ils sont effacés.

Avec ces trois tableaux supplémentaires, nous pouvons aller lire l'algorithme général fourni dans l'en-tête de SINPUTe, dans le fichier «bas.as».

CONCLUSION

Voilà, nous espérons vous avoir tout dit sur FINPUT. Cet ordre nous a demandé beaucoup de travail, mais nous pensons qu'il en vaut la peine.

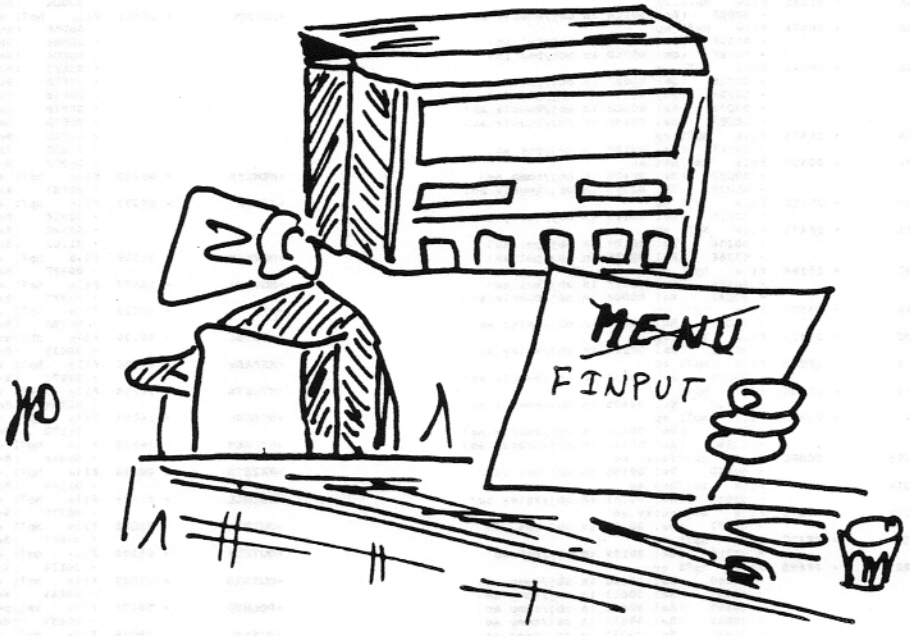
Depuis que nous l'avons fait, nous n'arrêtons pas de l'utiliser. Le programme d'enregistrement et de modification du fichier des adhérents du Club est réalisé sur la base de FINPUT. Et cela fournit un programme très agréable à utiliser, sans pour autant être très gourmand en octets.

Le listage du code est donné sous forme comparable à celle des IDS du HP-71. Nous savons que vous ne pourrez pas l'introduire dans votre HP-71 tel quel. Mais ceux d'entre vous qui désireront l'étudier le feront plus facilement.

Pour utiliser FINPUT, rendez-vous à la rubrique «le coin des Lhex», ou venez aux réunions. Dans tous les cas, FINPUT est déjà dans JPCLEX, et ceux qui ont acheté l'Eprom JPC l'auront dans leur Eprom.

FINPUT est à vous, maintenant. Amusez-vous bien, et surtout faites de beaux programmes...

