JPC
NOVEMBRE 1986
NUMERO 39
Le numéro 35 FF.

A PROPOS DU CLUB

P. David & J. Taillandier	Editorial	*1*
P. David	PPC Paris se réunit	*2*
J.J. Dhénin	Ah ! Vous écrivez	*2*
	Courrier du coeur	*3*
Hewlett-Packard	Le Support Technique	*4*

DUR ET MOU

P. Guez & J.P. Bondu	Le module "Translator Pac"	*5*

HP41

C. Jegouzo	Electronique de base	*10*

HP71

PPC-Paris	Récapitulation des tokens	*16*
P. David	Les débuts	*17*
J.J. Dhénin	Traitement de tableaux	*21*
J.P. Bondu	Encore un scan... dale !	*32*
G. Toublanc	Intégrales et complexes	*32*
X. Bille	Survol des fichiers de données	*34*
P. David	Le coin des Lhex	*38*

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

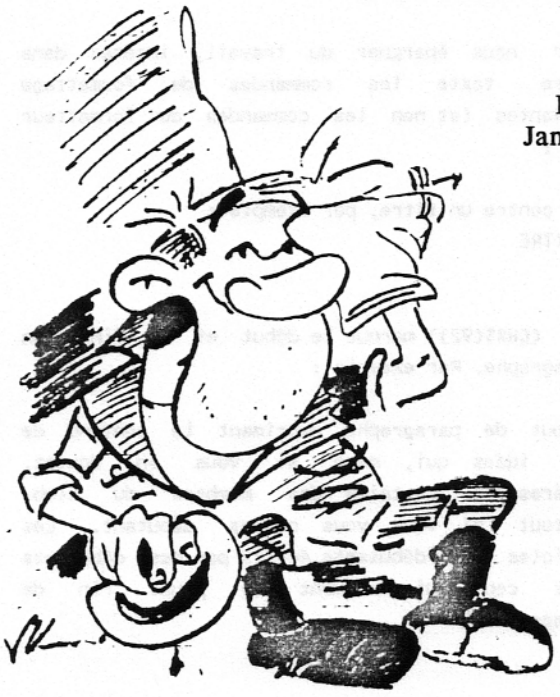
EDITORIAL

Chers amis,

Voici le quatrième JPC préparé depuis un mois, c'est à dire que nous travaillons actuellement à la cadence d'un journal par semaine... Tout est remis sur les rails, ce qui va nous permettre de prendre un repos bien mérité.

Dans ce mois un peu fou, nous avons également brassé les quelques 700 Ko de programmes (Basic, Forth, Lex et sources) parus dans le Journal depuis la sortie du HP71. Nous sommes heureux de vous annoncer la disponibilité de ces programmes dans un très proche avenir.

Nous reparlerons de tout ceci dans un mois...



Pierre David
Janick Taillandier

PPC PARIS SE REUNIT UNE FOIS PAR MOIS

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

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

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

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

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

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

- Métro 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 6 décembre 1986
Samedi 17 janvier 1987
Samedi 7 février 1987
Samedi 7 mars 1987
Samedi 4 avril 1987
Samedi 16 mai 1987
Samedi 6 juin 1987

Pierre DAVID
(P37, SIG1, CHHU616)

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

```
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)
```

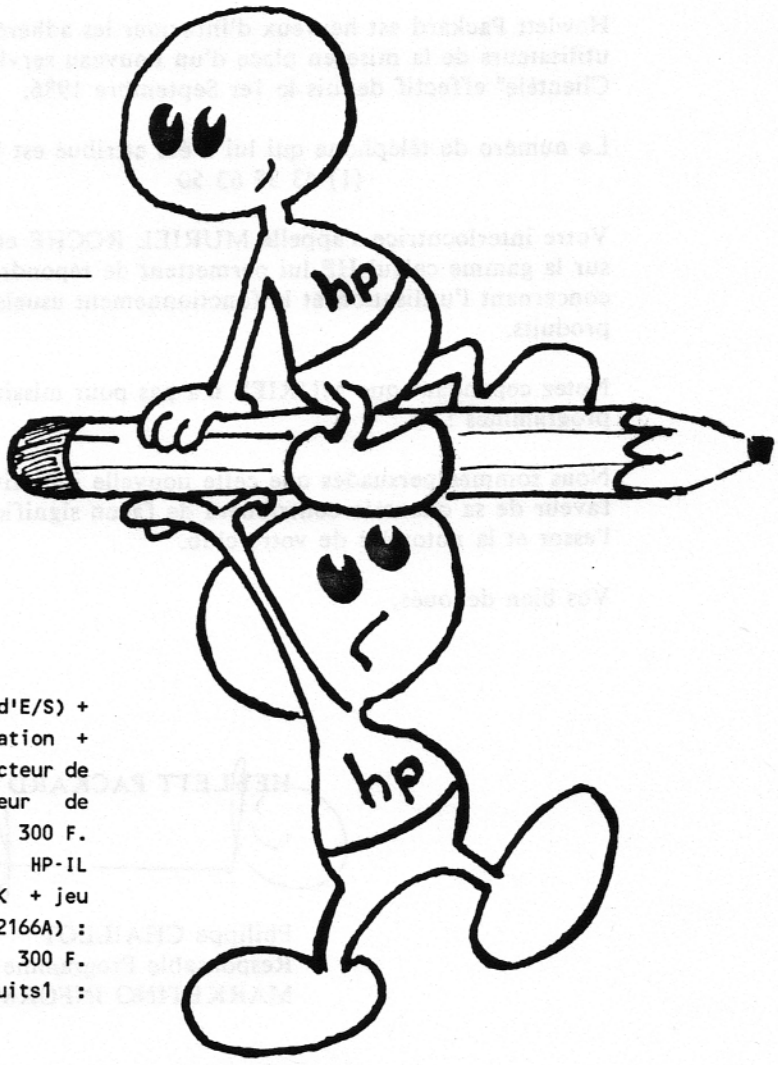
Didier JEHL
Appt 8
9 rue Utrillo
62000 ARRAS
Tel: 21 23 00 77

Je fais bâtir... PROFITEZ EN !!!

Pour HP41 :

MLDL wrappé (4K Ram + 4K Rom + Ports d'E/S) + programmateur d'Eprom + logiciel d'exploitation + 2 jeux d'Eprom (ASS-3, I/0) : 2000 F. Lecteur de cartes + 200 cartes : 400 F. Lecteur de codes-barre : 400 F. Port Extender : 300 F. Lecteur d'Eprom 16K + Jeu d'Eprom HP-IL Development 500 F. Lecteur d'Eprom 4K + jeu d'Eprom SM-1C : 300 F. Convertisseur (HP 82166A) : 1400 F. 2 XMémoires 700 F. XFonctions : 300 F. PPC Rom : 500 F. Math1 : 100 F. Circuits1 : 200 F.

Un lot de revues (PPC-US,TN,PPC-T,...) et de livres sur 41 sera joint à chaque commande.



HEWLETT
PACKARD

HEWLETT-PACKARD FRANCE : S. A. au capital de 304.000.000 de Francs régie par les articles 118 A 150 de la loi sur les sociétés commerciales - R.C.S. Corbeil-Essonnes : B 709805030.

SIÈGE SOCIAL Parc d'Activité du Bois Briard - 2, Avenue du Lac - 91040 EVRY CEDEX - Tél. : 60.77.83.83 - Télex : 692315 F.

Evry, le 3 Novembre 1986

Hewlett Packard est heureux d'informer les adhérents des clubs utilisateurs de la mise en place d'un nouveau service "Support Clientèle" effectif depuis le 1er Septembre 1986.

Le numéro de téléphone qui lui a été attribué est le suivant :
(1) 43 37 63 50

Votre interlocutrice s'appelle MURIEL ROCHE et ses compétences sur la gamme calcul HP lui permettent de répondre à vos questions concernant l'utilisation et le fonctionnement usuels de nos produits.

Notez cependant que MURIEL n'a pas pour mission de rédiger vos programmes !

Nous sommes persuadés que cette nouvelle initiative d'HP en faveur de sa clientèle contribuera de façon significative à l'essor et la notoriété de votre club.

Vos bien dévoués,

HEWLETT PACKARD FRANCE

Philippe CHAILLOT
Responsable Programme Calcul
MARKETING INFORMATIQUE PERSONNELLE

LE MODULE HP71-41 TRANSLATOR

Le HP41 Translator, comme son nom l'indique, transforme votre 71 en 41. Ce module de 48 K octets contient :

- 1 fichier FTH41ROM (Lex privé de 894 octets)
- 1 fichier TRANS41 (Basic de 3467 octets)
- 1 fichier READ41 (Basic de 1075 octets)
- 1 fichier KEYS41 (Key de 282 octets)
- 1 fichier EDTEXT (Basic de 6790 octets)
- 1 fichier EDKEYS (Key de 54 octets)
- 1 fichier EDLEX (Lex de 2557 octets)
- 1 noyau FORTH hard configuré à l'adresse E0000

Il a été développé pour HP par William C. WICKES (auteur du livre "SYNTHETIC PROGRAMMING" pour HP41). Le manuel de presque 200 pages regroupe 5 chapitres :

- Comment débuter (6 pages)
- L'environnement HP41 en détail (20 pages)
- Ecrire et Traduire les programmes 41 (12 pages)
- L'éditeur de texte (7 pages)
- L'environnement HP71 FORTH (23 pages)

Suivent 5 appendices contenant les messages d'erreur, les mots Basics supplémentaires (implémentés par EDLEX et FTH41ROM), le dictionnaire FORTH et un petit guide pour transformer et faire tourner un programme 41 sur le 71.

DEBUTONS...

Appuyons d'un doigt timide sur les touches H-P-4-1-[Endline]. "Tout va bien" répond le 71 : HP41 EMULATOR 1A. Enfin nous y sommes ! Suit l'affichage de : SIZE (max nnn)? Vous devez spécifier le nombre de registres de données que vous désirez réserver. C'est le SIZE bien connu de "Coconut". Maintenant si vous voulez exécuter :

1 [ENTER] 2 + 3 * , tapez :
1 2 + 3 * [ENDLINE]

La différence n'est pas trop grande, ça va ? Une bonne solution consiste à taper KEYS41 qui redéfinit le clavier. Les fonctions assignées (voir tableau 2) sont alors affichées puis exécutées. Cela est très pratique en mode calcul puisque l'on est dispensé de l'appui sur [ENDLINE]. Les fonctions non assignées sont obtenues en tapant leur nom suivi de [ENDLINE]. L'emploi de XEQ est réservé au lancement d'un programme 41. Le mot STACK permet de visualiser les registres de la pile à chaque opération :

La séquence : 1 2 3 4 + 5 STD STACK provoque l'affichage de : 1 2 7 5 | 4 qui sont respectivement la visualisation des registres T, Z, Y, X et L.

STD sélectionne le format standard d'affichage. A,X fonctionne de même en affichant le registre Alpha et le LastX. Pour afficher uniquement X il faut faire XONLY [ENDLINE]. Enfin la liste des programmes est obtenue par CAT, identique au CAT_1 de la 41.

PLUS EN DETAIL ...

Il est possible de calculer directement une expression algébrique sans avoir à la traduire en notation polonaise. Si vous introduisez SIN(1.234*2/COS(-31.9)), le résultat sera calculé et rangé dans le registre X. De la même manière les variables BASIC sont accessibles en lecture et en écriture.
La séquence A B C D1 [ENDLINE] rappelle la valeur de A dans le registre T, de B dans le registre Z, de C dans le registre Y, de D1 dans le registre X. Pour stocker le registre X dans une variable BASIC il faut faire précéder son nom d'un point d'exclamation. Par exemple : 45.6789 !G place 45.6789 dans X puis dans la variable G.

En fait une expression peut contenir des variables BASIC ou toute fonction numérique reconnue par le 71, Lex et modules enfichables compris.

Les flags utilisables sont les mêmes que ceux de la 41. De 0 à 29 l'utilisateur arme ou désarme (question de philosophie), de 30 à 55 il ne peut que tester. A l'exception des flags BEEP, USER, ON continu, les numéros de drapeaux correspondent. Les valeurs de trappe des exceptions mathématiques sont gérées par l'émulateur de la même manière que sous BASIC.

Quand l'HPIL est présent le 71 recherche si une imprimante est connectée. Si c'est le cas elle sera utilisée par les fonctions VIEW, AVIEW... et les flags 21 et 55 seront armés. Si la connection de l'imprimante a lieu après l'entrée dans l'environnement 41 il suffit de taper PRINTER pour la valider.

La fonction purge41 (en minuscules) permet de réinitialiser l'environnement 41. Tous les registres, les programmes et les mots Forth créés en mode 41 sont alors perdus et le 71 se place en FORTH. Quand on est en mode 41 on peut en sortir de 2 manières :

FORTH [ENDLINE] -> le 71 est en mode Forth.
BASIC [ENDLINE] -> " " " Basic.

Comme tout l'environnement 41 est stocké dans un fichier Forth appelé FTH41RAM, il est possible de le sauver, le renommer etc. Rien ne vous empêche d'avoir simultanément en mémoire plusieurs de ces fichiers et de les activer suivant les besoins. Seul le fichier FTH41RAM sera opérationnel, et il faut noter qu'il est incompatible avec le Forth du Forth/Assembleur.

ECRITURE ET TRADUCTION DES PROGRAMMES HP41

** Transfert 41 -> 71 **

Il existe dans le module 2 programmes BASIC qui lui sont propres. Le premier, READ41, permet le transfert d'un programme de 41 à 71. Pour ce faire Titan se travestit en imprimante et stocke chaque ligne reçue dans un fichier texte. Sur la 41 il suffit d'exécuter la séquence suivante :

MANIO	MANual Input/Output
CF 15	
CF 16	Format d'impression
x SELECT	x : adresse du 71
PRP 'Filename'	envoie le programme sous forme ASCII.

Toutes ces instructions font partie de l'HPIL de la 41 et ne nécessitent l'ajout d'aucun module.

** Introduction manuelle **

Si vous désirez introduire directement un programme 41 dans le 71, il vous faudra utiliser l'éditeur de textes présent dans ce module. Il est identique à celui du Forth/Assembleur et constitue un sous ensemble du module Text Editor. Vous rentrez le programme tel qu'il serait apparu sur un listing 41 en omettant cependant les numéros de ligne.

Maintenant que votre fichier source est en mémoire sous forme de fichier texte, il va devoir subir une précompilation qui lui permettra d'être ingurgité par l'émulateur. C'est ici qu'intervient le deuxième programme BASIC nommé TRANS41. Il va examiner le fichier source, le modifier éventuellement, rajouter les indications nécéssaires à la compilation. Le fichier obtenu est toujours de type Text et donc manipulable à volonté dans l'environnement BASIC (sauvegarde, édition...). De cette manière la phase de précompilation peut être évitée à chaque chargement du même programme. Fin de la deuxième étape et... début de la troisième !

Voici enfin venu le moment de charger le programme dans l'émulateur. Sous BASIC il suffit de lancer TRANS41 et de se laisser guider par les questions. Dans l'environnement 41 vous taperez LOAD et lancerez le programme par RUN ou XEQ. Comptez un temps de chargement d'environ une seconde par ligne.

MODIFICATION D'UN PROGRAMME

Pour modifier un programme il faut rappeler en mémoire le fichier "source" (c'est à dire de type Texte), lancer l'éditeur de texte pour effectuer vos corrections et recommencer la compilation depuis le début. Si vous aviez effectué une sauvegarde du fichier intermédiaire, n'oubliez pas de sauver la nouvelle. Enfin la fonction CLP effacera TOUS les programmes postérieurs - chronologiquement - à celui effacé.

CONCLUSION

Pour juger ce module il faut se mettre à la place de l'utilisateur auquel HP le destine. Celui-ci a développé des applications sur HP41, et voit dans le 71 le moyen de récupérer sa bibliothèque tout en augmentant la puissance des futurs programmes. Cet utilisateur peut être une société. A l'opposé l'acheteur d'un 71 se verra répondre que les programmes de la User's Library, fort nombreux, tournent sur sa machine.

Comme de bien entendu, des nuages viennent assombrir cette description trop idyllique.

D'abord, seuls les programmes développés sur une 41 nue fonctionneront. L'émulateur ne supporte, de manière générale, aucun module d'application, HPIL excepté. Il ne gère notamment pas la mémoire étendue. La seule instruction rescapée du module Temps est TIME. Voir le tableau 1 à ce sujet.

Ensuite le développement de programmes 41 dans l'environnement 71 est très lourd. Le débugage malaisé (pas de fonction SST). Il faudra vraiment se contenter de recopier les programmes.

Quelques détails sont irritants en usage quotidien. L'emploi du fichier KEYS41 détruit votre propre fichier Keys sans le sauvegarder comme le fait l'éditeur de texte. Bien sûr vous pouvez redéfinir un "mot" KEYS41 mais il faudra recharger le noyau FORTH à chaque réinitialisation. La fonction CLP efface tous les programmes enregistrés après le programme spécifié.

Par contre un avantage de ce module est la possibilité de documenter les programmes. Cela facilitera la maintenance logicielle, notamment de la pile opérationnelle. Puisque vous êtes dans un environnement Forth les commentaires figureront entre parenthèses. Pour les mêmes raisons il est possible d'étendre le vocabulaire de base et, peut-être, de compenser ainsi les défauts précités. Enfin vous bénéficiez de l'espace mémoire du 71, de l'accès à ses fonctions de calcul, de sa vitesse de traitement.

Voici les éléments que l'on doit avoir en tête pour décider ou non de son achat.
Mais ce n'est pas une 41 !

Philippe GUEZ (PPC 1)
et Jean-Pierre BONDU (PPC 33)

NDLR : Ce n'est pas fourni avec, mais ça existe ! Le "HP71 Translator Pac Programmers Tool Box", vendu séparément par la librairie de Corvallis sous la référence V7540-71 pour 60 dollars. L'auteur est également Bill Wickes.

Tableau 1 : le vocabulaire du Translator.

Les fonctions inclues dans le HP41 Translator comprennent la totalité des fonctions programmables de la HP41C/CV. Ont été rajoutées certaines fonctions propres à la 41CX et au module HPIL. La liste suivante vous permettra de constater quelles ont été les fonctions omises.

FONCTIONS	
CONNUES	OMISES
System/Format Functions :	
Toutes sauf	CLK12/24, CLKT, CLKTD, DMY, MDY, PASN.
Clearing Functions :	
CLD, CLKEYS, CLP, CLRG, CLRGX, CLsgma CLX, CLST.	CLALMA, CLALMX, CLFL, CLRALMS, DELCHR, DEL, PCLPS, PURFL, DELREC.
Stack/Data Register Functions :	
Toutes inclues.	
Numeric Functions :	
Toutes inclues.	
Extended Memory Functions :	
Aucune. En effet, la notion de mémoire étendue est déja supportée par la structure fichier du 71 (dixit HP |).	
Time Functions :	
TIME	Toutes les autres
Alpha Functions :	
ALENG, ANUM, AOFF, AON, ARCL, AROT, ASHF, ASTO, ATOX, AVIEW, CLA, POSA, PROMPT, XTOA.	ADATE, ATIME/24, ARCLREC, GETREC.
Interactive Functions :	
ADV,BEEP, PSE,TONE.	GETKEY, GETKEYX.
Printer Control :	
Flags 21 & 55 actifs.	Flags 12, 13, 15 & 16 sont sans effet.

Tableau 2 : les assignations du Translator.

Fichier KEYS41 :

```
DEF KEY '/',' /'
DEF KEY '*',' *'
DEF KEY '-',' -'
DEF KEY '+',' +'
DEF KEY ',',' -';
DEF KEY '#50',' RCL ';
DEF KEY '#51',' STO ';
DEF KEY 'fW',' XEQ ""';  (passage en insertion)
DEF KEY 'fR',' GTO ""';  (passage en insertion)
DEF KEY 'fY',' SF';
DEF KEY 'fU',' CF';
DEF KEY 'fO',' S+';
DEF KEY 'fP',' S-';
DEF KEY 'f7',' CLS';
DEF KEY 'f8',' MEAN';
DEF KEY 'f9',' SDEV';
DEF KEY 'f/',' SQRT';
DEF KEY 'fS',' XEQ';
DEF KEY 'fD',' RTN';
DEF KEY 'fF',' GTO ';
DEF KEY 'fH',' FS? ';
DEF KEY 'fJ',' FC? ';
DEF KEY 'fL',' BEEP '
DEF KEY 'f=',' X<>Y'
DEF KEY 'f4',' SIN'
DEF KEY 'f5',' COS'
DEF KEY 'f6',' TAN'
DEF KEY 'f*',' 10^X'
DEF KEY 'fX',' CAT'
DEF KEY 'fM',' 1/X'
DEF KEY 'f(',' E^X'
DEF KEY 'f)',' LN'
DEF KEY '#94',' AON'
DEF KEY 'f1',' ASIN'
DEF KEY 'f2',' ACOS'
DEF KEY 'f3',' ATAN'
DEF KEY 'f-',' LOG'
DEF KEY 'f.',' VIEW';
DEF KEY 'f+',' LASTX'
DEF KEY '#162',' RUP'
DEF KEY '#163',' RDN'
```



HP41

C. Jegouzo — Electronique de base — *10*

- Polarisation par pont de base — *10*
- Transistor monté en émetteur commun — *10*
- Transistor monté en base commune — *13*
- Transistor monté en collecteur commun — *14*

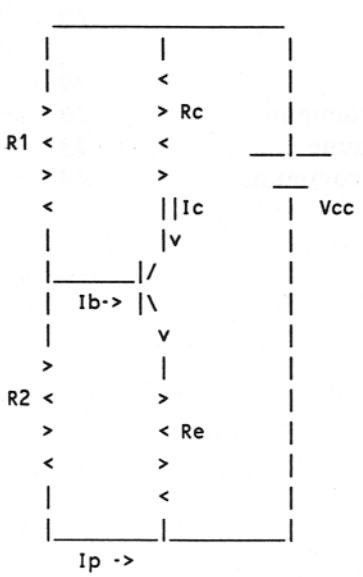
ELECTRONIQUE DE BASE

Les quatre programmes que je vous propose, devraient vous permettre de vérifier vos calculs en matière d'électronique de base. J'ai essayé de garder les mêmes spécifications dans les quatre programmes afin de faciliter leur emploi.

P-PB : POLARISATION PAR PONT DE BASE

Registres Programme : 223 octets
Registres Données : 8

La polarisation par pont diviseur de tension fait partie des polarisations classiques. Ce montage consomme de l'énergie, même au repos à cause du pont. Le courant Ib est fixé définitivement par les valeurs de R1 et R2.



Connaissant Ic0, VCE0, VBE0, BETA, et Vcc, les rapports X=Rc/Re et Y=IP/Ib, le programme donne Ib, IP, R1, R2, RB=R1/R2 .

Exemple:

"Ic0 ?"	E-3	R/S	(1mA)
"VCE0 ?"	5	R/S	(5 volts)
"VBE0 ?"	0,6	R/S	(0.6 volts)
"BETA ?"	100	R/S	(BETA=H21)
"Vcc"	10	R/S	(Tension d'alimentation)
"RE ?"		R/S	(mettre 0 si RE n'existe pas)
"RC=X.RE X?"	4	R/S	
"RE=1000"		R/S	(RE 1 kohms)
"RC=4000"		R/S	(RC 4 kohms)
"Ib=10 µA"		R/S	(10 micro-ampères)
"IP=Y.Ib Y?"	10	R/S	
"IP=100 µA"		R/S	(courant dans R2 : 100 µA)
"R2=16100"		R/S	(16,1 kohms)
"R1=76272"		R/S	(76.2 kohms)
"RB=13294"			(R1//R2=13,2 kohms)

Le programme :

```
01*LBL "P-PB"
FIX 0  "Ic0 ?"  PROMPT  STO 00  "VCE0 ?"  PROMPT
STO 01  "VBE0 ?"  PROMPT  STO 02  "BETA ?"  PROMPT
STO 03  "Vcc ?"  PROMPT  STO 04  "RE ?"  PROMPT
X=0?  GTO 00  RCL 04  RCL 01  -  RCL 00  /  GTO 01

28*LBL 00
"RC=X.RE X?"  PROMPT  1  RCL Y  +  RCL 00  *  1/X
RCL 04  RCL 01  -  *  STO 05  "RE="  ARCL X
PROMPT  *

46*LBL 01
"RC="  ARCL X  PROMPT  RCL 00  RCL 03  /  STO 06
E6  *  "Ib="  ARCL X  "µA"  PROMPT  "IP=Y.Ib Y?"
PROMPT  RCL 06  RCL Y  *  E6  *  "IP="  ARCL X
"├ µA"  PROMPT  E6  /  1/X  RCL 03  1  +  RCL 05
*  RCL 06  *  RCL 02  +  *  STO 07  "R2="  ARCL X
PROMPT  RCL Y  *  RCL 06  *  CHS  RCL 04  +  X<>Y
1  +  RCL 06  *  /  "R1="  ARCL X  PROMPT  RCL 07
RCL Y  *  X<>Y  RCL 07  +  /  "RB="  ARCL X
```

TR-Ec TRANSISTOR MONTE EN EMETTEUR COMMUN

Registres Programme : 895 octets
Registres données : 22

Le programme se décompose en 4 grandes options:

[A] : Initialisation

[B], [C], [D], [E] : Calcul des amplifications en tension (Av) et en courant (Ai), ainsi que des impédances d'entrée (Ze) et de sortie (ZS).

[F] : Détermination des équations des droites d'attaque et de charge statiques et dynamiques.

[G] : Calcul des coefficients de stabilité thermique.

Exemple:

[A] :

"R1^R2 ?"	76,4E3 ENTER^	
	16E3	R/S

```
                    (Si R/S la machine demande RB)
"RC^RE^RU ?"     4E3 ENTER^
                  E3 ENTER^
                 4E3 R/S                    (en ohms)
"H11 ?"         1000 R/S
"H21 ?"          100 R/S
"H22 ?"          E-5 R/S     (H12 toujours pris = 0)
"OK"
```

[a] : Permet de changer la valeur de H22

```
"H22 ?"            0 R/S
"OK"
```

[B], [C], [D], [E] : Les 4 options prennent en compte la valeur de H22, ainsi que la valeur de la résistance d'émetteur de couplage.

3 cas sont possibles:

- Condensateur aux bornes de RE : RE de couplage = 0 = RE1.
- Condensateur partiellement en parallèle avec RE : RE de couplage = RE1.
- Pas de condensateur : RE = RE1

```
                H22#0     H22=0      Couplage
[B] :
"RE COUPL ?"    R/S       R/S        222,0
                                     H22=0 (forcée)
"AV="          -196,08   -200,00     -8,54
[C] :
"AI="           45,57     46,49      18,05
[D] :
"Ze="          929,72    929,72    8454,24  (ohms)
[E] :
"ZS="         3846,15   4000,00    4000,00  (ohms)
[F] :
"R1^R2 ?"          76,4E3 ENTER^
                   16E3 R/S
    (demande si valeurs non entrées dans option A)
"BETA ?"             100 R/S
"Ic0 ?"              E-3 R/S                 (1mA)
"VBE0^VCE0 ?"        0,6 ENTER^
                       5 R/S
"Vcc ?"               10 R/S
"Vbb="              1,73 R/S
                           (potentiel de la base)
"M0 VBE= 1,73"       R/S
                  (droite d'attaque statique)
"M0 Ib= 0,00 µA"     R/S        (couple M0,M1)
"M1 VBE= 0,00"       R/S
"M1 Ib= 15,29 µA"    R/S
"D CHAR STATQ"       R/S
                  (droite de charge statique)
"M2 VCE= 10,00"      R/S        (couple M2,M3)
"M2 Ic= 0,00 MA"     R/S        (milliampères)
"M3 VCE= 0,00"       R/S                (volts)
"M3 Ic= 2,00 MA"     R/S        (milliampères)
```

```
"PO  VCE0 Ic0"
 (rappelle coordonnées du point de fonctionnement)
"D CHARG DYN"        R/S
                  (droite de charge dynamique)
"M4 VCE= 7,00"       R/S        (couple M4,M5)
"M4 Ic= 0,00 MA"     R/S
"M5 VCE= 0,00"       R/S                (volts)
"M5 Ic= 3,50 MA"     R/S        (milliampères)
```

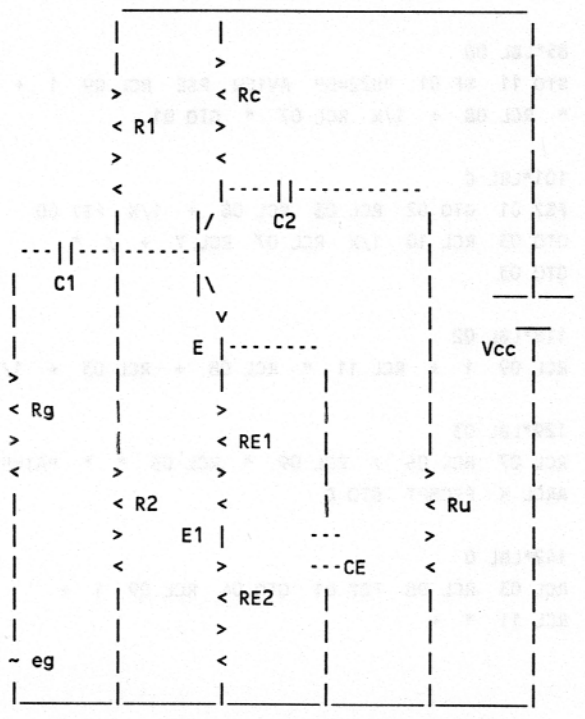
(amplitudes maxi >0 et <0 des grandeurs de sortie. Elles sont dissymétriques)

```
"EXCURS  MAXI"
"VCE+2,00 -5,00"
"Ic +2,50 -1,00"
```

(Pour obtenir des excursions symétriques, il faut placer le point de repos au milieu de la droite de charge dynamique. On détermine alors les nouvelles coordonnées M6 du point de repos et la nouvelle valeur de Ib1. Pour cela, on fait varier R2 d'ou R'2)

```
"EXCURS SYM"         R/S
"M6 VCE= 2,86        R/S
"M6 Ic= 1,43 MA"     R/S
"Ib1= 14,29 uA"      R/S
"GRAPH VBE1 ?"    0,65 R/S
"IP1= 89,40 uA"      R/S
"R'2= 23250,78"
```

```
[G] :
"S1=dIc/dIcb0        R/S  (rappel de formule)
"S1=12,58"           R/S
"S2=dIc/dVBE"        R/S
"S2=-8,75E-4"        R/S
"S3=dIc/dBETA"       R/S
"S3=1,26E-6"
```