Pierre David (37)
Janick Taillandier (246)



```
*ADDRSS     * 0F527  File : hp71.ep
                     + 00579   (Rel 001A2 in obj/bas.ao)
*ATNCLR     * 00510  File : hp71.ep
                     + 00F73   (Rel 002D8 in obj/boucle.ao)
                     + 01176   (Rel 004DB in obj/boucle.ao)
*AVE+D1     * 18BB8  File : hp71.ep
                     + 00760   (Rel 00389 in obj/bas.ao)
                     + 00CAA   (Rel 0000F in obj/boucle.ao)
                     + 01168   (Rel 004CD in obj/boucle.ao)
*AVMEMS     * 2F594  File : hp71.ep
                     + 0079C   (Rel 003C5 in obj/bas.ao)
*BSERR      * 0939A  File : hp71.ep
                     + 004AE   (Rel 000D7 in obj/bas.ao)
                     + 00589   (Rel 001B2 in obj/bas.ao)
                     + 00A1C   (Rel 001A8 in obj/comp.ao)
                     + 00D18   (Rel 0007D in obj/boucle.ao)
*CHEDIT     * 14C99  File : hp71.ep
                     + 00E3C   (Rel 001A1 in obj/boucle.ao)
*CKSREQ     * 00721  File : hp71.ep
                     + 00EA8   (Rel 0020D in obj/boucle.ao)
*COLLAP     * 091FB  File : hp71.ep
                     + 00665   (Rel 0028E in obj/bas.ao)
*COMCK*     * 032AE  File : hp71.ep
                     + 0006D   (Rel 0006D in obj/saisy.ao)
*CONVUC     * 152AA  File : hp71.ep
                     + 009F3   (Rel 0017F in obj/comp.ao)
*CRLFOF     * 02296  File : hp71.ep
                     + 00F68   (Rel 002CD in obj/boucle.ao)
*CURSFL     * 151DF  File : hp71.ep
                     + 00E1A   (Rel 0017F in obj/boucle.ao)
*CURSOR     * 2F47E  File : hp71.ep
                     + 0020F   (Rel 00090 in obj/pol.ao)
                     + 002FB   (Rel 0017C in obj/pol.ao)
*D1MSTK     * 1954E  File : hp71.ep
                     + 006AE   (Rel 002D7 in obj/bas.ao)
                     + 01092   (Rel 003F7 in obj/boucle.ao)
                     + 01145   (Rel 004AA in obj/boucle.ao)
*D+AVMS     * 1A460  File : hp71.ep
                     + 006E8   (Rel 00311 in obj/bas.ao)
*DEFADR     * 2F967  File : hp71.ep
                     + 0039D   (Rel 0021E in obj/pol.ao)
*DEST       * 0F7B0  File : hp71.ep
                     + 0113E   (Rel 004A3 in obj/boucle.ao)
*DOPEA      * 2F8DF  File : obj/bas.ao
                     + 00CB1   (Rel 00016 in obj/boucle.ao)
                     + 01109   (Rel 0046E in obj/boucle.ao)
*DOPEB$     * 2F8C5  File : obj/bas.ao
                     + 004DD   (Rel 00106 in obj/bas.ao)
                     + 007D1   (Rel 003FA in obj/bas.ao)
*DOPED$     * 2F8D2  File : obj/bas.ao
                     + 004D2   (Rel 000FB in obj/bas.ao)
                     + 00C74   (Rel 00400 in obj/comp.ao)
                     + 00D6C   (Rel 000D1 in obj/boucle.ao)
*DOPEI$     * 2F871  File : obj/bas.ao
                     + 004FE   (Rel 00127 in obj/bas.ao)
                     + 0064A   (Rel 00273 in obj/bas.ao)
                     + 008C8   (Rel 00054 in obj/comp.ao)
                     + 01019   (Rel 0037E in obj/boucle.ao)
                     + 010B4   (Rel 00419 in obj/boucle.ao)
*DOPEM$     * 2F87E  File : obj/bas.ao
                     + 004F3   (Rel 0011C in obj/bas.ao)
                     + 00673   (Rel 0029C in obj/bas.ao)
                     + 006B5   (Rel 002DE in obj/bas.ao)
                     + 0074D   (Rel 00376 in obj/bas.ao)
                     + 00C7B   (Rel 00407 in obj/comp.ao)
                     + 00FD3   (Rel 00338 in obj/boucle.ao)
```



```
*DOPEP$     * 2F8DF  File : obj/bas.ao
                     + 004C7   (Rel 000F0 in obj/bas.ao)
                     + 00767   (Rel 00390 in obj/bas.ao)
*DOPEU$     * 2F888  File : obj/bas.ao
                     + 004E8   (Rel 00111 in obj/bas.ao)
                     + 00851   (Rel 0047A in obj/bas.ao)
                     + 01012   (Rel 00377 in obj/boucle.ao)
*DRANGE     * 1B076  File : hp71.ep
                     + 00A0C   (Rel 00198 in obj/comp.ao)
*DSP$00     * 185DB  File : hp71.ep
                     + 0109C   (Rel 00401 in obj/boucle.ao)
*DSPBFE     * 2F540  File : hp71.ep
                     + 002AA   (Rel 0012B in obj/pol.ao)
                     + 0104E   (Rel 003B3 in obj/boucle.ao)
*DSPBFS     * 2F480  File : hp71.ep
                     + 00FA5   (Rel 0030A in obj/boucle.ao)
*DSPCHA     * 01C3E  File : hp71.ep
                     + 00DBF   (Rel 00124 in obj/boucle.ao)
*DSPCHX     * 2F674  File : hp71.ep
                     + 00339   (Rel 001BA in obj/pol.ao)
                     + 0038F   (Rel 00210 in obj/pol.ao)
*DSPMSK     * 2F540  File : hp71.ep
                     + 00226   (Rel 000A7 in obj/pol.ao)
                     + 002B4   (Rel 00135 in obj/pol.ao)
                     + 00D73   (Rel 000D8 in obj/boucle.ao)
                     + 00DE7   (Rel 0014C in obj/boucle.ao)
*DSPSTA     * 2F475  File : hp71.ep
                     + 0031F   (Rel 001A0 in obj/pol.ao)
*ELMCPY     * 003D7  File : obj/bas.ao
                     + 00C82   (Rel 0040E in obj/comp.ao)
                     + 01020   (Rel 00385 in obj/boucle.ao)
*EOLXC*     * 052EC  File : hp71.ep
                     + 00115   (Rel 00115 in obj/saisy.ao)
*ESCSTA     * 2F47B  File : hp71.ep
                     + 00356   (Rel 001D7 in obj/pol.ao)
                     + 0036E   (Rel 001EF in obj/pol.ao)
*EXPEXC     * 0F186  File : hp71.ep
                     + 00406   (Rel 0002F in obj/bas.ao)
                     + 00CA3   (Rel 00008 in obj/boucle.ao)
*EXPPAR     * 03FD9  File : hp71.ep
                     + 000EC   (Rel 000EC in obj/saisy.ao)
*EXPRDC     * 05922  File : hp71.ep
                     + 0012B   (Rel 0012B in obj/saisy.ao)
*F-R1-0     * 2F8AB  File : hp71.ep
                     + 00CCB   (Rel 00030 in obj/boucle.ao)
*F-R1-3     * 2F8BA  File : hp71.ep
                     + 01110   (Rel 00475 in obj/boucle.ao)
*FINDA      * 023E3  File : hp71.ep
                     + 00E5F   (Rel 001C4 in obj/boucle.ao)
                     + 00EBC   (Rel 00221 in obj/boucle.ao)
*FINPUTd    * 000FC  File : obj/saisy.ao
                     + 0056D   (Rel 00196 in obj/bas.ao)
*FINPUTe    * 00577  File : obj/bas.ao
                     + 00041   (Rel 00041 in obj/saisy.ao)
*FINPUTp    * 0007C  File : obj/saisy.ao
                     + 00572   (Rel 0019B in obj/bas.ao)
*FIRSTC     * 2F47C  File : hp71.ep
                     + 00318   (Rel 00199 in obj/pol.ao)
*FUNCRO     * 2F89B  File : hp71.ep
                     + 00880   (Rel 0000C in obj/comp.ao)
                     + 00887   (Rel 00013 in obj/comp.ao)
                     + 00895   (Rel 00021 in obj/comp.ao)
                     + 008A9   (Rel 00035 in obj/comp.ao)
                     + 008B5   (Rel 0003F in obj/comp.ao)
```



```
                     + 008C1   (Rel 0004D in obj/comp.ao)
                     + 008E3   (Rel 0006F in obj/comp.ao)
                     + 008FB   (Rel 00087 in obj/comp.ao)
                     + 0090C   (Rel 00098 in obj/comp.ao)
                     + 00922   (Rel 000AE in obj/comp.ao)
                     + 00929   (Rel 000B5 in obj/comp.ao)
                     + 00953   (Rel 000DF in obj/comp.ao)
                     + 0095E   (Rel 000EA in obj/comp.ao)
                     + 009CD   (Rel 00159 in obj/comp.ao)
                     + 009DF   (Rel 0016B in obj/comp.ao)
                     + 00A41   (Rel 001CD in obj/comp.ao)
                     + 00A50   (Rel 001DC in obj/comp.ao)
                     + 00AB7   (Rel 00213 in obj/comp.ao)
                     + 00AA3   (Rel 0022F in obj/comp.ao)
                     + 00AB0   (Rel 0023C in obj/comp.ao)
                     + 00AC4   (Rel 00250 in obj/comp.ao)
                     + 00ACF   (Rel 0025B in obj/comp.ao)
                     + 00AD6   (Rel 00262 in obj/comp.ao)
                     + 00AF5   (Rel 00281 in obj/comp.ao)
                     + 00B14   (Rel 002A0 in obj/comp.ao)
                     + 00B1F   (Rel 002AB in obj/comp.ao)
                     + 00B39   (Rel 002C5 in obj/comp.ao)
                     + 00B48   (Rel 002D4 in obj/comp.ao)
                     + 00B58   (Rel 002E4 in obj/comp.ao)
                     + 00B7F   (Rel 00308 in obj/comp.ao)
                     + 00B90   (Rel 00319 in obj/comp.ao)
                     + 00BA0   (Rel 00329 in obj/comp.ao)
                     + 00BB5   (Rel 00341 in obj/comp.ao)
                     + 00BCB   (Rel 00357 in obj/comp.ao)
                     + 00BD6   (Rel 00362 in obj/comp.ao)
                     + 00BE9   (Rel 00375 in obj/comp.ao)
                     + 00BF4   (Rel 00380 in obj/comp.ao)
                     + 00C02   (Rel 0038E in obj/comp.ao)
                     + 00C10   (Rel 0039C in obj/comp.ao)
                     + 00C6A   (Rel 003F6 in obj/comp.ao)
                     + 00C88   (Rel 00414 in obj/comp.ao)
*FiLeNd     * 01189  File : eof.ep
                     + 00020   (Rel 00020 in obj/saisy.ao)
*GETB$I     * 004DB  File : obj/bas.ao
                     + 008EC   (Rel 00078 in obj/comp.ao)
                     + 00DDE   (Rel 00143 in obj/boucle.ao)
*GETD$I     * 004D0  File : obj/bas.ao
                     + 01048   (Rel 003AD in obj/boucle.ao)
*GETFST     * 0052D  File : obj/bas.ao
                     + 008A4   (Rel 00030 in obj/comp.ao)
                     + 00959   (Rel 000E5 in obj/comp.ao)
                     + 00DA0   (Rel 00105 in obj/boucle.ao)
                     + 00F92   (Rel 002F7 in obj/boucle.ao)
*GETI$I     * 004FC  File : obj/bas.ao
*GETM$I     * 004F1  File : obj/bas.ao
                     + 00932   (Rel 000BE in obj/comp.ao)
                     + 00CF3   (Rel 00058 in obj/boucle.ao)
                     + 00F84   (Rel 002E9 in obj/boucle.ao)
*GETP$I     * 004C5  File : obj/bas.ao
                     + 00890   (Rel 0001C in obj/comp.ao)
*GETU$I     * 004E6  File : obj/bas.ao
                     + 00915   (Rel 000A1 in obj/comp.ao)
                     + 00C0B   (Rel 00397 in obj/comp.ao)
                     + 010D2   (Rel 00437 in obj/boucle.ao)
*GETx$I     * 00503  File : obj/bas.ao
                     + 00D92   (Rel 000F7 in obj/boucle.ao)
*HDFLT      * 1B31B  File : hp71.ep
                     + 01158   (Rel 004BD in obj/boucle.ao)
*HEXDEC     * 0ECAF  File : hp71.ep
```



```
                     + 006D3   (Rel 002FC in obj/bas.ao)
*ILFART     * 00337  File : obj/pol.ao
                     + 00F62   (Rel 002C7 in obj/boucle.ao)
*INSINP     * 2F98B  File : obj/saisy.ao
                     + 001CE   (Rel 0004F in obj/pol.ao)
                     + 003C4   (Rel 00245 in obj/pol.ao)
                     + 00E26   (Rel 0018B in obj/boucle.ao)
                     + 00E49   (Rel 001AE in obj/boucle.ao)
*INSMSK     * 00001  File : obj/saisy.ao
                     + 001D9   (Rel 0005A in obj/pol.ao)
                     + 003C1   (Rel 00242 in obj/pol.ao)
                     + 00E31   (Rel 00196 in obj/boucle.ao)
                     + 00E43   (Rel 001A8 in obj/boucle.ao)
*IVPARe     * 02E3F  File : hp71.ep
                     + 0005B   (Rel 0005B in obj/saisy.ao)
*IVVARe     * 02E66  File : hp71.ep
                     + 000DE   (Rel 000DE in obj/saisy.ao)
*LDCSPC     * 2F6C1  File : hp71.ep
                     + 00D5E   (Rel 000C3 in obj/boucle.ao)
                     + 00D88   (Rel 000ED in obj/boucle.ao)
                     + 00DD4   (Rel 00139 in obj/boucle.ao)
                     + 00EF5   (Rel 0025A in obj/boucle.ao)
                     + 00F08   (Rel 0026D in obj/boucle.ao)
                     + 00F3B   (Rel 002A0 in obj/boucle.ao)
                     + 00F4D   (Rel 002B2 in obj/boucle.ao)
                     + 00F7A   (Rel 002DF in obj/boucle.ao)
                     + 01030   (Rel 00395 in obj/boucle.ao)
                     + 0103E   (Rel 003A3 in obj/boucle.ao)
                     + 010C8   (Rel 0042D in obj/boucle.ao)
*MEMERR     * 0944D  File : hp71.ep
                     + 007B3   (Rel 003DC in obj/bas.ao)
*MFERR      * 09393  File : hp71.ep
                     + 004C0   (Rel 000E9 in obj/bas.ao)
                     + 0094C   (Rel 000D8 in obj/comp.ao)
                     + 01102   (Rel 00467 in obj/boucle.ao)
*MOVE+M     * 01308  File : hp71.ep
                     + 003FF   (Rel 00028 in obj/bas.ao)
*MOVEU3     * 1B177  File : hp71.ep
                     + 010F7   (Rel 0045C in obj/boucle.ao)
*MPY        * 0ECBB  File : hp71.ep
                     + 007BA   (Rel 003E3 in obj/bas.ao)
*MSGTBL     * 00130  File : obj/msg.ao
                     + 00035   (Rel 00035 in obj/saisy.ao)
*MSPARe     * 02E5C  File : hp71.ep
                     + 00077   (Rel 00077 in obj/saisy.ao)
*MTHSTK     * 2F599  File : hp71.ep
                     + 00484   (Rel 000AD in obj/bas.ao)
*NOSCRL     * 14C8A  File : hp71.ep
                     + 0117D   (Rel 004E2 in obj/boucle.ao)
*NTOKEN     * 0493B  File : hp71.ep
                     + 00066   (Rel 00066 in obj/saisy.ao)
*NXTSTM     * 08A48  File : hp71.ep
                     + 01184   (Rel 004E9 in obj/boucle.ao)
*OBCOLL     * 01435  File : hp71.ep
                     + 00775   (Rel 0039E in obj/bas.ao)
*OUTBY+     * 02CE5  File : hp71.ep
                     + 000F7   (Rel 000F7 in obj/saisy.ao)
*OUTEL1     * 05300  File : hp71.ep
                     + 00124   (Rel 00124 in obj/saisy.ao)
*OUTNIB     * 02D28  File : hp71.ep
                     + 000A4   (Rel 000A4 in obj/saisy.ao)
*POLHMD     * 0017F  File : obj/pol.ao
                     + 00039   (Rel 00039 in obj/saisy.ao)
*POP1S      * 0BD38  File : hp71.ep
```

```
AREUH LINKER V2.0 - Page 005 - File: saisylex

                        + 00450   (Rel 00079 in obj/bas.ao)
                        + 010A3   (Rel 00408 in obj/boucle.ao)
*POPBUF         + 010EE  File : hp71.ep
                        + 00EB3   (Rel 00218 in obj/boucle.ao)
*R<RSTK         + 014DD  File : hp71.ep
                        + 0034F   (Rel 001D0 in obj/pol.ao)
*RDATTY         + 17CC6  File : hp71.ep
                        + 004B5   (Rel 000DE in obj/bas.ao)
*READF5         + 0323B  File : hp71.ep
                        + 00081   (Rel 00081 in obj/saisy.ao)
*RESPTR         + 03172  File : hp71.ep
                        + 000E5   (Rel 000E5 in obj/saisy.ao)
*RPTKY          + 152BA  File : hp71.ep
                        + 00E94   (Rel 001F9 in obj/boucle.ao)
*RSTK<R         + 014A8  File : hp71.ep
                        + 0037E   (Rel 001FF in obj/pol.ao)
*S-R1-2         + 2F88B  File : hp71.ep
                        + 0114C   (Rel 004B1 in obj/boucle.ao)
*SCRLLR         + 0212E  File : hp71.ep
                        + 00E9E   (Rel 00203 in obj/boucle.ao)
*STKCHR         + 18504  File : hp71.ep
                        + 006A1   (Rel 002CA in obj/bas.ao)
*STORE          + 0F5F8  File : hp71.ep
                        + 0116F   (Rel 004D4 in obj/boucle.ao)
*TBLJMC         + 02426  File : hp71.ep
                        + 00A33   (Rel 001BF in obj/comp.ao)
*TRFMBF         + 2F8C5  File : hp71.ep
                        + 006C2   (Rel 002EB in obj/bas.ao)
                        + 00727   (Rel 00350 in obj/bas.ao)
                        + 008DC   (Rel 00068 in obj/comp.ao)
                        + 00AEB   (Rel 00277 in obj/comp.ao)
                        + 0086F   (Rel 002FB in obj/comp.ao)
*WINDLN         + 2F473  File : hp71.ep
                        + 00308   (Rel 00189 in obj/pol.ao)
*eIDIM          + 00006  File : obj/msg.ao
                        + 004A8   (Rel 000D1 in obj/bas.ao)
*eIFMT          + 00005  File : obj/msg.ao
                        + 00A16   (Rel 001A2 in obj/comp.ao)
*eINOVR         + 00007  File : obj/msg.ao
                        + 00583   (Rel 001AC in obj/bas.ao)
*eIPRMP         + 00004  File : obj/msg.ao
                        + 00D12   (Rel 00077 in obj/boucle.ao)
*eSTROV         + 00025  File : hp71.ep
                        + 004BC   (Rel 000E5 in obj/bas.ao)
                        + 00948   (Rel 000D4 in obj/comp.ao)
                        + 010FE   (Rel 00463 in obj/boucle.ao)
*id             + 000E1  File : obj/saisy.ao
                        + 004A8   (Rel 000D1 in obj/bas.ao)
                        + 00583   (Rel 001AC in obj/bas.ao)
                        + 00A16   (Rel 001A2 in obj/comp.ao)
                        + 00D12   (Rel 00077 in obj/boucle.ao)
*kcFRT          + 00006  File : hp71.ep
                        + 001F3   (Rel 00074 in obj/pol.ao)
*kcRT           + 00009  File : hp71.ep
                        + 001EA   (Rel 0006B in obj/pol.ao)
*kcRUN          + 0000F  File : hp71.ep
                        + 001FC   (Rel 0007D in obj/pol.ao)
*n              + 2F8FA  File : obj/bas.ao
                        + 0048E   (Rel 000B7 in obj/bas.ao)
                        + 0065C   (Rel 00285 in obj/bas.ao)
                        + 00784   (Rel 003AD in obj/bas.ao)
                        + 007CA   (Rel 003F3 in obj/bas.ao)
                        + 00876   (Rel 00002 in obj/comp.ao)
                        + 00CE6   (Rel 0004B in obj/boucle.ao)

AREUH LINKER V2.0 - Page 006 - File: saisylex

                        + 00F1F   (Rel 00284 in obj/boucle.ao)
                        + 00F54   (Rel 002B9 in obj/boucle.ao)
                        + 00FEC   (Rel 00351 in obj/boucle.ao)
                        + 01005   (Rel 0036A in obj/boucle.ao)
*pCONFG         + 000FB  File : hp71.ep
                        + 0018A   (Rel 0000B in obj/pol.ao)
*pKYDF          + 0001B  File : hp71.ep
                        + 00181   (Rel 00002 in obj/pol.ao)
*tSINPUT        + 0005D  File : obj/saisy.ao

AREUH LINKER V2.0 - Page 007 - File: saisylex

Output module : saisylex

Source module : obj/saisy.ao
          Start + 00000, End + 0012F, Length + 00130

Source module : obj/msg.ao
          Start + 00130, End + 0017E, Length + 0004F

Source module : obj/pol.ao
          Start + 0017F, End + 003D6, Length + 00258

Source module : obj/bas.ao
          Start + 003D7, End + 00873, Length + 0049D

Source module : obj/comp.ao
          Start + 00874, End + 00C9A, Length + 00427

Source module : obj/boucle.ao
          Start + 00C9B, End + 01188, Length + 004EE


Date : Mon Mar 16 11:53:59 1987

Errors : 000

Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986  Paris, Fran
```

```
Page 001          FINPUT : MODULE PRINCIPAL (saisy.as)
AREUH ASS. V2.0

0001 00000                TITLE  FINPUT : MODULE PRINCIPAL (saisy.as)
0002 00000
0003 00000 35149435       NIBASC 'SAISYLEX' File Name
     00008 95C45485
0004 00010 802E           NIBHEX 802E     File Type * Lex
0005 00014 00             CON(2) 0        Copy code * 0
0006 00016 0000           CON(4) 0        Time * 0
0007 0001A 000000         CON(6) 0        Date * 0
0008 00020 00000          REL(5) *FiLeNd  End of File Chain
0009 00025
0010 00025          *id      EQU   #E1
0011 00025          *tSINPUT EQU   93
0012 00025
0013 00025          *INSINP EQU    #2F98B
0014 00025          *INSMSK EQU    %0001
0015 00025
0016 00025 1E             CON(2) *id      Id
0017 00027 D5             CON(2) *tSINPUT Lowest Token
0018 00029 D5             CON(2) *tSINPUT Highest Token
0019 0002B 00000          CON(5) 0        Next Lex table link
0020 00030
0021 00030 F              NIBHEX F        No speed table
0022 00031 7100           REL(4) 1+TxTbSt Text Table offset
0023 00035 0000           REL(4) *MSGTBL  Message Table offset
0024 00039 00000          REL(5) *POLHND  Poll handler offset
0025 0003E
0026 0003E 000            CON(3) (TxEn01)-(TxTbSt)
0027 00041 00000          REL(5) *FINPUTe
0028 00046 D              CON(1) #D
0029 00047          TxTbSt
0030 00047 B        TxEn01 CON(1) 11
0031 00048 6494E405       NIBASC 'FINPUT' 86/11/29 : SINPUT mue en FINPUT
     00050 5545
0032 00054 D5             CON(2) *tSINPUT
0033 00056 1FF            NIBHEX 1FF
0034 00059
```

```
Page 002          FINPUT : MODULE PRINCIPAL (saisy.as)
AREUH ASS. V2.0   PARSE ROUTINE

0036 00059          *
0037 00059          * Syntaxe: SAISY I$,M$, [P$,] A
0038 00059          *
0039 00059
0040 00059 8D00000  ivpare GOVLNG *IVPARe
0041 00060
0042 00060 7F70     comma  GOSUB  resptr
0043 00064 8F00000         GOSBVL *NTOKEN
0044 0006B 8F00000         GOSBVL *COMCK*  check & output tCOMMA
0045 00072 400             RTNC            Ok
0046 00075 8D00000  mspare GOVLNG *MSPARe
0047 0007C
0048 0007C          *FINPUTp
0049 0007C          *       ST=0   8        Dummy arrays are valid
0050 0007C 859              ST=1   9        Single string variable parse
0051 0007F 8F00000          GOSBVL *READP5  supporte (mais oui !)
0052 00086 76DF             GOSUB  comma
0053 00088          * M$
0054 0008A 7C50             GOSUB  exppar   M$
0055 0008E 873              ?ST=1  3
0056 00091 8C               GOYES  ivpare
0057 00093 79CF             GOSUB  comma
0058 00097          * P$ ou A
0059 00097 136              CD0EX           ;
0060 0009A 10A              R2=C            ; Quartet de reconnaissance := 0
0061 0009D 134              D0=C            ;   si il n'y a pas P$
0062 000A0 D2               C=0    A        ;
0063 000A2 8F00000          GOSBVL *OUTNIB  ;
0064 000A9 7D30             GOSUB  exppar
0065 000AD 873              ?ST=1  3        Valid String expression
0066 000B0 D1               GOYES  SINp10
0067 000B2 11A              C=R2            ;
0068 000B5 136              CD0EX           ; Quartet de reconnaissance := F
0069 000B8 AC0              A=0    S        ;   si il y a P$
0070 000BB A4C              A=A-1  S        ;
0071 000BE 1504             DAT0=A S        ;
0072 000C2 134              D0=C            ;
0073 000C5 779F             GOSUB  comma
0074 000C9          * A
0075 000C9 7D10             GOSUB  exppar
0076 000CD 870      SINp10  ?ST=1  0        No valid expression ?
0077 000D0 C0               GOYES  ivvare
0078 000D2 863              ?ST=0  3        Valid string expression ?
0079 000D5 70               GOYES  ivvare   Invalid var
0080 000D7 831              ?XM=0           Expression + passable par reference
0081 000DA 90               GOYES  resptr
0082 000DC 8D00000  ivvare GOVLNG *IVVARe
0083 000E3 8D00000  resptr GOVLNG *RESPTR
0084 000EA
0085 000EA 8D00000  exppar GOVLNG *EXPPAR
0086 000F1
```

```
Page 003          FINPUT : MODULE PRINCIPAL (saisy.as)
AREUH ASS. V2.0   DECOMPILE ROUTINE

0088 000F1
0089 000F1 31C2      virgud LCASC  ','
0090 000F5 8D00000          GOVLNG *OUTBY*
0091 000FC
0092 000FC          *FINPUTd
0093 000FC 7920             GOSUB  exprdc
0094 00100 7DEF             GOSUB  virgud
0095 00104 7120             GOSUB  exprdc
0096 00108 75EF             GOSUB  virgud
0097 0010C 170              D1=D1+ 1
0098 0010F 7610             GOSUB  exprdc
0099 00113 8F00000          GOSBVL *EOLXC*
0100 0011A 73DF             GOSUB  virgud
0101 0011E 7700             GOSUB  exprdc
0102 00122 8D00000          GOVLNG *OUTEL1
0103 00129
0104 00129 8D00000  exprdc GOVLNG *EXPRDC
0105 00130
0106 00130                  END
```

```
Page 004          FINPUT : MODULE PRINCIPAL (saisy.as)
AREUH ASS. V2.0   **** SYMBOL TABLE ****

*COMCK*      Extrn Ukn       -  0044
*EOLXC*      Extrn Ukn       -  0099
*EXPPAR      Extrn Ukn       -  0085
*EXPRDC      Extrn Ukn       -  0104
*FINPUTd     000FC Rel  0092 -
*FINPUTe     Extrn Ukn       -  0027
*FINPUTp     0007C Rel  0048 -
*FiLeNd      Extrn Ukn       -  0008
 FiLeNd      00130 Rel  0106 -
*INSINP      2F98B Abs  0013 -
*INSMSK      00001 Abs  0014 -
*IVPARe      Extrn Ukn       -  0040
*IVVARe      Extrn Ukn       -  0082
*MSGTBL      Extrn Ukn       -  0023
*MSPARe      Extrn Ukn       -  0046
*NTOKEN      Extrn Ukn       -  0043
*OUTBY*      Extrn Ukn       -  0090
*OUTEL1      Extrn Ukn       -  0102
*OUTNIB      Extrn Ukn       -  0063
*POLHND      Extrn Ukn       -  0024
*READP5      Extrn Ukn       -  0051
*RESPTR      Extrn Ukn       -  0083
 SINp10      000CD Rel  0076 -  0066
 TxEn01      00047 Rel  0030 -  0026
 TxTbSt      00047 Rel  0029 -  0022 0026
 comma       00060 Rel  0042 -  0052 0057 0073
 exppar      000EA Rel  0085 -  0054 0064 0075
 exprdc      00129 Rel  0104 -  0093 0095 0098 0101
*id          000E1 Abs  0010 -  0016
 ivpare      00059 Rel  0040 -  0056
 ivvare      000DC Rel  0082 -  0077 0079
 mspare      00075 Rel  0046 -
 resptr      000E3 Rel  0083 -  0042 0081
*tSINPUT     0005D Abs  0011 -  0017 0018 0032
 virgud      000F1 Rel  0089 -  0094 0096 0100



Source  : saisy.as

Object  : obj/saisy.ao

Listing : list/saisy.al

Date    : Mon Mar 16 11:51:36 1987

Errors  : 000


Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986 Paris, France
```

```
Page 001            FINPUT : MESSAGE TABLE (msg.as)
AREUH ASS. V2.0

0001 00000                  TITLE  FINPUT : MESSAGE TABLE (msg.as)
0002 00000
0003 00000           MBASE  EQU    4
0004 00000          *eIPRMP EQU    (MBASE)+0    Invalid Prompt
0005 00000          *eIFMT  EQU    (MBASE)+1    Invalid Format
0006 00000          *eIDIM  EQU    (MBASE)+2    #Dims
0007 00000          *eINOVR EQU    (MBASE)+3    Var Not Found
0008 00000
0009 00000          *MSGTBL
0010 00000 40               CON(2) 4            Lowest message #
0011 00002 70               CON(2) 7            Highest message #
0012 00004
0013 00004          * #Dims
0014 00004 01               CON(2) 16
0015 00006 60               CON(2) *eIDIM   Message # 2
0016 00008 4                CON(1) 4
0017 00009 324496D6         NIBASC '#Dims'
     00011 37
0018 00013 C                CON(1) 12
0019 00014
0020 00014          * Invalid Prompt
0021 00014 51               CON(2) 21
0022 00016 40               CON(2) *eIPRMP  Message # 0
0023 00018 E                CON(1) 14
0024 00019 CE               CON(2) 236
0025 0001B 5                CON(1) 5
0026 0001C 0527F6D6         NIBASC 'Prompt'
     00024 0747
0027 00028 C                CON(1) 12
0028 00029
0029 00029          * Invalid Format
0030 00029 51               CON(2) 21
0031 0002B 50               CON(2) *eIFMT   Message # 1
0032 0002D E                CON(1) 14
0033 0002E CE               CON(2) 236
0034 00030 5                CON(1) 5
0035 00031 64F627D6         NIBASC 'Format'
     00039 1647
0036 0003D C                CON(1) 12
0037 0003E
0038 0003E          * Var Not Found
0039 0003E F0               CON(2) 15
0040 00040 70               CON(2) *eINOVR  Message # 3
0041 00042 2                CON(1) 2
0042 00043 651627           NIBASC 'Var'
0043 00049 E                CON(1) 14
0044 0004A 8E               CON(2) 232
0045 0004C C                CON(1) 12
0046 0004D
0047 0004D FF               NIBHEX FF       Table terminator
0048 0004F
0049 0004F                  END
```

```
Page 002            FINPUT : MESSAGE TABLE (msg.as)
AREUH ASS. V2.0     **** SYMBOL TABLE ****

 FiLeNd        0004F Rel   0049 -
*MSGTBL        00000 Rel   0009 -
 MBASE         00004 Abs   0003 -   0004 0005 0006 0007
*eIDIM         00006 Abs   0006 -   0015
*eIFMT         00005 Abs   0005 -   0031
*eINOVR        00007 Abs   0007 -   0040
*eIPRMP        00004 Abs   0004 -   0022



Source  : msg.as

Object  : obj/msg.ao

Listing : list/msg.al

Date    : Mon Mar 16 11:51:45 1987

Errors  : 000


Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986 Paris, France
```

```
Page 001            FINPUT : POLL HANDLER (pol.as)
AREUH ASS. V2.0

0001 00000                  TITLE  FINPUT : POLL HANDLER (pol.as)
0002 00000
0003 00000           *
0004 00000           * Bits d'etat du display driver du HP71.
0005 00000           *
0006 00000            BitsOK EQU    1
0007 00000            NoChFC EQU    11
0008 00000
0009 00000            EscSt0 EQU    0
0010 00000            EscSt1 EQU    1
0011 00000
0012 00000           *****************************************************
0013 00000           * POLHND
0014 00000           *
0015 00000           * But: traiter les polls arrivant dans le Lex.
0016 00000           * Entree:
0017 00000           *    - B(B) = numero du poll
0018 00000           *    - Mode = Hex
0019 00000           *    - P = 0
0020 00000           *    - si fast poll : D(A) a preserver
0021 00000           * Sortie:
0022 00000           *    - voir les poll
0023 00000           * Detail:
0024 00000           *    Polls interceptes : .pVER, pKYDF, pCONFG
0025 00000           * Historique:
0026 00000           *    86/11/16: P.D. & J.T. conception & codage
0027 00000           *    86/11/24: P.D.        ajout de documentation
0028 00000           *    86/12/02: P.D. & J.T. pCONFG pour mettre a 0 le flag
0029 00000           *****************************************************
0030 00000
0031 00000           *POLHND
0032 00000 3100             LC(2)  =pKYDF
0033 00004 961              ?B=C   B
0034 00007 64               GOYES  hKYDEF
0035 00009 3100             LC(2)  =pCONFG
0036 0000D 961              ?b=c   b
0037 00010 90               goyes  hCONFG
0038 00012 969              ?b=0   b
0039 00015 80               goyes  hVER$0
0040 00017 00        rtnsxm RTNSXM
0041 00019
0042 00019 6622      hCONFG GOTO   hCFG00
0043 0001D
0044 0001D           *****************************************************
0045 0001D           * hVER$0
0046 0001D           *
0047 0001D           * But: renvoyer le numero de version de SAISYLEX
0048 0001D           * Entree: -
0049 0001D           * Sortie: -
0050 0001D           * Detail:
0051 0001D           *    Pompe sur EDLEX (merci HP !)
0052 0001D           *    Mis la a la demande explicite (mais ferme) de J.T.,
0053 0001D           *    pour lui eviter de se melanger les pinceaux...
0054 0001D           *    Le mauvais francais du commentaire qu'est au dessus est
0055 0001D           *    d'P.D. (vive moi !).
0056 0001D           * Historique:
0057 0001D           *    86/11/29: P.D. & J.T. conception & pompage
0058 0001D           *    87/01/25: P.D. & J.T. V3: Voir PP$010 (bas.as)
0059 0001D           *    87/02/17: P.D.        FINP:A
0060 0001D           *****************************************************
0061 0001D
0062 0001D 11B        hVER$0 c=r3
0063 00020 135              d1=c
```

```
Page 002          FINPUT : POLL HANDLER (pol.as)
AREUH ASS. V2.0

0064 00023 112              a=r2
0065 00026 1CD              d1=d1- (VER$en)-(VER$st)-2
0066 00029 137              cd1ex
0067 0002C 8B6              ?A>c    a
0068 0002F C1               goyes   hVER$1
0069 00031 135              d1=c
0070 00034 10B              r3=c
0071 00037 3D14A305  VER$st lcasc   \ FINP:A\
     0003F E4946402
0072 00047 15DD      VER$en dat1=c (VER$en)-(VER$st)-2
0073 0004B 00        hVER$1 rtnsxm
0074 0004D
0075 0004D          ********************************************************
0076 0004D          * hKDYDF
0077 0004D          *
0078 0004D          * But: traiter le poll pKYDF (definition de touche)
0079 0004D          * Entree:
0080 0004D          *    R0(A) = code logique de la touche
0081 0004D          * Sortie:
0082 0004D          *   2 cas possibles :
0083 0004D          *     non interception
0084 0004D          *      - on n'est pas dans SINPUT
0085 0004D          *      - la touche n'est ni [->], ni [g][->]
0086 0004D          *      - la touche est [->], mais on l'autorise (pas arrive
0087 0004D          *        au bout de l'affichage)
0088 0004D          *      - la touche est [g][->], mais c'est un cas special
0089 0004D          *     interception et action (sans definition)
0090 0004D          *      - la touche est [RUN]
0091 0004D          *      - la touche est [->], et on est arrive a la fin de
0092 0004D          *        l'affichage
0093 0004D          *      - la touche est [g][->], c'est nous qui manipulons le
0094 0004D          *        curseur pour l'amener a la fin de l'affichage.
0095 0004D          * Abime: A=C, D0, D1, ST
0096 0004D          * Niveaux: 0
0097 0004D          * Historique:
0098 0004D          *   86/11/16: P.D. & J.T. conception & codage
0099 0004D          *   86/11/24: P.D.        ajout de documentation
0100 0004D          *   86/12/02: P.D. & J.T. ajout de [RUN]
0101 0004D          ********************************************************
0102 0004D
0103 0004D           hKYDEF
0104 0004D 1B00000          D0=(5) =INSINP
0105 00054 15A0             A=DAT0 1
0106 00058
0107 00058 300              LC(1)  =INSMSK
0108 0005B 0ED6             A=A&C  P
0109 0005F 908              ?A=0   P
0110 00062 5B               GOYES  rtnsxm
0111 00064 110              A=R0
0112 00067 D2               C=0    A
0113 00069 3100             LC(2)  =kcRT
0114 0006D 8A2              ?A=C   A
0115 00070 E1               GOYES  hKD010
0116 00072 3100             LC(2)  =kcFRT
0117 00076 8A2              ?A=C   A
0118 00079 D0               GOYES  hKD000
0119 0007B 3100             LC(2)  =kcRUN
0120 0007F 8A2              ?A=C   A
0121 00082 80               GOYES  hKD001
0122 00084 00        RTnsxm RTNSXM
0123 00086
0124 00086 6790      hKD000 GOTO   hKD050   Touche [g][->]
0125 0008A 6191      hKD001 GOTO   hKD100
```

```
Page 003          FINPUT : POLL HANDLER (pol.as)
AREUH ASS. V2.0

0126 0008E
```

```
Page 004          FINPUT : POLL HANDLER (pol.as)
AREUH ASS. V2.0   TOUCHE [->]

0128 0008E          ********************************************************
0129 0008E          * hKD010
0130 0008E          *
0131 0008E          * But: traiter le deplacement a droite (touche [->]) dans
0132 0008E          *   SINPUT.
0133 0008E          * Entree: -
0134 0008E          * Sortie:
0135 0008E          *    - interception de la touche signifie que nous
0136 0008E          *      n'autorisons pas le deplacement a droite.
0137 0008E          * Abime: A=C, D0, D1, ST
0138 0008E          * Niveaux: 0
0139 0008E          * Historique:
0140 0008E          *   86/11/16: P.D. & J.T. conception & codage
0141 0008E          *   86/11/24: P.D.        ajout de documentation
0142 0008E          ********************************************************
0143 0008E
0144 0008E           hKD010
0145 0008E          *
0146 0008E          * Recherche s'il existe un caractere non protege dans le