Le programme :

```
01*LBL "TR-Ec"
SIZE?  22  X>Y?  PSIZE

06*LBL A
0  X<>F  SF 02  FIX 2  CF 22  "R1^R2 ?"  PROMPT
FS? 22  GTO 06  "RB ?"  PROMPT  STO 03  GTO 07

20*LBL 06
CF 02  STO 02  X<>Y  STO 01  XEQ 20  STO 03

27*LBL 07
"RC^RE^RU ?"  PROMPT  STO 06  RDN  STO 05  RDN
STO 04  RCL Z  XEQ 20  STO 07  SF 03

39*LBL J
CF 22  "H11 ?"  PROMPT  FC? 22  GTO F  STO 08
"H21 ?"  PROMPT  STO 09

49*LBL a
CF 00  "H22 ?"  PROMPT  STO 10  X=0?  SF 00  CF 03
"OK"  PROMPT  GTO A

60*LBL B
FS? 03  GTO J  CF 01  CF 22  "RE COUPL ?"  PROMPT
FS? 22  GTO 00  RCL 10  SF 25  1/X  RCL 07  FC? 00
XEQ 20  RCL 08  /

77*LBL 01
RCL 09  *  CHS  "AV="  ARCL X  PROMPT  GTO B

85*LBL 00
STO 11  SF 01  "H22=0"  AVIEW  PSE  RCL 09  1  +
*  RCL 08  +  1/X  RCL 07  *  GTO 01

101*LBL C
FS? 01  GTO 02  RCL 03  RCL 08  +  1/X  FS? 00
GTO 03  RCL 10  1/X  RCL 07  RCL Y  +  /  *
GTO 03

118*LBL 02
RCL 09  1  +  RCL 11  *  RCL 08  +  RCL 03  +  1/X

129*LBL 03
RCL 07  RCL 06  /  RCL 09  *  RCL 03  *  *  "AI="
ARCL X  PROMPT  GTO C

142*LBL D
RCL 03  RCL 08  FC? 01  GTO 04  RCL 09  1  +
RCL 11  *  +

153*LBL 04
XEQ 20  "Ze="  ARCL X  PROMPT  GTO D

159*LBL 05
RCL 10  SF 25  1/X  RCL 04  FS? 00  GTO 05  FS? 01
GTO 05  XEQ 20

169*LBL 05
"ZS="  ARCL X  PROMPT  GTO E

174*LBL F
FS? 02  GTO A  "BETA ?"  PROMPT  STO 00  "Ic0 ?"
PROMPT  STO 12  "VBE0^VCE0 ?"  PROMPT  STO 14  RDN
STO 13  "Vcc ?"  PROMPT  STO 15  RCL 02  *  LASTX
RCL 01  +  /  STO 16  "Vbb="  FIX 2  ARCL X
PROMPT  "D ATTQ STATQ"  CF 04  0  RCL 16  0
XEQ 21  RCL 16  RCL 00  RCL 05  *  RCL 03  +  /  0
1  XEQ 21  "D CHAR STATQ"  PROMPT  SF 04  0
RCL 15  2  XEQ 21  RCL 15  RCL 04  RCL 05  +  /  0
3  XEQ 21  "P0 VCE0 Ic0"  PROMPT  "D CHARG DYN"
PROMPT  RCL 07  RCL 12  *  STO 17  RCL 14  +  0
X<>Y  4  XEQ 21  RCL 14  RCL 07  /  STO 18  RCL 12
+  0  5  XEQ 21  "EXCURS MAXI"  PROMPT  E3
ST* 18  ST* 12  "VCE+"  ARCL 17  "├ -"  ARCL 14
PROMPT  "IC +"  ARCL 18  "├ -"  ARCL 12  ST/ 12
PROMPT  "EYCURS SYM"  PROMPT  SF 04  RCL 15
RCL 07  *  LASTX  RCL 04  +  RCL 05  +  /  RCL X
RCL 07  /  STO 19  X<>Y  6  XEQ 21  E3  *  RCL 00
/  "Ib1="  ARCL X  STO 20  "├µA"  PROMPT
"GRAPH VBE1 ?"  PROMPT  RCL 05  RCL 19  *  +
RCL X  CHS  RCL 15  +  RCL 01  /  E6  *  RCL 20
-  "IP1="  ARCL X  "├µA"  PROMPT  /  E6  *
"R'2="  ARCL X  PROMPT  GTO F

327*LBL 20
RCL Y  RCL Y  +  RDN  *  R^  /  RTN

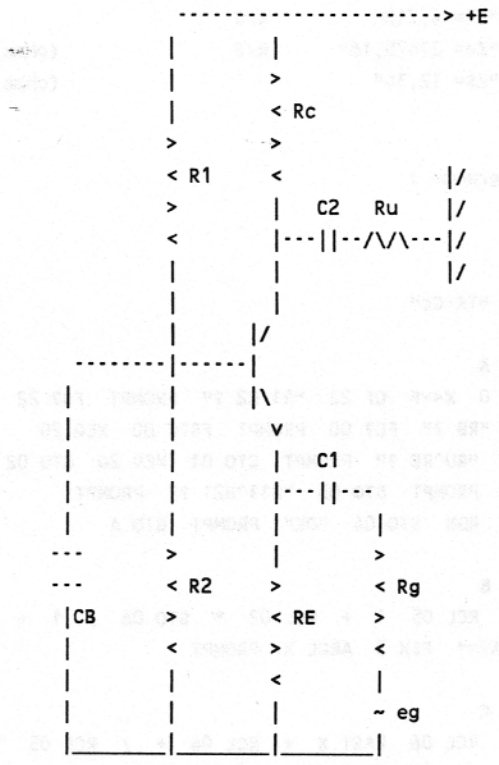
336*LBL 21
"M"  FIX 0  ARCL X  FC? 04  "├ VBE="  FS? 04
"├ VCE="  FIX 2  ARCL Y  PROMPT  "M"  FIX 0
ARCL X  FC? 04  "├ Ib="  FS? 04  "├ Ic="  RCL Z
FS? 04  E3  FC? 04  E6  *  FIX 2  ARCL X  FC? 04
"├µA"  FS? 04  "├MA"  PROMPT  RTN

368*LBL G
"S1=dIc/dIcb0"  PROMPT  RCL 05  RCL 00  1  +
RCL Y  RCL Y  *  RCL 03  +  LASTX  R^  +  RCL Z  *
X<>Y  /  STO 21  "S1="  ARCL X  PROMPT  RCL 00
"S2=dIc/dVbe"  PROMPT  RCL 05  RCL Y  1  +  *
RCL 03  +  /  CHS  "S2="  ARCL X  PROMPT
"S3=dIc/dBETA"  PROMPT  RCL 00  RCL 05  RCL Y  *
RCL 03  +  RCL Y  *  X<>Y  X^2  RCL 05  RCL 03  +
/  1/X  RCL 12  *  RCL 16  +  RCL 13  -  /  1/X
RCL 21  *  "S3="  ARCL X  PROMPT  END
```

TR-Bc: TRANSISTOR MONTE EN BASE COMMUNE

Registres Programme : 448 octets
Registres Données : 10

On considère que le transistor est utilisé en basse fréquence. Les condensateurs sont des court-circuits. On transforme le générateur de Norton en générateur de Thévenin dépendant de VBE et e=u VBE. (u=H21/H11.H22)



Le programme donne l'amplification en tension (Av), en courant (Ai), les impédances d'entrée (Ze) et de sortie (ZS).

Les options [b], [c], [d] et [e] rappellent les formules (utiliser le pas à pas SST).

Exemple :

```
[A] :
"RC?"              3335 R/S              (ohms)
"RE?"               667 R/S
"RG?"                50 R/S
"RU?"              3335 R/S
"H11?"              800 R/S
"H21?"               80 R/S
"H22?"              5E-5 R/S
"u=2000"
"u=&.H21/H11"                          (&=1/H22)

[a] : pour changer H22

[B] : "AV= 153,99"
[b] : formule par SST Av=(μ+1) RL/&+RL  RL=RC//RU

[C] : AI= -0,49
[c] : Ai=(-RC/RC=RU)
      (1/1+(H11+RE).(&+RL)/RE(μ+1).H11
[D] : Ze= 10,52       (ohms)
[d] : 1/Ze= 1/RE + (μ+1/&+RL) + 1/H11

[E] : ZS= 3235,06     (ohms)
[e] : 1/ZS= 1/RC + (1/(&+(μ+1)RO)   RO=RG//RE//H11
```

Le programme :

```
01*LBL "TR-Bc"

02*LBL A
0  X<>F  "RC"  PROMPT  STO 01  "RC"  PROMPT
STO 02  "RG"  PROMPT  STO 03  "RU"  PROMPT  STO 04
"H11"  PROMPT  STO 05  "H21"  PROMPT  STO 07
RCL 04  RCL 01  XEQ 20  STO 09

27*LBL a
SF 00  "H22?"  PROMPT  STO 08  X=0?  GTO 06
RCL 05  RCL 08  *  1/X  RCL 07  *  "μ="  ARCL X
AVIEW  PSE  1  +  STO 06  "μ=&H21/H11"  AVIEW  PSE
PSE  CF 00

52*LBL 06
"OK"  PROMPT  GTO A

56*LBL 20
STO Z  RCL Y  +  1/X  RDN  *  R^  *  RTN

66*LBL B
FS? 00  GTO 02  RCL 06  RCL 08  1/X  RCL 09  +
LASTX  /  1/X  *

78*LBL 03
"AV="  FIX 2  ARCL X  PROMPT  GTO B

84*LBL 02
RCL 07  RCL 09  *  RCL 05  /  GTO 03

91*LBL b
"AV=μ+1 RL/"  AVIEW  "&+RL"  PROMPT

96*LBL C
FS? 00  GTO 04  RCL 06  RCL 05  *  RCL 02  *  1/X
```

```
RCL 08  1/X  RCL 09  +  *  RCL 02  RCL 05  +  *  1
+  1/X  CHS  RCL 01  RCL 04  +  /  RCL 01  *

124*LBL 05
"AI="  ARCL X  PROMPT  GTO C

129*LBL 04
RCL 05  RCL 02  +  LASTX  /  RCL 07  /  1  +  -2
*  1/X  GTO 05

143*LBL c
"-RC/RC+RU 1/"  AVIEW  "1+ H11+RE*&+"  AVIEW
"RL/RE.μ+1.H11"  PROMPT  GTO c

151*LBL D
FS? 00  GTO 07  RCL 08  1/X  RCL 09  +  1/X
RCL 06  *  RCL 05  1/X  +  RCL 02  1/X  +  1/X

168*LBL 08
"Ze="  FIX 2  ARCL X  PROMPT

173*LBL 07
RCL 07  1  +  RCL 05  /  RCL 02  1/X  +  1/X
GTO 08

184*LBL d
"1/Ze=1/RE +"  AVIEW  "1/H11+ μ+1/"  AVIEW  "&+RL"
PROMPT

191*LBL E
FS? 00  GTO 09  RCL 03  RCL 02  XEQ 20  RCL 05
XEQ 20  RCL 06  *  RCL 08  1/X  +  1/X  RCL 01
1/X  +  1/X

209*LBL 10
"ZS="  ARCL X  PROMPT

213*LBL 09
RCL 01  GTO 10

216*LBL e
"1/ZS=1/RC 1"  AVIEW  "/&+ μ+1.R0"  AVIEW
"R0=RG/RE/H11"  PROMPT  END
```

TR-Cc: TRANSISTOR MONTE EN COLLECTEUR COMMUN

Registres Programme : 201 octets
Registres Données : 7

Comme pour les autres montages, le programme calcule Av, Ai, Ze, et Zs. On remarquera que l'étage collecteur commun n'amplifie pas en tension, mais amplifie en courant et possède une résistance d'entrée assez forte et une faible impédance de sortie. Il peut être utilisé comme étage adaptateur d'impédances.

Exemple :

```
[A] :
"R1^R2 ?"                R/S
"RB ?"                 23700 R/S
                  (demande si R1 et R2 non entrés)
"RU^RE ?"              5E3 ENTER^
                       5E3 R/S                (ohms)
"RG ?"                  50 R/S                (ohms)
"H11^H21 ?"           1200 ENTER^
                       100 R/S
"OK"

[B] : "AV= 0,995"        R/S
[C] : "AI= 4,718"        R/S
[D] : "Ze= 21675,16"     R/S                   (ohms)
[E] : "ZS= 12,34"                              (ohms)
```

Le programme :

```
01*LBL "TR-Cc"

02*LBL A
FIX 2  0  X<>F  CF 22  "R1^R2 ?"  PROMPT  FS? 22
SF 00  "RB ?"  FC? 00  PROMPT  FS?C 00  XEQ 20
STO 00  "RU^RE ?"  PROMPT  STO 01  XEQ 20  STO 02
"RG ?"  PROMPT  STO 03  "H11^H21 ?"  PROMPT
STO 05  RDN  STO 04  "OK"  PROMPT  GTO A

33*LBL B
RCL 04  RCL 05  1  +  RCL 02  *  STO 06  /  1  +
1/X  "AV="  FIX 3  ARCL X  PROMPT

49*LBL C
RCL 00  RCL 06  LAST X  +  RCL 04  +  /  RCL 05  1
+  *  2  /  "AI="  FIX 3  ARCL X  PROMPT

67*LBL D
RCL 06  RCL 04  +  RCL 00  XEQ 20  "Ze"  FIX 2
ARCL X  PROMPT

77*LBL E
RCL 00  RCL 03  XEQ 20  RCL 04  +  1/X  RCL 05  1
+  *  RCL 01  1/X  +  1/X  "ZS="  FIX 2  ARCL X
PROMPT  GTO E

97*LBL 20
STO Z  RCL Y  +  1/X  RDN  *  R^  *  RTN  END
```

C. JEGOUZO

ASSEMBLEUR

PPC-Paris	Récapitulation des tokens	*16*
P. David	Les débuts	*17*
J.J. Dhénin	Traitement de tableaux	*21*

BASIC

J.P. Bondu	Encore un scan... dale	*32*
G. Toublanc	Intégrales et complexes	*32*
X. Bille	Survol des fichiers de données	*34*
J.P. Bondu	Programme "SCANALEX"	*36*
G. Toublanc	Programme "INTEG"	*37*

LE COIN DES LHEX

38

RECAPITULATION DES TOKENS

Comme chaque année maintenant, nous publions la liste complète des tokens attribués dans notre Lex (ID #E1).

Certains mots-clefs, ne présentant plus beaucoup d'intérêt ou étant dépassés, il a été décidé de les supprimer, afin de laisser la place aux nouveaux venus. Ce sont :

```
FKEY
CMD16                    (moins puissant que STACK)
XERR
PRIM            (remplacé par PRIM de G. Toublanc)
CURSDOWN
CURSKEYS
INDATE$                    (voir JPC No 36 page 4)
KBD             (écrasé sans précaution par DATE+)
```

Quant à FRAC$ (XFN 225088), il n'est pas encore paru. Vous le verrez prochainement surgir dans nos colonnes...

```
DESAL
XFN   225001 ADBUF$       $=ADBUF$($)
XFN   225002 ASC$         $=ASC$($)
XFN   225003 ATH$         $=ATH$($)
XFN   225004 HTA$         $=HTA($)
XFN   225005 RED$         $=RED($)

REPLEX
XFN   225006 REPLACE$     $=REPLACE$($,$,$ [,$])

FILELEX
XFN   225007 FILE?        n=FILE?($)

ATTNLEX
XWORD 225008 ATTN         ATTN ON|OFF

DRIVELEX
XWORD 225009 DISABLE      DISABLE $
XWORD 225010 ENABLE       ENABLE $

FKEYLEX
* XWORD 225011 FKEY         FKEY $

DESLEX
XFN   225012 CONTRAST     n=CONTRAST()
XWORD 225013 INVERSE      INVERSE
XFN   225014 INV$         $=INV$($)
XFN   225015 PAINT        n=PAINT(n,n,n)

ENDUPLEX
XFN   225016 ENDUP$       $=ENDUP$()
XWORD 225017 ENDUP        ENDUP $
XFN   225018 STARTUP$     $=STARTUP$()
XWORD 225019 EXECUTE      EXECUTE $

CMD16LEX
* XWORD 225019 CMD16         CMD16

COMBARR
XFN   225020 ARR          n=ARR(n,n)
XFN   225021 COMB         n=COMB(n,n)

HMSLEX
XFN   225022 HMS+         n=HMS+(n,n)
XFN   225023 HMS-         n=HMS-(n,n)
XFN   225024 HMS          n=HMS(n)
XFN   225025 HR           n=HR(n)

XERRLEX
* XFN   225026 XERRLEX       n=XERR(n)

STKLEX
XWORD 225027 STACK        STACK n

KSPEEDLX
XWORD 225028 KSPEED       KSPEED n

MARGELEX
XWORD 225029 MARGIN       MARGIN n

PRIMLEX
* XFN   225030 PRIM          n=PRIM(n)

ONKEYLEX
* XFN   225031 CURSDOWN      n=CURSDOWN()
* XFN   225032 CURSKEYS      n=CURSKEYS()

MENULEX
XFN   225033 MENU         n=MENU(n [,n])

FORMALEX
XFN   225034 CENTER$      $=CENTER$($,n)
XFN   225035 CESURE       n=CESURE($,n)
XFN   225036 FORMAT$      $=FORMAT$($,n)
XFN   225037 REDUCE$      $=REDUCE$($)
XFN   225038 SPACE$       $=SPACE$(n)

PRINTLEX
XWORD 225039 BELL         BELL
XWORD 225040 BOLD         BOLD ON|OFF
XWORD 225041 CR           CR
XFN   225042 ESC$         $=ESC$([$])
XWORD 225043 FF           FF
XWORD 225044 LF           LF [n]
XWORD 225045 MODE         MODE n
XWORD 225046 PERF         PERF ON|OFF
```

```
XWORD 225047 PL           PL [n [,n]]
XWORD 225048 UNDERLINE    UNDERLINE ON|OFF
XWORD 225049 WRAP         WRAP ON|OFF

INDATLEX
* XFN   225050 INDATE$     $=INDATE$()

KBDLEX
* XWORD 225051 KBD         KBD n

DATELEX
XFN   225051 DATE+        n=DATE+(n,n)
XFN   225052 DDAYS        n=DDAYS(n,n)
XFN   225053 DMY          DMY
XFN   225054 DOW$         $=DOW$(n)
XFN   225055 DOW          n=DOW(n)
XFN   225056 MDY          MDY

MMLEX
XFN   225057 MAXD         MAXD(n|$)
XFN   225058 MAXM         MAXM(n|$)
XFN   225059 MEMD         MEMD(n|$)
XFN   225060 MEMM         MEMM(n|$)

EXITLEX
XWORD 225061 EXIT         EXIT var

PPOLL
XFN   225062 NLOOP        n=NLOOP([n])
XFN   225063 PPOLL        n=PPOLL([n])
XWORD 225064 SLEEP        SLEEP
XFN   225065 SRQ          n=SRQ([n])

STRUC1
XWORD 225066 END          END WHILE
XWORD 225067 WHILE        WHILE n
XWORD 225068 REPEAT       REPEAT
XWORD 225069 UNTIL        UNTIL n
XWORD 225070 LEAVE        LEAVE

SWAPLEX
XWORD 225071 SWAP         SWAP var,var

SCANLEX
XFN   225072 ENTRY$       $=ENTRY$($ [,n])
XFN   225073 TOKEN        n=TOKEN($ [,n])
XFN   225074 TYPE         n=TYPE($ [,n])

FINDLEX
XWORD 225075 FIND         FIND $

LINKLEX
XFN   225076 LABEL$       $=LABEL$($)
XFN   225077 MNEMO        n=MNEMO($)

GRAPHLEX
XWORD 225078 GLINE        GLINE n,n,n,n,n
XWORD 225079 GPSET        GPSET n

RPLCLEX
XFN   225080 RPLC$        $=RPLC$($,$,$)

SRINKLX
XWORD 225081 SHRINK       SHRINK $

DIVILEX
XFN   225082 FPRM         n=FPRM(n [,n])
XFN   225083 NPRM         n=NPRM(n,n)
XFN   225084 PGCD         n=PGCD(n,n...n)
XFN   225085 PHI          n=PHI(n)
XFN   225086 PPCM         n=PPCM(n,n...n)
XFN   225087 PRIM         n=PRIM(n [,n])

FRACLEX
XFN   225088 FRAC$        $=FRAC$(n [,n])

POSILEX
XFN   225089 POSI         n=POSI($,n [,n])

BASICLEX
XWORD 225090 BLIST        BLIST [$ [,n [,n]]]
XWORD 225091 PBLIST       PBLIST [$ [,n [,n]]]
XWORD 225092 RENUMREM     RENUMREM [n[,n[,n[,n]]]]
```

LES DEBUTS

Cela fait déjà longtemps que vous adhérez à PPC-Paris. Vous voyez avec envie les articles traitant d'assembleur HP71B. Vous avez acquis les IDS par un moyen ou par un autre.