0147 0008E          * reste du display buffer
0148 0008E          *
0149 0008E          * for i := CURSOR + 1 to 96 do
0150 0008E          *   if dspmask[i] = unprotected then exit ;
0151 0008E          *
0152 0008E 1B00000          D0=(5) =CURSOR
0153 00095 AF0              A=0    W
0154 00098 14A              A=DAT0 B        A(A) := (CURSOR)
0155 0009B E4               A=A+1  A        A(A) := next char.
0156 0009D D8               B=A    A        B(A) := next char.
0157 0009F 81C              ASRB
0158 000A2 81C              ASRB            A(B) := (next char) / 4
0159 000A5 3400000          LC(5)  (=DSPMSK)*23
0160 000AC E2               C=C-A  A
0161 000AE 134              D0=C            D0 := ^ dspmsk [next char]
0162 000B1          *
0163 000B1          * Il est bien evident que la boucle reelle ne sera pas une
0164 000B1          * boucle caractere par caractere. La boucle va traiter un
0165 000B1          * quartet de DSPMSK a la fois, c'est a dire quatre
0166 000B1          * caracteres. Il y a donc un cas particulier a tester au
0167 000B1          * prealable : le cas ou il n'y a pas exactement un multiple
0168 000B1          * de 4.
0169 000B1          *
0170 000B1 D4               A=B    A        A(A) := next char.
0171 000B3 31F5             LC(2)  95
0172 000B7 B62              C=C-A  B        C(B) := nb de char. restant
0173 000BA 49C              GOC    RTnsxm   curseur a la fin
0174 000BD AEA              A=C    B
0175 000C0 AE8              B=A    B        B(B) := sauvegarde du nb de char.
0176 000C3 3130             LC(2)  %0011
0177 000C7 0E66             A=A&C  B        A(B) := X * nb de char mod 4
0178 000CB 962              ?A=C   B
0179 000CE A2               GOYES  reste
0180 000D0 21               P=     %0001
0181 000D2 A6C              A=A-1  B
0182 000D5 4C0              GOC    msk
0183 000D8 23               P=     %0011
0184 000DA A6C              A=A-1  B
0185 000DD 440              GOC    msk
0186 000E0 27               P=     %0111
0187 000E2 80CF      msk    C=P    15
0188 000E6 20               P=     0
0189 000E8 1524             A=DAT0 S
0190 000EC 0E46             A=A&C  S
```

```
Page 005          FINPUT : POLL HANDLER (pol.as)
AREUH ASS. V2.0   TOUCHE [->]

0191 000F0 946              ?A#C   S
0192 000F3 92               GOYES  Rtnsxm
0193 000F5          *
0194 000F5          * Maintenant, le cas particulier a ete teste, il ne reste
0195 000F5          * qu'a tester que tous les autres quartets de DSPMSK ont
0196 000F5          * comme valeur #F.
0197 000F5          *
0198 000F5 180              D0=D0- 1
0199 000F8           reste
0200 000F8 81D              BSRB
0201 000FB 81D              BSRB            B(B) := (nb de car.) / 4
0202 000FE 5F0              GONC   rst20    B.E.T.
0203 00101 1564      rst10  C=DAT0 S
0204 00105 846              C=C+1  S
0205 00108 531              GONC   Rtnsxm
0206 0010B 180              D0=D0- 1
0207 0010E A6D       rst20  B=B-1  B
0208 00111 5FE              GONC   rst10
0209 00114
0210 00114          ********************************************************
0211 00114          * RIEN
0212 00114          *
0213 00114          * But: ne rien faire (ahhhhhhhhhhh)
0214 00114          * Entree: -
0215 00114          * Sortie: Par RTNSXM, a CHEDIT, en declarant qu'on a egit
0216 00114          *   et qu'il n'y a pas besoin que le systeme continue.
0217 00114          * Historique:
0218 00114          *   86/12/02: P.D. & J.T. conception & codage
0219 00114          ********************************************************
0220 00114 840       RIEN   ST=0   0        inhiber la touche
0221 00117 821              XM=0
0222 0011A 01               RTN
0223 0011C
0224 0011C 00        Rtnsxm RTNSXM
```

```
0226 0011E          ********************************************************
0227 0011E          * hKD050
0228 0011E          *
0229 0011E          * But: traiter la touche [g][·>] quand on est dans SINPUT.
0230 0011E          * Entree: -
0231 0011E          * Sortie: -
0232 0011E          *   La touche est interceptee, c'est nous qu'on a traite
0233 0011E          *   l'appui sur la touche, et c'est pas HP. Na !
0234 0011E          * Detail:
0235 0011E          *   2 cas :
0236 0011E          *     - Le dernier caractere du display buffer est protege.
0237 0011E          *       Dans ce cas, [g][·>] nous amene sur le dernier
0238 0011E          *       caractere du dernier champ non protege.
0239 0011E          *     - le dernier caractere du display n'est pas protege.
0240 0011E          *       Alors, [g][·>] nous amene apres le dernier
0241 0011E          *       caractere introduit si c'est possible (c'est a dire
0242 0011E          *       que le display buffer n'est pas plein).
0243 0011E          * Historique:
0244 0011E          *   86/11/21: P.D. & J.T. conception & codage
0245 0011E          *   86/11/24: P.D.        ajout de documentation
0246 0011E          *   86/11/29: P.D. & J.T. gestion de la video HPIL
0247 0011E          ********************************************************
0248 0011E
0249 0011E           hKD050
0250 0011E          *
0251 0011E          * Modification du 86/11/29 :
0252 0011E          *
0253 0011E          * Ce poll est destine a court-circuiter le display-driver
0254 0011E          * (de m...) du HP71. Quand la touche [g][·>] est detectee,
0255 0011E          * c'est nous qui choisissons la position d'arrivee du
0256 0011E          * curseur. Le probleme est de faire comprendre ca au module
0257 0011E          * HPIL.
0258 0011E          *
0259 0011E          * Ce driver se comporte d'une maniere plus propre que celui
0260 0011E          * interne au HP71. Il sait jusqu'ou ne pas aller quand la
0261 0011E          * sequence ESC CTRL C lui arrive. Il n'y a donc pas besoin
0262 0011E          * de l'empecher d'agir. Mais ceci suppose de l'activer
0263 0011E          * explicitement. Il n'est plus appele par le driver du HP71
0264 0011E          * mais par nous.
0265 0011E          *
0266 0011E
0267 0011E          *
0268 0011E          * Comme dans le display-driver du HP71, nous appelons
0269 0011E          * d'abord l'HPIL :
0270 0011E          *
0271 0011E 7690              gosub  =ILFART
0272 00122
0273 00122          *
0274 00122          * Puis, nous gerons le LCD.
0275 00122          *
0276 00122          * Recherche dans DSPMSK pour avoir la position du premier
0277 00122          * caractere non protege a partir de la fin.
0278 00122          *
0279 00122          * j := 0 ;
0280 00122          * i := 95 ;
0281 00122          * while (i >= 0) and
0282 00122          *       ((dspmsk[i] = protected) or (display [i] = 0))
0283 00122          * do
0284 00122          *   begin
0285 00122          *     if dspmsk[i] # protected then j := i ;
0286 00122          *     i := i-1 ;
0287 00122          *   end ;
0288 00122          * if i<0 then exit poll ;
```



```
0289 00122          * if j#0 then curseur := j
0290 00122          *        else curseur := i ;
0291 00122          *
0292 00122 31F5              LC(2)  95
0293 00126 AE5               B=C    B        B(B) := i
0294 00129 1F00000           D1=(5) (=DSPBFE)-2    caractere courant
0295 0012D AF0               A=0    W        A(S) := 0 : masque courant := 0
0296 00133                   =      W        A(B) := 0 : j := 0
0297 00133 1B00000           D0=(5) (=DSPMSK)
0298 0013A 6F20              GOTO   hKD070
0299 0013E
0300 0013E          *
0301 0013E          * Dans la boucle, les assertions suivantes sont vraies :
0302 0013E          *
0303 0013E          *   A(S) = masque courant (mis a jour a la fin)
0304 0013E          *   A(B) = j
0305 0013E          *   B(B) = i
0306 0013E          *   C(S) = DSPMSK [IP(i/4)]
0307 0013E          *   D0 = ^ DSPMSK [IP(i/4)]
0308 0013E          *   D1 = ^ DSPBFS [i]
0309 0013E          *
0310 0013E           hKD060
0311 0013E          *
0312 0013E          * Obtention d'un nouveau DSPMSK si necessaire, ainsi qu'une
0313 0013E          * regeneration du masque courant.
0314 0013E          *
0315 0013E 94C               ?A#0   S
0316 00141 C0                GOYES  hKD062
0317 00143 B44               A=A+1  S
0318 00146 1564              C=DAT0 S
0319 0014A 160               D0=D0- 1
0320 0014D           hKD062
0321 0014D          *
0322 0014D          * Test de la protection
0323 0014D          *
0324 0014D AC5               B=C    S        On abime B(S)
0325 00150 3E44              B=B&A  S        B(S) := si dspmsk[i] = protected
0326 00154                   =               alors #0
0327 00154                   =               sinon 0
0328 00154 94D               ?B#0   S
0329 00157 D0                GOYES  hKD065   Pas la peine de continuer pour la
0330 00159                   =               position i si le caractere est
0331 00159                   =               protege.
0332 00159
0333 00159          *
0334 00159          * La position i n'est pas protegee.
0335 00159          *
0336 00159 14F               C=DAT1 B
0337 0015C 96E               ?C#0   B        il y a un caractere
0338 0015F 31                GOYES  hKD080
0339 00161
0340 00161 AE4               A=B    B        j := i ;
0341 00164
0342 00164 1C1       hKD065  D1=D1- 2
0343 00167 A44               A=A+A  S        masque mis a jour
0344 0016A           hKD070
0345 0016A A6D               B=B-1  B
0346 0016D 50D               GONC   hKD060
0347 00170          *
0348 00170          * On est sorti de la boucle sans avoir rien trouve. Tout le
0349 00170          * display est donc protege, ou aucun caractere n'a encore
0350 00170          * ete introduit.
0351 00170          *
```



```
0352 00170 00                RTNSXM
0353 00172
0354 00172          *
0355 00172          * On est sorti du WHILE.
0356 00172          *
0357 00172           hKD080
0358 00172          *
0359 00172          * if j#0 then j := i ;
0360 00172          *
0361 00172 96C               ?A#0   B
0362 00175 50                GOYES  hKD090
0363 00177
0364 00177 AE4               A=B    B        j := i
0365 0017A          *
0366 0017A          * curseur := j
0367 0017A          *
0368 0017A           hKD090
0369 0017A 1B00000           D0=(5) =CURSOR
0370 00181 B64               A=A+1  B        Programmation heuristique !
0371 00184 148               DAT0=A B
0372 00187          *
0373 00187          * Calcul de FIRSTC (premier caractere a afficher sur la
0374 00187          * fenetre de 22 caracteres du HP71) :
0375 00187          *
0376 00187          * FIRSTC := CURSOR - WINDLN si c'est possible
0377 00187          *
0378 00187 1900              D0=(2) =WINDLN
0379 0018B 14E               C=DAT0 B
0380 0018E B6E               C=A-C  B        C(B) := CURSOR - WINDLN
0381 00191 550               GONC   FST#0
0382 00194 AE2               C=0    B
0383 00197 1900      FST#0   D0=(2) =FIRSTC
0384 0019B 14C               DAT0=C B
0385 0019E
0386 0019E          *
0387 0019E          * Les instructions suivantes (modifications sur ST) sont
0388 0019E          * des ordres donnes au display driver :
0389 0019E          *
0390 0019E 1900              D0=(2) (=DSPSTA)+3
0391 001A2 1563              C=DAT0 X
0392 001A6 0A                ST=C
0393 001A8
0394 001A8 84B               ST=0   NoChFC   On a change FIRSTC
0395 001AB 841               ST=0   BitsOK   Il faut reconstruire le display
0396 001AE 09                C=ST
0397 001B0 1543              DAT0=C X        Dans les 12 bits du display driver
0398 001B4
0399 001B4 6F5F              GOTO   RIEN     interception
0400 001B8
0401 001B8          ********************************************************
0402 001B8          * ILFART
0403 001B8          *
0404 001B8          * But: Envoyer le curseur a l'extreme droite sur la video
0405 001B8          *   HPIL courante.
0406 001B8          * Entree: -
0407 001B8          * Sortie:
0408 001B8          * Abime: A-D, D0, D1, R3, ST
0409 001B8          * Appelle: BDISPJ (HPIL display driver), R=RSTK, RSTK=R
0410 001B8          * Niveaux: 0 (5 sauves par R=RSTK)
0411 001B8          * Detail: D est sauvegarde dans R3
0412 001B8          * Historique:
0413 001B8          *   86/11/29: P.D. & J.T. conception & codage
0414 001B8          ********************************************************
```



```
0415 001B8
0416 001B8           =ILFART
0417 001B8 1B00000           d0=(5) =DSPCHX
0418 001BF 142               a=dat0 a
0419 001C2 8A8               ?a#0   a
0420 001C5 00                rtnyes
0421 001C7
0422 001C7 DB                c=d    a
0423 001C9 10B               r3=c
0424 001CC
0425 001CC 24                p=     4        Sauvegarde 5 niveaux de pile
0426 001CE 8F00000           gosbvl =R=RSTK
0427 001D5
0428 001D5 1B00000           d0=(5) =ESCSTA
0429 001DC 301               lc(1)  EscSt1
0430 001DF 15C0              dat0=c 1
0431 001E3
0432 001E3 3130              lc(2)  3        CTRL C
0433 001E7 7020              gosub  ILSEND
0434 001EB 20                p=     0
0435 001ED
0436 001ED 1B00000           d0=(5) =ESCSTA
0437 001F4 300               lc(1)  EscSt0
0438 001F7 15C0              dat0=c 1
0439 001FB
0440 001FB 24                p=     4        Restaure les niveaux de pile
0441 001FD 8F00000           gosbvl =RSTK=R
0442 00204
0443 00204 11B               c=r3
0444 00207 D7                d=c    a
0445 00209 01                rtn
0446 0020B
0447 0020B          ********************************************************
0448 0020B          * ILSEND
0449 0020B          *
0450 0020B          * But: appeler directement le display-driver du module HPIL
0451 0020B          * Entree:
0452 0020B          *   - C(B) = caractere a afficher
0453 0020B          * Sortie: -
0454 0020B          * Abime: A-D, D0, D1, ST
0455 0020B          * Appelle: BDISPJ (#F3637 dans l'HPIL:1B)
0456 0020B          * Niveaux: 5
0457 0020B          * Detail:
0458 0020B          *   le controle est rendu a la routine appelante directement
0459 0020B          * Historique:
0460 0020B          *   86/11/29: P.D. & J.T. conception & codage
0461 0020B          ********************************************************
0462 0020B
0463 0020B AEA        ILSEND a=c    b        DATA BYTE
0464 0020E 1B00000           d0=(5) =DSPCHX
0465 00215 146               c=dat0 a
0466 00218 06                rstk=c
0467 0021A 01                rtn             GOVLNG =BDSIPJ dans l'HPIL
0468 0021C
0469 0021C          ********************************************************
0470 0021C          * hKD100
0471 0021C          *
0472 0021C          * But: traiter la touche [RUN] quand on est dans SINPUT.
0473 0021C          * Entree: -
0474 0021C          * Sortie: -
0475 0021C          *   La touche est interceptee, pour inhiber la definition
0476 0021C          *   eventuelle de la touche [RUN]. Chez moi, j'ai toujours
0477 0021C          *   un DEF KEY '#46', 'RUN ';
```

```
Page 010          FINPUT : POLL HANDLER <pol.as>
AREUH ASS. V2.0   TOUCHE [g][->]

0478 0021C          *   Pas vous ?
0479 0021C          * Appelle: kRUN (tombe dedans)
0480 0021C          * Historique:
0481 0021C          *   86/12/02: P.D. & J.T. conception & codage
0482 0021C          ************************************************
0483 0021C
0484 0021C 1F00000   hKD100 d1*(5) (*DEFADR)*2
0485 00223 AC2              c*0    &
0486 00226 1554             dat1*c &
0487 0022A 170              d1*d1* 1
0488 0022D 7200             gosub  hKD110
0489 00231 F0               con(2) 15       RUN
0490 00233 07        hKD110 c*rstk
0491 00235 145              dat1*c a
0492 00238 850              st*1   0
0493 0023B 821              xm*0
0494 0023E 01               rtn
0495 00240
0496 00240          ************************************************
0497 00240          * hCFG00
0498 00240          *
0499 00240          * But: traiter le poll de configuration
0500 00240          * Entree: -
0501 00240          * Sortie: -
0502 00240          * Detail:
0503 00240          *   Ce poll est la pour traiter le cas (peu frequent) ou
0504 00240          *   l'utilisateur fait INIT quand il est sous FINPUT. Si
0505 00240          *   l'on ne fait rien, les touches de curseur ([g][->] et
0506 00240          *   [->]), ainsi que la touche [RUN] sont toujours
0507 00240          *   interceptees. Lorsqu'une configuration intervient, on
0508 00240          *   est sur que FINPUT n'est pas activee. On peut donc
0509 00240          *   mettre le flag a 0 sans autre precaution.
0510 00240          * Historique:
0511 00240          *   86/12/02: P.D. & J.T. conception & codage
0512 00240          *   86/12/02: P.D.        ajout de documentation
0513 00240          ************************************************
0514 00240
0515 00240 300       hCFG00 lc(1)  `*INSMSK complement a un de *INSMSK
0516 00243 1B00000          d0*(5) *INSINP
0517 0024A 15A0             a*dat0 1
0518 0024E 0E06             a*a6c  p
0519 00252 15C0             dat0*c 1
0520 00256 00               rtnsxm
0521 00258
0522 00258                  end
```

```
Page 011          FINPUT : POLL HANDLER <pol.as>
AREUH ASS. V2.0   **** SYMBOL TABLE ****

 BitsOK       00001 Abs  0006 -  0395
*CURSOR       Extrn Ukn       -  0152 0369
*DEFADR       Extrn Ukn       -  0484
*DSPBFE       Extrn Ukn       -  0294
*DSPCHX       Extrn Ukn       -  0417 0464
*DSPMSK       Extrn Ukn       -  0159 0297
*DSPSTA       Extrn Ukn       -  0390
*ESCSTA       Extrn Ukn       -  0428 0436
 EscSt0       00000 Abs  0009 -  0437
 EscSt1       00001 Abs  0010 -  0429
*FIRSTC       Extrn Ukn       -  0383
 FST#0        00197 Rel  0383 -  0381
 FiLeNd       00258 Rel  0522 -
*ILFART       001B8 Rel  0416 -  0271
*INSINP       Extrn Ukn       -  0104 0516
*INSMSK       Extrn Ukn       -  0107 0515
 ILSEND       0020B Rel  0463 -  0433
 NoChFC       0000B Abs  0007 -  0394
*POLHND       00000 Rel  0031 -
*R<RSTK       Extrn Ukn       -  0426
*RSTK<R       Extrn Ukn       -  0441
 RIEN         00114 Rel  0220 -  0399
 RTnsxm       00084 Rel  0122 -  0173
 Rtnsxm       0011C Rel  0224 -  0192 0205
 VERsen       00047 Rel  0072 -  0065 0072
 VERsst       00037 Rel  0071 -  0065 0072
*WINDLN       Extrn Ukn       -  0378
 hCFG00       00240 Rel  0515 -  0042
 hCONFG       00019 Rel  0042 -  0037
 hKD000       00086 Rel  0124 -  0118
 hKD001       0008A Rel  0125 -  0121
 hKD010       0008E Rel  0144 -  0115
 hKD050       0011E Rel  0249 -  0124
 hKD060       0013E Rel  0310 -  0346
 hKD062       0014D Rel  0320 -  0316
 hKD065       00164 Rel  0342 -  0329
 hKD070       0016A Rel  0344 -  0298
 hKD080       00172 Rel  0357 -  0338
 hKD090       0017A Rel  0368 -  0362
 hKD100       0021C Rel  0484 -  0125
 hKD110       00233 Rel  0490 -  0488
 hKYDEF       0004D Rel  0103 -  0034
 hVER60       0001D Rel  0062 -  0039
 hVERs1       0004B Rel  0073 -  0068
*kcFRT        Extrn Ukn       -  0116
*kcRT         Extrn Ukn       -  0113
*kcRUN        Extrn Ukn       -  0119
 msk          000E2 Rel  0187 -  0182 0185
*pCONFG       Extrn Ukn       -  0035
*pKYDF        Extrn Ukn       -  0032
 reste        000F8 Rel  0199 -  0179
 rst10        00101 Rel  0203 -  0208
 rst20        0010E Rel  0207 -  0202
 rtnsxm       00017 Rel  0040 -  0110


Source  : pol.as

Dirset  : sd:\pol.ao
```

```
Page 012          FINPUT : POLL HANDLER <pol.as>
AREUH ASS. V2.0   **** SYMBOL TABLE ****

Listing : list/pol.al

Date    : Mon Mar 16 11:52:06 1987

Errors  : 000


Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986 Paris, France
```

```
Page 001          FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0

0001 00000                title  FINPUT, EVALUATION DES ARGUMENTS (bas.as)
0002 00000          *
0003 00000          * Format des "dope vecteuuuurs" made in France
0004 00000          *
0005 00000          * 00-03 : nombre d'elements (1..n)
0006 00000          * 04-08 : pointeur vers les donnees
0007 00000          * 09-12 : longueur max des elements
0008 00000          *
0009 00000          *DOPEB$ EQU    (*TRFMBF)+00      13 quartets
0010 00000          *DOPED$ EQU    (*TRFMBF)+13      13 quartets
0011 00000          *DOPEP$ EQU    (*TRFMBF)+26      13 quartets
0012 00000          *DOPEA  EQU    (*TRFMBF)+26      27 quartets
0013 00000          *n      EQU    (*TRFMBF)+53       5 quartets
0014 00000          *
0015 00000          * il reste 2 quartets a (*TRFMBF)+58 qui sont utilises lors
0016 00000          * de la compilation (fichier comp.as)
0017 00000          *
0018 00000
0019 00000          *DOPEI$ EQU    (*STMTR0)+00      13 quartets
0020 00000          *DOPEM$ EQU    (*STMTR0)+13      13 quartets
0021 00000          *DOPEU$ EQU    (*STMTR0)+26      13 quartets
0022 00000          * il reste 3 quartets a (*STMTR0)+39
0023 00000          * correpondant a (*STMTD1)+02
0024 00000
```

```
Page 002          FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0   UTILITAIRES

0026 00000          *************************************************************
0027 00000          * ELMCPY
0028 00000          *
0029 00000          * But: copier l'element Y$(I) dans X$(I)
0030 00000          * Entree:
0031 00000          *    - C(A) = I
0032 00000          *    - D0 = ^ dope vecteuuuuuuuur de Y$
0033 00000          *    - D1 = ^ dope vecteuuuuuuuur de X$
0034 00000          * Sortie:
0035 00000          *    - A(A) = adresse de Y$(I)
0036 00000          *    - B(A) = (LEN(Y$(I)) + 2)*2
0037 00000          *    - C(A) = adresse de X$(I)
0038 00000          *    - P = 0
0039 00000          * Abime: A-D, D0, D1
0040 00000          * Appelle: GETxsI, MOVE+M
0041 00000          * Niveaux: 1
0042 00000          * Historique:
0043 00000          *   86/09/05: F.D. & J.T. conception & codage
0044 00000          *   86/11/24: F.D.        integration dans bas.as
0045 00000          *************************************************************
0046 00000
0047 00000 D7       *ELMCPY D=C    A        D(A) := I
0048 00002 7621             GOSUB  *GETxsI  D0 := ^ Y$(I)
0049 00006 136              CD0EX
0050 00009 137              CD1EX           D1 := ^ Y$(I)
0051 0000C 134              D0=C            D0 := dope-vecteuuuuur de X$
0052 0000F DB               C=D    A
0053 00011 7711             GOSUB  *GETxsI  C(A) = D0 := ^ X$(I)
0054 00015          *
0055 00015          * C(A) = D0 = ^ X$(I) = dest. address.
0056 00015          * D1 = ^ Y$(I)
0057 00015          * et il faut faire X$(I) := Y$(I)
0058 00015          *
0059 00015
0060 00015          *
0061 00015          * B(A) := (LEN(Y$(I)) + 2) * 2
0062 00015          *
0063 00015 D0               A=0    A
0064 00017 15B3             A=DAT1 4
0065 0001B E4               A=A+1  A
0066 0001D E4               A=A+1  A
0067 0001F C4               A=A+A  A
0068 00021 D8               B=A    A
0069 00023
0070 00023 133              AD1EX           A(A) := source address (Y$(I))
0071 00026
0072 00026 8D00000          GOVLNG *MOVE+M
0073 0002D
0074 0002D          *************************************************************
0075 0002D          * evlary
0076 0002D          *
0077 0002D          * But: evaluer ce qui est apres la virgule pointee par D0.
0078 0002D          *   et selon le cas :
0079 0002D          *   si vecteur de chaines, fabriquer le "dope vecteuuuuur"
0080 0002D          *   si chaine simple, fabriquer un pseudo-vecteur d'un seul
0081 0002D          *   element
0082 0002D          *
0083 0002D          * Entree:
0084 0002D          *    - D0 = PC
0085 0002D          *    - MTHSTK = sommet de la Math-Stack
0086 0002D          * Sortie:
0087 0002D          *    - D0 = ^ passee l'expression
0088 0002D          *      B(W) = dope-vecteuuuuuuuur fabrique
```

```
Page 003          FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0   UTILITAIRES

0089 0002D          *    - MTHSTK actualise pour prendre en compte une eventuelle
0090 0002D          *      longueur d'element de tableau, ou pour "oublier" le
0091 0002D          *      "dope-vecteuuuuuuur".
0092 0002D          * Abime: A-D, R0-R4, ST, D0, D1, Function Scratch
0093 0002D          * Appelle: EXPEXC, RCL4, RCL5, BSL4, BSL5, POP1S
0094 0002D          * Niveaux: 5 (EXPEXC)
0095 0002D          * Historique:
0096 0002D          *   86/08/30: F.D. & J.T. conception & codage
0097 0002D          *   86/11/24: F.D.        ajout de documentation
0098 0002D          *************************************************************
0099 0002D
0100 0002D 8F00000   evlary GOSBVL *EXPEXC
0101 00034 AF1              B=0    W
0102 00037 31F1             LCHEX  1F       tableau alpha ?
0103 0003B 966              ?A#C   B        non
0104 0003E 03               GOYES  evry20   (2 avenue du Lac, en fait...)
0105 00040          *
0106 00040          * string array descriptor
0107 00040          *
0108 00040 172              D1=D1+ 3        D1 := ^ maxlen
0109 00043 7631             GOSUB  RCL4
0110 00047 177              D1=D1+ 8        D1 := ^ pointer
0111 0004A 7D31             GOSUB  RCL5
0112 0004E 1C3              D1=D1- 4        D1 := ^ dim
0113 00051 7821             GOSUB  RCL4
0114 00055 1C4              D1=D1- 5        D1 := ^ option base
0115 00058 1574             C=DAT1 S
0116 0005C 94E              ?C#0   S
0117 0005F 90               GOYES  evry10
0118 00061 23               P=     3
0119 00063 B15              B=B+1  WP
0120 00066 20               P=     0
0121 00068 17D       evry10 D1=D1+ 14
0122 0006B 5C3              GONC   evry90   B.E.T.
0123 0006E
0124 0006E          *
0125 0006E          * Attention !
0126 0006E          * Cette erreur ne devrait jamais arriver...
0127 0006E          *
0128 0006E 31F0      evry20 LCHEX  0F       Chaine alpha ?
0129 00072 966              ?A#C   B
0130 00075 76               GOYES  datatp   Non, mais je ne sais pas comment
0131 00077                                  cela serait possible...
0132 00077          *
0133 00077          * On a trouve une expression alphanumerique sur la pile.
0134 00077          * Nous allons la transformer en vecteur de un element.
0135 00077          *
0136 00077 8F00000          GOSBVL *POP1S   D1 := ^ sommet de la Math-Stack
0137 0007E          *
0138 0007E          * La longueur de la chaine doit tenir sur 2 octets, d'ou le
0139 0007E          * test de la longueur (20000 quartets.
0140 0007E          *
0141 0007E 3400002          LC(5)  #20000
0142 00085 8BE              ?A>=C  A
0143 00088 B5               GOYES  strovf
0144 0008A          *
0145 0008A          * C'est une chaine, et elle est de bonne longueur
0146 0008A          *
0147 0008A 81C              ASRB            A(A) := longueur en octets
0148 0008D 1C3              D1=D1- 4        D1 := ^ longueur de V$(1)
0149 00090 1593             DAT1=A 4        LEN(V$(1)) := LEN(chaine)
0150 00094 D8               B=A    A
0151 00096 72D0             GOSUB  BSL5
```

```
Page 004          FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0   UTILITAIRES

0152 0009A 137              CD1EX
0153 0009D 135              D1=C            C(A) = D1 := ^ math-stack
0154 000A0 D5               B=C    A
0155 000A2 79C0             GOSUB  BSL4
0156 000A6 E5               B=B+1  A
0157 000A8
0158 000A8          *
0159 000A8          * terminaison de evlary. On attend dans B(W) le "dope-
0160 000A8          * vecteuuuuur" et dans D1 l'adresse du sommet de la
0161 000A8          * Math-Stack. Cette adresse sera sauvegardee en MTHSTK.
0162 000A8          *
0163 000A8 137       evry90 CD1EX           Evry Cedex en fait
0164 000AB 1F00000          D1=(5) *MTHSTK
0165 000B2 145              DAT1=C A        Sommet de la Math-Stack
0166 000B5          *
0167 000B5          * Verification du nombre d'elements du tableau en cours
0168 000B5          * avec I$
0169 000B5          *
0170 000B5 1F00000          D1=(5) =n
0171 000BC 143              A=DAT1 A
0172 000BF 23               P=     3
0173 000C1 914              ?A#B   WP
0174 000C4 90               GOYES  erdim
0175 000C6 20               P=     0
0176 000C8          *
0177 000C8          * On remet les choses en place
0178 000C8          *
0179 000C8 135              D1=C            D1 := sommet de la Math-Stack
0180 000CB 01               RTN
0181 000CD
0182 000CD 20        erdim  P=     0
0183 000CF 330000           LC(4)  (=id)-(=eIDIM)  "#Dims"
0184 000D5 8D00000          GOVLNG *BSERR
0185 000DC
0186 000DC 8D00000   datatp GOVLNG *BDATTY
0187 000E3 3100      strovf LC(2)  =eSTROV
0188 000E7 8D00000          GOVLNG *MFERR
0189 000EE
0190 000EE          *************************************************************
0191 000EE          * GETM$I, GETI$I, GETU$I, GETB$I, GETD$I, GETP$I, GETxsI
0192 000EE          *
0193 000EE          * But: renvoyer l'adresse de l'element I du tableau M$
0194 000EE          *   (resp. I$, U$, B$, D$, P$).
0195 000EE          * Entree:
0196 000EE          *    - C(A) = I
0197 000EE          * Sortie:
0198 000EE          *    - C(A) = D0 = ^ long. de T$(I) (T$=M$,I$,U$,B$,D$,P$)
0199 000EE          * Abime: A-C, D0
0200 000EE          * Appelle: MPY
0201 000EE          * Niveaux: 1
0202 000EE          * Detail: La routine GETxsI permet de retrouver l'element I
0203 000EE          *   du tableau dont le dope-vecteuuuuur est pointe par D0
0204 000EE          * Historique:
0205 000EE          *   86/08/31: F.D. & J.T. conception & codage
0206 000EE          *************************************************************
0207 000EE
0208 000EE 1B00000  *GETP$I D0=(5) *DOPEP$
0209 000F5 6630             GOTO   GET010
0210 000F9 1B00000  *GETD$I D0=(5) *DOPED$
0211 00100 6B20             GOTO   GET010
0212 00104 1B00000  *GETB$I D0=(5) *DOPEB$
0213 0010B 6020             GOTO   GET010
0214 0010F 1B00000  *GETU$I D0=(5) *DOPEU$
```

```
Page 005            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     UTILITAIRES

0215 00116 6510             GOTO   GET010
0216 0011A 1B00000  *GETM$I D0=(5) =DOPEM$
0217 00121 6A00             GOTO   GET010
0218 00125 1B00000  *GETI$I D0=(5) =DOPEI$
0219 0012C
0220 0012C          *GETx$I
0221 0012C CE       GET010 C=C-1   A
0222 0012E D5               B=C    A          !
0223 00130 AF2              C=0    W          ! C(15-5) := 0
0224 00133 D9               C=B    A          !
0225 00135 168              D0=D0= 9          D0 := ^ maxlen
0226 00138 AF0              A=0    W
0227 0013B 15A3             A=DAT0 4          A(W) := maxlen
0228 0013F E4               A=A+1  A
0229 00141 E4               A=A+1  A          A(W) := maxlen * longueur
0230 00143 7A92             GOSUB  mpy        A,B,C := offset en octets
0231 00147 C4               A=A+A  A          A(A) := offset en quartets
0232 00149 184              D0=D0- 5          D0 := ^ pointeur
0233 0014C 146              C=DAT0 A          C(A) := ^ T$(1)
0234 0014F C2               C=C+A  A          C(A) := ^ T$(I)
0235 00151 134              D0=C              D0: = ^ T$(I)
0236 00154 01               RTN
0237 00156
0238 00156          ********************************************************
0239 00156          * GETFST
0240 00156          *
0241 00156          * But: renvoyer l'adresse du premier caractere d'un element
0242 00156          *   de tableau.
0243 00156          * Entree:
0244 00156          *   - D0 = adresse de la longueur de l'element
0245 00156          * Sortie:
0246 00156          *   - C(A) = D0 = adresse du premier caractere de l'element
0247 00156          *     sus-cite. (cf Math-Stack, il faut faire D0=D0- 2 avant
0248 00156          *     utilisation)
0249 00156          *   - A(A) = D0 en entree
0250 00156          * Abime: A(A), C(A), D0
0251 00156          * Niveaux: 0
0252 00156          * Historique:
0253 00156          *   86/09/02: P.D. & J.T. conception & codage
0254 00156          ********************************************************
0255 00156
0256 00156 D2       *GETFST C=0    A
0257 00158 15E3             C=DAT0 4
0258 0015C E6               C=C+1  A
0259 0015E E6               C=C+1  A
0260 00160 C6               C=C+C  A
0261 00162 132              ADOEX
0262 00165 C2               C=A+C  A
0263 00167 134              D0=C
0264 0016A 01               RTN
0265 0016C
0266 0016C          ********************************************************
0267 0016C          * BSL4, BSL5
0268 0016C          *
0269 0016C          * But: decaler B a gauche de 4 (resp. 5) quartets)
0270 0016C          * Entree:
0271 0016C          *   - B(W)
0272 0016C          * Sortie:
0273 0016C          *   - B(W)
0274 0016C          * Abime: B(W)
0275 0016C          * Niveaux: 0
0276 0016C          * Historique:
0277 0016C          *   86/08/31: P.D. & J.T. conception & codage
```

```
Page 006            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     UTILITAIRES

0278 0016C          ********************************************************
0279 0016C
0280 0016C BF1      BSL5   BSL    W
0281 0016F BF1      BSL4   BSL    W
0282 00172 BF1             BSL    W
0283 00175 BF1             BSL    W
0284 00178 BF1             BSL    W
0285 0017B 01              RTN
0286 0017D
0287 0017D          ********************************************************
0288 0017D          * RCL4, RCL5
0289 0017D          *
0290 0017D          * But: decale B(W) de 4 (resp. 5) quartets, et y place les 4
0291 0017D          *   (resp. 5) quartets pointes par D1.
0292 0017D          * Entree:
0293 0017D          *   - B(W) = registre a decalage
0294 0017D          *   - D1 = ^ zone a inserer dans B
0295 0017D          * Sortie:
0296 0017D          *   - B(W) = decale
0297 0017D          * Abime: B(W), C(A)
0298 0017D          * Appelle: BSL4 (resp. BSL5)
0299 0017D          * Niveaux: 1
0300 0017D          * Historique:
0301 0017D          *   86/08/30: P.D. & J.T. conception & codage
0302 0017D          ********************************************************
0303 0017D
0304 0017D 7EEF     RCL4   GOSUB  BSL4
0305 00181 D2              C=0    A
0306 00183 15F3            C=DAT1 4
0307 00187 C1              B=B+C  A
0308 00189 01              RTN
0309 0018B 7DDF     RCL5   GOSUB  BSL5
0310 0018F 147             C=DAT1 A
0311 00192 D5              B=C    A
0312 00194 01              RTN
0313 00196
```

```
Page 007            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0315 00196          ********************************************************
0316 00196          * SINPUTe
0317 00196          *
0318 00196          * But: implementer un gestionnaire de masque d'ecran
0319 00196          * Abime: tout ce qui est abimable par un statement
0320 00196          * Niveaux: tous ceux qui sont abimables par un statement
0321 00196          * Algorithme:
0322 00196          *   evaluer I$, creer le dope-vecteuuuuuuuur
0323 00196          *   evaluer M$, creer le dope-vecteuuuuuuuur
0324 00196          *   si P$ n'existe pas
0325 00196          *     alors creer un pseudo P$ : P$(1):=STR$(LEN(M$(1))&"F")
0326 00196          *     sinon evaluer P$, creer le dope-vecteuuuuuuuur
0327 00196          *   finsi
0328 00196          *   initialiser le tableau D$ (D$(1) := M$(1))
0329 00196          *   initialiser le tableau U$ (U$(1) := NULL$)
0330 00196          *   compiler le format dans le tableau B$ (Bit Map)
0331 00196          *   evaluer A
0332 00196          *   verifier que M$ ne contient pas de car. non affichables
0333 00196          *   i := 1 ;
0334 00196          *   repeter
0335 00196          *     afficher la chaine D$(i)
0336 00196          *     (-)
0337 00196          *     placer B$(i) dans DSPMSK
0338 00196          *     editer la ligne
0339 00196          *     selon la derniere touche appuyee :
0340 00196          *       [^]: i := si i>1 alors i-1 ;
0341 00196          *       [v]: i := si i<n alors i+1 ;
0342 00196          *       [g][^]: i := 1 ;
0343 00196          *       [g][v]: i := n ;
0344 00196          *       [OFF]: sortir avec A := 0 ;
0345 00196          *       [ATTN]: si display = M$(i)
0346 00196          *                 alors sortir avec A := 0 ;
0347 00196          *                 sinon
0348 00196          *                   afficher la chaine M$(i) ;
0349 00196          *                   reprendre a (-) ;
0350 00196          *                 finsi ;
0351 00196          *       [ENDLINE]: U$(i) := display
0352 00196          *                  si i=1 alors continuer dans [RUN]
0353 00196          *       [RUN]: U$(i) := display
0354 00196          *              sortir avec A := i ;
0355 00196          *     fin selon
0356 00196          *   fin repeter
0357 00196          * Historique:
0358 00196          *   86/08/30: P.D. & J.T. conception & codage
0359 00196          *   86/11/24: P.D.        ajout de documentation
0360 00196          ********************************************************
0361 00196
0362 00196 00000            REL(5) =FINPUTd
0363 0019B 00000            REL(5) =FINPUTp
0364 001A0          =FINPUTe
0365 001A0          *
0366 001A0          * Traitement de I$
0367 001A0          *
0368 001A0 8F00000          GOSBVL =ADDRSS
0369 001A7          *
0370 001A7          * A la sortie de ADDRSS,
0371 001A7          * si Cy = 1, la variable n'est pas trouvee
0372 001A7          * si Cy = 0, la variable est trouvee, et :
0373 001A7          *   D0 = B(A) = adresse du registre de la variable
0374 001A7          *   A(A) = ^ passee la tokenisation de la variable
0375 001A7          *
0376 001A7 5F0              GONC   I$005
0377 001AA 330000           LC(4)  (=id)-(=eINOVR)     "Var Not Found"
```

```
Page 008            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0378 001B0 8D00000          GOVLNG =BSERR
0379 001B7
0380 001B7 136      I$005  CD0EX            C(A) := ^ registre de la variable
0381 001BA 135             D1=C             D1 := ^ registre de la variable
0382 001BD 130             D0=A             D0 := PC
0383 001C0          *
0384 001C0          * Boucle pour trouver le registre reel de la variable
0385 001C0          *
0386 001C0 30E             LCHEX  E
0387 001C3 170      I$010  D1=D1+ 1
0388 001C6 1530            A=DAT1 P         A(P) := type de la variable
0389 001CA 906             ?A#C   P         Tableau indirect
0390 001CD E0              GOYES  I$020
0391 001CF          * la variable est un tableau indrect
0392 001CF 170             D1=D1+ 1         D1 := ^ "pointer"
0393 001D2 143             A=DAT1 A         A(A) := ^ reg. de la var. suivant
0394 001D5 131             D1=A             D1 := ^ reg. de la var. suivant
0395 001D8 5AE             GONC   I$010     B.E.T.
0396 001DB          *
0397 001DB          * Fin de chainage
0398 001DB          *
0399 001DB 1C0      I$020  D1=D1- 1         D1 := ^ deux premiers quartets
0400 001DE AF1             B=0    W
0401 001E1 14B             A=DAT1 B         A(B) := type de la variable
0402 001E4 31F1            LCHEX  1F        Tableau alphanumerique
0403 001E8 966             ?A#C   B
0404 001EB C3              GOYES  I$040
0405 001ED          *
0406 001ED          * Tableau (direct ou indirect)
0407 001ED          *
0408 001ED 172             D1=D1+ 3
0409 001F0 798F            GOSUB  RCL4      Maxlen
0410 001F4 177             D1=D1+ 8         D1 := ^ "pointer"
0411 001F7 709F            GOSUB  RCL5      relative pointer
0412 001FB 133             AD1EX            A(A) := adresse du relative pointer
0413 001FE 131             D1=A
0414 00201 E0              A=A-B  A         A(A) := ^ premier element
0415 00203 D8              B=A    A         B(8-5) := maxlen ; B(4-0) := pointer
0416 00205 1C3             D1=D1- 4         D1 := ^ dim
0417 00208 717F            GOSUB  RCL4
0418 0020C 1C4             D1=D1- 5         D1 := ^ option base
0419 0020F 1574            C=DAT1 S
0420 00213 94E             ?C#0   S
0421 00216 90              GOYES  I$030
0422 00218 23              P=     3
0423 0021A B15             B=B+1  WP        = 1 en option base 0
0424 0021D 20              P=     0
0425 0021F 6E40     I$030  GOTO   I$100
0426 00223
0427 00223 68BE     Datatp GOTO   datatp
0428 00227
0429 00227 31F0     I$040  LCHEX  0F
0430 0022B 966             ?A#C   B
0431 0022E B1              GOYES  I$050
0432 00230          *
0433 00230          * variable directe
0434 00230          *
0435 00230 172             D1=D1+ 3         D1 := ^ maxlen
0436 00233 764F            GOSUB  RCL4
0437 00237 177             D1=D1+ 8         D1 := ^ relative pointer
0438 0023A 7D4F            GOSUB  RCL5      relative pointer
0439 0023E 133             AD1EX            A(A) := adresse du relative pointer
0440 00241 E0              A=A-B  A         A(A) := ^ premier element
```

```
Page 009            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0441 00243 D8               B=A     A       B(8-5) := maxlen ; B(4-0) := pointer
0442 00245 6A10             GOTO    I$060
0443 00249
0444 00249          *
0445 00249          * Cette erreur ne devrait jamais arriver :
0446 00249          *
0447 00249 31FF      I$050  LCHEX   FF
0448 0024D 966              ?A#C    B
0449 00250 3D               GOYES   Datatp
0450 00252          *
0451 00252          * variable indirecte
0452 00252          *
0453 00252 176              D1=D1+  7       D1 := ^ maxlen
0454 00255 742F             GOSUB   RCL4
0455 00259 1C4              D1=D1-  5       D1 := ^ absolute pointer
0456 0025C 7B2F             GOSUB   RCL5
0457 00260 BF1       I$060  BSL     W
0458 00263 BF1              BSL     W
0459 00266 BF1              BSL     W
0460 00269 BF1              BSL     W
0461 0026C E5               B=B+1   A       Dim := 1
0462 0026E
0463 0026E AF9       I$100  C=B     W
0464 00271 1F00000          D1=(5)  =DOPEI$
0465 00278 15DC             DAT1=C  13
0466 0027C 24               P=      4
0467 0027E A82              C=0     P
0468 00281 20               P=      0
0469 00283 1E0000           D1=(4)  =n      Tant que =n reste en TRFMBF+53,
0470 00289 145              DAT1=C  A         D1=(2) aurait suffi
0471 0028C          *
0472 0028C          * A ce stade, la variable I$ est traitee (ouf)
0473 0028C          *
0474 0028C          * J.T. : "YapuquaM$" (je cite)
0475 0028C          *
0476 0028C 8F00000          GOSBVL  =COLLAP Vide la Math-Stack
0477 00293 161              D0=D0-  2
0478 00296 739D             GOSUB   evlary  evaluation de M$
0479 0029A 1F00000          D1=(5)  =DOPEM$
0480 002A1 AF4              A=B     W
0481 002A4 159C             DAT1=A  13
0482 002A8          *
0483 002A8          * P$ ou A ?
0484 002A8          *
0485 002A8 161              D0=D0-  2       On passe tCOMMA
0486 002AB 1524             A=DAT0  S       A(S) := quartet de reconnaissance
0487 002AF 160              D0=D0+  1
0488 002B2 948              ?A=0    S       A ?
0489 002B5 A1               GOYES   PP$010  Pseudo P$
0490 002B7          *
0491 002B7          * On est devant P$.
0492 002B7          *
0493 002B7 727D             GOSUB   evlary  evaluation de P$
0494 002BB 161              D0=D0+  2       J.T. & P.D. bugfix (861123.1415)
0495 002BE 136              CD0EX           Sauvegarde de PC dans R3 pour la
0496 002C1 10B              R3=C            compilation
0497 002C4 69C0             GOTO    compilerP$
0498 002C8
0499 002C8 8D00000   stkchr GOVLNG  =STKCHR
0500 002CF          *
0501 002CF          * On est devant A. Il faut creer un pseudo P$ par la boucle:
0502 002CF          *  FOR I = nombre-d'elements(M$) STEP -1
0503 002CF          *     P$(I) = STR$(LEN(M$(I)))&"PU"
```

```
Page 010            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0504 002CF          *   NEXT I
0505 002CF          * Ce tableau P$ est construit sur la Math-Stack.
0506 002CF          *
0507 002CF
0508 002CF          ***************************************************
0509 002CF          * PP$010
0510 002CF          *
0511 002CF          * But: creer un pseudo-P$ dans le cas ou ce tableau est
0512 002CF          *    omis par l'utilisateur.
0513 002CF          * Historique:
0514 002CF          *    86/08/30: F.D. & J.T. conception & codage
0515 002CF          *    87/01/25: F.D. & J.T. remplace S-R0-0 par TRFMBF
0516 002CF          ***************************************************
0517 002CF
0518 002CF 136       PP$010 CD0EX           Sauvegarde de PC dans R3
0519 002D2 10B              R3=C
0520 002D5 8F00000          GOSBVL  =D1MSTK
0521 002DC 1B00000          D0=(5)  =DOPEM$
0522 002E3 D2               C=0     A
0523 002E5 15E3             C=DAT0  4       C(A) := nombre d'elements de M$
0524 002E9 1900             D0=(2)  =TRFMBF
0525 002ED          *
0526 002ED          * Debut de la boucle ci-dessus expliquee.
0527 002ED          * D0 = TRFMBF
0528 002ED          * TRFMBF + I
0529 002ED          * P$(I) sera depose sur la pile (pointee par D1)
0530 002ED          *
0531 002ED           PP$050
0532 002ED 144              DAT0=C  A       C(A) := I
0533 002F0 762E             GOSUB   =GETM$I D0 := ^ M$(I) (sur la longueur)
0534 002F4 D0               A=0     A
0535 002F6 15A3             A=DAT0  4       A(A) := longueur en hexa
0536 002FA 8F00000          GOSBVL  =HEXDEC  A(W) = B(W) = C(W) := long. en dec.
0537 00301 04               SETHEX
0538 00303 814              ASRC
0539 00306 814              ASRC
0540 00309 814              ASRC
0541 0030C 814              ASRC
0542 0030F
0543 0030F 8F00000          GOSBVL  =D=AVMS  D(A) := AVMEMS
0544 00316 304              LC(1)   4
0545 00319 816              CSRC            C(S) := 4
0546 0031C          *
0547 0031C          * Boucle interne de formattage ascii de la longueur de M$(I)
0548 0031C          * On aura, si M$="AREUH" : P$ := "00005PU"
0549 0031C          *
0550 0031C 3103      PP$070 LCASC   '0'
0551 00320 A62              C=C+A   B
0552 00323 D0               A=0     A
0553 00325 7F9F             GOSUB   stkchr
0554 00329 810              ASLC
0555 0032C A4E              C=C-1   S
0556 0032F 5CE              GONC    PP$070
0557 00332
0558 00332 3105             LCASC   'P'
0559 00336 7E8F             GOSUB   stkchr
0560 0033A 3155             LCASC   'U'
0561 0033E 768F             GOSUB   stkchr
0562 00342          *
0563 00342          * On met la longueur de P$(I) sur la pile
0564 00342          *
0565 00342 D2               C=0     A
0566 00344 307              LC(1)   7       LEN(P$(I)) := 7
```

```
Page 011            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0567 00347 1C3              D1=D1-  4
0568 0034A 15D3             DAT1=C  4
0569 0034E          *
0570 0034E          * On passe a l'element suivant de M$  (M$(I-1))
0571 0034E          *
0572 0034E 1B00000          D0=(5)  =TRFMBF
0573 00355 146              C=DAT0  A
0574 00358 CE               C=C-1   A
0575 0035A 8AE              ?C#0    A
0576 0035D 09               GOYES   PP$050
0577 0035F          *
0578 0035F          * Creation du "dope-vecteuuuuuur"
0579 0035F          *
0580 0035F AF1              B=0     W
0581 00362 307              LC(1)   7
0582 00365 AB5              B=C     P       B(3-0):=maxlen
0583 00368
0584 00368 137              CD1EX
0585 0036B 135              D1=C
0586 0036E 7AFD             GOSUB   BSL5
0587 00372 D5               B=C     A       B(8-5):=maxlen;B(4-0):=pointer
0588 00374
0589 00374 1B00000          D0=(5)  =DOPEM$
0590 0037B D2               C=0     A
0591 0037D 15E3             C=DAT0  4
0592 00381 7AED             GOSUB   BSL4
0593 00385 C1               B=B+C   A       B(12-9):=maxlen ;
0594 00387                  "               B(8-4):=pointer ;
0595 00387                  "               B(3-0):=dim
0596 00387 8F00000          GOSBVL  =AVE+D1
0597 0038E
0598 0038E          ***************************************************
0599 0038E          * compilerP$
0600 0038E          *
0601 0038E          * But: compiler le format donne par l'utilisateur (ou
0602 0038E          *    construit par SINPUT) en une Bit Map utilisable par le
0603 0038E          *    display driver du HP71 (DSPMSK).
0604 0038E          * Entree:
0605 0038E          *    - DOPEP$ contient le dope-vecteuuuuuur de P$
0606 0038E          * Sortie:
0607 0038E          *    - B$ contient le tableau des Bits map.
0608 0038E          * Abime:
0609 0038E          * Appelle: fichier comp.as
0610 0038E          * Niveaux: 1
0611 0038E          * Detail et Algorithme: voir fichier comp.as
0612 0038E          * Historique:
0613 0038E          *    86/11/24: F.D.         ajout de documentation
0614 0038E          ***************************************************
0615 0038E
0616 0038E           compilerP$
0617 0038E 1B00000          D0=(5)  =DOPEP$
0618 00395 AF4              A=B     W
0619 00398 158C             DAT0=A  13
0620 0039C
0621 0039C
0622 0039C          *
0623 0039C          * Preparation des tableaux
0624 0039C          * B$ (bit map) : Bit mask de DSPMSK
0625 0039C          *      chargement : lors de la compilation (seulement lors)
0626 0039C          *      utilisation: POKE dans DSPMASK
0627 0039C          *      DIM B$(n)[12]
0628 0039C          * D$ (Display) : copie des caracteres du DSPBUF
0629 0039C          *      chargement : apres appui sur [ENDLINE]
```

```
Page 012            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0630 0039C          *      utilisation: reaffichage du champ par une routine de
0631 0039C          *       type DSPCNA, mais inversee
0632 0039C          *      DIM D$(n)[96]
0633 0039C          * U$ (User) : donnee saisie par l'utilisateur
0634 0039C          *      chargement : apres appui sur [ENDLINE], par DSP$00
0635 0039C          *      utilisation: lors de l'appui sur [RUN], chargement de
0636 0039C          *       I$
0637 0039C          *      DIM U$(n)[96]
0638 0039C          *
0639 0039C
0640 0039C          *
0641 0039C          * Les tableaux sont ranges entre OUTBS et AVMEMS
0642 0039C          *
0643 0039C 8F00000          GOSBVL  =OBCOLL
0644 003A3          *
0645 003A3          * Tout d'abord, y-a-t'il assez d'espace ?
0646 003A3          *
0647 003A3 AF2              C=0     W
0648 003A6 324A1            LC(3)   (96+2*96+2+12+2)*2
0649 003AB 1B00000          D0=(5)  =n
0650 003B2 AF0              A=0     W
0651 003B5 142              A=DAT0  A
0652 003B8 7520             GOSUB   mpy     A, B, C := taille memoire necessair=
0653 003BC D1               B=0     A
0654 003BE 97D              ?B#0    W
0655 003C1 91               GOYES   mererr
0656 003C3          *
0657 003C3          * Oui, il y a suffisamment d'espace...
0658 003C3          *
0659 003C3 1B00000          D0=(5)  =AVMEMS
0660 003CA 146              C=DAT0  A
0661 003CD C2               C=C+A   A       C(A) := nouvel AVMEMS
0662 003CF 164              D0=D0+  5       D0=(5) AVMEME
0663 003D2 142              A=DAT0  A       A(A) := AVMEME
0664 003D5 9B6              ?C(A    A
0665 003D8 01               GOYES   cmp10   Ok
0666 003DA
0667 003DA 8D00000   mererr GOVLNG  =MEMERR
0668 003E1 8D00000   mpy    GOVLNG  =MPY
0669 003E8
0670 003E8 184       cmp10  D0=D0-  5       D0=(5) AVMEMS
0671 003EB 142              A=DAT0  A       A(A) := adresse de B$
0672 003EE 144              DAT0=C  A       AVMEMS := adresse de la fin de U$
0673 003F1 1B00000          D0=(5)  =n
0674 003F8 1F00000          D1=(5)  =DOPEB$
0675 003FF
0676 003FF          * A(A) = ^ premier element de B$
0677 003FF
0678 003FF AF1              B=0     W
0679 00402 D2               C=0     A
0680 00404 31C0             LC(2)   12      Taille d'Un element
0681 00408 D5               B=C     A
0682 0040A 7E5D             GOSUB   BSL5
0683 0040E D8               B=A     A       B(A) := adresse de B$(1)
0684 00410 7B5D             GOSUB   BSL4
0685 00414 146              C=DAT0  A       C(A) := n
0686 00417 C1               B=B+C   A
0687 00419 AF9              C=B     W
0688 0041C 15DC             DAT1=C  13
0689 00420 17C              D1=D1+  13
0690 00423
0691 00423 AF2              C=0     W
0692 00426 31C1             LC(2)   (12+2)*2
```

```
Page 013            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     SINPUT, EXECUTION ROUTINE (debut)

0693 0042A AFE              ACEX   W
0694 0042D D7               D+C    A        D(A) :* adresse de B$(1)
0695 0042F AF2              C+0    W
0696 00432 146              C+DAT0 A
0697 00435 78AF             GOSUB  mpy
0698 00439 CB               C+C+D  A
0699 0043B DA               A+C    A
0700 0043D            * D$
0701 0043D AF1              B+0    W
0702 00440 D2               C+0    A
0703 00442 3106             LC(2)  96
0704 00446 D5               B+C    A
0705 00448 702D             GOSUB  BSL5
0706 0044C D8               B+A    A        B(A) :* adresse de D$(1)
0707 0044E 7D1D             GOSUB  BSL4
0708 00452 146              C+DAT0 A        C(A) :* n
0709 00455 C1               B+B+C  A
0710 00457 AF9              C+B    W
0711 0045A 15DC             DAT1+C 13
0712 0045E
0713 0045E AF2              C+0    W
0714 00461 314C             LC(2)  (96+2)*2
0715 00465 AFE              ACEX   W
0716 00468 D7               D+C    A        D(A) :* adresse de D$(1)
0717 0046A AF2              C+0    W
0718 0046D 146              C+DAT0 A
0719 00470 7D6F             GOSUB  mpy
0720 00474 CB               C+C+D  A
0721 00476 DA               A+C    A
0722 00478            * U$
0723 00478 1D00             D1+(2) =DOPEU$
0724 0047C AF1              B+0    W
0725 0047F D2               C+0    A
0726 00481 3106             LC(2)  96
0727 00485 D5               B+C    A
0728 00487 71EC             GOSUB  BSL5
0729 0048B D8               B+A    A        B(A) :* adresse de U$(1)
0730 0048D 7EDC             GOSUB  BSL4
0731 00491 146              C+DAT0 A        C(A) :* n
0732 00494 C1               B+B+C  A
0733 00496 AF9              C+B    W
0734 00499 15DC             DAT1+C 13
0735 0049D
0736 0049D            *
0737 0049D            * Le code continue dans le fichier comp.as
0738 0049D            *
0739 0049D
0740 0049D                  END
```

```
Page 014            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     **** SYMBOL TABLE ****

*ADDRSS         Extrn Ukn         -  0368
*AVE+D1         Extrn Ukn         -  0596
*AVMEMS         Extrn Ukn         -  0659
*BSERR          Extrn Ukn         -  0184 0378
 BSL4           0016F Rel  0281   -  0155 0304 0592 0684 0707 0730
 BSL5           0016C Rel  0280   -  0151 0309 0586 0682 0705 0728
*COLLAP         Extrn Ukn         -  0476
*D1MSTK         Extrn Ukn         -  0520
*D+AVMS         Extrn Ukn         -  0543
*DOPEA          Unkwn Ukn  0012   -
*DOPEB$         Unkwn Ukn  0009   -  0212 0674
*DOPED$         Unkwn Ukn  0010   -  0210
*DOPEI$         Unkwn Ukn  0019   -  0218 0464
*DOPEM$         Unkwn Ukn  0020   -  0216 0479 0521 0589
*DOPEP$         Unkwn Ukn  0011   -  0208 0617
*DOPEU$         Unkwn Ukn  0021   -  0214 0723
 Datatp         00223 Rel  0427   -  0449
*ELMCPY         00000 Rel  0047   -
*EXPEXC         Extrn Ukn         -  0100
*FINPUTd        Extrn Ukn         -  0362
*FINPUTe        001A0 Rel  0364   -
*FINPUTp        Extrn Ukn         -  0363
 FileHd         0049D Rel  0740   -
*GETB$!         00104 Rel  0212   -
*GETD$!         000F9 Rel  0210   -
*GETFST         00156 Rel  0256   -
*GETI$!         00125 Rel  0218   -
*GETM$!         0011A Rel  0216   -  0533
*GETP$!         000EE Rel  0208   -
*GETU$!         0010F Rel  0214   -
*GETx$!         0012C Rel  0220   -  0048 0053
 GET010         0012C Rel  0221   -  0209 0211 0213 0215 0217
*HEXDEC         Extrn Ukn         -  0536
 !$005          00187 Rel  0380   -  0376
 !$010          001C3 Rel  0387   -  0395
 !$020          001DB Rel  0399   -  0390
 !$030          0021F Rel  0425   -  0421
 !$040          00227 Rel  0429   -  0404
 !$050          00249 Rel  0447   -  0431
 !$060          00260 Rel  0457   -  0442
 !$100          0026E Rel  0463   -  0425
*MEMERR         Extrn Ukn         -  0667
*MFERR          Extrn Ukn         -  0188
*MOVE+M         Extrn Ukn         -  0072
*MPY            Extrn Ukn         -  0668
*MTHSTK         Extrn Ukn         -  0164
*OBCOLL         Extrn Ukn         -  0643
*POP1S          Extrn Ukn         -  0136
 PP$010         002CF Rel  0518   -  0489
 PP$050         002ED Rel  0531   -  0576
 PP$070         0031C Rel  0550   -  0556
*RDATTY         Extrn Ukn         -  0186
 RCL4           0017D Rel  0304   -  0109 0113 0409 0417 0436 0454
 RCL5           0018B Rel  0309   -  0111 0411 0438 0456
*STKCHR         Extrn Ukn         -  0499
*STMTRC         Extrn Ukn         -  0019 0020 0021
*TREMBF         Extrn Ukn         -  0009 0010 0011 0012 0013 0524 0572
 cmpl0          003E8 Rel  0670   -  0665
 compilerP$     0038E Rel  0616   -  0497
 datalp         000DC Rel  0186   -  0130 0427
*eIDIM          Extrn Ukn         -  0183
*eINOVR         Extrn Ukn         -  0377
*eTTRCV         Extrn Ukn         -  0187
```

```
Page 015            FINPUT, EVALUATION DES ARGUMENTS (bas.as)
AREUH ASS. V2.0     **** SYMBOL TABLE ****

 erdim          000CD Rel  0182   -  0174
 evlary         0002D Rel  0100   -  0478 0493
 evry10         00068 Rel  0121   -  0117
 evry20         0006E Rel  0128   -  0104
 evry90         000A8 Rel  0163   -  0122
*id             Extrn Ukn         -  0183 0377
 memerr         003DA Rel  0667   -  0655
 mpy            003E1 Rel  0668   -  0230 0652 0697 0719
*n              Unkwn Ukn  0013   -  0170 0469 0649 0673
 stkchr         002C8 Rel  0499   -  0553 0559 0561
 strovf         000E3 Rel  0187   -  0143



Source  : bas.as

Object  : obj/bas.ao

Listing : list/bas.al

Date    : Mon Mar 16 11:52:34 1987

Errors  : 000


Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986 Paris, France
```

```
Page 001          FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0

0001 00000                title  FINPUT : COMPILATION DU FORMAT (comp.as)
0002 00000          *
0003 00000          * La boucle suivante est la boucle de compilation de P$
0004 00000          *
0005 00000          * FOR I = 1 TO nombre-d'elements(P$)
0006 00000          *   appliquer l'automate :
0007 00000          *   D$(I) := M$(I) ;
0008 00000          * NEXT I
0009 00000          *
0010 00000
0011 00000           i       equ     (*FUNCR0)+00  5 quartets
0012 00000           cptp$   equ     (*FUNCR0)+05  5 quartets
0013 00000           addsp$  equ     (*FUNCR0)+10  5 quartets
0014 00000           addsu$  equ     (*FUNCR0)+15  5 quartets
0015 00000           cptm$   equ     (*FUNCR0)+20  2 quartets
0016 00000           addsm$  equ     (*FUNCR0)+22  5 quartets
0017 00000           mask    equ     (*FUNCR0)+27  1 quartet
0018 00000           admask  equ     (*FUNCR0)+28  5 quartets
0019 00000           cptmsk  equ     (*FUNCR0)+33  2 quartets
0020 00000           mult    equ     (*FUNCR0)+35  2 quartets
0021 00000           lastUP  equ     (*FUNCR0)+37  1 quartet
0022 00000          * Il reste 1 quartet a (*FUNCR0)+41
0023 00000           credit equ     (*TRFMBF)+58  2 quartets
0024 00000          * Il ne reste plus rien en TRFMBF !
0025 00000
0026 00000          *
0027 00000          * Les variables k et p utilisees dans "bitmap" sont en
0028 00000          * vark : (*FUNCR0)+38 sur 2 quartets
0029 00000          * varp : (*FUNCR0)+40 sur 1 quartet
0030 00000          *
0031 00000
0032 00000          *                        '0'..'9'
0033 00000          *                          ----
0034 00000          *                         |c   |
0035 00000          *                         |    |
0036 00000          *           '0'..'9'      v    |
0037 00000          *           ----------)    2
0038 00000          *         |
0039 00000          *       b|                 |                 '0'..'9'
0040 00000          *        |                  |'P'|'U'          ----
0041 00000          *        |                 e|                |c   |
0042 00000          *        |                  |                |    |
0043 00000          *   a      'P'|'U'         v  b  '0'..'9'  v    |
0044 00000          *  ---> 1 ----------> 3 ----------> 4
0045 00000          *             d        ->|| ^              |
0046 00000          *                       |  || |    'P'|'U'  e |
0047 00000          *                     d --  |  --------------
0048 00000          *                  'P'|'U'|
0049 00000          *                        ||
0050 00000          *                        / fin de P$
0051 00000          *                        |
0052 00000          *
0053 00000          *
0054 00000          * Actions de l'automate :
0055 00000          *
0056 00000          * a: Initialisation de l'automate
0057 00000          *
0058 00000          *   cptp$  := LEN (GETP$I) ;
0059 00000          *   addsp$ := GETFST (GETP$I) ;
0060 00000          *   mask   := 2^3 ;
0061 00000          *   cptmsk := 96 ;
0062 00000          *   credit := min (MAXLEN(I$), 96) ;
0063 00000          *   admask := GETB$I + (12*2 + 4 - 1) quartets ;
```

```
Page 002          FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0

0064 00000          *   (admask) := 0 ;
0065 00000          *   addsu$ := GETU$I + 4 quartets ;
0066 00000          *   l      := LEN(GETM$I) ;
0067 00000          *   si l>96 alors "String Overflow" finsi ;
0068 00000          *   cptm$  := l ;
0069 00000          *   addsm$ := GETFST (GETM$I) ;
0070 00000          *
0071 00000          * b: Initialisation explicite du multiplicateur
0072 00000          *
0073 00000          *   mult   := carlu + '0' ;
0074 00000          *
0075 00000          * c: Calcul du multiplicateur
0076 00000          *
0077 00000          *   si mult>10 alors "Invalfd format" finsi ;
0078 00000          *   mult := (mult * 10) + (carlu - '0') ;
0079 00000          *
0080 00000          * d: Initialisation implicite du multiplicateur
0081 00000          *
0082 00000          *   mult := 1 ;
0083 00000          *   executer e ;
0084 00000          *
0085 00000          * e: Protection
0086 00000          *
0087 00000          *   p := si carlu='P' alors 1 sinon 0 ;
0088 00000          *   lastUP := p ;
0089 00000          *   bitmap (mult, p) ;
0090 00000          *
0091 00000          * 1: Sortie de l'automate
0092 00000          *
0093 00000          *   bitmap (cptmsk, lastUP) ;
0094 00000          *   Us(I) := REVs(Us(I)) ;
0095 00000          *
0096 00000          *
0097 00000          * bitmap (n, p)
0098 00000          *   si n=0 alors retour ;
0099 00000          *   pour k := n jusque 1
0100 00000          *     faire
0101 00000          *       si cptmsk = 0 alors "Invalid format" ; finsi
0102 00000          *       cptmsk-- ;
0103 00000          *       si credit = 0 alors p := 1 ; finsi
0104 00000          *       si p
0105 00000          *         alors (admask) := (admask) ou mask ;
0106 00000          *         sinon
0107 00000          *           si cptm$>0 alors (addsu$++) := (addsm$) ; finsi;
0108 00000          *           credit-- ;
0109 00000          *       finsi
0110 00000          *       addsm$++ ;
0111 00000          *       si cptm$>0 alors cptm$-- ; finsi
0112 00000          *       mask := mask >> 2 ;
0113 00000          *       si mask=0
0114 00000          *         alors
0115 00000          *           mask := 2^3 ;
0116 00000          *           admask-- ;
0117 00000          *           (admask) := 0 ;
0118 00000          *       finsi
0119 00000          *   finpour
0120 00000          *
0121 00000
0122 00000 1F00000          d1*(5) *n
0123 00007 147              c*dat1 A
0124 0000A 1D00             d1*(2) 1
0125 0000E          *
0126 0000E          * la boucle "compiler" attend dans
```

```
Page 003          FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0

0127 0000E          *   D1 = l'adresse de la variable i
0128 0000E          *   C(A) = la valeur de cette variable
0129 0000E          *
0130 0000E
0131 0000E           compiler
0132 0000E 145              dat1=c a        i := c(a)
0133 00011
0134 00011          *
0135 00011          * Action a : initialisation de l'automate
0136 00011          *
0137 00011          * cptp$ := LEN (GETP$I) ;
0138 00011          *
0139 00011 1F00000          d1=(5) 1
0140 00018 147              c=dat1 A
0141 0001B 7000             gosub  =GETP$I  Commentaire
0142 0001F 1F00000          d1=(5) cptp$
0143 00026 D2               c=0    A
0144 00028 15E3             c=dat0 4
0145 0002C 145              dat1=c A
0146 0002F          *
0147 0002F          * addsp$ := GETFST (GETP$I) ;
0148 0002F          *
0149 0002F 7000             gosub  =GETFST  D0 est toujours comme apres GETP$I
0150 00033 1D00             d1=(2) addsp$
0151 00037 145              dat1=c A
0152 0003A          *
0153 0003A          * mask := 2^3 ;
0154 0003A          *
0155 0003A 308              lc(1)  8  2^3 (ou encore 8, ou plutot %1000)
0156 0003D 1D00             d1=(2) mask
0157 00041 1550             dat1=c P
0158 00045          *
0159 00045          * cptmsk := 96 ;
0160 00045          *
0161 00045 D2               c=0    A        Pour l'instruction suivante...
0162 00047 3106             lc(2)  96
0163 0004B 1D00             d1=(2) cptmsk
0164 0004F 14D              dat1=c B
0165 00052          *
0166 00052          * credit := MIN (MAXLEN(I$), 96) ;
0167 00052          *
0168 00052 1B00000          d0=(5) (=DOPE1$)+9 D0 := ^ MAXLEN(I$)
0169 00059 D0               a=0    a
0170 0005B 15A3             a=dat0 4        A(A) := MAXLEN(I$)
0171 0005F 8B2              ?c>a   a
0172 00062 40               goyes  actiona-05  on ne change rien
0173 00064 DA               a=c    a
0174 00066                 actiona-05
0175 00066 1D00             d1=(2) credit
0176 0006A 149              dat1=a b        credit := MIN(MAXLEN(I$), 96) ;
0177 0006D          *
0178 0006D          * admask := GETB$I + (12*2 + 4 - 1) quartets ;
0179 0006D          *
0180 0006D 1B00000          d0=(5) 1
0181 00074 146              c=dat0 A
0182 00077 7000             gosub  =GETB$I
0183 0007B DA               a=c    A
0184 0007D D2               c=0    A
0185 0007F 31B1             lc(2)  12*2+4-1
0186 00083 C2               c=c=a  A
0187 00085 1D00             d1=(2) admask
0188 00089 145              dat1=c A
0189 0008C          *
```

```
Page 004          FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0

0190 0008C          * (admask) := 0 ;
0191 0008C          *
0192 0008C          *   * Bug fix (861129.1115)
0193 0008C 134              d0=c
0194 0008F AC2              c=0    s
0195 00092 1544             dat0=c s
0196 00096          *
0197 00096          * addsu$ := GETU$I + 4 quartets ;
0198 00096          *
0199 00096 1B00000          d0=(5) 1
0200 0009D 146              c=dat0 A
0201 000A0 7000             gosub  =GETU$I
0202 000A4 E6               c=c-1  A
0203 000A6 E6               c=c-1  A
0204 000A8 E6               c=c-1  A
0205 000AA E6               c=c-1  A
0206 000AC 1D00             d1=(2) addsu$
0207 000B0 145              dat1=c A
0208 000B3          *
0209 000B3          * 1 := LEN (GETM$I) ;
0210 000B3          *
0211 000B3 1B00000          d0=(5) 1
0212 000BA 146              c=dat0 A
0213 000BD 7000             gosub  =GETM$I
0214 000C1 D0               a=0    A
0215 000C3 15A3             a=dat0 4
0216 000C7 D2               c=0    A
0217 000C9 3106             lc(2)  96
0218 000CD 8BA              ?a(=c  A
0219 000D0 D0               goyes  actiona-10
0220 000D2
0221 000D2 3100             lc(2)  =eSTROV
0222 000D6 8D00000          goving =MFERR
0223 000DD
0224 000DD                 actiona-10
0225 000DD          *
0226 000DD          * cptm$ := 1 ;
0227 000DD          *
0228 000DD 1D00             d1=(2) cptm$
0229 000E1 149              dat1=a B
0230 000E4          *
0231 000E4          * addsm$ := GETFST (GETM$I) ;
0232 000E4          *
0233 000E4 7000             gosub  =GETFST
0234 000E8 1D00             d1=(2) addsm$