Mais voilà : vous n'arrivez pas à vous y mettre, le sujet est difficile, très difficile. Et vous ne savez pas par quel bout l'appréhender.

Il est de l'intérêt général que le plus de gens possible se mettent à l'assembleur du HP71B. Et c'est pour vous, pour vous encourager que je rédige cet article. J'espère que ce sera le petit coup de pouce qui vous fera passer à l'acte.

QUE FAUT-IL ?

Pour faire de l'assembleur sur HP71B, il faut :

1 HP71B
1 module Forth/Assembleur
1 moyen de sauver ses programmes (cartes, K7...)

IDS I (fonctionnement général du HP71B)
IDS III (listing commenté des 64Ko de rom interne)
Les JPC à partir du No 20 (déc. 84/Janv. 85)

et de la matière grise !

PAR QUOI COMMENCER ?

Là, il y a deux solutions. Le Forth ou le Basic.

Les primitives Forth sont généralement faciles à écrire, et conviennent bien pour apprendre à manipuler les instructions du microprocesseur. Néanmoins, et c'est mon avis personnel, on apprend beaucoup plus en faisant des fonctions Basic. Et en plus (c'est toujours mon avis personnel), c'est bien plus rigolo...

Donc, vous avez choisi librement d'écrire de l'assembleur à utiliser en basic. Par quoi commencer ?

Le plus simple, ici, ce sont les fonctions. Il existe deux sortes de programmes assembleur pour le basic :

Les ordres (ou statements)
Ex: BEEP, GOTO, INPUT...

Les fonctions
Ex: LOG, UPRC$, POS...

Les fonctions renvoient toujours un résultat. C'est le mécanisme le plus simple pour débuter. L'exemple de cet article est une fonction.

OU SONT LES DONNEES ?
OU SERA LE RESULTAT ?

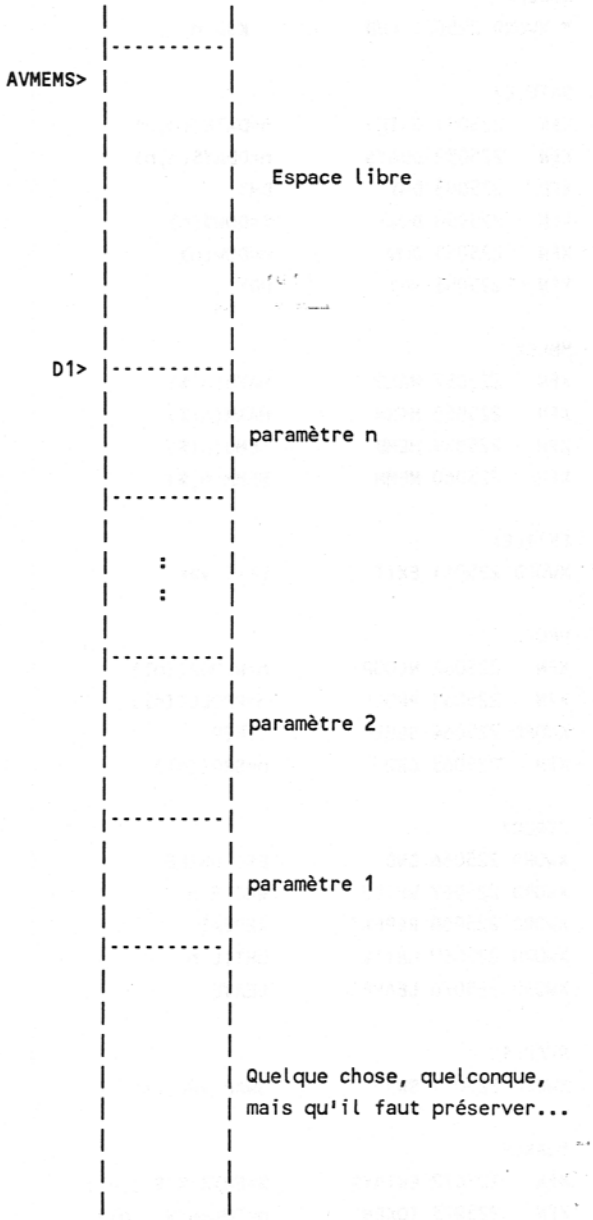
Dans le HP71, les fonctions prennent les données sur la Math-Stack, et laissent le résultat sur celle-ci, en ayant pris soin d'enlever les données.

Exemple : imaginons que nous cherchions à évaluer LOWC$(UPRC$("areuh")) (conversion en minuscules de la conversion en majuscules de "areuh").

Nous trouverons la chaîne "AREUH" (c'est à dire déjà traitée par UPRC$) sur la Math-Stack. Et nous mettrons la chaîne "areuh" sur cette même Math-Stack, à la place de "AREUH".

La Math-Stack est un espace libre qui sert pendant l'évaluation des fonctions. Grosso modo, cela ressemble à :

Adresses basses



Adresses hautes

Chaque paramètre contient un en-tête (header) qui identifie son type (réel, tableau, chaîne, complexe). En pratique, cet en-tête ne nous importe pas quand on passe par les routines HP (POP1S, POP1N...), qui le testent pour nous.

Si votre fonction admet n paramètres (n<16), alors elle devra enlever les n paramètres et mettre le résultat à la place.

En pratique, cela revient à déplacer le pointeur D1 (pointeur de Math-Stack).

Venons-en maintenant à notre Lex LOWCLEX. La seule et unique fonction est destinée à convertir une chaîne en minuscules.

Il faut donc accepter une chaîne alphanumérique, enlever son en-tête, remplacer tous les caractères par les mêmes en minuscules, et reconstruire l'en-tête.

En fait, nous nous trouvons ici devant une fonction particulière : la longueur de la chaîne résultat est la même que celle de l'argument. Il n'y aurait donc pas besoin d'enlever l'en-tête. Mais pour illustrer la méthode générale, j'ai choisi de procéder de manière classique.

Le header est évalué par POP1S, qui vérifie en même temps que l'élément est bien une chaîne, et retourne la longueur de celle-ci.

Ensuite, une boucle est faite pour la conversion.

L'en-tête est enfin reconstitué par ADHEAD, qui renverra le contrôle au Basic.

Voilà, vous en savez autant que moi. J'espère que vous étudierez et comprendrez le programme. J'ai essayé de mettre suffisamment de commentaires pour le rendre compréhensible.

Et n'oubliez pas d'envoyer vos oeuvres à JPC !

A Bientôt

Pierre David (SIG, PC37, CHHU616)

```
      LEX    'LOWCLEX'
************************************************
*
* Lex "LOWCLEX" : fichier Lex didactique à l'usage
* des membres de PPC-Paris qui veulent en savoir
* plus sur l'assembleur du HP71.
*
* Fonction présente :
*   LOWC$ ( <chaîne numérique> )
*   convertit une chaîne de caractères en
*   minuscules (pendant de UPRC$ pour les
*   majuscules).
*
* Note : ne pas rentrer les commentaires dans le
* fichier source. Cela prend de la place, et du
* moment que vous avez ces commentaires dans votre
* Journal préféré...
************************************************

*
* Id du Lex.
*   '#' indique qu'il s'agit d'un nombre en
*     hexadécimal.
*   analogie : l'Id est au HP71 ce que le premier
*     numéro d'un XROM est à la HP41. L'Id
*     identifie le fichier Lex.
*   5C (92 en décimal) est parmi les Id de scratch
*     alloués par HP. C'est à dire les Id
*     utilisables pour nos essais.
*   Si jamais une fonction est d'intérêt suffisant
*     PPC-Paris lui attribue un numéro dans l'Id
*     de PPC-Paris (#E1, ou 225), ce qui est
*     généralement le cas.
*   Les Id de scratch sont : 5C, 5D et 5E
*     C'est ceux-ci que vous utiliserez pour vos
*     premiers essais
*
      ID     #5C
*
* La table de message
*   Le '0' indique qu'il n'y a pas de table de
*     messages.
*   Nous verrons ça une autre fois, peut-être.
*   En tous cas, sachez que la construction d'une
*     table de messages est fastidieuse, et qu'il
*     vaut mieux la laisser à des programmes (il
*     en existe un, en Basic)
*
      MSG    0
*
* La routine d'interception de poll
*   cf ci-dessus : le '0' indique qu'il n'y a pas
*   de routine d'interception de poll.
*
      POLL   0
*
* La partie constituant l'en-tête du fichier est
* finie. Maintenant, il y a les descriptions des
* fonctions, au nombre de une ici :
*
*
* L'ordre ENTRY indique l'adresse ou commencera
* l'exécution de la fonction.
*
      ENTRY  lowc
*
* CHAR #F indique qu'il s'agit d'une fonction.
* (et non d'un "statement").
*
      CHAR   #F
```

```
*
* KEY est suivi du nom de la fonction. Le '$' est
* le seul moyen pour le système Basic de savoir
* qu'il s'agit d'une fonction alphanumérique.
* Le nom doit être limité à 8 caractères (y
* compris le '$').
*
      KEY    'LOWC$'
*
* Vient enfin le numéro de token de la fonction,
* c'est à dire le deuxième numéro d'un XROM HP41
* pour garder l'analogie ci-dessus.
*
      TOKEN  28
*
* ENDTXT = fin de la partie obligatoire du Lex.
* C'est la fin de la déclaration des noms et des
* caractéristiques des fonctions présentes dans ce
* Lex.
*
      ENDTXT

*
* C'est ici que commence véritablement le Lex.
*



*
* Ici, nous définissons les noms des routines
* internes HP que nous allons utiliser :
*
ADHEAD EQU    #181B7
D=AVMS EQU    #1A460
POP1S  EQU    #0BD38
RANGE  EQU    #1B07C

*
* Juste avant le point d'entrée de la fonction
* (défini par ENTRY), il y a la description des
* paramètres. Il y a ici :
*
      NIBHEX 4        Une chaîne alphanumérique
      NIBHEX 1        Un paramètre au minimum
      NIBHEX 1        Un paramètre au maximum
*
* Le point d'entrée de la fonction. C'est ici où
* arrive le système, quand il va essayer d'évaluer
* LOWC$("AREUH").
*
* A cet instant là, nous aurons :
* P=0
* Mode = Hex
* 4 niveaux de pile (RSTK) disponibles
* la chaîne ("AREUH") est sur la Math-Stack
* C(S) contient le nombre de paramètres (ici 1)
* D0 contient le pointeur programme
* D1 contient le pointeur sur la Math-Stack
* D(A) = AVMEMS
* Le premier travail à faire sera de sauvegarder
* D0, pour le restituer en bon état à la fin.
*
lowc
*
* D0 est sauvegardé dans la pile de retours, car
* nous n'allons pas beaucoup l'utiliser dans ce
* Lex. Si ça n'avait pas été le cas, nous aurions
* sauvé D0 dans FUNCD0 qui est prévu pour ça.
*
      CD0EX
      RSTK=C
*
* Il faut maintenant regarder ce que nous avons
* sur la Math-Stack. L'utilitaire POP1S est fait
* pour cela : il examine le "stack-header", et
* en déduit la longueur de la chaîne. En sortie,
* D1 = adresse de 'H'
* A(A) = 10 (= 2x5 caractères)
*
      GOSBVL POP1S
*
* D1 pointe sur le 'H'. Nous allons le faire
* pointer sur le 'A'.
* L'adresse ainsi obtenue sera sauvegardée dans
* R1 pour être utilisée par ADHEAD.
* La chaîne sera parcourue de 'A' à 'H'.
*
*         | #00000  |
*         |         |
*         |         |
*         |---------|
*  (1) -> |  Stack  |
*         |  header |
*         |---------|
*  (2) -> |    H    |
*         |---------|
*         |    U    |
*         |---------|
*         |    E    |
*         |---------|
*         |    R    |
*         |---------|
*         |    A    |
*         |---------|
*         |         |<- (3)
*         |         |
*         |         |
*         | #FFFFF  |
*
* (1) : avant le POP1S
* (2) : après le POP1S
* (3) : après le C=C+A
*
      CD1EX          C(A) := adr. de 'H'
      C=C+A  A       C(A) := adr. juste avant 'A'
      D1=C           D1 := adr. juste avant 'A'
      R1=C           R1 := début de la chaîne
```

```
* Nous allons maintenant calculer la longueur en
* octets, par une division par 2 de A(A).
* A(A)=xxxxxxxxxxxllllll
* Nous allons faire une divison par 2, c'est à
* dire un décalage à droite de 1 bit. Mais alors,
* le premier 'x' va interférer avec 'llllll' (on ne
* peut décaler que le registre complet).
*
       B=0    M        premier x:=0 (entre autres)
       B=A    A        B(W)=x0000000000llllll
       BSRB            B(A)=longueur en octets
*
* B(A) est maintenant la longueur de la chaîne en
* octets. Cela va nous servir de compteur de
* boucle.
* FOR B(A)=B(A) TO 1 STEP -1
*
*
* La boucle qui suit est d'un type que vous voyez
* souvent dans mes programmes :
* Vous voulez compter n fois
*
*      LC(5)  n
*      GOTO   b20
* b10
*       <code de la boucle>
*
* b20  C=C-1  A
*      GONC   b10
*
* Si si, essayez, ça marche !
*
       GOTO   lowc20

lowc10 D1=D1- 2        D1 := adr. exacte du 'A'
       A=DAT1 B        A(B) := 'A'

       LCASC  'ZA'     } A(B) dans l'intervalle
       GOSBVL RANGE    } 'A'..'Z' ?
       GOC    lowc20   Non. On ne fait rien

       LC(2)  32       Conversion en majuscule
       A=A+C  B
       DAT1=A B        Et remise en mémoire

lowc20 B=B-1  A        Décrémentation du compteur
       GONC   lowc10   Saut si non fini
*
* La boucle est maintenant finie.
* D1 pointe sur 'h'
* R1 pointe juste avant 'a'
*
* Il faut renvoyer ces informations au système,
* par l'intermédiaire de la routine ADHEAD.
* Mais auparavant, vous vous souvenez que nous
* avons mis D0 dans la pile de retours ?
* Il faudrait peut-être penser à le rapatrier...
       C=RSTK
       D0=C
*
* ADHEAD attend dans D(A) la valeur de AVMEMS
*
       GOSBVL D=AVMS
*
* Pour l'explication exacte de ce ST=0, voir la
* documentation de ADHEAD.
*
       ST=0   0
       GOVLNG ADHEAD
*
* Et voilà 74 octets d'assembleur HP71. Alors, ça
* ne vous donne pas, vous aussi, l'envie de vous
* y mettre ?
*
       END
```

TRAITEMENT DE TABLEAUX
VECTLEX

Voici 5 nouvelles instructions pour votre HP-71. La préoccupation qui est à la base de ce travail est simple à saisir : la manipulation des colonnes d'un tableau n'est pas aisée et prend bien du temps à l'exécution. Il était donc nécessaire de travailler en assembleur.

Les cinq instructions sont :

Une fonction :
SOMME des éléments d'une colonne

Quatre ordres :
MULTIPLICATION d'une colonne par une valeur
ADDITION d'une valeur à chaque terme d'une colonne
COPIE d'une colonne
TRI d'une colonne

Quelques remarques sur ces instructions :

- le tableau peut être de n'importe quel type (réel, entier ou court),

- la copie peut être effectuée dans le tableau lui-même ou sur un autre tableau,

- la colonne n'est pas nécessairement considérée dans sa totalité : des paramètres définissent les numéros de ligne pris en compte.

Il a été nécessaire de rompre avec l'habitude de désignation des tableaux pour définir totalement les instructions. Donc, au début, soyez attentifs à la syntaxe : LE NOM DU TABLEAU EST A METTRE ENTRE LES PARENTHESES.

VSUM

Syntaxe : S=VSUM (V,L1,C1,L2)

Renvoie la somme des éléments du tableau V, de la colonne C1, de la ligne L1 à la ligne L2 incluses.

VADD

Syntaxe : VADD (X)+(V,L1,C1,L2)

Effectue l'addition sur chacun des termes de la colonne C1 du tableau V, de la ligne L1 à la ligne L2 par la constante X. Les nouvelles valeurs remplacent les valeurs initiales.

VMULT

Syntaxe : VMULT (X)*(V,L1,C1,L2)

Effectue le produit de chaque terme de la colonne C1 du tableau V, de la ligne L1 à la ligne L2 comprises, par la constante (X). Les nouvelles valeurs remplacent les valeurs initiales.

VCOPY

Syntaxe : VCOPY (V1,L1,C1,L2) TO (V2,N1,M1,N2)

Effectue la copie de la colonne C1 du tableau V1 sur la colonne M1 du tableau V2. Les tableaux peuvent être de types différents. Les différences L2-L1 et N2-N1 ne sont pas nécessairement égales ; l'instruction s'arrête d'elle-même lorsque la plus petite colonne est copiée. Il est possible de copier une colonne partiellement sur elle-même sans précautions.

VSORT

Syntaxe : VSORT (V,L1,C1,L2)

Trie les lignes d'un tableau suivant les valeurs d'une colonne C1. Le tri est stable.

Pour vous mettre dans le bain, voici quelques informations sur le traitement des valeurs numériques dans le HP-71.

REPRESENTATION INTERNE DES DONNEES

Exposé des formats de représentation des variables numériques dans la mémoire du HP-71 ou dans les registres du CPU.

1 Types de données

Le HP-71 supporte sept types de données. Le type de donnée d'une variable est identifié par l'apparence du registre de variable. Les valeurs réelles simples sont conservées directement dans le registre de variable et peuvent être identifiées par le quartet de poids faible qui appartient à l'ensemble [0 à 9] (et qui est également interprété comme le quartet de poids faible de l'exposant). Si ce quartet de poids faible du registre de variable n'appartient pas à cet ensemble, le registre est utilisé comme descripteur de l'un des six types de données suivants :

A	Entier	(simple ou tableau)
B	Réel court	(simple ou tableau)
C	Tableau de réels	
D	Complexe court	(simple ou tableau)
E	Complexe	(simple ou tableau)
F	Chaîne	(simple ou tableau)

2 Registres

Ce paragraphe expose la représentation des variables en mémoire. Il contient une introduction à la représentation des nombres dans le CPU.

Les nombres dans les registres du CPU :

Quand un nombre est transporté dans un registre du CPU, le processus de rappel (recherche dans la mémoire et chargement dans le CPU en passant par la mathstack) doit le convertir en un ou deux nombres de représentation standard (respectivement pour les valeurs de type réel ou complexe) ; la représentation standard est la suivante :

```
15                              0
+-+-------------------------+-----+
|S|   Mantisse              | Exp |
+-+-------------------------+-----+
 1           12               3       16 quartets
```

l'ordre des chiffres correspond à l'ordre des quartets du registre :

```
15                              0
+-+-------------------------+-----+
|S| M11 . . .  . . . . .M0 |E2.E0|
+-+-------------------------+-----+
 1           12               3       16 quartets
```

La mantisse n'est pas signée, un champ séparé (le champs S) représentant le signe (0=+, 9=-). L'exposant est exprimé sous la forme d'un complément à 10. Cette représentation est la forme normale attendue par tous les utilitaires utilisant des arguments de 12 chiffres en virgule flottante.

Beaucoup d'utilitaires travaillent avec des valeurs exprimées sur 15 chiffres pendant les calculs intermédiaires afin d'obtenir une plus grande précision. Typiquement, lors de l'implantation d'une fonction, le paramètre est prélevé (sous sa forme 12 chiffres) réexprimé sous sa forme 15 chiffres, l'utilitaire de calcul appelé et enfin le résultat est réexprimé arrondi sous sa forme 12 chiffres.

La représentation 15 chiffres est disposée ainsi dans les registres :

```
15                              0
+-+-------------------------+-----+
| |        Mantisse         |  !  |
+-+-------------------------+-----+
 1           15                      16 quartets

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

où l'exposant a été réexprimé sur 5 chiffres (incluant le signe si l'exposant est négatif).

3.4 Accès aux variables à partir d'un programme binaire.

Voir JPC No 29 (Nov 1985), Page 29.

Jean-Jacques Dhénin
(PC177 SIG5)

Nota : Le Lex est placé en id #5C car il va certainement évoluer dans l'avenir. Le placer en #E1 aurait posé des problèmes de compatibilité avec les versions futures.

```
        LEX     'VECTLEX'

        TITLE   Vecteurs manipulations
*       Jean-Jacques DHENIN  (SIG#5 P#177)
*                43 36 12 05
*
*       1ère partie  : TPOURC    29/12/85
*
*********************************************
*
id      EQU     #5C
t       EQU     100

Vcfg0   ID      id
        MSG     0
        POLL    pvect

        ENTRY   Sum
        CHAR    #F
        ENTRY   Pro
        CHAR    #D
        ENTRY   Copy
        CHAR    #D
        ENTRY   Tri
        CHAR    #D
        ENTRY   Plus
        CHAR    #D

        KEY     'VSUM'
        TOKEN   t
        KEY     'VMULT'
        TOKEN   1+t
        KEY     'VCOPY'
        TOKEN   2+t
        KEY     'VSORT'
        TOKEN   3+t
        KEY     'VADD'
        TOKEN   4+t

        ENDTXT
```

```
=A-MULT EQU   #1B349
=AD2-12 EQU   #0C35F
=AD2-15 EQU   #0C363
=ARGERR EQU   #0BF19
=ARYDC  EQU   #05178
=ARYELM EQU   #0B5A7
=ARYSIZ EQU   #0B61B
=AVMEMS EQU   #2F594
=COLLAP EQU   #091FB
=COMCK  EQU   #036CD
=CSLC5  EQU   #1B435
=CSRC10 EQU   #1B432
=CSRC3  EQU   #1B421
=CSRC4  EQU   #1B41E
=CSRC5  EQU   #1B41B
=CSRC6  EQU   #1B418
=CSRC8  EQU   #1B42C
=D1FSTK EQU   #1955D
=D1MSTK EQU   #1954E
=EXAB2  EQU   #0D40E
=EXPEX- EQU   #0F178
=EXPEXC EQU   #0F186
=FDO    EQU   #2F8BB
=FNRTN4 EQU   #0F238
=FORSTK EQU   #2F59E
=FRO    EQU   #2F89B
=FR1    EQU   #2F8AB
=FUNCR1 EQU   #2F8AB
=GNXTCR EQU   #03064
=GTEXT++ EQU  #05192
=INTGR  EQU   #0F99B
=MEMERR EQU   #0944D
=MFERR  EQU   #09393
=MOVEDM EQU   #1B0EE
=MOVEUM EQU   #1B15C
=MP2-12 EQU   #0C432
=MSTKD1 EQU   #1953C
=NXTELM EQU   #148AC
=NTOKEN EQU   #0493B
=NUMCK  EQU   #0369D
=NXTSTM EQU   #08A48
=OAGNXT EQU   #03060
=OUTBY+ EQU   #02CE5
=OUTBYT EQU   #02CE8
=OUTELA EQU   #05303
=POP1N  EQU   #0BD1C
=RCCD2  EQU   #0D41C
=RDATTY EQU   #17CC6
=RESTPR EQU   #03172
=RNDAHX EQU   #136CB
=SCRTCH EQU   #2F901
=SFLAGC EQU   #13601
=SHRT   EQU   #0F96C
=SPLITA EQU   #0C6BF
=STAB2  EQU   #0D400
=STKVCT EQU   #1470C
=STMTR0 EQU   #2F871
=SYNTXe EQU   #02E2B
=TST12A EQU   #0D476
=VARP   EQU   #0350E
=eSUBSC EQU   #1C
=pCONFG EQU   #FB
=tTO    EQU   #F3
=uRES12 EQU   #0C994

pvect  ?B=0   B
       GOYES  hVER$

       GONC   hVER$1

hVER$  C=R3
       D1=C
       A=R2
       D1=D1- (VER$en)-(VER$st)-2
       CD1EX
       ?A>C   A
       GOYES  hVER$1
       D1=C
       R3=C
VER$st LCASC  ' VT:C'
VER$en DAT1=C (VER$en)-(VER$st)-2
hVER$1 RTNSXM
*
Ures12 GOVLNG =uRES12
cslc5  GOVLNG =CSLC5
csrc4  GOVLNG =CSRC4
csrc5  GOVLNG =CSRC5
stab2  GOVLNG =STAB2
d1fstk GOVLNG =D1FSTK
d1mstk GOVLNG =D1MSTK
invpar GOVLNG =ARGERR
ivvarE GOVLNG =RDATTY
mstkd1 GOVLNG =MSTKD1
movedm GOVLNG =MOVEDM
moveum GOVLNG =MOVEUM
nxtstm GOVLNG =NXTSTM
outby+ GOVLNG =OUTBY+
splita GOVLNG =SPLITA
*********************************************
* Nom : argan   - préparation aux
*                 manipulations,
*                 analyse des arguments.
* Nom : velm    - Calcul de l'adresse d'un
*                 élément dont on connait les
*                 coordonnées dans le
*                 tableau.
* Remarques :
*       Sous la mstk on trouve le dope
*        vector,
*       se reporter aux routines STKVCT et
*       NXTELM pour conditions d'utilisation.
* Historique :
* Date        Programmeurs      Modifications
* --------- --------------- ---------------
* 29/11/85    JJD             Création
*********************************************
```

```
arg
        GOSBVL =RNDAHX  -> A[A] = arg en Hex
        GONC   invpar   Si négatif -> err
        DAT0=A A        place arg dans F-R0-
        D1=D1+ 16       Passe au paramètre
*                       suivant
        D0=D0+ 5        Passe F-R0 suivant
        RTN

vexp-   GOSBVL =COLLAP  @ MSTK = @ FORSTK
vexpc   GOSBVL =EXPEXC  Evaluation
argan   CD0EX           Sauve D0 ...
        D0=(5) =FD0     ...dans FD0
        DAT0=C A        .
        D0=(2) =FR0     Début de l'analyse
vect1   GOSUB  arg      F-R0-0 = L2
        GOSUB  arg      F-R0-1 = C1
        GOSUB  arg      F-R0-2 = L1
        GOSUB  mstkd1   Préserve le dope vect.
        A=DAT1 W        Lit le 'Dope vector'
        LCHEX  #A
        ?A<C   P        Si variable simple
        GOYES  ivvarE   | alors erreur
        C=A    W
        CSRC            Passe le type
        ST=0   9        ST=(9) 1 dim
        P=C    0        P = Nb de dim
        ?P=    2        Si base 2
        GOYES  vect2    | alors ST(9)= 0
        ST=1   9        | sinon ST(9)= 1
vect2   P=     0
        CSRC            Passe le nb de dim
        ST=0   8        ST(8)= base 0
        ?C=0   P        Base 0 ?
        GOYES  vect11   Oui ST=0 8
        ST=1   8        ST8= base 1
vect11  CSR    A        Passe et A[A]= nb col
        D0=D0- 10       F-R0-1 = C1
        ?ST=0  9        Si base 2
        GOYES  vect22   | alors continue
        C=0    A        | sinon
        ?ST=0  8        | si base 0
        GOYES  vect22   | | alors A[A]= 0
        C=C+1  A        | | sinon A[A]= 1
vect22  ST=0   0        ST0  pour 2 boucles
        A=DAT0 A        A(A)= C1
        ?A>C   A        Si dépassement
        GOYES  Esubsc   |alors ERR:Subcript
        ?ST=0  8        Si base 0
        GOYES  vect12   | alors ok
        ?A=0   A        | Sinon Si Col 0
        GOYES  Esubsc   | | alors ERR
        C=C-1  A        | Max col -1 si base 1
        A=A-1  A        | C1-1
vect12  C=C+1  A        C[A] = max colonnes
        DAT0=A A        F-R0-1 No col. norm
        D1=D1+ 3
        DAT1=C 3
        D1=D1- 3
        GOSUB  csrc4    C[A] = nb lignes
        CSR    A
        D0=D0+ 5        D0= #2F8A5 F-R0-2
vect13  A=DAT0 A        1 : F-R0-2 ; 2 : F-R0-0
        ?A>C   A        L1 ou L2 > lgn max ?
        GOYES  Esubsc
        ?ST=0  8        Base 0 ?
        GOYES  vect14
        ?A=0   A        Base 1 : Ligne 0 ?
        GOYES  Esubsc
        A=A-1  A        L1 ou L2 -1
        DAT0=A A
vect14  ?ST=1  0        2ème boucle ?
        GOYES  vect15   Oui passe  la suite
        R0=A
        ST=1   0        Prépare 2ème boucle
        D0=D0- 10       D0= #2F89B F-R0-0
        GONC   vect13
Esubsc  P=     0
        LC(2)  =eSUBSC
mferr   GOVLNG =MFERR

vect15  C=R0
        A=A-C  A        L2>L1 ?
        GOC    Esubsc
        DAT0=A A        F-R0-0 = N = L2-L1
        ST=0   8        Pas de réécriture
*                       Dopvect
        GOSBVL =STKVCT  sinon écrase STMTR =
*                       DEST
        GOSUB  d1mstk
        SETHEX
        D1=D1+ 7
        P=     3
        C=0    WP
        C=C-1  WP
        P=     0
        DAT1=C 3
        D0=(5) =FUNCR1
        D1=D1- 7
        C=DAT1 W
        DAT0=C W
        D0=D0- 16
        C=DAT0 W
        R0=C
        GOSBVL =CSRC10
        A=C    A
        GOSUB  cslc5
        D=C    A
        GOSUB  cslc5
        C=A    A
        R1=C
        C=D    A

* Maintenant R0[15-10]= Addr et R0(A)= N
*           R1(A)= L1  et R1(9-5)= C1
        GOTO   velm
```

```
vnlmt
       C=R0
       C=C-1  A
       RTNC
       R0=C
       C=R1
       C=C+1  A
       R1=C
       A=C    A
       GOSUB  csrc5

* Calcul de l'adresse d'un élément
* A= numéro de ligne , C= numéro de colonne
*
* Sortie : R0(15-11)= adresse de l'élément
*          B= élément 12 digits

velm   SETHEX
       D=C    A        D= n. de col
       C=0    A
       D0=(5) (=FUNCR1)+3
       P=     0
       C=DAT0 3        C(A)=  nb de col
       D0=D0- 3
       GOSUB  a-mult   l,0
       C=D    A        C= n. col
       C=C+A  A
       B=C    A
velm2  GOSUB  ofst
       D0=D0+ 11
       C=DAT0 A
       C=C+A  A        Calcul de l'adresse
       D1=D1+ 11
       DAT1=C A
       C=R0
       GOSUB  cslc5
       C=DAT1 A
       GOSUB  csrc5
       R0=C            R0[15-11] =addr de
*                      l'élmt
       ST=0   7
       GOSUB  d1mstk
       GOSBVL =NXTELM
       ?ST=0  5
       GOYES  velm21
       GOTO   mferr
velm21 D1=D1+ 16
       GOSUB  mstkd1   MSTK au dessus de
*                      Dopvect.
       P=     0
       SETHEX
       C=R0
       GOSUB  cslc5
       A=C    A
       RTN

* R0=(15-11) Addr de l'élmt; B=élmt; élmt on
* MSTK

* ofst
* calcule le déplacement de n. éléments
* B(A)= nb d'éléments
* D0= FR1


ofst   A=DAT0 P
       C=0    A
       LCHEX  #B
       ?A=C   P
       GOYES  vshr
       ?A<C   P
       GOYES  vint
       LCHEX  #10
       GOTO   mult
vshr   LCHEX  #9
       GOTO   mult
vint   LCHEX  #6
mult   ABEX   A
a-mult GOVLNG =A-MULT