0235 000EC 145              dat1=c A
0236 000EF
0237 000EF          ********************************************************
0238 000EF          * Etat 1
0239 000EF          ********************************************************
0240 000EF
0241 000EF 7460            etat1  gosub  getchr
0242 000F3 DA0              rel(3) erreur   EOL
0243 000F6 600              rel(3) t1-3     P | U
0244 000F9 B00              rel(3) t1-2     0..9
0245 000FC
0246 000FC 7B01             t1-3   gosub  actiond
0247 00100 6020             goto   etat3
0248 00104
0249 00104 7CB0             t1-2   gosub  actionb
0250 00108          *       goto   etat2
0251 00108          *
0252 00108          * ATTENTION : LE CODE CONTINUE
```

```
Page 005            FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0

0253 00108          *
0254 00108
0255 00108          ************************************************************
0256 00108          * Etat 2
0257 00108          ************************************************************
0258 00108
0259 00108 7B40      etat2  gosub  getchr
0260 0010C 490              rel(3) erreur   EOL
0261 0010F E00              rel(3) t2-3     P | U
0262 00112 300              rel(3) t2-2     0..9
0263 00115
0264 00115 7EB0      t2-2   gosub  actionc
0265 00119 6EEF             goto   etat2
0266 0011D
0267 0011D 7AF0      t2-3   gosub  actione
0268 00121          *       goto   etat3
0269 00121          *
0270 00121          * ATTENTION : LE CODE CONTINUE
0271 00121          *
0272 00121
0273 00121          ************************************************************
0274 00121          * Etat 3
0275 00121          ************************************************************
0276 00121
0277 00121 7230      etat3  gosub  getchr
0278 00125 E42              rel(3) actionl  EOL
0279 00128 600              rel(3) t3-3     P | U
0280 0012B B00              rel(3) t3-4     0..9
0281 0012E
0282 0012E 79D0      t3-3   gosub  actiond
0283 00132 6EEF             goto   etat3
0284 00136
0285 00136 7A80      t3-4   gosub  actionb
0286 0013A          *       goto   etat4
0287 0013A          *
0288 0013A          * ATTENTION : LE CODE CONTINUE
0289 0013A          *
0290 0013A
0291 0013A          ************************************************************
0292 0013A          * Etat 4
0293 0013A          ************************************************************
0294 0013A
0295 0013A 7910      etat4  gosub  getchr
0296 0013E 260              rel(3) erreur   EOL
0297 00142 600              rel(3) t4-3     P | U
0298 00144 B00              rel(3) t4-4     0..9
0299 00147
0300 00147 70D0      t4-3   gosub  actione
0301 0014B 65DF             goto   etat3
0302 0014F
0303 0014F 7480      t4-4   gosub  actionc
0304 00153 66EF             goto   etat4
0305 00157
0306 00157
```

```
Page 006            FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0     traitement du flot d'entree

0308 00157
0309 00157          ************************************************************
0310 00157          * getchr
0311 00157          *
0312 00157          * But: lit un caractere et branche sur erreur s'il n'est pas
0313 00157          *   reconnu, puis le classe en trois categories et opere un
0314 00157          *   branchement a une adresse fournie par le module appelant
0315 00157          * Entree:
0316 00157          *   - RSTK = adresse de la table de debranchements
0317 00157          *   - addsp$ pointe sur deux quartets plus haut que le
0318 00157          *     caractere a lire.
0319 00157          *   - cptp$ = nombre de caracteres restant a lire (1..n)
0320 00157          * Sortie:
0321 00157          *   - A(B) = caractere lu
0322 00157          *   - P = 0
0323 00157          *   - cptp$ et addsp$ ajustes
0324 00157          * Abime: C(A)
0325 00157          * Appelle: CONVUC, DRANGE, TBLJMC
0326 00157          * Niveaux: 2 (CONVUC)
0327 00157          * Detail:
0328 00157          *   Les trois classes de lexemes sont :
0329 00157          *   - Fin de ligne
0330 00157          *   - P / U
0331 00157          *   - 0..9
0332 00157          *   Donc, l'appel doit etre de la forme :
0333 00157          *      GOSUB  getchr
0334 00157          *      REL(3) transition si fin de ligne
0335 00157          *      REL(3) transition si P ou U
0336 00157          *      REL(3) transition si 0..9
0337 00157          * Historique:
0338 00157          *   86/08/31: J.T. & P.D. conception & codage
0339 00157          ************************************************************
0340 00157
0341 00157 1F00000   getchr d1=(5) cptp$
0342 0015E 147              c=dat1 A
0343 00161 CE               c=c-1  A
0344 00163 145              dat1=c A
0345 00166 445              goc    saut0    Fin de Ligne
0346 00169
0347 00169 1D00             d1=(2) addsp$
0348 0016D 147              c=dat1 A
0349 00170 CE               c=c-1  A
0350 00172 CE               c=c-1  A
0351 00174 145              dat1=c A
0352 00177 135              d1=c
0353 0017A 14B              a=dat1 B
0354 0017D 8F00000          gosbvl =CONVUC
0355 00184 3105             lcasc  \F\
0356 00188 962              ?a#c   B
0357 0018B 92               goyes  saut1
0358 0018D 3155             lcasc  \U\
0359 00191 962              ?a#c   B
0360 00194 32               goyes  saut1
0361 00196 8F00000          gosbvl =DRANGE
0362 0019D 5F0              gonc   saut2
0363 001A0
0364 001A0 330000    erreur lc(4)  (=id)-(=eIFMT)  "Invalid Format"
0365 001A6 8D00000          goving =BSERR
0366 001AD
0367 001AD D2        saut2  c=0    A
0368 001AF E6               c=c-1  A
0369 001B1 540              gonc   st       B.E.T.
0370 001B4 D2        saut1  c=0    A
```

```
Page 007            FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0     traitement du flot d'entree

0371 001B6 E6               st     c=c-1  A
0372 001B8 540              gonc   saut
0373 001BB D2        saut0  c=0    A
0374 001BD 8D00000          saut   goving =TBLJMC
0375 001C4
```

```
Page 008            FINPUT : COMPILATION DU FORMAT (comp.as)
AREUH ASS. V2.0     Actions de l'automate

0377 001C4
0378 001C4          ************************************************************
0379 001C4          * actionb
0380 001C4          ************************************************************
0381 001C4
0382 001C4           actionb
0383 001C4 3103             lcasc  \0\
0384 001C8 B6A              a=a-c  B
0385 001CB 1F00000          d1=(5) mult
0386 001D2 149              dat1=a B
0387 001D5 01               rtn
0388 001D7
0389 001D7          ************************************************************
0390 001D7          * actionc
0391 001D7          ************************************************************
0392 001D7
0393 001D7           actionc
0394 001D7          *
0395 001D7          * si mult>10 alors "invalid format" finsi ;
0396 001D7          *
0397 001D7 AE8              b=a    B
0398 001DA 1F00000          d1=(5) mult
0399 001E1 14B              a=dat1 B
0400 001E4 31A0             lc(2)  10
0401 001E8 9E6              ?a>c   B
0402 001EB 5B               goyes  erreur
0403 001ED          *
0404 001ED          * mult := (mult*10) ...
0405 001ED          *
0406 001ED A64              a=a+a  B
0407 001F0 AE6              c=a    B
0408 001F3 A66              c=c+c  B       c(a) := mult * 4
0409 001F6 A66              c=c+c  B       c(a) := mult * 8
0410 001F9 A6A              a=a+c  B       a(a) := mult * 10
0411 001FC          *
0412 001FC          *                  ... + (carlu-'0') ;
0413 001FC          *
0414 001FC 3103             lcasc  \0\
0415 00200 B61              b=b-c  B
0416 00203 A60              a=a+b  B
0417 00206 149              dat1=a B
0418 00209 01               rtn
0419 0020B
0420 0020B          ************************************************************
0421 0020B          * actiond
0422 0020B          ************************************************************
0423 0020B
0424 0020B           actiond
0425 0020B          *
0426 0020B          * mult := 1 ;
0427 0020B          *
0428 0020B AE2              c=0    B
0429 0020E B66              c=c+1  B
0430 00211 1F00000          d1=(5) mult
0431 00218 14D              dat1=c B
0432 0021B          *       goto   actione     (en fait: gosub ; rtn)
0433 0021B          *
0434 0021B          * ATTENTION : LE CODE CONTINUE '
0435 0021B          *
0436 0021B
0437 0021B          ************************************************************
0438 0021B          * actione
0439 0021B          ************************************************************
```

```
Page 009            FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0     Actions de l'automate

0440 0021B
0441 0021B           actione
0442 0021B          *
0443 0021B          * p :* si carlu * 'P' alors 1 sinon 0 ;
0444 0021B          *
0445 0021B 3155              lcasc  \U\
0446 0021F B62               c*c-a  B
0447 00222 96A               ?c*0   B
0448 00225 80                goyes  actione-010
0449 00227 AE2               c*0    B
0450 0022A B66               c*c+1  B
0451 0022D           actione-010
0452 0022D          *
0453 0022D          * lastUP :* p ;
0454 0022D          *
0455 0022D 1F00000           d1*(5) lastUP
0456 00234 15D0              dat1*c 1
0457 00238          *
0458 00238          * bitmap (mult, p) ;
0459 00238          *
0460 00238 DA                a*c    A
0461 0023A 1D00              d1*(2) mult
0462 0023E 14F               c*dat1 B
0463 00241 6700              goto   bitmap  (en fait gosub ; rtn)
0464 00245
0465 00245 6A5F      Erreur  goto   erreur
0466 00249
```

```
Page 010            FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0     bitmap

0468 00249
0469 00249           ***************************************************
0470 00249           * bitmap
0471 00249           *
0472 00249           * But: voir algorithme en debut de fichier
0473 00249           * Entree:
0474 00249           *   - A(0) = p
0475 00249           *   - C(B) = n
0476 00249           * Abime: A(A), C(A), D0, D1
0477 00249           * Niveaux: 0
0478 00249           * Historique:
0479 00249           *   86/08/31: J.T. & F.D. conception & codage
0480 00249           *   86/11/29: J.T. & F.D. ajout du credit
0481 00249           ***************************************************
0482 00249
0483 00249            vark   equ    (=FUNCR0)+38
0484 00249            varp   equ    (=FUNCR0)+40
0485 00249
0486 00249            bitmap
0487 00249           *
0488 00249           * si n=0 alors retour ; finsi
0489 00249           *
0490 00249 96A               ?c*0   B
0491 0024C 00                rtnyes
0492 0024E           *
0493 0024E           * pour k := n jusque 1
0494 0024E           *
0495 0024E 1F00000           d1*(5) varp
0496 00255 1590              dat1*a 1
0497 00259 1D00              d1*(2) vark
0498 0025D
0499 0025D 14D        bm010  dat1*c B
0500 00260           *
0501 00260           *   faire
0502 00260           *     si cptmsk=0 alors "invalid format" ; finsi
0503 00260           *
0504 00260 1F00000           d1*(5) cptmsk
0505 00267 14F               c*dat1 b
0506 0026A 96A               ?c*0   B
0507 0026D 8D                goyes  Erreur
0508 0026F           *
0509 0026F           *     cptmsk-- ;
0510 0026F           *
0511 0026F A6E               c*c-1  B
0512 00272 14D               dat1*c B
0513 00275           *
0514 00275           *     si credit = 0 alors p := 1 ;
0515 00275           *
0516 00275 1F00000           d1*(5) credit
0517 0027C 14F               c*dat1 b        C(B) := credit en "Unprotected"
0518 0027F 1F00000           d1*(5) varp
0519 00286 96E               ?c=0   b
0520 00289 C0                goyes  bm015
0521 0028B A80               a*0    p
0522 0028E B04               a*a+1  p
0523 00291 1590              dat1*a 1
0524 00295           *
0525 00295           *     finsi
0526 00295           *
0527 00295            bm015
0528 00295           *
0529 00295           *     si p
0530 00295           *
```

```
Page 011            FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0     bitmap

0531 00295 15B0              a*dat1 1
0532 00299 908               ?a*0   P
0533 0029C 72                goyes  bm020
0534 0029E           *
0535 0029E           *       alors (admask) :* (admask) ou mask ;
0536 0029E           *
0537 0029E 1F00000           d1*(5) mask
0538 002A5 1574              c*dat1 S
0539 002A9 1D00              d1*(2) admask
0540 002AD 147               c*dat1 a
0541 002B0 135               d1*c
0542 002B3 1534              a*dat1 S
0543 002B7 0E4E              a*a!c  S
0544 002BB 1514              dat1*a S
0545 002BF 6940              goto   bm035
0546 002C3           *
0547 002C3           *       sinon
0548 002C3           *         si cptm$>0
0549 002C3           *
0550 002C3 1F00000    bm020  d1*(5) cptm$
0551 002CA 14F               c*dat1 B
0552 002CD 96A               ?c*0   B
0553 002D0 92                goyes  bm030
0554 002D2           *
0555 002D2           *           alors (addsu$**) :* (addsm$)
0556 002D2           *
0557 002D2 1D00              d1*(2) addsm$
0558 002D6 147               c*dat1 A
0559 002D9 135               d1*c
0560 002DC 1C1               d1*d1- 2
0561 002DF 14B               a*dat1 B          (addsm$)
0562 002E2 1F00000           d1*(5) addsu$
0563 002E9 147               c*dat1 A
0564 002EC 134               d0*c
0565 002EF E6                c*c+1  A
0566 002F1 E6                c*c+1  A
0567 002F3 145               dat1*c A          addsu$**
0568 002F6 14B               dat0*a B
0569 002F9                        '
0570 002F9           *         finsi ;
0571 002F9           *
0572 002F9           *
0573 002F9            bm030
0574 002F9           *
0575 002F9           *         credit-- ;
0576 002F9           *
0577 002F9 1F00000           d1*(5) credit
0578 00300 14F               c*dat1 b
0579 00303 A6E               c*c-1  b
0580 00306 14D               dat1*c b
0581 00309           *
0582 00309           *     finsi
0583 00309           *
0584 00309            bm035
0585 00309           *
0586 00309           *     addsm$** ;
0587 00309           *
0588 00309 1F00000           d1*(5) addsm$
0589 00310 147               c*dat1 A
0590 00313 CE                c*c-1  A
0591 00315 CE                c*c-1  A
0592 00317 145               dat1*c A
0593 0031A           *
```

```
Page 012            FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0     bitmap

0594 0031A           *     si cptm$>0 alors cptm$-- ; finsi
0595 0031A           *
0596 0031A 1D00              d1*(2) cptm$
0597 0031E 14F               c*dat1 B
0598 00321 A6E               c*c-1  B
0599 00324 450               goc    bm040
0600 00327 14D               dat1*c B
0601 0032A            bm040
0602 0032A           *
0603 0032A           *     mask :* mask >> 2 ;
0604 0032A           *
0605 0032A 1D00              d1*(2) mask
0606 0032E D2                c*0    A
0607 00330 15F0              c*dat1 1
0608 00334 81E               csrb             C(0) :* mask
0609 00337           *
0610 00337           *     si mask * 0
0611 00337           *       alors
0612 00337           *         mask :* 2^3 ;
0613 00337           *
0614 00337 90E               ?c#0   F
0615 0033A B1                goyes  bm050
0616 0033C 308               lc(1)  (%10)^3  C(0) * mask :* 2^3
0617 0033F           *
0618 0033F           *         admask-- ;
0619 0033F           *
0620 0033F 1D00              d1*(2) admask
0621 00343 143               a*dat1 A
0622 00346 CC                a*a-1  A
0623 00348 141               dat1*a A
0624 0034B           *
0625 0034B           *         (admask) :* 0 ;
0626 0034B           *
0627 0034B 131               d1*A
0628 0034E AC0               a*0    S
0629 00351 1514              dat1*a S
0630 00355           *
0631 00355           *     finsi
0632 00355           *
0633 00355 1F00000    bm050  d1*(5) mask
0634 0035C 15D0              dat1*c 1
0635 00360           *
0636 00360           *   finpour
0637 00360           *
0638 00360 1D00              d1*(2) vark
0639 00364 14F               c*dat1 B
0640 00367 A6E               c*c-1  B
0641 0036A 96A               ?c*0   B
0642 0036D 90                rtnyes
0643 0036F 6DEE              goto   bm010
0644 00373
```

```
Page 013          FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0   Action 1

0646 00373
0647 00373          ********************************************************
0648 00373          * action1                                              *
0649 00373          ********************************************************
0650 00373
0651 00373           action1
0652 00373          *
0653 00373          *   bitmap (cptmsk, lastUP) ;
0654 00373          *
0655 00373 1F00000          d1*(5) lastUP
0656 00378 15B0             a*dat1 1
0657 0037E 1F00000          d1*(5) cptmsk
0658 00385 14F              c*dat1 B
0659 00388 7DBE             gosub  bitmap
0660 0038C          *
0661 0038C          * U$(I) :* REV$(U$(I)) ;
0662 0038C          *
0663 0038C          *
0664 0038C          * len(U$(I)) :* ( addsu$ - GETU$I - 4) / 2
0665 0038C          *
0666 0038C 1F00000          d1*(5) 1
0667 00393 147              c*dat1 A
0668 00396 7000             gosub  *GETU$I
0669 0039A 1D00             d1*(2) addsu$
0670 0039E 143              a*dat1 a
0671 003A1 131              d1*a
0672 003A4 EE               c*a-c  A          adddu$ - GETU$I
0673 003A6 CE               c*c-1  A
0674 003A8 CE               c*c-1  A
0675 003AA CE               c*c-1  A
0676 003AC CE               c*c-1  A
0677 003AE AD0              a*0    M
0678 003B1 DA               a*c    A          a(a) :* 2 * LEN(U$(I)) ;
0679 003B3 81C              asrb              a(a) :* LEN(U$(I)) ;
0680 003B6 1583             dat0*a 4
0681 003BA 163              d0*d0* 4
0682 003BD          *
0683 003BD          * U$(I) :* REV$(U$(I)) ;
0684 003BD          *
0685 003BD AF1              b*0    W
0686 003C0 D8               b*a    A
0687 003C2 822              sb*0
0688 003C5 81D              bsrb
0689 003C8 136              cd0ex
0690 003CB C9               c*b*c  A
0691 003CD C9               c*b*c  A
0692 003CF 134              d0*c
0693 003D2 135              d1*c
0694 003D5 832              ?sb*0
0695 003D8 71               goyes  pair
0696 003DA 511              gonc   impair
0697 003DD 1C1       rptrev d1*d1- 2
0698 003E0 14A              a*dat0 B
0699 003E3 14F              c*dat1 B
0700 003E6 149              dat1*a B
0701 003E9 14C              dat0*c B
0702 003EC 161       impair d0*d0* 2
0703 003EF CD        pair   b*b-1  A
0704 003F1 5BE              gonc   rptrev
0705 003F4
0706 003F4 1F00000          d1*(5) 1
0707 003FB 147              c*dat1 a
0708 003FE 1F00000          d1*(5) *DOPED$
```

```
Page 014          FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0   Action 1

0709 00405 1B00000          d0*(5) *DOPEM$
0710 0040C 8E0000           gosubl *ELMCPY  D$(I) :* M$(I)
0711 00412
0712 00412 1F00000          d1*(5) 1
0713 00419 147              c*dat1 a
0714 0041C CE               c*c-1  a
0715 0041E 8AA              ?c*0   a
0716 00421 60               goyes  compfin
0717 00423 6AEB             goto   compiler
0718 00427
0719 00427           compfin
0720 00427          *
0721 00427          * Le code continue dans le fichier boucle.as
0722 00427          *
0723 00427                end
```

```
Page 015          FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0   **** SYMBOL TABLE ****

*BSERR          Extrn Ukn         -  0365
*CONVUC         Extrn Ukn         -  0354
*DOPED$         Extrn Ukn         -  0708
*DOPEI$         Extrn Ukn         -  0168
*DOPEM$         Extrn Ukn         -  0709
*DRANGE         Extrn Ukn         -  0361
*ELMCPY         Extrn Ukn         -  0710
 Erreur         00245 Rel  0465   -  0507
*FUNCR0         Extrn Ukn         -  0011 0012 0013 0014 0015 0016 0017 0018 0019
                                  *  0020 0021 0483 0484
 FiLeNd         00427 Rel  0723   -
*GETB$I         Extrn Ukn         -  0182
*GETFST         Extrn Ukn         -  0149 0233
*GETM$I         Extrn Ukn         -  0213
*GETP$I         Extrn Ukn         -  0141
*GETU$I         Extrn Ukn         -  0201 0668
*MFERR          Extrn Ukn         -  0222
*TBLJMC         Extrn Ukn         -  0374
*TRFMBF         Extrn Ukn         -  0023
 actiona-05     00066 Rel  0174   -  0172
 actiona-10     000DD Rel  0224   -  0219
 actionb        001C4 Rel  0382   -  0249 0285
 actionc        001D7 Rel  0393   -  0264 0303
 actiond        0020B Rel  0424   -  0246 0282
 actione        0021B Rel  0441   -  0267 0300
 actione-010    0022D Rel  0451   -  0448
 action1        00373 Rel  0651   -  0278
 addsm$         Unkwn Ukn  0016   -  0234 0557 0588
 addsp$         Unkwn Ukn  0013   -  0150 0347
 addsu$         Unkwn Ukn  0014   -  0206 0562 0669
 admask         Unkwn Ukn  0018   -  0187 0539 0620
 bitmap         00249 Rel  0486   -  0463 0659
 bm010          0025D Rel  0499   -  0643
 bm015          00295 Rel  0527   -  0520
 bm020          002C3 Rel  0550   -  0533
 bm030          002F9 Rel  0573   -  0553
 bm035          00309 Rel  0584   -  0545
 bm040          0032A Rel  0601   -  0599
 bm050          00355 Rel  0633   -  0615
 compfin        00427 Rel  0719   -  0716
 compiler       0000E Rel  0131   -  0717
 cptm$          Unkwn Ukn  0015   -  0228 0550 0596
 cptmsk         Unkwn Ukn  0019   -  0163 0504 0657
 cptp$          Unkwn Ukn  0012   -  0142 0341
 credit         Unkwn Ukn  0023   -  0175 0516 0577
*eIFMT          Extrn Ukn         -  0364
*eSTROV         Extrn Ukn         -  0221
 erreur         001A0 Rel  0364   -  0242 0260 0296 0402 0465
 etat1          000EF Rel  0241   -
 etat2          00108 Rel  0259   -  0265
 etat3          00121 Rel  0277   -  0247 0283 0301
 etat4          0013A Rel  0295   -  0304
 getchr         00157 Rel  0341   -  0241 0259 0277 0295
*id             Extrn Ukn         -  0364
 i              Unkwn Ukn  0011   -  0124 0139 0180 0199 0211 0666 0706 0712
 impair         003EC Rel  0702   -  0696
 lastUP         Unkwn Ukn  0021   -  0455 0655
 mask           Unkwn Ukn  0017   -  0156 0537 0605 0633
 mult           Unkwn Ukn  0020   -  0385 0398 0430 0461
*n              Extrn Ukn         -  0122
 pair           003EF Rel  0703   -  0695
 rptrev         003DD Rel  0697   -  0704
 saut           001BD Rel  0374   -  0372
```

```
Page 016          FINPUT : COMPILATION DU FORMAT <comp.as>
AREUH ASS. V2.0   **** SYMBOL TABLE ****

 saut0          001BB Rel  0373   -  0345
 saut1          001B4 Rel  0370   -  0357 0360
 saut2          001AD Rel  0367   -  0362
 st             001B6 Rel  0371   -  0369
 t1-2           00104 Rel  0249   -  0244
 t1-3           000FC Rel  0246   -  0243
 t2-2           00115 Rel  0264   -  0262
 t2-3           0011D Rel  0267   -  0261
 t3-3           0012E Rel  0282   -  0279
 t3-4           00136 Rel  0285   -  0280
 t4-3           00147 Rel  0300   -  0297
 t4-4           0014F Rel  0303   -  0298
 vark           Unkwn Ukn  0483   -  0497 0638
 varp           Unkwn Ukn  0484   -  0495 0518



Source   : comp.as

Object   : obj/comp.ao

Listing  : list/comp.al

Date     : Mon Mar 16 11:52:59 1987

Errors   : 000


Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986 Paris, France
```

```
Page 001            FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0

0001 00000                  title  FINPUT : BOUCLE PRINCIPALE <boucle.as>
0002 00000                  *
0003 00000                  * i contient le numero de la ligne en cours (1..n)
0004 00000                  *
0005 00000                   i      equ    (*LDCSPC)
0006 00000                  *
0007 00000                  * <861202.2120> : merci HP !
0008 00000                  * La variable i etait placee en FUNCR0. Comme la
0009 00000                  * documentation d'HP ne l'indique pas, CHEDIT ABIME TOUTE
0010 00000                  * LA FUNCTION SCRATCH, ainsi que DEUX QUARTETS DANS LA
0011 00000                  * STATEMENT SCRATCH.
0012 00000                  * Ceci expliquait un comportement "peu normal" de FINPUT.
0013 00000                  *
0014 00000
0015 00000                  *
0016 00000                  * menu est un flag pour indiquer si tous le display est
0017 00000                  * protege.
0018 00000                  *
0019 00000                   menu   equ    0
0020 00000
0021 00000                  *
0022 00000                  * Si, si ! Rappelez-vous, R3 contient toujours D0 !
0023 00000                  *
0024 00000                  * On va maintenant evaluer la variable A (dernier parametre
0025 00000                  * variable numerique). Les elements caracteristiques (pour
0026 00000                  * utiliser apres DEST) vont etre places a l'endroit DOPEA.
0027 00000                  *
0028 00000 11B              *       c=r3            c(a) :* PC
0029 00003 134              *       d0=c            d0 :* PC
0030 00006 8F00000          *       gosbvl =EXPEXC
0031 0000D 8F00000          *       gosbvl =AVE=D1
0032 00014 1B00000          *       d0=(5) =DOPEA
0033 0001B
0034 0001B AF4              *       a=b     w
0035 0001E 1507             *       dat0=a  w
0036 00021 16F              *       d0=d0+ 16       DOPEA + 0 :* B(W)    (16 q.)
0037 00025
0038 00025 137              *       cdiex
0039 00028 144              *       dat0=c  a
0040 0002B 164              *       d0=d0+ 5        DOPEA + 16 :* D1     (5 q.)
0041 0002E
0042 0002E 1F00000          *       d1=(5) =F-R1-0
0043 00035 147              *       c=dat1  a
0044 00038 144              *       dat0=c  a
0045 0003B 164              *       d0=d0+ 5        DOPEA + 21 :* F-R1-0   (5 q.)
0046 0003E
0047 0003E 17E              *       d1=d1+ 15
0048 00041 15F0             *       c=dat1  1
0049 00045 15C0             *       dat0=c  1       DOPEA + 26 :* F-R1-3   (1 q.)
0050 00049                  *
0051 00049                  * Voila, les elements caracteristiques de A sont sauves.
0052 00049                  *
0053 00049
0054 00049                  *
0055 00049                  * Pour que [ATTN] se comporte bien, il faut verifier que
0056 00049                  * les elements de M$ ne comportent pas de caracteres non
0057 00049                  * affichables (NULL, CR, LF, ESC, BS). Rappelons que
0058 00049                  * [ATTN] comparera le contenu du display avec M$(1).
0059 00049                  *
0060 00049 1F00000          *       d1=(5) =n
0061 00050 147              *       c=dat1  a
0062 00053
0063 00053 108              ver10   r0=c
```

```
Page 002            FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0

0064 00056 8E0000           *       gosubl =GETM$I  D0 :* ^ longueur M$(1)
0065 0005C D2               *       c=0     a
0066 0005E 15E3             *       c=dat0  4
0067 00062 D5               *       b=c     a
0068 00064 D2               *       c=0     a
0069 00066 3106             *       lc(2)   96
0070 0006A 8B1              *       ?b>c    a
0071 0006D 80               *       goyes   invcar  "Invalid Prompt"
0072 0006F 163              *       d0=d0+ 4
0073 00072 5E3              *       gonc    ver30   B.E.T.
0074 00075
0075 00075 330000    invcar lc(4)   (=id)-(=eIPRMP)  "Invalid Prompt"
0076 0007B 8D00000          *       govlng =BSERR
0077 00082                  *
0078 00082                  * Boucle interne de verification des caracteres un par un.
0079 00082                  *
0080 00082 14A       ver20  a=dat0 b
0081 00085 968              *       ?a=0    b       NULL
0082 00088 DE               *       goyes   invcar  "Invalid Prompt"
0083 0008A 31B1             *       lc(2)   27      C(B) :* ESC
0084 0008E 962              *       ?a=c    b
0085 00091 4E               *       goyes   invcar  "Invalid Prompt"
0086 00093 31D0             *       lc(2)   13      C(B) :* CR
0087 00097 962              *       ?a=c    b
0088 0009A BD               *       goyes   invcar  "Invalid Prompt"
0089 0009C 31A0             *       lc(2)   10      C(B) :* LF
0090 000A0 962              *       ?a=c    b
0091 000A3 2D               *       goyes   invcar  "Invalid Prompt"
0092 000A5 3180             *       lc(2)   08      C(B) :* BS
0093 000A9 962              *       ?a=c    b
0094 000AC 9C               *       goyes   invcar  "Invalid Prompt"
0095 000AE 161              *       d0=d0+ 2
0096 000B1 A6D       ver30  b=b-1   b
0097 000B4 5DC              *       gonc    ver20
0098 000B7                  *
0099 000B7 118              *       c=r0
0100 000BA CE               *       c=c-1   a
0101 000BC 8AE              *       ?c#0    a
0102 000BF 49               *       goyes   ver10
0103 000C1                  *
0104 000C1                  * Maintenant, on prepare la boucle principale de SINPUT.
0105 000C1                  *
0106 000C1 1F00000          *       d1=(5) i
0107 000C8 D2               *       c=0     a
0108 000CA E6               *       c=c+1   a
0109 000CC 145              *       dat1=c a        i :* 1 (numero de la ligne courante
0110 000CF
0111 000CF
```

```
Page 003            FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0     CORPS DE LA BOUCLE

0113 000CF                  ********************************************************
0114 000CF                  * BP:, BP-
0115 000CF                  *
0116 000CF                  * But: Boucle principale de SINPUT.
0117 000CF                  * Entree:
0118 000CF                  *    - BP:
0119 000CF                  *      - i (FUNCR0) = numero de la ligne courante (0..n-1)
0120 000CF                  *    - BP-
0121 000CF                  *      - i (FUNCR0) = numero de la ligne courante (0..n-1)
0122 000CF                  *      - D0 = ^ dope-vecteuuuuuuur du tableau a afficher
0123 000CF                  * Sortie:
0124 000CF                  *    - suivant les touches :
0125 000CF                  *      [^], [v], [g][^], [g][v] : passage a la ligne suivant
0126 000CF                  *        la direction, sans changer la ligne courante.
0127 000CF                  *      [ENDLINE] : valide la ligne courante. Si l'ecran est
0128 000CF                  *        constitue d'une seule ligne, termine SINPUT (comme
0129 000CF                  *        (L)INPUT). Dans ce dernier cas, la variable A est
0130 000CF                  *        mise a la valeur 1.
0131 000CF                  *      [RUN] : valide la ligne courante, et termine SINPUT.
0132 000CF                  *        La variable A contiendra le numero de cette ligne
0133 000CF                  *      [ATTN] :
0134 000CF                  *        1: remet la valeur par defaut de la ligne courante
0135 000CF                  *        2: si la ligne courante contient deja la valeur
0136 000CF                  *           par defaut, [ATTN] sort de SINPUT, la variable A
0137 000CF                  *           est mise a 0. La variable/tableau IS n'est pas
0138 000CF                  *           modifiee par SINPUT.
0139 000CF                  * Abime: CPU
0140 000CF                  * Appelle: CHEDIT, DSPCHA, GETxs1, GETFST, CURSFL
0141 000CF                  * Niveaux: 7 (CHEDIT)
0142 000CF                  * Note: le point d'entree BP- est utilise par kATTN
0143 000CF                  * Historique:
0144 000CF                  *   86/09/03: P.D. & J.T. conception & codage
0145 000CF                  *   86/11/22: P.D. & J.T. documentation
0146 000CF                  *   86/11/22: P.D. & J.T. gestion des touches speciales
0147 000CF                  *   86/11/24: P.D.        Modification pour ajout de BP-
0148 000CF                  ********************************************************
0149 000CF
0150 000CF                   BP:
0151 000CF 1B00000          *       d0=(5) =DOPED$
0152 000D6                   BP-
0153 000D6                  *
0154 000D6                  * Mise a zero du masque avant d'envoyer D$(i) a l'affichage
0155 000D6                  *
0156 000D6 1F00000          *       d1=(5) =DSPMSK
0157 000DD AF2              *       c=0     w
0158 000E0 15DF             *       dat1=c 16
0159 000E4 17F              *       d1=d1+ 16
0160 000E7 15D7             *       dat1=c 8
0161 000EB                  *
0162 000EB                  * Affichage du tableau D$(i). Les caracteres de D$(i) sont
0163 000EB                  * ranges en sens inverse, ce qui oblige a utilise une
0164 000EB                  * boucle speciale.
0165 000EB                  *
0166 000EB 1F00000          *       d1=(5) i
0167 000F2 147              *       c=dat1  a
0168 000F5                  *
0169 000F5                  * D0 = ^ DOPExs
0170 000F5                  *
0171 000F5 8E0000           *       gosubl =GETxs1
0172 000FB D2               *       c=0     a
0173 000FD 15E3             *       c=dat0  4
0174 00101 D7               *       d=c     a       D(A) :* LEN(x$(I))
0175 00103 8E0000           *       gosubl =GETFST
```

```
Page 004          FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0   CORPS DE LA BOUCLE

0176 00109 DB             c=d    a       C(A) := LEN(x$(I))
0177 0010B 132            ad0ex
0178 0010E 6020           goto   anpc10
0179 00112
0180 00112          *
0181 00112          * Invariant de cette boucle :
0182 00112          *   a(a) = pointeur dans x$(I)
0183 00112          *   c(a) = nombre de caracteres restant a afficher
0184 00112          *
0185 00112
0186 00112          *
0187 00112          * ancpsd = rev$("dspcma")
0188 00112          *
0189 00112 06        ancpsd rstk=c          RSTK := LEN(x$(I))
0190 00114 CC              a=a-1  a
0191 00116 CC              a=a-1  a
0192 00118 D6              c=a    a
0193 0011A 06              rstk=c         RSTK := " x$(I)
0194 0011C 130             d0=a