*********************************************
*
* Nom : stelm
*
* Objet :
*       Chargement des éléments calculés dans
*       un tableau.
*
* Historique :
*
* Date       Programmeurs     Modifications
* --------- --------------- ----------------
* 29/11/85   JJD             Création
*
*********************************************

stelm  GOSUB  Ures12
       SETHEX
st12   B=C    W
       C=R0
       GOSUB  cslc5
       D0=C
       GOSUB  d1mstk
       A=DAT1 B
       P=     0
       LCHEX  #B
       ?A=C   P
       GOYES  shorts
       ?A<C   P
       GOYES  intgrs
       A=B    W
       DAT0=A W
       GOTO   cfl-4
shorts
       A=B    W
       GOSBVL =SHRT
       GOTO   cfl-4
```

```
intgrs A=B    W
      GOSBVL =INTGR
cfl-4
      SETHEX
      P=     0
      RTN

*********************************************
*
* Nom : mov1
*
* Objet :
*      Placer une colonne d'un tableau sur la
*      mathstack.
*
* Historique :
*
* Date         Programmeurs     Modifications
* --------- ----------------- -------------
* 10/12/85    JJD              Création
*
*********************************************

mov1   C=DAT1 W
       A=B    W
       DAT1=A W
       D1=D1- 16
       DAT1=C W
       GOSUB  mstkd1
       GOSUB  vnlmt
       RTNC
       GONC   mov1

*********************************************
*
* Routines de parsing et décompilation
*
*********************************************

Plusd  GOSUB  arydc
       LCASC  '+'
       GONC   Prod1    (B.E.T)
Prod   GOSUB  arydc
       LCASC  '*'
Prod1  GOSBVL =OUTBYT
       GOSUB  arydc
       GONC   outela   (B.E.T)
Copyd  GOSUB  arydc-
       LC(2)  =tTO
       A=C    B
       GOSBVL =GTEXT++
Trid   GOSUB  arydc-
outela GOVLNG =OUTELA

arydc  D1=D1+ 2
arydc- LCASC  '('
       ST=0   5
       GOVLNG =ARYDC

Copyp  GOSUB  vectp
       GOSBVL =NTOKEN
       LC(2)  =tTO
       ?A#C   B
       GOYES  restpr
       GOSUB  outby+
       GOSUB  gnxtcr
       GOTO   vectp
restpr GOVLNG =RESTPR
ctep   GOSUB  gnxtcr
       LCASC  '('
       GOSUB  tken?
       GOSUB  outby+
       GOSUB  numck
       LCASC  ')'
       GOSUB  tken?
       GOVLNG =OAGNXT
Plusp  GOSUB  ctep
       LCASC  '+'
       GOTO   Prop1
Prop   GOSUB  ctep
       LCASC  '*'
Prop1  GOSUB  tken?
       D1=D1+ 2
vectp  GOSUB  gnxtcr
       LCASC  '('
       GOSUB  tken?
       D1=D1+ 2
       GOSBVL =VARP
       GOSBVL =COMCK
       LC(2)  2
       RSTK=C
arg3   GOSUB  numck
       C=RSTK
       C=C-1  B
       GOC    P+
       RSTK=C
       LCHEX  #F1
       GOSUB  tken?
       GONC   arg3
P+     LCASC  ')'
tken?  ?A#C   B
       GOYES  syntxE
       RTNCC
syntxE GOVLNG =SYNTXe
numck  GOVLNG =NUMCK
gnxtcr GOVLNG =GNXTCR
*********************************************
* Nom : VSUM    syntaxe : VSUM (V,L1,C1,L2)
*
* Objet : calculer la somme des éléments d'une
*         colonne d'un tableau, de la ligne L1
*         la ligne L2.
* Historique :
*    Date   Programmeur    Modifications
* -------- ------------ --------------------
* 29/11/85  JJD           Création
*********************************************
```

```
        NIBHEX 888A44
Sum
        GOSUB  argan
        SETDEC
        A=B    W
        GOSUB  splita
        GOSUB  stab2
        SETHEX
sum1
        GOSUB  vnlmt
        GOC    sum2
        ABEX   W
        SETDEC
        GOSUB  splita
        GOSBVL =RCCD2
        GOSBVL =AD2-15
        GOSUB  stab2
        SETHEX
        GOTO   sum1
sum2
        D0=(5) =FD0
        C=DAT0 A
        D0=C
        SETDEC
        GOSBVL =EXAB2
        GOSUB  Ures12
        GOVLNG =FNRTN4

*********************************************
*
* Nom : VPLUS   Syntaxe : VPLUS (X)+(V,L1,C1,L2)
*
* Objet :
*       Effectuer l'addition sur chacun des
*       termes d'une colonne d'un tableau par
*       une constante. Les nouveaux éléments
*       remplacent les valeurs initiales.
*
* Historique :
*
*    Date    Programmeur   Modifications
* -------- ------------- ------------------
* 28/12/85  JJD           Création
*
*********************************************

ctepr
        D0=D0+ 2
        GOSBVL EXPEX-   Collapse la mstk et pose
*                       la constante.
        GOSBVL =POP1N
        SETHEX
        D1=(5) =STMTR0
        DAT1=A W        SCRTCH= Cte
        D0=D0+ 2        Passe TRMNTR )
        GOTO   vexp-
```

```
nxtval
        SETHEX
        GOSUB  d1mstk   pour uRES12
        ABEX   W
        D0=(5) =STMTR0
        C=DAT0 W
        SETDEC
        RTN

        REL(5) Plusd
        REL(5) Plusp
Plus
        GOSUB  ctepr

plus1
        GOSUB  nxtval
        GOSBVL =AD2-12
        GOSUB  stelm
        GOSUB  vnlmt
        GONC   plus1
        GOTO   nxtstm


*********************************************
*
* Nom : VPROD   Syntaxe : VPROD
*       (X)*(V,L1,C1,L2)
*
* Objet :
*       Effectuer le produit de chacun des
*       termes d'une colonne d'un tableau par
*       une constante. Les produits
*       remplacent les valeurs initiales de
*       la colonne.
* Historique :
*
*    Date    Programmeur   Modifications
* -------- ------------- ------------------
* 29/11/85  JJD           Création
*
*********************************************

        REL(5) Prod
        REL(5) Prop

Pro
        GOSUB  ctepr

Pro1
        GOSUB  nxtval
        GOSBVL =MP2-12
        GOSUB  stelm
        GOSUB  vnlmt
        GONC   Pro1
        GOTO   nxtstm
```

```
*********************************************
*
* Nom : VCOPY
* Syntaxe : VCOPY (V1,L1,C1,L2) TO (V2,L3,C3,L4)
*
* Objet :
*       Effectuer une copy d'une colonne d'un
*       tableau sur une autre colonne. Les
*       tableaux peuvent être de types
*       différents
*       Les différences L2-L1 et L4-L3 ne
*       sont pas nécessairement égales, et il
*       est possible de copier une colonne sur
*       elle-même.
*
* Historique
*
*
*    Date      Programmeur     modifications
* --------- ---------------- ---------------
* 15/12/85   JJD              Création
*
*********************************************

       REL(5) Copyd
       REL(5) Copyp
Copy
       GOSUB  vexp-
       GOSUB  mov1
       D1=D1+ 16
       GOSUB  mstkd1
       D0=(5) =FD0
       C=DAT0 A
       D0=C
       D0=D0+ 2
       GOSUB  vexpc
       D1=(5) =FORSTK
       C=DAT1 A
       R2=C
       GOSUB  d1mstk
mov3
       A=R0
       C=R2
       CD1EX
       D1=D1- 16
       D1=D1- 16
       D1=D1- 16
       AD1EX
       ?A#C   A
       GOYES  more
       D1=(5) #00001
more   AD1EX
       R0=A
       A=C    A
       D1=D1+ 16
       D1=D1+ 16
       C=DAT1 W
       AD1EX
       R2=A
       GOSUB  st12
       GOSUB  vnlmt
       GONC   mov3
nxstmx GOTO   nxtstm

*********************************************
*
* Nom : VSORT      Syntaxe : VSORT (V,L1,C1,L2)
*
* Objet :
*       Trier les lignes d'un tableau suivant
*       les valeurs d'une colonne.
* Algorithme :
*       La colonne à trier est recopiée sur
*        la mstk oK à lieu le tri avec
*        actualisation d'une représentation
*        de l'ordre des lgns.
*        Les lgns du tableaux sont placées
*        dans l'ordre selon la
*        représentation.
*
* Historique :
*
*   Date    Programmeur          Modifications
* ------- --------------- ------------------
* 28/12/85      JJD          Création
* 14/01/86      JJD          Inversion du test
*                            et sortie si 1 lg
*
*********************************************

       REL(5) Trid
       REL(5) vectp
Tri
       GOSUB  vexp-
       GOSUB  mov1
       D0=(5) =FR0
       C=DAT0 A
       ?C=0   A
       GOYES  nxstmx
       D0=(2) (=FR0)+10
       C=DAT0 A
       D=C    A
       D0=D0- 10
       C=DAT0 A
       B=C    A
putnb
       D1=D1- 16
       AD1EX
       D1=(5) =AVMEMS
       C=DAT1 A
       C=C-A  A
       D1=A
       GOC    Tri1
memerr GOVLNG =MEMERR
```

```
Tri1   C=D    W
       DAT1=C W
       D1=D1+ 5
       DAT1=C A
       D1=D1- 5
       C=C+1  A
       D=C    A
       B=B-1  A
       GONC   putnb
       GOSUB  mstkd1
       C=DAT0 A
       C=C+1  A
       C=C+1  A
       CSL    A
       R3=C
       D1=D1- 16
       AD1EX
       C=C+A  A
       R0=C          R0= Fin de tri
       D1=(5) =FORSTK
       C=DAT1 A
dep    D1=C          Positione les pointeurs
       D1=D1- 16     Fin de boucle ext ?
       CD1EX
       D1=C
       A=R0
       ?C#A   A      Pour D1= mstk to (top
*                    -1)
       GOYES  ext
       GOTO   res
ext    A=DAT1 W      Lit la valeur min
*                    supposée
       R2=A          R2=  val initiale
       R1=C          R1=  adr de val
       D0=C          D but de boucle interne

sup    D0=D0- 16     Val suivante
       AD0EX
       D0=A
       C=R0
       ?C>=A  A      Pour D0=D1 to top
       GOYES  nxt    Si fin de boucle int.
       A=R2          Rappel min supposé
       C=DAT0 W      Lit val act
       P=     6
       SB=0
       XM=0
       SETDEC
       GOSBVL =TST12A
       SETHEX
       GOC    sup
       CD0EX
       D0=C
       R1=C          R1 =>  pointe nouv min
       C=DAT0 W      Lit le nouveau min
       R2=C
       GOTO   sup    --->

nxt    C=R1          Fin de boucle interne
       D0=C          Pointe le min
       C=DAT1 W      Lit val initiale
       DAT0=C W      Remplace min par val
       A=R2          Rappel min
       DAT1=A W      Remplace val par min
       A=R3          Rappel offset
       CD0EX         Rappel adr de min
       C=C-A  A      Calcule adr de No de
*                    min
       D0=C          Pointe No de min
       CD1EX         Rappel adr de val
*                    initiale
       B=C    A      Sauve adr de val init.
       C=C-A  A      Adr de No de val
*                    initiale
       D1=C          Pointe No de val
*                    initiale
       A=DAT0 A      Lit le No de min
       C=DAT1 A      Lit le No initiale
       DAT0=C A      No init.   la place de
*                    min
       DAT1=A A      No min   la place No
*                    init.
       C=B    A      Rappel  adr val
       GOTO   dep
res
       GOSUB  d1fstk
       CD1EX         C[A] @ end of dest
       B=C    A      B[A] @ end of dest
       A=R3          A[A] = block length+16
       C=C-A  A      C[A] end of source
       AD1EX
       D1=D1- 16
       AD1EX         A[A]=block length
       ABEX   A      A[A] @ end of dest
       GOSUB  movedm
       D1=D1- 16
       GOSUB  mstkd1
       D0=(5) (=FR1)+3
       A=0    A
       A=DAT0 4
       B=A    A
       D0=(2) #AB
       GOSUB  ofst
       B=A    A
       GOSUB  d1mstk
       AD1EX
       A=A-B  A
       D1=(5) =AVMEMS
       C=DAT1 A
       C=C-A  A
       GOC    res1
       GOTO   memerr
res1   A=B    A
       GOSUB  d1fstk
       D1=D1- 11
       C=DAT1 A
```

```
        GOSUB  cslc5
        C=A    A
        R3=C           R3[10-5] No lgn;
*                      R3[A]=bl
        D1=D1- 5
        CD1EX
        D1=C
        RSTK=C
perm
        C=DAT1 W
        GOSUB  csrc5
        R1=C           R1=B...A
        C=R3
        GOSUB  csrc5
        A=C    A
        C=0    A
        GOSUB  velm
        C=R3
        B=C    A
        GOSUB  d1mstk
        D1=D1- 16
        AD1EX
        A=A-B  A
        CD1EX
        GOSUB  moveum

* D1 pointe @ end of dest
* D0 @ end of source ->
* -> @ end of dest pour le prochain move

        D1=D1+ 16      = D1MSTK [ passe
*                      element ]
        D1=D1+ 16
        C=RSTK
        RSTK=C
        CD0EX
        R2=C           R2[A]  @ end of dest
        AD1EX
        D1=A
        C=R3
        GOSUB  csrc5
        B=C    A       A[A]= No lgn à déplacer
        C=R1
        GOSUB  cslc5
        D=C    A
        D0=D0+ 5
b1      C=DAT0 A
        ?C=D   A
        GOYES  b2
        D0=D0- 16
        B=B+1  A
        CD0EX
        D0=C
        ?C>A   A
        GOYES  b1
        GOTO   mferr

b2      C=R1
        DAT0=C A
        A=B    A
        C=0    A
        GOSUB  velm    C[A] @ start of source
        A=R3
        B=A    A       B[A]= bl lgth
        A=R2
        A=A-B  A
        GOSUB  moveum

* Maintenant la nouvelle ligne est à sa place
* L'ancienne est sur la mathstack.
* RSTK @ descripteur courant
* R1= No triée....No lgn dest
* R3[9-5] = No lgn courante R3[A]= bl lgth

        C=R3
        B=C    A
        GOSUB  d1mstk
        D1=D1- 16
        CD1EX
        AD0EX
        GOSUB  movedm

* Maintenant l'ancienne lgn occupe la place
* de la lgn deplacée.

        GOSUB  d1mstk
        D1=D1+ 16
        AD1EX
        C=RSTK
        D1=C
        D1=D1- 16
        CD1EX
        ?C=A   A
        GOYES  nxtst2
        RSTK=C
        D1=C
        C=R3
        GOSUB  csrc5
        C=C+1  A
        GOSUB  cslc5
        R3=C
        GOTO   perm

nxtst2 GOVLNG =NXTSTM

        END
```

ENCORE UN SCAN... DALE !

Avec le programme SCANALEX, paru dans JPC No 34, je commettais l'une de mes plus belles erreurs de programmation. L'algorithme est complètement faux. Je préfère encore le dire haut et fort que de l'entendre murmurer tout bas dans mon dos ! La version précédente ne fonctionne correctement que si l'ordre numérique des tokens correspond à l'ordre alphabétique des intitulés. Ce qui n'est pas toujours le cas. Bref, voici LA bonne version, totalement revue.

Afin de me faire pardonner, j'y ai ajouté des raffinements dont vous me saurez gré (mais si...). D'abord, si votre Lex possède une "speed table", cela vous est signalé. S'il intercepte un poll, il est possible de désassembler le code concerné par appel direct du désassembleur de Michel Martinet. Pour être plus clair, voici ce qui se passe lors de l'examen d'un Lex :

Nous prendrons en exemple EDLEX. Tapez CALL SCAN("EDLEX"). S'affichent les numéros des Lex contenus (F0/52), suivis du nombre de mots clefs contenus dans le premier (F0-7 mots clefs), puis du premier mot :

```
1 DELETE  [D]  e5D3C8 p5D818 d5D876
```

Cela signifie que DELETE a pour token 1, est une commande ([D]), que le code d'exécution débute en 5D3C8, de parse en 5D818, de décompilation en 5D876. Bien sûr, vous trouverez des adresses différentes suivant votre configuration, ceci n'est qu'un exemple. Vous pouvez alors appuyer sur :

- [E] pour désassembler le code Exécutable
- [P] " " " " de Parse
- [D] " " " de Décompilation
- une autre touche pour passer au mot suivant
- [ON] pour sortir du programme

Admettons que vous arriviez ainsi au mot :

```
1 FILESZR [F]  e5D322   11   $
```

Cela signifie que ce mot est une fonction ([F]), dont le code exécutable débute en 5D322, qu'il admet un et un seul paramètre (11), en l'occurence une chaîne ($). Voici les symboles que vous pouvez rencontrer et leur signification :

$	chaîne de caractères
($)	tableau alphabétique
#	valeur numérique
(#)	tableau numérique
A	alphanumérique : $ ou #

Par exemple, la fonction MEMD de MMLEX de Michel Martinet (toujours lui) admet pour paramètre soit une chaîne (ex : "TAPE(2)"), soit un nombre (ex : 2). Il en est de même pour DEVADDR du module HPIL, et ainsi de suite. Le nombre d'arguments est un intervalle mini-maxi. Ainsi 23 signifie que la fonction admet au minimum 2 arguments, et au maximum 3.

Je crois avoir dit tout ce qu'il est nécéssaire de savoir pour utiliser efficacement ce programme, je vous laisse donc aux mains de votre 71.

Bonnes fêtes !

Jean-Pierre BONDU (PPC 33, SIG 4)

INTEGRALES ET COMPLEXES

Le module Math pour HP71 permet de calculer des approximations d'intégrales avec variables réelles.

Pour ceux qui ne possèdent pas le module, trois auteurs ont fourni des fichiers basic (JPC Nos 22, 23 et 24) appliquant les méthodes de Gauss et Romberg.

Mais une lacune existe pour le champ complexe. INTEGRAL (module Math) est inopérant pour les variables complexes, comme DEF FNF d'ailleurs.

Aussi, voici une solution en basic : l'algorithme est celui de Romberg (voir JPC No 24). Il n'a pas l'avantage de la rapidité, mais la précision étant optionnelle, ceci assure en général un meilleur contrôle et les résultats incompatibles avec la précision demandée sont signalés.

Le module Math est nécessaire pour les calculs de nombres complexes.

Donc, soit à déterminer la valeur de :

$$I = \int_{a1+ib1}^{a2+ib2} F(Z)dZ$$

Exemples :

1) F(Z) = SQR(iZ)/(Z+1/Z)
pour z0 = a1 + ib1 = 1
et z1 = a2 + ib2 = i

```
RUN
      Affichage :          Input :

 -> Intégrale complexe
 -> Programmation (O/N)     O
 -> <- ligne 16 : F = f(Z)  16 F = SQR(Z)/(Z+1/Z)

RUN
 -> Intégrale ....
 -> Program....             N
 -> z0 = ( , )              (1,0)
 -> z1 = ( , )              (0,1)
 -> Précision (1 à 8) ?     7
 -> Je calcule....
 -> I = ( -0.5857864 , 0.5328400 )
```

Valeur exacte : SQR(2)-2 + i[SQR(2)-LOG(1+SQR(2))]
= -0.58478643763 + i0.532839975352

2) soit F(Z) = Z^2 / (2*Z - 1) à intégrer le long du cercle :
X^2 + Y^2 -X = 0
parcouru dans le sens positif

Ce cercle coupe l'axe des x en 0 et 1. Avec la précision de 10^(-5) on obtient pour les 4 intervalles d'intégration :

```
(0,0) à (0.5,0.5)     -> I = ( 0.125 , -0.05365 )
(0.5,-0.5) à (1,0)    -> I = ( 0.375 , 0.44635 )
(0,1) à (0.5,0.5)     -> I = ( -0.375 , 0.44635 )
(0.5,0.5) à (0,0)     -> I = ( -0.125 , -0.05365 )
d'où                     I = i 0.7854
valeur exacte     i PI/4 = i 0.785398163398
                    delta = 1.8366602 E-6
```

3) Si des intégrales avec variables réelles présentent des problèmes de lenteur de convergence, une tranposition dans le champ complexe peut permettre de résoudre le problème.

Exemple tiré du manuel des fonctions mathématiques HP-15C :

$$I1 = \int_{1}^{\inf} COS(X)/(X+1/X)\ dX$$

$$I2 = \int_{1}^{\inf} SIN(X)/(X+1/X)\ dX$$

Dans le champ complexe on a :

$$I = I1 + i\ I2 = \int_{1}^{1+i\ INF} e^{iz} / (Z + 1/Z)\ dZ$$

Donc F = EXP((0,1)*Z)/(Z+1/Z)

Sous cette forme, la convergence est rapide, ce qui n'est pas le cas de I1 et I2 pour lesquelles la méthode de Romberg est interminable et la méthode de Gauss donne des résultats aberrants. Avec I, l'intervalle d'intégration peut être réduit considérablement.

```
Intervalles          Précision demandée : 10^(-3)
d'intégration :
(1,0) à (1,10)   ->  I = ( -0.324 , 0.382 )
                 ->  Précision assurée : 10^(-2)
(1,0) à (1,5)    ->  I = ( -0.325 , 0,381 )
(1,0) à (1,6)    ->  I = ( -0,324 , 0.382 )
Donc     I1 = -0.324  et  I2 = 0.382
```

Remarque : ce fichier basic permet de calculer des intégrales avec variables réelles il suffit d'avoir des bornes sans partie imaginaire (qui peut le plus peut le moins).

Exemple : soit à intégrer
1 / X^2 dX de 1 à 4
F = 1/Z^2 z0 = (1,0) z1 = (4,0)
Précision : 10^(-4)
-> I = (0.7500 , 0.0000)
donc I = 3/4

Il ne restera plus qu'un LEX pour remplacer cela. Un des SIGouilleurs du peloton de tête s'en chargera peut-être. En attendant, soyez patients, et intégrez réellement ou complexement...

Guy TOUBLANC (P276)

SURVOL DES FICHIERS DE DONNEES

Les quelques 500 pages de lecture qui accompagnent la machine ne sont pas très explicites sur le sujet. Au début (et même après, si, si) on perd beaucoup de temps, on se décourage, pour comprendre l'usage et le fonctionnement des fichiers de données. Comme toutes les machines de poche ne possèdent pas ces facilités, il est valable de les connaitre.

Tout d'abord quelques définitions :

- fichier de données type séquentiel. Les éléments n'y sont accessibles que les uns après les autres, dans l'ordre. Exemple : pour lire le 25ème enregistrement, il faut d'abord lire les 24 premiers ; pour écrire au 17ème enregistrement, il faut faire les 16 premières écritures. Ce n'est guère souple.

Quels fichiers ? Les fichiers TEXT par excellence, les fichiers DATA parfois. Ici il faut introduire la notion de pointeur. Le pointeur est une variable de référence pour la machine, il indique à quel endroit se fera la prochaine lecture/écriture. L'utilisateur n'y a pas accès.

- fichier de données de type relatif. Les enregistrements sont accessibles au gré de l'utilisateur. Ecrire au 17ème, puis lire le 23ème, lire le 25ème et décider d'écrire au premier.

Quels fichiers ? Les SDATA et les DATA aussi. Le pointeur est modifiable par l'utilisateur, puisqu'on peut lire et écrire en n'importe quel endroit du fichier.

Il faut savoir qu'un fichier relatif peut être traité en séquentiel, puisqu'on gère écritures et lectures. Tous les types de fichiers de donnés du 71B sont de type relatif. C'est pourquoi, on peut dire que les fichiers DATA ou TEXT sont aussi des séquentiels.

C'est vu ? A présent, à quoi servent les fichiers et leurs accès ? Pour la création, je vous renvoie au manuel de référence (page 60). C'est une partie assez bien expliquée.

LES FICHIERS TEXT

Par exemple, le support de cet article est un fichier TEXT (avant impression, bien sûr). Destiné à stocker des chaînes de caractères, on peut aussi y ranger des variables numériques, des tableaux.

A savoir : si vous stockez une variable numérique, par la suite, vous pouvez lire l'enregistrement comme une chaîne alphanumérique ou un nombre. Si vous stockez une chaîne, vous ne pouvez la relire que sous forme alphanumérique.

On peut se servir de fichiers TEXT pour ranger le mode d'emploi d'un programme, lequel sera affiché à l'exécution : gain de place. Résumer toutes les utilisations possibles, c'est dire que les TEXT sont des fichiers fourre-tout.

Pour lire, l'ordre est READ #c, n; v
où c=canal, n=numéro de l'enregistrement et v=variable ou une constante.

Autre possibilité : RESTORE #c,n @ READ #c;v. Pour écrire l'ordre est RESTORE#c,n @ PRINT#c;v et non pas PRINT#c,n;v, ordre qui entraine l'erreur No 63. Regarder le manuel de référence page 250 ; en fait, RESTORE# positionne le pointeur, sur un numéro d'enregistrement.

Lors de la création, il est possible de choisir la taille du fichier en octets, mais non pas la taille de ses enregistrements ni le nombre.

Usage séquentiel : pour écrire RESTORE#c puis faire les écritures par PRINT#c;v. Pour lire, RESTORE#c puis faire les lectures par READ#c;v.

LES FICHIERS DATA

Il sont plus destinés au stockage d'objets de même type. Voila une raison pour laquelle il est possible, à la création, de choisir la taille de chaque enregistrement. Remarquez que le fichier DATA est propice au stockage pèle-mèle de chaînes et de nombres. La différence avec le fichier de type TEXT ? La souplesse. Au clavier ou en programme, ASSIGN#1 TO TOTO crée le fichier TOTO et l'ouvre. Puissant non ? Le seul défaut : pas de fonction testant la fin de fichier.

Que fait-on d'un fichier DATA ? Stocker les données nécessaires à la bonne marche du programme ; ranger les 2800 chiffres de SQR(2) calculés. Plus astucieux, le transfert du contenu du tableau A$() vers B$() : écrire le contenu de A$() dans un fichier et relire l'enregistrement en direction de B$() (A$ et B$ ont même dimension). C'est très rapide.

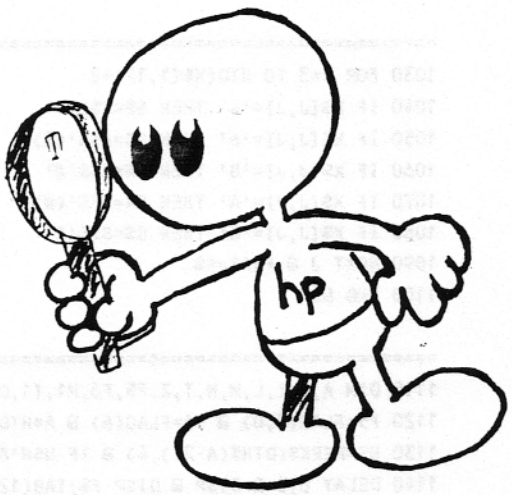
A savoir : les fichiers DATA sont cycliques. Vous avez créé TOTO, CREATE DATA TOTO,25. 25 enregistrements, vous les avez remplis. Où se fera le 26ème ? Non, non, il n'y aura pas de message d'erreur, le pointeur repasse à zéro. Le 26ème enregistrement écrasera le contenu du premier.

Se méfier : Lorsqu'à la création, on donne le nombre d'enregistrements, il est préférable d'indiquer la taille de ces derniers. Sinon, par défaut, la machine prend la taille maximale (255 octets). Les fichiers DATA sont parfaits pour la constitution de base de données.

Pour lire, l'ordre est : READ#c,n;v ou bien RESTORE#c,n @ READ#c;v.

Pour écrire, l'ordre est : PRINT#c,n;v ou encore RESTORE#c,n @ PRINT#c;v.

En usage séquentiel, pour écrire RESTORE#c puis faire les écritures avec PRINT#c;v. Pour lire, RESTORE#c puis faire les lectures avec READ#c;v.



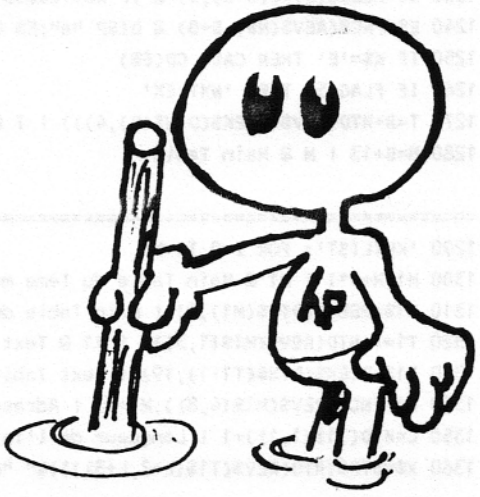
LES FICHIERS SDATA

Grosso modo, c'est un fichier DATA dont on a limité la taille des enregistrements à 8 octets. Il ne peut servir qu'au stockage de données numériques. Le manuel explique l'existence du SDATA par le dialogue avec la HP-41 et celle du TEXT par le dialogue avec le HP-75.

A propos, au clavier essayez CREATE LIF1 TOTO.

Voila un rapide tour d'horizon du fichier de données. Tout connaître dans les moindres détails nécessite de se plonger dans la documentation et les exemples de programmes parus ; effectuer des "essais sur le terrain" est tout aussi instructif.

Xavier Bille



Programme "SCANALEX" (Exploration d'un fichier LEX; necessite DESAL)

```
   - HTA$,REV$
  20 INPUT "Lex File ";F$ @ CALL SCAN(F$)

================================================================
1000 SUB SCAN(F$) @ ON ERROR GOTO 1540 @ POKE '2F441','F'

================================================================
1010 DEF FNC$(C$,B)=DTH$(MOD(HTD(C$)+B,16^5))

================================================================
1020 DEF FNA$(X$) @ S$="" ! '()' signifie 'tableau'

================================================================
1030 FOR J=3 TO HTD(X$[1,1])+2
1040 IF X$[J,J]='4' THEN S$=S$&'$'
1050 IF X$[J,J]='6' THEN S$=S$&'($)'
1060 IF X$[J,J]='8' THEN S$=S$&'#'
1070 IF X$[J,J]='A' THEN S$=S$&'(#)'
1080 IF X$[J,J]='C' THEN S$=S$&'A'
1090 NEXT J @ FNA$=S$
1100 END DEF

================================================================
1110 DIM A,B,I,L,M,N,T,Z,F5,F6,M1,T1,D$,E$,N$,P$,M1$,T1$
1120 F5=FLAG(5,0) @ F6=FLAG(6) @ A=HTD(ADDR$(F$))+37 @ B=A @ OFF ERROR
1130 N$=PEEK$(DTH$(A-21),4) @ IF N$#'802E' AND N$#'FF00' THEN DISP MSG$(63) @ END
1140 DELAY 0,0 @ DISP @ DISP F$;TAB(12);"LEX#: ";

================================================================
1150 'CNTLEX': DISP REV$(PEEK$(DTH$(B),2));" ";
1160 B=B+6 @ Z=HTD(REV$(PEEK$(DTH$(B),5)))
1170 IF Z THEN B=B+Z @ GOTO 'CNTLEX' ELSE DISP @ WAIT .5

================================================================
1180 'MAINLOOP': DISP @ DISP REV$(PEEK$(DTH$(A),2));" -"; ! A @ Lex ID
1190 N$=REV$(PEEK$(DTH$(A+2),4)) @ N=HTD(N$[1,2])-HTD(N$[3])
1200 IF N$="0000" THEN DISP " pas de mot clef" @ SFLAG 5 ELSE DISP N+1;"mot(s) clef" @ CFLAG 5
1210 WAIT .5 @ B=A+12 @ IF PEEK$(DTH$(A+11),1)='0' THEN B=B+79 @ DISP "Speed Table"
1220 DISP "Poll ..... ";
1230 N$=PEEK$(DTH$(B+8),5) @ IF N$='00000' THEN DISP 'NO' @ GOTO 1260
1240 E$=FNC$(REV$(N$),B+8) @ DISP "e";E$ @ K$=KEYWAIT$
1250 IF K$='E' THEN CALL CD(E$)
1260 IF FLAG(5) THEN 'NXTLEX'
1270 T=B+HTD(REV$(PEEK$(DTH$(B),4))) ! T @ Text Table
1280 M=B+13 ! M @ Main Table

================================================================
1290 'KEYLIST': FOR I=0 TO N
1300 M1=M+9*I ! M1 @ Main Table du Ième mot
1310 M1$=PEEK$(DTH$(M1),9) ! Main Table du mot
1320 T1=T+HTD(REV$(M1$[1,3])) ! T1 @ Text Table du mot
1330 T1$=PEEK$(DTH$(T1-1),19) ! Text Table du mot
1340 E$=FNC$(REV$(M1$[4,8]),M1+3) ! Adresse du code executable
1350 L=HTD(T1$[1,1])+1 ! Longueur de l'intitulé
1360 X$=STR$(HTD(REV$(T1$[L+2,L+3])))&" "&HTA$(T1$[2,L+1])
```