0195 0011F 14A             a=dat0 b
0196 00122 8F00000         gosbvl =DSPCHA
0197 00129 07              c=rstk         C(A) := " x$(I)
0198 0012B DA              a=c    a
0199 0012D 07              c=rstk         C(A) := LEN(x$(I))
0200 0012F CE        anpc10 c=c-1  a
0201 00131 50E             gonc   ancpsd
0202 00134          *
0203 00134          * Injection de B$(1) dans DSPMSK :
0204 00134          *
0205 00134 840             st=0   menu
0206 00137 1B00000         d0=(5) 1
0207 0013E 146             c=dat0 a
0208 00141 8E0000          gosubl =GETB$I
0209 00147 163             d0=d0- 4
0210 0014A 1F00000         d1=(5) =DSPMSK
0211 00151 15EF            c=dat0 16
0212 00155 15DF            dat1=c 16
0213 00159
0214 00159 B76             c=c-1  w
0215 0015C 550             gonc   inj10
0216 0015F          * les 64 premiers caracteres sont tous proteges
0217 0015F 850             st=1   menu
0218 00162           inj10
0219 00162 16F             d0=d0+ 16
0220 00165 17F             d1=d1+ 16
0221 00168 15E7            c=dat0 8
0222 0016C 15D7            dat1=c 8
0223 00170
0224 00170 27              p=     7
0225 00172 B16             c=c-1  wp
0226 00175 20              p=     0
0227 00177 450             goc    inj20
0228 0017A          * les 32 derniers caracteres ne sont pas proteges
0229 0017A B40             st=0   menu
0230 0017D           inj20
0231 0017D          *
0232 0017D          * Curseur a gauche avant l'edition
0233 0017D          *
0234 0017D 8F00000         gosbvl =CURSFL
0235 00184
0236 00184          ***************************************************
0237 00184          * edit
0238 00184          *
```

```
Page 005          FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0   CORPS DE LA BOUCLE

0239 00184          * But: faire l'edition de la ligne deja affichee.
0240 00184          * Entree:
0241 00184          *   - la ligne est deja affichee.
0242 00184          * Sortie:
0243 00184          *   - par les routines particulieres :
0244 00184          *       deplacement du curseur : kUP, kDOWN, kTOP, kBOT
0245 00184          *       validation et envoi : kENDLN, kRUN
0246 00184          *       sortie : kATTN, kOFF
0247 00184          * Appelle: BP10 ou BP20
0248 00184          * Historique:
0249 00184          *   86/11/23: P.D. & J.T. conception & codage
0250 00184          ***************************************************
0251 00184
0252 00184           edit
0253 00184 870             ?st=1  menu     tous les car. sont proteges ?
0254 00187 07              goyes  BP20     oui : affichage sans introduction
0255 00189                   *             non : INPUT normal
0256 00189
0257 00189          ***************************************************
0258 00189          * BP10
0259 00189          *
0260 00189          * But: realiser l'edition de la ligne deja affichee, et
0261 00189          *   envoyer sur les routines particulieres.
0262 00189          * Entree:
0263 00189          * Sortie:
0264 00189          *   - par les routines particulieres.
0265 00189          * Abime: A-D, D0, D1, R0, R3, ST, INSINP (flag pour le poll)
0266 00189          *   DEFADR, USRSTA, 32 nibbles at scratch
0267 00189          * Appelle: CHEDIT, FINDA
0268 00189          * Niveaux: 7
0269 00189          * Historique:
0270 00189          *   86/11/23: P.D. & J.T. conception & codage
0271 00189          ***************************************************
0272 00189
0273 00199           BP10
0274 00189          *
0275 00199          * Previent le Poll handler que c'est nous qu'on travaille !
0276 00189          *
0277 00189 1B00000         d0=(5) =INSINP
0278 00190 15A0            a=dat0 1
0279 00194 300             lc(1)  =INSMSK
0280 00197 0E0E            a=a!c  p
0281 0019B 1580            dat0=a 1
0282 0019F
0283 0019F          *
0284 0019F          * Y-a-queq'un ?
0285 0019F          *
0286 0019F 8F00000         gosbvl =CHEDIT
0287 001A6
0288 001A6          *
0289 001A6          * Previent le Poll handler que c'est nous qu'on travaille
0290 001A6          * plus !
0291 001A6          *
0292 001A6 300             lc(1)  -=INSMSK  (Complement a un de =INSMSK)
0293 001A9 816             c=rc
0294 001AC 1B00000         d0=(5) =INSINP
0295 001B3 1524            a=dat0 s
0296 001B7 0E46            a=a&c  s        Carry is not affected
0297 001BB 1544            dat0=c s        A(B) n'est pas modifie
0298 001BF          *
0299 001BF          * La carry est encore dans l'etat apres CHEDIT :
0300 001BF          *   Cy = 1 : touche speciale [ATTN], [^], etc.
0301 001BF          *   Cy = 0 : user terminating key
```

```
Page 006          FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0   CORPS DE LA BOUCLE

0302 001BF          *
0303 001BF 54C             gonc   edit
0304 001C2          *
0305 001C2          * Et dans A(A) : le code de la touche
0306 001C2          *
0307 001C2 8F00000         gosbvl =FINDA
0308 001C9 D0              con(2) 13
0309 001CB 081             rel(3) kENDLN
0310 001CE E0              con(2) 14
0311 001D0 201             rel(3) kATTN
0312 001D3 F0              con(2) 15
0313 001D5 F81             rel(3) kRUN
0314 001D8 21              con(2) 18
0315 001DA D80             rel(3) kUP
0316 001DD 31              con(2) 19
0317 001DF 570             rel(3) kDOWN
0318 001E2 41              con(2) 20
0319 001E4 6B0             rel(3) kTOP
0320 001E7 51              con(2) 21
0321 001E9 3C0             rel(3) kBOT
0322 001EC 81              con(2) 24
0323 001EE 351             rel(3) kOFF
0324 001F1 00              nibhex 00
0325 001F3          *
0326 001F3          * Aucune de celles-ci : on ne reconnait pas, on ignore.
0327 001F3          *
0328 001F3 609F            goto   edit
0329 001F7
0330 001F7          ***************************************************
0331 001F7          * BP20
0332 001F7          *
0333 001F7          * But: traiter le cas ou la ligne est entierement protegee
0334 001F7          *   C'est le cas particulier de MENU.
0335 001F7          * Entree: -
0336 001F7          * Sortie:
0337 001F7          *   - par les routines particulieres.
0338 001F7          * Abime: A-D, D0, D1, 32 nibbles at scratch
0339 001F7          * Appelle: CKSREQ, SCRLLR, POPBUF, FINDA
0340 001F7          * Niveaux: 6 (SCRLLR)
0341 001F7          * Historique:
0342 001F7          *   86/11/23: P.D. & J.T. conception & codage
0343 001F7          *   86/11/29: P.D. & J.T. ajout de RPTKY pour la repetition
0344 001F7          ***************************************************
0345 001F7
0346 001F7           BP20
0347 001F7 8F00000         gosbvl =RPTKY
0348 001FE 4E1             goc    BP25     Go if repeated key
0349 00201 8F00000   BP12  gosbvl =SCRLLR
0350 00208 5D0             gonc   BP15
0351 0020B 8F00000         gosbvl =CKSREQ
0352 00212 6EEF            goto   BP12
0353 00216 8F00000   BP15  gosbvl =POPBUF
0354 0021D D4        BP25  a=b    a       A(B) := keycode
0355 0021F 8F00000         gosbvl =FINDA
0356 00226 62              con(2) 38
0357 00228 321             rel(3) kENDLN
0358 0022B B2              con(2) 43
0359 0022D 5A0             rel(3) kATTN
0360 00230 E2              con(2) 46
0361 00232 231             rel(3) kRUN
0362 00235 23              con(2) 50
0363 00237 030             rel(3) kUP
0364 0023A 33              con(2) 51
```

```
Page 007          FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0   CORPS DE LA BOUCLE

0365 0023C 810             rel(3) kDOWN
0366 0023F 2A              con(2) 162
0367 00241 950             rel(3) kTOP
0368 00244 3A              con(2) 163
0369 00246 660             rel(3) kBOT
0370 00249 36              con(2) 99
0371 0024B 6F0             rel(3) kOFF
0372 0024E 00              nibhex 00
0373 00250          *
0374 00250          * La touche n'est pas reconnue.
0375 00250          *
0376 00250 633F            goto   edit
```

```
Page 008              FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0       TOUCHES DE DEPLACEMENT [^] [v] [g][^] [g][v]

0378 00254            ********************************************************
0379 00254            * kUP, kDOWN, kBOT, kDOWN
0380 00254            *
0381 00254            * But: traiter les touches de deplacement de curseur.
0382 00254            * Entree: -
0383 00254            * Sortie:
0384 00254            *    - par "curs"
0385 00254            *    - la variable 1 contient le numero de la nouvelle ligne.
0386 00254            * Abime: A-D, D0, D1, ST, 1
0387 00254            * Appelle: finlgn, curs
0388 00254            * Niveaux: 5 (finlgn)
0389 00254            * Historique:
0390 00254            *    86/11/23: P.D. & J.T. conception & codage
0391 00254            *    86/11/24: P.D.         ajout de documentation
0392 00254            ********************************************************
0393 00254
0394 00254            ********************************************************
0395 00254            * Touche [v]
0396 00254            ********************************************************
0397 00254
0398 00254 7D60        kDOWN  gosub  finlgn
0399 00258 1F00000     \DOWN  d1=(5) 1
0400 0025F 143                a=dat1 a
0401 00262 E4                 a+a-1  a
0402 00264 521                gonc   curs     B.E.T.
0403 00267
0404 00267            ********************************************************
0405 00267            * Touche [^]
0406 00267            ********************************************************
0407 00267
0408 00267 7A50        kUP    gosub  finlgn
0409 0026B 1F00000            d1=(5) 1
0410 00272 143                a=dat1 a
0411 00275 CC                 a+a-1  a
0412 00277            *
0413 00277            * Le code continue dans "curs"
0414 00277            *
0415 00277
0416 00277            ********************************************************
0417 00277            * curs
0418 00277            *
0419 00277            * But: gerer les deplacements dans les differentes lignes
0420 00277            *    d'une entree.
0421 00277            * Entree:
0422 00277            *    - A(A) = nombre de lignes
0423 00277            * Sortie:
0424 00277            *    - par BP: (Boucle Principale)
0425 00277            * Abime: A(A), C(A), D0
0426 00277            * Niveaux: 0
0427 00277            * Historique:
0428 00277            *    86/09/03: P.D. & J.T. conception & codage
0429 00277            *    86/11/22: P.D. & J.T. documentation
0430 00277            ********************************************************
0431 00277
0432 00277 D2          curs   c=0    a
0433 00279 E6                 c=c+1  a
0434 0027B 8BE                ?c(=a  a
0435 0027E 40                 goyes  curs10
0436 00280 DA                 a=c    a        a(a) := 1
0437 00282 1B00000     curs10 d0=(5) =n
0438 00289 146                c=dat0 a
0439 0028C 8BA                ?c>=a  a
0440 0028F 40                 goyes  curs20
```

```
Page 009              FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0       TOUCHES DE DEPLACEMENT [^] [v] [g][^] [g][v]

0441 00291 DA                 a=c    a
0442 00293 141         curs20 dat1=a a
0443 00296 683E               goto   BP:
0444 0029A
0445 0029A            ********************************************************
0446 0029A            * Touche [g][^]
0447 0029A            ********************************************************
0448 0029A
0449 0029A 7720        kTOP   gosub  finlgn
0450 0029E 1F00000            d1=(5) 1
0451 002A5 D0                 a=c    a
0452 002A7 E4                 a+a-1  a
0453 002A9 59E                gonc   curs20   B.E.T.
0454 002AC
0455 002AC            ********************************************************
0456 002AC            * Touche [g][v]
0457 002AC            ********************************************************
0458 002AC
0459 002AC 7510        kBOT   gosub  finlgn
0460 002B0 1F00000            d1=(5) 1
0461 002B7 1B00000            d0=(5) =n
0462 002BE 142                a=dat0 a
0463 002C1 61DF               goto   curs20
0464 002C5
0465 002C5            ********************************************************
0466 002C5            * finlgn
0467 002C5            *
0468 002C5            * But: terminer physiquement (pour l'affichage) une ligne
0469 002C5            *    saisie.
0470 002C5            * Entree: -
0471 002C5            * Sortie: -
0472 002C5            * Abime: A-D, D0, D1, ST, R3
0473 002C5            * Appelle: ILFART, CRLFND
0474 002C5            * Niveaux: voir la documentation de ILFART
0475 002C5            * Detail:
0476 002C5            *    La routine systeme FINLIN n'est pas utilisee. Ceci est
0477 002C5            *    du a son mode de fonctionnement : FINLIN "deprotege" le
0478 002C5            *    dernier caractere (le 96eme) et fait un CURSFR (cursor
0479 002C5            *    far right), ce qui a un desagreable effet sur la video.
0480 002C5            *    Le curseur balaye les 96 caracteres.
0481 002C5            *    Cette routine laisse le dernier caractere intact, l'HPIL
0482 002C5            *    ne balayera donc que ce qui est necessaire.
0483 002C5            * Historique:
0484 002C5            *    86/11/29: P.D. & J.T. conception & codage
0485 002C5            *    86/11/29: P.D. & J.T. documentation
0486 002C5            ********************************************************
0487 002C5
0488 002C5 8E0000      finlgn gosubl =ILFART  Voir le fichier pol.as
0489 002CB 8D00000            govlng =CRLFOF
```

```
Page 010              FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0       TOUCHES DE SORTIE [ATTN] [OFF]

0491 002D2            ********************************************************
0492 002D2            * kATTN, kOFF
0493 002D2            *
0494 002D2            * But: traiter les touches de sortie de SINPUT.
0495 002D2            * Entree: -
0496 002D2            * Sortie:
0497 002D2            *    - par EXIT, ou BP-
0498 002D2            *    - la variable A contiendra 0
0499 002D2            * Appelle: finlgn, BP- ou EXIT
0500 002D2            * Historique: -
0501 002D2            *    86/11/23: P.D. & J.T. conception & codage
0502 002D2            *    86/11/24: P.D.         ajout de documentation
0503 002D2            ********************************************************
0504 002D2
0505 002D2            ********************************************************
0506 002D2            * Touche [ATTN]
0507 002D2            ********************************************************
0508 002D2
0509 002D2 7FEF        kATTN  gosub  finlgn
0510 002D6 8F00000            gosbvl =ATNCLR  C'est magique ! Mais ca evite au
0511 002DD                      *             curseur de se mal retrouver.
0512 002DD            *
0513 002DD            * if display = M$(1) then EXIT avec 0 ;
0514 002DD            * DISP M$(1)
0515 002DD            *
0516 002DD 1B00000            d0=(5) 1
0517 002E4 146                c=dat0 a
0518 002E7 8E0000             gosubl =GETM$1  C(A) := D0 :* ^ LEN(M$(1))
0519 002ED D0                 a=0    a
0520 002EF 15A3               a=dat0 4        A(A) := LEN(M$(1))
0521 002F3 D8                 b=a    a        B(A) := LEN(M$(1))
0522 002F5 8E0000             gosubl =GETFST  D0 := ^ M$(1)[1,1]
0523 002FB
0524 002FB D2                 c=0    a
0525 002FD 3106               1c(2)  96
0526 00301 8BD                ?b(=c  a
0527 00304 40                 goyes  kATN10
0528 00306 D5                 b=c    a
0529 00308             kATN10
0530 00308            *
0531 00308            * B(A) = longueur en octets de ce qu'il y a a comparer
0532 00308            *
0533 00308 1F00000            d1=(5) =DSPBFS
0534 0030F 6910               goto   kATN30
0535 00313 181         kATN20 d0=d0- 2
0536 00316 14A                a=dat0 b
0537 00319 14F                c=dat1 b
0538 0031C 171                d1=d1- 2
0539 0031F 96A                ?c=0   b
0540 00322 41                 goyes  kATN40   arrive au bout de DSPBUF avant M$(1
0541 00324 966                ?c#a   b
0542 00327 F0                 goyes  kATN40   DSPBUF[j,j] # M$(1)[j,j]
0543 00329 CD          kATN30 b=b-1  a
0544 0032B 57E                gonc   kATN20
0545 0032E 14F                c=dat1 b
0546 00331 96A                ?c=0   b        arrive au bout de M$(1) avant DSPBUF
0547 00334 11                 goyes  exit
0548 00336             kATN40
0549 00336            *
0550 00336            * DISP M$(1)
0551 00336            *
0552 00336 1B00000            d0=(5) =DOPEM$  on reaffiche M$ sans rien changer
0553 0033D 689D               goto   BP-
```

```
Page 011              FINPUT : BOUCLE PRINCIPALE <boucle.as>
AREUH ASS. V2.0       TOUCHES DE SORTIE [ATTN] [OFF]

0554 00341
0555 00341            ********************************************************
0556 00341            * Touche [f][OFF]
0557 00341            ********************************************************
0558 00341
0559 00341             kOFF
0560 00341 708F               gosub  finlgn
0561 00345
0562 00345             exit
0563 00345 D0                 a=0    a
0564 00347 6421               goto   EXIT
```

```
Page 012            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     TOUCHES DE VALIDATION  [ENDLINE] [RUN]

0566 0034B          ********************************************************
0567 0034B          * kENDLN, kRUN
0568 0034B          *
0569 0034B          * But: traiter les touches de validation d'une ligne.
0570 0034B          * Entree: -
0571 0034B          * Sortie:
0572 0034B          *    - par EXIT, ou kDOWN-
0573 0034B          *    - la variable A contiendra le numero de la ligne
0574 0034B          *      courante pour [RUN] ou [ENDLINE] si une seule ligne.
0575 0034B          * Appelle: VALIDE, kDOWN- ou EXIT
0576 0034B          * Detail: si il n'y a qu'une seule ligne, SINPUT traite
0577 0034B          *   la touche [ENDLINE] comme [RUN].
0578 0034B          * Historique:
0579 0034B          *   86/11/23: P.D. & J.T. conception & codage
0580 0034B          *   86/11/24: P.D.        ajout de documentation
0581 0034B          ********************************************************
0582 0034B
0583 0034B          ********************************************************
0584 0034B          * Touche [ENDLINE]
0585 0034B          ********************************************************
0586 0034B
0587 0034B           kENDLN
0588 0034B 7250             gosub  VALIDE
0589 0034F 1B00000          d0=(5) =n
0590 00356 146              c=dat0 a
0591 00359 CE               c=c-1  a
0592 0035B 8AA              ?c=0   a
0593 0035E A0               goyes  kRUN00
0594 00360 67FE             goto   kDOWN-
0595 00364
0596 00364          ********************************************************
0597 00364          * Touche [RUN]
0598 00364          ********************************************************
0599 00364
0600 00364           kRUN
0601 00364 7930             gosub  VALIDE
0602 00368           kRUN00
0603 00368          *
0604 00368          * Copier le tableau U$ dans le tableau I$.
0605 00368          *
0606 00368          * Algorithme :
0607 00368          * for z:= n downto 1 do
0608 00368          *   begin
0609 00368          *     I$(z) := U$(z) ;
0610 00368          *   end ;
0611 00368          *
0612 00368 1F00000          d1=(5) =n
0613 0036F 147              c=dat1 a
0614 00372          *
0615 00372          * for z := n downto 0 do
0616 00372          *
0617 00372           kRUN10
0618 00372          *
0619 00372          * Assertion :
0620 00372          *   C(A) = z
0621 00372          *
0622 00372 108              r0=c
0623 00375
0624 00375 1B00000          d0=(5) =DOPEU$  D0 := ^ dope-vecteuuuuur source
0625 0037C 1F00000          d1=(5) =DOPEI$  D1 := ^ dope-vecteuuuuur destination
0626 00383 8E0000           gosubl =ELMCPY
0627 00389
0628 00389 118              c=r0
```

```
Page 013            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     TOUCHES DE VALIDATION  [ENDLINE] [RUN]

0629 0038C CE               c=c-1  a
0630 0038E 8AE              ?c#0   a
0631 00391 1E               goyes  kRUN10
0632 00393          *
0633 00393          *   end ;
0634 00393          *
0635 00393
0636 00393          *
0637 00393          * Renvoyer le numero de la ligne courante dans la var. A
0638 00393          * et revenir a Basic.
0639 00393          *
0640 00393 1F00000          d1=(5) 1
0641 0039A 143              a=dat1 a
0642 0039D 6EC0             goto   EXIT
```

```
Page 014            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     UTILITAIRES

0644 003A1          ********************************************************
0645 003A1          * VALIDE
0646 003A1          *
0647 003A1          * But: valider une ligne, c'est a dire copier l'entree de
0648 003A1          *   l'utilisateur dans U$(1)
0649 003A1          * Entree:
0650 003A1          *   - DSPBFS contient l'entree de l'utilisateur
0651 003A1          * Sortie:
0652 003A1          *   - U$(1) contient les seuls caracteres entres
0653 003A1          *   - D$(1) contient le display buffer tel quel
0654 003A1          * Abime:
0655 003A1          * Appelle: GETD$I, finlgn, DSP$00, POP1S, GETU$I, D1MSTK
0656 003A1          * Niveaux:
0657 003A1          * Detail:
0658 003A1          * Algorithme:
0659 003A1          *   adresse de D$(1)
0660 003A1          *   recopier DSPBFS-DSPBFE dans D$(1)
0661 003A1          *   finlgn
0662 003A1          *   DSP$00
0663 003A1          *   enlever le CR de la fin
0664 003A1          *   adresse de U$(1)
0665 003A1          *   mettre LEN(U$(1)) a jour
0666 003A1          *   recopier Math-Stack dans U$(1)
0667 003A1          * Historique:
0668 003A1          *   86/11/23: P.D. & J.T. conception & codage
0669 003A1          ********************************************************
0670 003A1
0671 003A1           VALIDE
0672 003A1          *
0673 003A1          * adresse de D$(1)
0674 003A1          *
0675 003A1 1B00000          d0=(5) 1
0676 003A8 146              c=dat0 a
0677 003AB 8E0000           gosubl =GETD$I
0678 003B1          *
0679 003B1          * recopier DSPBFS-DSPBFE dans D$(1)
0680 003B1          *
0681 003B1 1F00000          d1=(5) (=DSPBFE)-2
0682 003B8 D2               c=0    a
0683 003BA 3106             lc(2)  96          Nb de boucles
0684 003BE 5D0              gonc   vld020      B.E.T.
0685 003C1          *
0686 003C1          * boucle pour determiner la longueur de D$(1)
0687 003C1          *
0688 003C1 14B       vld010 a=dat1 b
0689 003C4 96C              ?a#0   b
0690 003C7 B0               goyes  vld030      sortie des qu'un caractere est
0691 003C9 1C1              d1=d1- 2           trouve
0692 003CC A6E       vld020 c=c-1  b
0693 003CF 51F              gonc   vld010
0694 003D2 B66       vld030 c=c+1  b
0695 003D5          *
0696 003D5          * C(A) = longueur en octets de la chaine lue dans DSPBUF
0697 003D5          *
0698 003D5 15C3             dat0=c 4           LEN(D$(1)) := C(B)
0699 003D9 163              d0=d0- 4
0700 003DC 5E0              gonc   vld050
0701 003DF          *
0702 003DF          * boucle de recopie
0703 003DF          *
0704 003DF 14B       vld040 a=dat1 b
0705 003E2 148              dat0=a b
0706 003E5 1C1              d1=d1- 2
```

```
Page 015            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     UTILITAIRES

0707 003E8 161              d0=d0- 2
0708 003EB A6E       vld050 c=c-1  b
0709 003EE 50F              gonc   vld040
0710 003F1          *
0711 003F1          * finlgn : DSP$00
0712 003F1          *
0713 003F1 70DE             gosub  finlgn
0714 003F5 8F00000          gosbvl =D1MSTK
0715 003FC 850              ST=1   0           Pour retour apres DSP$00
0716 003FF 8F00000          gosbvl =DSP$00
0717 00406 8F00000          gosbvl =POP1S
0718 0040D          *
0719 0040D          * Enlever le CR de la fin
0720 0040D          *
0721 0040D 171              d1=d1- 2
0722 00410 CC               a=a-1  a
0723 00412 CC               a=a-1  a           LEN(DSPBFS) := LEN(DSPBFS)-1 (octe
0724 00414 100              r0=a               sauver la longueur dans R0
0725 00417
0726 00417 1B00000          d0=(5) (=DOPEI$)+9 D0 := ^ longueur max des elemen
0727 0041E D2               c=0    a
0728 00420 15E3             c=dat0 4
0729 00424 C6               c=c+c  a           C(A) := LEN(DISP$) en quartets
0730 00426 8B6              ?a>c   a
0731 00429 83               goyes  strovf
0732 0042B          *
0733 0042B          * adresse de U$(1)
0734 0042B          *
0735 0042B 1B00000          d0=(5) 1
0736 00432 146              c=dat0 a
0737 00435 8E0000           gosubl =GETU$I  D0 := ^ LEN(U$(1))
0738 0043B          *
0739 0043B          * mettre LEN(U$(1)) a jour
0740 0043B          *
0741 0043B 118              c=r0               restaurer la longueur de la chaine
0742 0043E AD0              a=0    a           A(5) := 0
0743 00441 DA               a=c    a           A(A) := A(B) := longueur en octet
0744 00443 81C              asrb
0745 00446 1583             dat0=a 4
0746 0044A 163              d0=d0- 4
0747 0044D          *
0748 0044D          * recopier Math-Stack dans U$(1)
0749 0044D          *
0750 0044D          * recapitulatif :
0751 0044D          *   D0 = start of dest
0752 0044D          *   D1 = start of source
0753 0044D          *   A(A) = longueur en octets
0754 0044D          *
0755 0044D 132              ad0ex
0756 00450 133              ad1ex
0757 00453 DE               acex   a
0758 00455 136              cd0ex
0759 00458 C6               c=c+c  a
0760 0045A          *
0761 0045A          * D0 = start of source
0762 0045A          * D1 = start of dest
0763 0045A          * C(A) = longueur en quartets
0764 0045A          *
0765 0045A 8D00000          govlng =MOVEU3
0766 00461
0767 00461           strovf
0768 00461 3100             lc(2)  =eSTROV
0769 00465 8D00000          govlng =MFERR
```

```
Page 016            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     UTILITAIRES

0770 0046C
0771 0046C          ********************************************************
0772 0046C          * EXIT
0773 0046C          *
0774 0046C          * But: stocker la valeur de retour dans la variable A, et
0775 0046C          *   revenir a Basic par NXTSTM.
0776 0046C          * Entree:
0777 0046C          *   - A(A) = valeur a renvoyer (en hexa)
0778 0046C          *   - DOPEA = informations sauves lors de l'evaluation de A
0779 0046C          *   - AVMEME = sauvegarde du pointeur de Math-Stack sauve
0780 0046C          *         lors de l'evaluation de A.
0781 0046C          * Sortie:
0782 0046C          *   - par NXTSTM
0783 0046C          * Abime: ouh la la !
0784 0046C          * Appelle: DEST, STORE, NXTSTM, HDFLT, AVE=D1, ATNCLR,NOSCRL
0785 0046C          * Niveaux: 6 (STORE)
0786 0046C          * Historique:
0787 0046C          *   86/11/22: P.D. & J.T. conception & codage
0788 0046C          *   86/11/24: P.D.       ajout de documentation
0789 0046C          ********************************************************
0790 0046C
0791 0046C           EXIT
0792 0046C          *
0793 0046C          * Lors de l'evaluation de A, on avait sauvegarde ses
0794 0046C          * elements caracteristiques dans DOPEA. Il est maintenant
0795 0046C          * temps d'aller les rechercher, pour utiliser la sequence
0796 0046C          * DEST, STORE etc. cf IDS I, page 13-13
0797 0046C          *
0798 0046C 1B00000          d0=(5) (=DOPEA)+26   Variable A
0799 00473 1F00000          d1=(5) =F-R1-3
0800 0047A 15E0             c=dat0 1
0801 0047E 15D0             dat1=c 1
0802 00482
0803 00482 184              d0=d0- 5
0804 00485 1CE              d1=d1- 15
0805 00488 146              c=dat0 a
0806 0048B 145              dat1=c a
0807 0048E
0808 0048E 184              d0=d0- 5
0809 00491 146              c=dat0 a
0810 00494 135              d1=c
0811 00497
0812 00497 18F              d0=d0- 16
0813 0049A 1567             c=dat0 w
0814 0049E AF5              b=c    w
0815 004A1          *
0816 004A1          * Maintenant que tout est restaure, on reprend la sequence
0817 004A1          * habituelle de stockage dans une variable. cf IDS I.
0818 004A1          *
0819 004A1 8F00000          gosbvl =DEST    N'abime pas A(A)
0820 004A8 8F00000          gosbvl =D1MSTK
0821 004AF          *
0822 004AF          * TRACE OFF
0823 004AF          *
0824 004AF 1B00000          d0=(5) =S-R1-2
0825 004B6 D2               c=0    a
0826 004B8 144              dat0=c a
0827 004BB
0828 004BB 8F00000          gosbvl =HDFLT   A(W) := 1 en sortie (12 digits form)
0829 004C2 04               sethex
0830 004C4 1CF              d1=d1- 16
0831 004C7 1517             dat1=a w        Math-Stack := A(W)
0832 004CB 8F00000          gosbvl =AVE=D1
```

```
Page 017            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     UTILITAIRES

0833 004D2 8F00000          gosbvl =STORE   variable A := 1 en sortie
0834 004D9          *
0835 004D9          * C'est fini !
0836 004D9          *
0837 004D9 8F00000          gosbvl =ATNCLR  On n'a jamais appuye sur [ATTN]
0838 004E0 8F00000          gosbvl =NOSCRL  Le curseur est disponible
0839 004E7 8D00000          govlng =NXTSTM  Au revoir...
0840 004EE          *
0841 004EE          *   O U F !!!!!
0842 004EE          *
0843 004EE                  end
```

```
Page 018            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     **** SYMBOL TABLE ****

=ATNCLR         Extrn Ukn       -  0510 0837
=AVE=D1         Extrn Ukn       -  0031 0832
=BSERR          Extrn Ukn       -  0076
 BP             000D6 Rel  0152 -  0553
 BP10           00189 Rel  0273 -
 BP12           00201 Rel  0349 -  0352
 BP15           00216 Rel  0353 -  0350
 BP20           001F7 Rel  0346 -  0254
 BP25           0021D Rel  0354 -  0348
 BP:            000CF Rel  0150 -  0443
=CHEDIT         Extrn Ukn       -  0286
=CKSREQ         Extrn Ukn       -  0351
=CRLFOF         Extrn Ukn       -  0489
=CURSFL         Extrn Ukn       -  0234
=D1MSTK         Extrn Ukn       -  0714 0820
=DEST           Extrn Ukn       -  0819
=DOPEA          Extrn Ukn       -  0032 0798
=DOPED$         Extrn Ukn       -  0151
=DOPEI$         Extrn Ukn       -  0625 0726
=DOPEM$         Extrn Ukn       -  0552
=DOPEU$         Extrn Ukn       -  0624
=DSP$00         Extrn Ukn       -  0716
=DSPBFE         Extrn Ukn       -  0681
=DSPBFS         Extrn Ukn       -  0533
=DSPCHA         Extrn Ukn       -  0196
=DSPMSK         Extrn Ukn       -  0156 0210
=ELMCPY         Extrn Ukn       -  0626
=EXPEXC         Extrn Ukn       -  0030
 EXIT           0046C Rel  0791 -  0564 0642
=F-R1-0         Extrn Ukn       -  0042
=F-R1-3         Extrn Ukn       -  0799
=FINDA          Extrn Ukn       -  0307 0355
 FiLeNd         004EE Rel  0843 -
=GETB$1         Extrn Ukn       -  0208
=GETD$1         Extrn Ukn       -  0677
=GETFST         Extrn Ukn       -  0175 0522
=GETM$1         Extrn Ukn       -  0064 0518
=GETU$1         Extrn Ukn       -  0737
=GETx$1         Extrn Ukn       -  0171
=HDFLT          Extrn Ukn       -  0828
=ILFART         Extrn Ukn       -  0488
=INS1NP         Extrn Ukn       -  0277 0294
=INSMSK         Extrn Ukn       -  0279 0292
=LDCSPC         Extrn Ukn       -  0005
=MFERR          Extrn Ukn       -  0769
=MOVEU3         Extrn Ukn       -  0765
=NOSCRL         Extrn Ukn       -  0838
=NXTSTM         Extrn Ukn       -  0839
=POP1S          Extrn Ukn       -  0717
=POPBUF         Extrn Ukn       -  0353
=RPTKY          Extrn Ukn       -  0347
=S-R1-2         Extrn Ukn       -  0824
=SCRLLR         Extrn Ukn       -  0349
=STORE          Extrn Ukn       -  0833
 VALIDE         003A1 Rel  0671 -  0588 0601
 ancpsd         00112 Rel  0189 -  0201
 anpc10         0012F Rel  0200 -  0178
 curs           00277 Rel  0432 -  0402
 curs10         00282 Rel  0437 -  0435
 curs20         00293 Rel  0442 -  0440 0453 0463
=e!PRMP         Extrn Ukn       -  0075
=eSTROV         Extrn Ukn       -  0768
 edit           00184 Rel  0252 -  0303 0328 0376
```

```
Page 019            FINPUT : BOUCLE PRINCIPALE (boucle.as)
AREUH ASS. V2.0     **** SYMBOL TABLE ****

 exit           00345 Rel  0562 -  0547
 finign         002C5 Rel  0488 -  0398 0408 0449 0459 0509 0560 0713
=id             Extrn Ukn       -  0075
 i              Unkwn Ukn  0005 -  0106 0166 0206 0399 0409 0450 0460 0516 0640
                                +  0675 0735
 inj10          00162 Rel  0218 -  0215
 inj20          0017D Rel  0230 -  0227
 invcar         00075 Rel  0075 -  0071 0082 0085 0088 0091 0094
 kATN10         00308 Rel  0529 -  0527
 kATN20         00313 Rel  0535 -  0544
 kATN30         00329 Rel  0543 -  0534
 kATN40         00336 Rel  0548 -  0540 0542
 kATTN          002D2 Rel  0509 -  0311 0359
 kBOT           002AC Rel  0459 -  0321 0369
 kDOWN          00254 Rel  0398 -  0317 0365
 kDOWN-         00258 Rel  0399 -  0594
 kENDLN         0034B Rel  0587 -  0309 0357
 kOFF           00341 Rel  0559 -  0323 0371
 kRUN           00364 Rel  0600 -  0313 0361
 kRUN00         00368 Rel  0602 -  0593
 kRUN10         00372 Rel  0617 -  0631
 kTOP           0029A Rel  0449 -  0319 0367
 kUP            00267 Rel  0408 -  0315 0363
 menu           00000 Abs  0019 -  0205 0217 0229 0253
=n              Extrn Ukn       -  0060 0437 0461 0589 0612
 strovf         00461 Rel  0767 -  0731
 ver10          00053 Rel  0063 -  0102
 ver20          00082 Rel  0080 -  0097
 ver30          000B1 Rel  0096 -  0073
 vid010         003C1 Rel  0688 -  0693
 vid020         003CC Rel  0692 -  0684
 vid030         003D2 Rel  0694 -  0690
 vid040         003DF Rel  0704 -  0709
 vid050         003EB Rel  0708 -  0700



Source   : boucle.as

Object   : obj/boucle.ao

Listing  : list/boucle.al

Date     : Mon Mar 16 11:53:29 1987

Errors   : 000


Areuh Assembler/Linker V2.0, (c) P. David & J. Taillandier 1986 Paris, France
```