```
1370 N$=REV$(PEEK$(DTH$(HTD(E$)-17),17)) @ CFLAG 5,6 @ LC OFF
1380 P$=FNC$(N$[1,5],HTD(E$)-5) @ D$=FNC$(N$[6,10],HTD(E$)-10)
1390 DISP X$;TAB(13);"[";M1$[9];"]";TAB(17);"e";E$;" ";
1400 ON POS('58CDP',M1$[9])+1 GOTO 1430,'C5','CD','CD','CD','CF'
1410 POKE "2F443","0" @ POKE "2F441","0" @ DISP @ GOTO 1340

================================================================
1420 'CF': DISP TAB(26);REV$(N$[1,2]);TAB(31);FNA$(N$);
   - N$[1,2]= Min Max d'arguments
1430 DISP @ GOTO 'S'

================================================================
1440 'C5': SFLAG 5 @ DISP "p";P$ @ GOTO 'S'

================================================================
1450 'CD': SFLAG 5,6 @ DISP "p";P$;" d";D$

================================================================
1460 'S': SCROLL 1 @ K$=KEY$
1470 IF K$='P' AND FLAG(5) THEN CALL CD(P$) @ GOTO 1410
1480 IF K$='D' AND FLAG(6) THEN CALL CD(D$) @ GOTO 1410
1490 IF K$='E' THEN CALL CD(E$) @ GOTO 1410
1500 IF K$='#43' THEN 1540
1510 NEXT I

================================================================
1520 'NXTLEX': A=A+6 @ Z=HTD(REV$(PEEK$(DTH$(A),5)))
1530 IF Z THEN A=A+Z @ GOTO 'MAINLOOP'
1540 POKE '2F94A','0' @ I=FLAG(5,F5)+FLAG(6,F6) @ POKE '2F441','0'
1550 END SUB

================================================================
2000 SUB CD(A$)
2010 DISP @ CALL DESAS(A$) @ POKE '2F441','0' @ POKE '2F443','0' @ DISP @ END SUB
```

****

Progamme "INTEG" (Calcul d'integrales complexes par la methode de Romberg)

```
  1 DISP "Intégrale complexe" @ INPUT "Programmation (O/N) ";R$ @ IF R$="N" THEN 3
  2 DISP "<- ligne 16 : F=F(Z)" @ END
  3 DESTROY ALL @ OPTION BASE 0 @ COMPLEX A,B,F,F1,Z,X,H,T(9,9) @ STD
  4 INPUT "z0 = ","( , )";A @ INPUT "z1 = ","( , )";B
  5 INPUT "Précision (1 à 8) ? ";D @ E=10^(-D) @ DISP "Je calcule..."
  6 N=1 @ Z=A @ GOSUB 16 @ F1=F @ Z=B @ GOSUB 16 @ F1=(F+F1)/2
  7 T(0,0)=(B-A)*F1
  8 H=(B-A)/2^N @ Z=A @ T(0,N)=F1 @ FOR I=1 TO 2^N-1 @ Z=Z+H
  9 GOSUB 16 @ T(0,N)=T(0,N)+F @ NEXT I @ T(0,N)=T(0,N)*H @ FOR P=1 TO N @ K=N-P
 10 T(P,K)=(T(P-1,K+1)*4^P-T(P-1,K))/(4^P-1) @ NEXT P
 11 Z1=MAX(ABS(REPT(T(N,0)-T(N-1,0))),ABS(IMPT(T(N,0)-T(N-1,0))))
 12 IF Z1<E THEN 14 ELSE IF N<5 THEN N=N+1 @ GOTO 8
 13 BEEP @ DISP "Precision <= 1E";1+EXPONENT(Z1) @ WAIT 2
 14 BEEP @ FIX D @ DISP "I = (";REPT(T(N,0));",";IMPT(T(N,0));")" @ STD @ END
 16 F=EXP((0,1)*Z)/(Z+1/Z)
 20 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 trouverez, outre les Lex de la rubrique "assembleur", le Lex CHARLEX nécessaire à la rédaction de votre article. Voir l'article "Ah ! Vous écrivez !".

P. David (PC37 SIG1 CHHU616)

LOWC	LOWC$	92028
VECTLEX	VSUM	92100
	VMULT	92101
	VCOPY	92102
	VSORT	92103
	VADD	92104

DESAL contient HTA$ et REV$ pour le programme SCANALEX

CHARLEX

```
 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
```

```
LOWCLEX  ID#5C   74 octets

     0123456789ABCDEF sm

000: C4F47534C4548502 383
001: 802E002291720168 6D8
002: 89000C5C1C100000 A39
003: F710000000000000 D67
004: 091000F9C4F47534 0E4
005: 42C11FF411136068 459
006: F83DB0137C213510 7DA
007: 9AD1D881D63201C1 B72
008: 14B3314A58FC70B1 F05
009: 4C03102A6A149CD5 294
00A: DD071348F064A184 621
00B: 08D7B181F        82F

VECTLEX  ID#5C  1115 octets

     0123456789ABCDEF sm

000: 65543445C4548502 363
001: 802E002291720168 6B8
002: AB800C5468600000 A23
003: FB30000002700000 D67
004: 0A9400FB00C5500D 0EE
005: 810B7500D5208F50 463
006: 0D23091500D76535 7C9
007: 55D446965D455C44 B56
008: 55696534F4059566 ECA
009: 96535F4254576765 23F
00A: 144444861FF96950 5BC
00B: 5C211B1351121C91 925
00C: 378B68113510B393 C93
00D: 4A345650215D9008 FFC
00E: D499C08D534B18DE 3B5
00F: 14B18DB14B18D004 743
010: D08DD55918DE4591 AEF
011: 8D91FB08D6CC718D EB9
012: C35918DEE0B18DC5 27B
013: 1B18D84A808D5EC2 62B
014: 08DFB6C08FBC6315 9EA
015: 0C14017F164018FB D6C
016: F1908F681F01361B 0FC
017: BB8F214419B978CF 4B6
018: 74CF70CF729F1537 862
019: 30A98218AF681684 BEA
01A: 980D089250859208 F57
01B: 1684890A50858F61 2D3
01C: 89869B0D286840E6 668
01D: 8401428B605868B0 9D8
01E: 8A864CECCE614017 D89
01F: 215D21C271FEF616 123
020: 41428B622868C08A 4A4
021: 881CC140870B1100 813
022: 8501895AD2031C18 B8E
023: D39390118EA4DE14 F2C
024: 08488FC07417ABE0 2CC
025: 417623A92A1E2015 63E
026: D21BBA8F21C61577 9E8
027: 154718F15671088F D62
028: 234B1DA7B5ED7755 112
029: ED6109DB6C10118C 4B8
02A: E400108119E6109D 82E
02B: A7F3E04D7D21BEA8 BFA
02C: F22015E21827D70D F82
02D: BC2D57C4016A146C 326
02E: 217A14511878FD14 6A3
02F: 77FFD1088477A0E8 A52
030: FCA84186560632F1 DE4
031: 7F780E200411875C 164
032: DDA011520D230B90 4EA
033: 2F09821131016D00 846
034: 3096600306DC8D94 BC7
035: 3B1788D04AF51187 F59
036: 38D1347F9D14B203 2E6
037: 0B90221982B1AF41 668
038: 5076B10AF48FC69F A1E
039: 06D00AF48FB99F00 DCB
03A: 420011577AF41517 12F
03B: 1CF1557736D7DDE4 4E7
03C: 0052E7A3031B25A0 85C
03D: 7F2031A28F8EC207 BF9
03E: 0205817C10313FAE F77
03F: A8F291507A008D30 2FE
040: 3501713182845 8D8 662
041: 715071608FB39403 9D0
042: 13F966E07E0D7E90 D79
043: 68408D271307F803 0EC
044: 18272707FEC78703 475
045: 19272608D0603079 7D5
046: DF31B26B007DCF31 B87
047: A273401717150318 EDA
048: 274301718FE05308 246
049: FDC6303120067720 5B8
04A: 07A6E4F006311F77 944
04B: 0059E31929664003 CA2
04C: 8DB2E208DD96308D 057
04D: 46030888A447C8C0 3DA
04E: 5AF4795C7F0C047A 78E
04F: AD432AFC05734C8F B4A
050: C14D08F363C07BEB F03
051: 046BDF1BBB8F2146 2BB
052: 134058FE04D07FAB 663
053: 8D832F01618F871F 9FE
054: 08FC1DB0041F178F DA7
055: 21517161640C0477 0FE
056: ABAFC1B178F21567 4B9
057: 050115EFF5EEFF75 87D
058: BF76DF8FF53C072C C57
059: D770D5CE6A9B43EF 02E
05A: F9CEFF7D8F7EAF8F 44B
05B: 234C07A9D7FDC5CE 81D
05C: 627B22EFFB4EFF7B C03
05D: 8B7FCD17F714B1BB FD2
05E: B8F2146134161737 340
05F: B1FE95F214710A77 6E0
060: 0B11011A1371CF1C A60
061: F1CF1338A6901F10 DF7
062: 000133100DA17F17 15A
063: F15771331027A1D7 4D2
064: 95C5EB6CEAEADFFA 8E5
065: 2EFF750B794D1BB9 CAC
066: 8F21468AADD195A1 05A
067: 46D7189146D51CF1 3F0
068: 331F495F2147E213 76E
069: 14908DD4490AFB15 B0C
06A: 571741451C4E6D7C E9F
06B: D5AC766A146E6E6F 267
06C: 210B1CF133C21081 5DC
06D: FE95F21471351CF1 97D
06E: 371351108A6606E8 CEA
06F: 0153710210913418 028
070: F1321301188BA331 394
071: 1215672682282105 6DE
072: 8F674D00444D1361 A5C
073: 34109156710A61CF DD2
074: 1191341577154711 11C
075: 21517113136E2134 46A
076: 137D5E2135142147 7D1
077: 144141D9606F7289 B49
078: 137D5113E21331CF ECA
079: 133DC7C891CF7E79 289
07A: 1BEA8F2D015A3D81 63D
07B: 9BA7E6BD87C49133 9F7
07C: E01F495F2147E246 D8A
07D: 062CED477291CA14 125
07E: 77109D610B1C4137 49D
07F: 13506157777F8109 806
080: 11B7DE8DAD276AA1 BCD
081: 1BD571F81CF133E0 F75
082: 137750917F17F070 2E8
083: 613610A133131113 63A
084: 70B8D51197998D71 9CC
085: 641468A36118FE51 D4E
086: 361348B2AE64C911 0D9
087: 9144D4D2793A113D 466
088: 8112E072A811BD57 7ED
089: 6781CF1371327588 B69
08A: 756817F133071351 EC9
08B: CF1378A2B1061351 24B
08C: 1B7E28E67A1810B6 5E9
08D: 52F8D84A80F      878

DESAL    ID#E1   407 octets

     0123456789ABCDEF sm

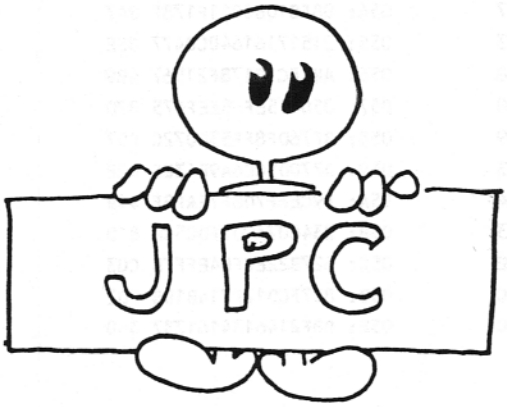
000: 44543514C4020202 34B
001: 802E004291720168 6A2
002: 233001E10506E200 9F3
003: FB30000000000000 D2E
004: 012100FF0019100F 08F
005: A10BD100F5200500 3FA
```

```
006: 0F03025200FB1444 761
007: 2455644210714353 AA9
008: 4422071445844230 DEF
009: 7844514424072554 141
00A: 4442501FF4118F83 4C0
00B: DB0D60A871A51360 851
00C: 6137C2134135AD2D BD3
00D: 681E81ED7C6BF2BF FAB
00E: 2A0E109CF4811831 330
00F: 5A32172201C11495 68F
010: 7E071341CF1198D8 A23
011: 32F08D91FB080CF2 DCF
012: 08F670B15F18FAA2 17B
013: 513314648FC70B14 4F5
014: 4D309A0ABF1A88BF 8BD
015: 4BF4A4E59CAF4034 C7A
016: 118F13DB0D23068A 00C
017: 63A15B5175137062 373
018: 2769FD68F1C81158 714
019: 11C613324AC280FF AA6
01A: 8F84171071355901 E08
01B: C915991098508F06 187
01C: 4A18F7B1818FE83B 53E
01D: 18DC32F041113713 8AE
01E: 5068F83DB0AD1D88 C62
01F: 1D31E20A3302E7D7 FF2
020: CD4C114BDB8FC70B 3C8
021: 15700914D17153E0 72D
022: 71358DC32F04118F ABB
023: 83DB013606137135 E20
024: 1348F064A11CF137 1A7
025: E28B37413506D6C6 538
026: BF2BF230F155717F 8E5
027: AF2D681ED7A4ECF4 CC8
028: 3214ABCA8F841711 05D
029: 59316117353E8DD4 3E2
02A: 490071350713480C 74D
02B: 32F04117B007D207 ABB
02C: 3004248F13DB0AD1 E46
02D: D881D3102CD48214 1CF
02E: B966001715FED4E4 570
02F: C4BF0BF0A0C1CF15 93B
030: 17010772EF8DC32F CD9
031: 025C0C000000F710 034
032: 0000000000000710 34C
033: 00F725546542C01F 6C4
034: F4118FE83B18DC32 A75
035: F0F              B34

CHARLEX  ID#E1   624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E005291720168 6B6
002: 5E4001E000000000 9FA
003: FE0000000800001F D54
004: F31BF961400032BF 0E7
005: 38F14A11DB10AD23 481
006: 07D532BFB8FD7911 834
007: 11AD754D7A101743 BB7
008: 11014D1CB15D0000 F22
009: 71450375FF864834 29F
00A: 5655581008355654 5F6
00B: 5810002455565870 943
00C: 0026555658700836 C95
00D: 5556581008364545 FEB
00E: 4A30000A49724000 33E
00F: 0808094A2C180814 6A7
010: A464242008355455 A01
011: 581000054C714000 D47
012: 0C3142404C700832 0A3
013: 41414A70002078A0 3FB
014: 2F30000000000000 726
015: 0000000000000000 A36
016: 0000000000000000 D46
017: 0000000000000000 056
018: 0000000000000000 366
019: 0000000000000000 676
01A: 0000000000000000 986
01B: 0000000000000000 C96
01C: 0000000000000000 FA6
01D: 0000000000000000 2B6
01E: 0000000000000000 5C6
01F: 0000000000000000 8D6
020: 0000000000000000 BE6
021: 000000000000080C F11
022: 1A280800080080A2C 27B
023: 180008040E340800 5C4
024: 08001E3018000000 8FE
025: 0000000000000000 C0E
026: 0000000000000000 F1E
026: 0000000000000000 F1E
027: 0000000000000000 22E
028: 0201000000010200 544
029: 0000000201020000 859
02A: 0001000100000002 B6D
02B: 0102010000000000 E81
02C: 0000000000000000 191
02D: 045E755142400101 4DD
02E: 0101010000000000 7F0
02F: 0000000000000000 B00
030: 0000070507000000 E23
031: 00000000083444C4 161
032: 44400D7901112D70 4C1
033: 050D750509700000 80B
034: 0D70000000384540 B4E
035: 4020014E322E3140 EA2
036: 084E794142400000 1F2
037: 000000000002E4559 530
038: 3200000000000000 845
039: 0000000000000026 B5D
03A: 5556587008365556 EBC
03B: 5810083645464830 20D
03C: 0832414248700024 54E
03D: 5655587008345655 8AB
03E: 5810083446454830 BFA
03F: 0C3042414C700024 F4F
040: 5556587008355654 2AC
041: 5810083546444830 5FB
042: 0C3142404C700025 951
043: 5455587008355455 CAB
044: 5810083544454830 FF9
045: 0C3140414C700875 35B
046: 14141870000A4972 6AC
047: 40000E3159454E30 A0C
048: 0C7A0F7949400024 D84
049: 5554587000084A71 0E0
04A: 40000C523A262D10 441
04B: 0424587458400875 798
04C: 1415187000094A70 AE8
04D: 4000083544454830 E2C
04E: 0C3140414C300C74 194
04F: 5655545000054C71 4EB
050: 40000            5E4
```



PPC PARIS CHAPTER

Novembre 1986

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

BULLETIN D'ADHESION

NOM |_|_|_|

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

La maquette de ce numéro a été préparée et réalisée par Pierre David, Janick Taillandier et Philippe Guez, 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.