ANAGRAMMES ET PERMUTATIONS

Dans cet article nous nous proposons de résoudre un problème qui s'énonce simplement mais dont la solution n'est pas immédiate. Il s'agit, étant donnés n éléments, de trouver les (n!) combinaisons possibles (par exemple pour n=3 : 012-021-102...). Ceci peut avoir de nombreuses applications quand on veut couvrir tous les cas de figure ; nous verrons pour notre part comme application un programme qui donne tous les anagrammes d'un prénom.

Le problème se ramène à trouver tous les nombres de n chiffres comportant une fois et une seule fois les chiffres 0, 1, 2 ... n-1.

Voyons tout d'abord une première méthode, simpliste et longue d'exécution, mais de programmation facile. Si l'on se place dans le cas n=5, le premier nombre sera 01234 et le suivant sera le premier nombre dont le produit des chiffres ajoutés d'un fait 5! et ainsi de suite, le dernier nombre étant 43210. Nous avons donc intérêt dans le programme à nous placer en base 5. Il faut dans un premier temps créer une routine qui convertit un nombre en base dix en une autre base. Cette routine étant effectuée de nombreuses fois il est conseillé de la faire la plus rapide possible (si possible en assembleur pour ceux qui savent). Voici celle que je vous propose :

```
130 SUB BASE(X,B,Y) @ Y=0
140 IF X#0 THEN M=INT(LOG(X)/LOG(B)) @
    Y=Y+INT(X/B^M)*10^M @ X=MOD(X,B^M) @ GOTO 150
150 END SUB
```

Prenons un exemple :

```
CALL BASE(3,2,Y) @ Y
```

nous donne la valeur de 3 en base 2. Le programme en entier est donc :

```
10 DESTROY ALL @ INPUT "Nbre de chiffres : ";N
20 Z=FACT(N)
30 FOR I=1 TO N-1 @ A=A+I*N^I @ B=B+I*N^(N-I-1) @
   NEXT I
40 I=B
50 X=I @ CALL BASE(X,N,Y) @ A$=STR$(Y) @
   A$="000000000"[1,N-LEN(A$)]&A$
60 Y=1 @ FOR J=1 TO N
70 Y=Y*(VAL(A$[J,J])+1)
80 NEXT J
90 IF Y=Z THEN 110
100 I=I+N-1 @ GOTO 50
110 K=K+1 @ PRINT STR$(K)&" : "&A$
120 IF I>=A THEN END
```

les lignes 130, 140 et 150 ont déjà été données.

Comme on s'en rend compte en exécutant ce programme, sa lenteur le condamne définitivement.

La seconde méthode est très différente de la première. Reprenons le cas n=5. Il s'agit d'augmenter le $i^{ème}$ chiffre de notre nombre de manière à ce qu'il soit différent de ceux qui le précédent et de telle sorte qu'il soit inférieur à 5 (dans notre exemple), puis on passe au rang i+1 et ainsi de suite. Le programme devient alors :

```
10 INPUT "Nbre de chiffres :";N @ IF N>=10 OR N<=1
   THEN 10
20 A$="0123456789"[1,N] @ I=N-1 @ B=0
30 IF I=N-1 AND B=0 THEN DISP A$
40 IF B=1 THEN K=-1 ELSE K=VAL(A$[I,I])
50 K=K+1 @ IF POS(A$[1,I-1],STR$(K)) AND K#N
   THEN 50
60 IF K#N THEN B=1 @ A$=A$[1,I-1]&STR$(K) ELSE B=0
70 IF B=1 THEN B=I#N @ I=I-(I=N)+(I#N) @ GOTO 30
80 I=I-1 @ IF I=0 THEN END ELSE 30
```

Enfin une dernière remarque pour que tout le monde puisse comprendre le programme : la différence de deux nombres composés uniquement des n-1 premiers entiers est supérieure à n-1 (ceci nous évite des boucles inutiles).

L'application promise est simplement la transposition du programme précédent à des chaînes de caractères. Les modifications à faire sur le programme sont les suivantes :

```
10 INPUT "Prénom : ";P$ @ N=LEN(P$) @
   IF N>=10 OR N<=1 THEN 10
```

Changer au pas `30 DISP A$` par `GOSUB 90` et ajouter :

```
90 C$="" @ FOR W=1 TO N
100 C$=C$&P$[VAL(A$[W,W])+1,VAL(A$[W,W])+1]
110 NEXT W @ DISP C$
120 RETURN
```

Voilà ce sera tout pour aujourd'hui, vous pouvez aller jouer maintenant.

Alain Herreman (200)

POUR DEBUTER
AVEC GOSUB, GOTO, RETURN

Soit un jeu de questions à réponses multiples. Dix questions et trois solutions sont proposées. On y répond par un chiffre qui correspond au rang de la réponse qui vous paraît bonne. Si la réponse est incorrecte, Titan vous donnera le bon numéro.

Après chaque réponse, un GOSUB vous envoie sur la ligne où commence le traitement de votre réponse. A l'intérieur du traitement de votre réponse, un GOTO permet de vous indiquer la bonne réponse. Un deuxième GOTO vous renvoie sur la suite des questions.

Vous trouverez le programme JEU listé avec les autres Basic. En espérant que cet article aidera à la compréhension du Basic. Le programme est commenté ce qui permet de suivre les GOSUB, GOTO et RETURN.

Alain Gillet (199)

Programme "JEUX" (Questionnaire a Choix Multiples)

```
  - JEUX
    PROGRAMME DE CONNAISSANCE
    N : Variable contenant vôtre score
    R : Variable contenant le numéro de la réponse choisie
    T$ : Variable stop-encore
    On règle Titan
 70 DELAY .9,.250
 80 FIX 0
  - On initialise la variable du compteur
100 N=0
  - On affiche la règle du jeux
120 PRINT "       Bonjour.Ce programme va vous permettre de tester vos connaissances."
130 PRINT "    Régles du jeux."
140 PRINT "    Dix questions d'ordre général."
150 PRINT "    Trois solutions proposées."
160 PRINT "    Répondre par 1,2,3."
170 PRINT "    Chaque bonne réponse donne deux points."
  - Le jeux commence. Bonne chance
190 PRINT "    1Q: Quel est le président actuel de la république?"
200 PRINT "    1)Mr Miterrand 2)Mr Rocard 3)Mr Chirac"
210 WAIT 5
  - Le temps de réflexion est fini, allons voir si la réponse est bonne.
230 GOSUB 710
240 PRINT "    2Q: Durée d'un mandat législative?"
250 PRINT "    1)4 ans 2)7ans 3)5ans"
260 WAIT 5
270 GOSUB 870
280 PRINT "    3Q: Vitesse commerciale du T.G.V.?"
290 PRINT "    1)260 2)300 3)200"
300 WAIT 5
310 GOSUB 710
320 PRINT "    4Q: Quel est le nom du moteur de Concorde?"
330 PRINT "    1)Olympe 2)Olympus 3)Olympia"
340 WAIT 5
350 GOSUB 810
360 PRINT "    5Q: Quel est l'âge maximal de la scolarité?"
370 PRINT "    1)14 ans 2)18 ans 3)16 ans"
380 WAIT 5
390 GOSUB 870
400 PRINT "    Quel est l'âge civile de la majorité?"
410 PRINT "    1)18 ans 2)21 ans 3)16ans"
420 WAIT 5
430 GOSUB 810
440 PRINT "    7Q: Un kilo octet c'est?"
450 PRINT "    1)1024 octets 2)1000 octets 3)992 octets"
460 WAIT 5
470 GOSUB 710
480 PRINT "    7Q: Un  général une étoile est?"
490 PRINT "    1)Un général d'armée 2)Un préfet mobilisé 3)Ca n'existe pas"
500 WAIT 5
510 GOSUB 810
520 PRINT "    9Q: Lorsqu'il est 12h à Paris il est à Tokyo:"
530 PRINT "    1)21h 2)20h 3)9h"
540 WAIT 5
550 GOSUB 710
```

```
560 PRINT "10Q:     Mr Freud médecin et psychologue est né le?"
570 PRINT "    1)1856 2)1910 3)1800"
580 WAIT 5
590 GOSUB 710
  - On affiche votre score final
610 PRINT "    Votre note est ";N;"/20"
  - On recommence!
630 INPUT "    Voulez-vous essayer à nouveau (O/N)?";T$
  - Si oui on y va
650 IF T$="O" THEN GOTO 100
  - Ici on remet Titan dans son état préféré.
670 FIX 4
680 DELAY .5,.125
690 END
```

==========

```
  - Votre réponse s.v.p. .
710 INPUT "Réponse :";R
  - Si votre réponse est 2 ou 3 alors perdu
730 IF R=2 OR R=3 THEN GOTO 780
  - Bravo pour votre bonne réponse
750 IF R=1 THEN PRINT "Gagné" @ BEEP 1500
  - On tient le score à jour
770 N=N+2 @ GOTO 800
780 PRINT "Perdu.C'était 1" @ BEEP 2500
  - Titan a digéré votre réponse, les questions continuent.
800 RETURN
810 INPUT "Réponse :";R
820 IF R=1 OR R=3 THEN GOTO 850
830 IF R=2 THEN PRINT "Gagné" @ BEEP 1500
840 N=N+2 @ GOTO 860
850 PRINT "Perdu.C'était 2." @ BEEP 2500
860 RETURN
870 INPUT "Réponse :";R
880 IF R=1 OR R=2 THEN GOTO 910
890 IF R=3 THEN PRINT "Gagné" @ BEEP 1500
900 N=N+2 @ GOTO 920
910 PRINT "Perdu.C'était 3." @ BEEP 2500
920 RETURN
```

LE COIN DES LHEX

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

Rappelons ce qu'est un fichier Lex : c'est un programme pour le HP71, en assembleur, qui apporte de nouvelles fonctions. Celles-ci sont utilisables directement, ou dans des programmes Basic.

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

Il suffit de recopier le petit programme basic "MAKELEX" ci-dessous, de le lancer et de recopier les codes du fichier Lex désiré. Quand vous avez fini, les nouvelles fonctions sont accessibles, après avoir éteint et rallumé votre HP71.

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

Vous trouverz donc, outre le Lex SAISYLEX de la rubrique assembleur, le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !").

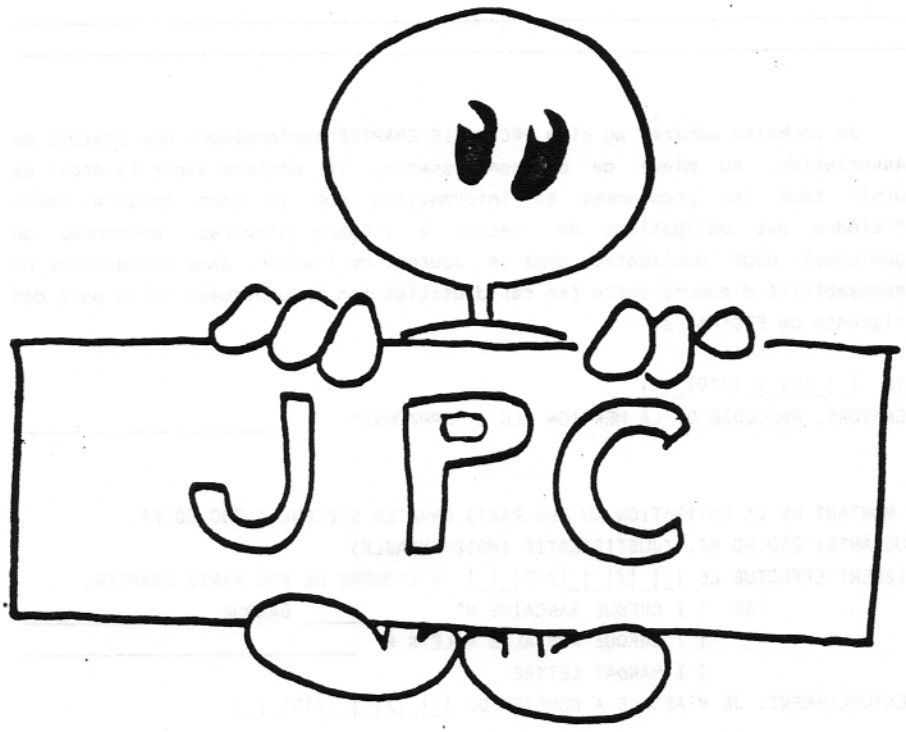
```
CHARLEX

SAISYLEX  FINPUT    225093
```

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

```
CHARLEX  ID#E1  624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E008012713078 6B2
002: 5E4001E000000000 9F6
003: FE00000000800001F D50
004: F31BF961400032BF 0E3
005: 38F14A11DB10AD23 47D
006: 07D532BFB8FD7911 830
007: 11AD754D7A101743 BB3
008: 11014D1CB15D0000 F1E
009: 71450375FF864834 29B
00A: 5655581008355654 5F2
00B: 5810002455565870 93F
00C: 0026555658700836 C91
00D: 5556581008364545 FE7
00E: 4A30000A49724000 33A
00F: 0808094A2C180814 6A3
010: A464242008355455 9FD
011: 581000054C714000 D43
012: 0C3142404C700832 09F
013: 41414A70002078A0 3F7
014: 2F30000000000000 722
015: 0000000000000000 A32
016: 0000000000000000 D42
017: 0000000000000000 052
018: 0000000000000000 362
019: 0000000000000000 672
01A: 0000000000000000 982
01B: 0000000000000000 C92
01C: 0000000000000000 FA2
01D: 0000000000000000 2B2
01E: 0000000000000000 5C2
01F: 0000000000000000 8D2
020: 0000000000000000 BE2
021: 000000000000080C F0D
022: 1A28080008080A2C 277
023: 180008040E340800 5C0
024: 08001E3018000000 8FA
025: 0000000000000000 C0A
026: 0000000000000000 F1A
027: 0000000000000000 22A
028: 0201000000010200 540
029: 0000000201020000 855
02A: 0001000100000002 B69
02B: 0102010000000000 E7D
02C: 0000000000000000 18D
02D: 045E755142400101 4D9
02E: 0101010000000000 7EC
02F: 0000000000000000 AFC
030: 0000070507000000 E1F
031: 0000000008344C4 15D
032: 44400D7901112D70 4BD
033: 050D750509700000 807
034: 0D70000000384540 B4A
035: 4020014E322E3140 E9E
036: 084E794142400000 1EE
037: 000000000002E4559 52C
038: 3200000000000000 841
039: 0000000000000026 B59
03A: 5556587008365556 EB8
03B: 5810083645464830 209
03C: 0832414248700024 54A
03D: 5655587008345655 8A7
03E: 5810083446454830 BF6
03F: 0C3042414C700024 F4B
040: 5556587008355654 2A8
041: 5810083546444830 5F7
042: 0C3142404C700025 94D
043: 5455587008355455 CA7
044: 5810083544454830 FF5
045: 0C3140414C700875 357
046: 14141870000A4972 6A8
047: 40000E3159454E30 A08
048: 0C7A0F7949400024 D80
049: 5554587000084A71 0DC
04A: 40000C523A262D10 43D
04B: 0424587458400875 794
04C: 1415187000094A70 AE4
04D: 4000083544454830 E28
04E: 0C3140414C300C74 190
04F: 5655545000054C71 4E7
050: 40000            5E0

SAISYLEX ID#E1 2226 octets

     0123456789ABCDEF sm

000: 3514943595C45485 36D
001: 802E009012713078 6C2
002: 961101ED5D500000 A2B
003: F7100BF006410000 D8C
004: 063500DB6494E405 105
005: 5545D51FF8DF3E20 4BA
006: 7F708FB39408FEA2 874
007: 304008DC5E208598 BF4
008: FB323076DF7C5087 F99
009: 38C79CF13610A134 324
00A: D28F82D207D30873 6B8
00B: D111A136AC0A4C15 A4C
00C: 04134779F7D10870 DC0
00D: C086370831908D66 138
00E: E208D271308D9DF3 4DA
00F: 031C28D5EC207920 860
010: 7DEF712075EF1707 C06
011: 6108FCE25073DF77 FAC
012: 008D003508D22950 30E
013: 407001604324496D 664
014: 637C5140ECE50527 9F1
015: F6D60747C5150ECE DA4
016: 564F627D61647CF0 13D
017: 702651627E8ECFF3 4E5
018: 1B19616431BF9619 868
019: 09698000662211B1 BBD
01A: 351121CD1378B6C1 F40
01B: 13510B3D14A305E4 2BA
01C: 94640215DD001BB8 63E
01D: 9F215A03010E0690 9AE
01E: 85B110D231908A2E D32
01F: 131608A2D031F08A 0AF
020: 28000679061911BE 418
021: 74F2AF014AE4D881 7C6
022: C81C34755F2E2134 B54
023: D431F5B6249CAEAA F1D
024: E831300E66962A22 298
025: 1A6C4C023A6C4402 623
026: 780CF2015240E469 9A1
027: 469218081D81D5F0 D24
028: 1564B46531180A6D 09D
029: 5FE8408210100769 40B
02A: 031F5AE51FE35F2A 7C2
02B: F01B045F26F2094C B5A
02C: C0B441564160AC50 ED7
02D: E4494DD014F96E31 27C
02E: AE41C1A44A6D50D0 628
02F: 096C50AE41BE74F2 9D4
030: B64148193714EB6E D68
031: 550AE219C714C198 0F8
032: 715630A84B841091 464
033: 5436F5F1B476F214 7F8
034: 28A800DB10B248FD B9C
035: D4101BB74F230115 F17
036: C031307020201BB7 278
037: 4F230015C0248F8A 5FD
038: 41011BD701AEA1B4 990
039: 76F214606011F969 D06
03A: F2AC215541707200 074
03B: F007145850821013 3C7
03C: 0E1BB89F215A00E0 766
03D: 615C000D77621136 ACA
03E: 137134DB7711D015 E3D
03F: B3E4E4C4D81338D8 1F2
040: 03108F681F0AF131 575
041: F196603172763117 8D7
042: 77D311C378211C41 C4F
043: 57494E9023B15201 FBA
044: 7D5C331F0966768F 35A
045: 83DB034000028BEB 6E5
046: 581C1C31593D872D A78
047: 0137135D579C0E51 DF3
048: 371F995F21451FAF 19B
049: 8F21432391490201 4F2
04A: 35012033601E8DA9 865
04B: 3908D6CC7131528D BF8
04C: 393901BFD8F26630 F8C
04D: 1B2D8F26B201B5C8 332
04E: F260201BB88F2651 6BB
04F: 01BE78F26A001B17 A4C
050: 8F2CED5AF2D9168A E23
051: F015A3E4E47A92C4 1CF
052: 184146C213401D21 52C
053: 5E3E6E6C6132C213 8C7
```

```
054: 401BF1BF1BF1BF1B C92
055: F1017EEFD215F3C1 04A
056: 017DDF147D501F8B 3F6
057: FFA0BFF8F725F05F 7DE
058: 033701E8DA939013 B57
059: 613513030E170153 EA3
05A: 0906E01701431315 1F1
05B: AE1C0AF114B31F19 59E
05C: 66C3172798F17770 91E
05D: 9F133131E0D81C37 CA8
05E: 17F1C4157494E902 02C
05F: 3B15206E4068BE31 3B1
060: F0966B1172764F17 738
061: 77D4F133E0D86A10 AD4
062: 31FF9663D176742F E71
063: 1C47B2FBF1BF1BF1 245
064: BF1E5AF91F178F21 608
065: 5DC24A82201EAF8F 9C2
066: 1458FBF190161739 D47
067: D1FE78F2AF4159C1 10C
068: 611524160948A172 466
069: 7D16113610B69C08 7E0
06A: D4058113610B8FE4 B6A
06B: 5911BE78F2D215E3 F0C
06C: 195C144762ED015A 296
06D: 38FFACE004814814 634
06E: 8148148F064A1304 99F
06F: 8163103A62D07F9F D2E
070: 810A4E5CE31057E8 0CB
071: F3155768FD23071C 45E
072: 315D31B5C8F2146C 7F7
073: E8AE09AF1307A851 BA4
074: 371357AFDD51BE78 F59
075: F2D215E37AEDC18F 328
076: 8BB811BFD8F2AF41 6F6
077: 58C8F53410AF2324 A83
078: A11BAF8F2AF01427 E34
079: 520D197D911B495F 1C9
07A: 2146C21641428B60 52D
07B: 18DD44908DBBCE01 8E8
07C: 841421441BAF8F21 C6F
07D: F5C8F2AF1D231C0D 036
07E: 57E5DD87B5D146C1 3ED
07F: AF915DC17CAF231C 7B5
080: 1AFED7AF214678AF B87
081: CBDAAF1D23106D57 F49
082: 02DD87D1D146C1AF 301
083: 915DCAF2314CAFED 6DB
084: 7AF21467D6FCBDA1 AA8
085: DB8AF1D23106D571 E4F
086: ECD87EDC146C1AF9 235
087: 15DC1FAF8F21471D 5F1
088: B91451FB98F21477 98B
089: 23C1F0A8F2D215E3 D2F
08A: 145768C1D5A14530 0A9
08B: 81D6B1550D231061 41A
08C: DCB14D1BA78F2D01 7DC
08D: 5A38B240DA1DFF14 B90
08E: 91BB98F21467CEBD F57
08F: AD231B1C21D7B145 2F2
090: 134AC215441BB98F 68A
091: 21467ECBE6E6E6E6 A54
092: 1DAA1451BB98F214 DF8
093: 67CBBD015A3D2310 194
094: 68BAD031528D3939 528
095: 01DFA14971DB1D1B 8D8
096: 1457460DA0600B00 C40
097: 7B0160207CB07B40 FBB
098: 490E003007EB06EE 34E
099: F7AF07230E42600B 6E1
09A: 0079D06EEF7A8079 A8B
09B: 10260600B0070D06 DDD
09C: 5DF748066EF1F0A8 196
09D: F2147CE1454451D5 523
09E: A147CECE14513514 8B8
09F: B8FAA25131059629 C45
0A0: 23155962028F670B FB5
0A1: 15F033501E8DA939 344
0A2: 0D2E6540D2E6540D 6DC
0A3: 28D624203103B6A1 A49
0A4: FEB8F214901AE81F E0A
0A5: EB8F214B31A09E65 1B6
0A6: BA64AE6A66A66A6A 581
0A7: 3103B61A6014901A 8E8
0A8: E2B661FEB8F214D3 CA7
0A9: 155B6296A80AE2B6 044
0AA: 61F0C8F215D0DA1D 3F8
0AB: EB14F67006A5F96A 7A9
0AC: 001F3C8F215901D1 B2B
0AD: C14D1FCB8F214F96 EED
0AE: A8DA6E14D1FFF8F2 2D2
0AF: 14F1F3C8F296EC0A 692
0B0: 80B04159015B0908 9F8
0B1: 721F6B8F215741D7 D8D
0B2: B14713515340E4E1 100
0B3: 51469401FFA8F214 490
0B4: F96A921D1B147135 81E
0B5: 1C114B1FAA8F2147 BBF
0B6: 134E6E61451481FF F51
0B7: F8F214FA6E14D1F1 30F
0B8: B8F2147CECE1451D 6CC
0B9: FA14FA6E45014D1D A81
0BA: 6BD215F081E90EB1 E2A
0BB: 3081D7B143CC1411 1A8
0BC: 31AC015141F6B8F2 53A
0BD: 15D01D1C14FA6E96 8E3
0BE: A006DEE1F0C8F215 C98
0BF: B01FCB8F214F7DBE 073
0C0: 1FB98F214778D81D 421
0C1: AA143131EECECECE 802
0C2: CEAD0DA81C158316 BB8
0C3: 3AF1D882281D136C F55
0C4: 9C91341358327151 2B6
0C5: 11C114A14F14914C 633
0C6: 161CD5BE1FB98F21 9F1
0C7: 471F2D8F21BE78F2 DA8
0C8: 8E157F1FB98F2147 156
0C9: CE8AA606AEB11B13 514
0CA: 48F681F08F8BB811 8BF
0CB: BFD8F2AF4150716F C80
0CC: 1371441641FBA8F2 009
0CD: 14714416417E15F0 372
0CE: 15C01FAF8F214710 706
0CF: 88EAF7FD215E3D5D AE0
0D0: 231068B1801635E3 E46
0D1: 33401E8DA939014A 1CE
0D2: 968DE31B19624E31 563
0D3: D0962BD31A09622D 8FA
0D4: 31809629C161A6D5 C7B
0D5: DC118CE8AE491F1C 04A
0D6: 6F2D2E61451B2D8F 3FA
0D7: 21F045F2AF215DF1 79E
0D8: 7F15D71F1C6F2147 B41
0D9: 8ED67FD215E3D78E F11
0DA: 987FDB132602006C 29C
0DB: CCCD60613014A8FE 652
0DC: 3C1007DA07CE50E8 9F6
0DD: 401B1C6F21468E9F D96
0DE: 6F1631F045F215EF 135
0DF: 15DFB7655085016F 4C8
0E0: 17F15E715D727B16 85B
0E1: 204508408FFD1518 BD9
0E2: 70071BB89F215A03 F5F
0E3: 010E0E15808F99C4 2EF
0E4: 130E8161BB89F215 67B
0E5: 240E46154454C8F3 9FB
0E6: E320D0081E0201F0 D70
0E7: F8121D8031570416 0D9
0E8: B0513C0813510060 42F
0E9: 9F8FAB2514E18FE2 7F6
0EA: 1205D08F127006EE B7A
0EB: F8FEE010D48F3E32 F3C
0EC: 062321B25A0E2231 2A1
0ED: 23030338102A9503 5EC
0EE: A660366F000633F7 967
0EF: D601F1C6F2143E45 D00
0F0: 217A501F1C6F2143 081
0F1: CCD2E68BE40DA1BA 464
0F2: F8F21468BA40DA14 80E
0F3: 1683E77201F1C6F2 B9E
0F4: D0E459E75101F1C6 F3C
0F5: F21BAF8F214261DF 2F6
0F6: 8E1D3F8D692207FE 6B2
0F7: F8F015001B1C6F21 A42
0F8: 468E965FD015A3D8 DED
0F9: 8E795FD231068BD4 197
0FA: 0D51F084F2691018 514
0FB: 114A14F17196A419 88D
0FC: 66F0CD57E14F96A1 C43
0FD: 11BE78F2689D708F FF3
0FE: D0642172501BAF8F 38A
0FF: 2146CE8AAA067FE7 749
100: 9301FAF8F2147108 AD8
101: 1BB88F21F178F28E E91
```

```
102: 3B3F118CE8AE1E1F 25C
103: 1C6F21436EC01B1C 5FB
104: 6F21468E484F1FE3 9A6
105: 5F2D231065D014B9 D2C
106: 6CB01C1A6E51FB66 0E2
107: 15C31635E014B148 456
108: 1C1161A6E50F70DE 7FA
109: 8FE45918508FBD58 BAE
10A: 18F83DB0171CCCC1 F64
10B: 001BA78F2D215E3C 306
10C: 68B6831B1C6F2146 697
10D: 8E014F118AD0DA81 A3C
10E: C1583163132133DE DB0
10F: 136C68D771B13152 12C
110: 8D393901B9F8F21F 4D9
111: AB8F215E015D0184 86E
112: 1CE1461451841461 BD5
113: 3518F1567AF58F0B F7B
113: 3518F1567AF58F0B F7B
114: 7F08FE45911BB88F 339
115: 2D21448FB13B1041 6B6
116: CF15178F8BB818F8 A76
117: F5F08F015008FA8C E25
118: 418D84A80        024
```



PPC PARIS CHAPTER

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

BULLETIN D'ADHESION

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

PROFESSION ______________________________
INTERETS ______________________________
____

MATERIEL HP EN VOTRE POSSESSION ______________________________
____
____

AUTRE MATERIEL MICRO-INFORMATIQUE ______________________________
____
____

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

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

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

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

LE MONTANT DE LA COTISATION AU PPC PARIS CHAPTER S'ELEVE A 300.00 FF.
ETUDIANTS: 250.00 FF. (JUSTIFICATIF INDISPENSABLE)
PAIEMENT EFFECTUE LE |_|_|/|_|_|/19|_|_| A L'ORDRE DE PPC PARIS CHAPTER.
PAR [ ] CHEQUE BANCAIRE N° __________ BANQUE __________
[ ] CHEQUE POSTAL 3 VOLETS N° __________
[ ] MANDAT LETTRE
EVENTUELLEMENT: JE M'ABONNE A COMPTER DU |_|_|/|_|_|/19|_|_|

VEUILLEZ ENVOYER TOUTE CORRESPONDANCE A :
PPC-PARIS, 56 RUE J.J. ROUSSEAU, 75001 PARIS (FRANCE)

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

La maquette de ce numéro a été préparée et réalisée par Pierre David, Jean-Jacques Dhénin et Janick Taillandier, grâce à un système comprenant un HP71B, deux lecteurs de disquettes HP9114A, une imprimante HP2225B et une imprimante LaserJet.

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

ENGLISH SUMMARY

JPC 43 - AVRIL 1987

This issue is bigger than usual. The reason is to be found from page 16 to page 41. This paper describes a new, very sophisticated statement for the HP-71B providing formatted input capabilities ("à la" INPUT USING). The source files are presented in a way similar to the IDS Vol. III format. This is intended to help you study the code if you wish. Any feedback on this statement is welcomed.

Our first HP-28C program is presented by Pierre David on page 6. Its purpose is to simplify the expression expanding process.

Guy Toublanc, in his series about "approximations of real numbers by fractions", gives you his HP-41 version.

You will find a presentation of a lesser known peripheral for the HP-75 : the PMS. Eric Gengoux tells us all secrets about the device itself, the keywords to drive it and the problems he has encountered.

Given a problem (permutations in a set), Alain Herreman describes two ways to solve it. He uses two different algorithms, and a base conversion program. Follow him in this approach.

Last, Alain Gillet, a beginner, try to help his fellow beginners with programming a simple game.

Happy reading.