JPC

SEPTEMBRE 1987
NUMERO 47

Le numéro 40 F

A PROPOS DU CLUB

P. David	Editorial	*1*
	PPC Paris se réunit	*2*
	Ah ! Vous écrivez !	*2*
	La conférence de Copenhague	*3*
	Choc en retour	*4*
	Courrier des lecteurs	*4*
	Courrier du coeur	*6*
	Nouveaux produits	*7*

HP28

W. Mier-Jedrzejowicz	Programmation Synthétique	*10*
W. Mier-Jedrzejowicz	La fonction PEEK	*14*
H.P.	Les anomalies du HP-28C	*14*

HP41

J. Elhay	Probabilités	*18*
R. Dine	Les nombres premiers	*19*

HP71

X. Bille	Tracé de courbes (acte II)	*22*
P. David & J. Taillandier	Assembleur pour HP-71	*24*
J. Buitenhuis & F. van der Windt	Module de développement	*24*
J. Baudier	L'assembleur du HP-71 (acte III)	*26*
J. Elhay	Inversez vos drapeaux	*29*
E. Gengoux	Retrouvez vos fichiers	*30*
	Le coin des Lhex	*39*

EDITORIAL

Chers revenants (de vacances),

Ne soyez pas trop tristes, la rentrée va bien commencer puisque voici votre journal favori... Comme d'habitude, il est plein de bonnes choses, de délicieux programmes à déguster, de trucs et d'astuces en tous genres.

Ce mois-ci, il faut signaler l'article de Wlodek Mier-Jedrzejowicz, de Londres, sur le HP-28C. Pour la première fois, on peut découvrir la structure interne de ce calculateur. Nous devons un grand merci à Wlodek, et espérons que les prochaines découvertes soient votre oeuvre. A vos claviers !

Pendant le mois d'août, le Club a vécu au ralenti. Nous allons maintenant reprendre le rythme habituel. Nous nous sommes fixés comme objectif de répondre au courrier que vous nous adressez dans un délai d'une semaine. Aidez-nous, soyez clairs, et indiquez toujours votre numéro d'adhérent. Merci.

Bonne lecture,

Pierre David (37)

PPC PARIS SE REUNIT UNE FOIS PAR MOIS

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

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

Si donc cela vous intéresse, n'hésitez plus un seul instant, venez nous rejoindre tous les premiers samedis de chaque mois (sauf en période de vacances scolaires) au :
Centre de Jeunesse et de Loisirs Jean Verdier
11 rue de Lancry
75010 Paris
et en montant au deuxième étage, vous entendrez des éclats de rire et des discussions passionnées vers la salle 215. Attention, toutefois, de venir entre 16 et 19h.

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

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

Les dates des prochaines réunions sont :
Samedi 3 octobre 1987
Samedi 21 novembre 1987
Samedi 5 décembre 1987
Samedi 16 janvier 1988
Samedi 20 février 1988
Samedi 5 mars 1988
Samedi 16 avril 1988
Samedi 7 mai 1988
Samedi 4 juin 1988

Pierre David (37)

AH ! VOUS ECRIVEZ

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

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

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

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

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

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

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

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

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

Jean-Jacques Dhénin (177)

```
DEF KEY 'fW', CHR$(197);      (é)
DEF KEY 'fE', CHR$(193);      (ê)
DEF KEY 'fR', CHR$(201);      (è)
DEF KEY 'fY', CHR$(203);      (ù)
DEF KEY 'fU', CHR$(195);      (û)
DEF KEY 'fI', CHR$(209);      (î)
DEF KEY 'fO', CHR$(194);      (ô)
DEF KEY 'f/', CHR$(92);       (\)
DEF KEY 'fA', CHR$(192);      (â)
DEF KEY 'fS', CHR$(200);      (à)
```

```
DEF KEY 'fD', CHR$(205);      (ë)
DEF KEY 'fJ', CHR$(207);      (ü)
DEF KEY 'fK', CHR$(221);      (ï)
DEF KEY 'f*', CHR$(124);      (|)
DEF KEY 'fC', CHR$(181);      (ç)
```

LA CONFERENCE DE COPENHAGUE

Les 8 et 9 août 1987 se tenait à Copenhague la conférence internationale des utilisateurs de matériel HP, organisée par PPC Danemark.

Cette conférence, je dois le dire tout de suite, était extrêmement bien organisée. Flemming Madsen et Steen Petersen, pour ne citer qu'eux, ont accompli un travail remarquable.

La conférence a eû un large succès, puisque pas moins de 11 pays étaient représentés. Je ne connais pas la fréquentation exacte, mais il y avait au moins 50 personnes, vraissemblablement plus. Nombre de sociétés fabriquant ou commercialisant des produits pour nos calculateurs étaient présentes. Citons, par exemple, David Lin, le président de CMT, venu spécialement des Etats-Unis. Notons que toute la conférence se déroulait en anglais.

A 9 heures, le samedi 8 août au matin, après nous avoir distribué des documents papiers et des badges, la conférence a débuté par une présentation de la stratégie commerciale de Hewlett-Packard, par Kermit Yensen, ainsi que par une description des orientations pour les nouveaux produits par Michel Maupoux. Tous deux venaient spécialement de l'usine HP de Grenoble. Pas d'annonce fracassante, pas de nouvelle spectaculaire, mais une confirmation des volontés d'HP sur le marché : plus large diffusion de ses machines en grignotant sur le bas de gamme. Peut-être s'agit-il des prémisses de l'annonce d'une nouvelle machine de moins haut de gamme que le HP-28C ? D'autre part, l'architecture matérielle et logicielle des HP-28C et HP-18C reservira : elle va devenir l'architecture unifiée des calculatrices de demain...

Après une pause, David Lin, de Corvallis Micro Technology, nous a présenté « Marys II », le nouvel ordinateur compatible HP-41. En fait, il s'agit d'une machine portable, alimentée par batteries, de taille non encore connue, à base de micro-processeur 80C88. Le système d'exploitation, quant à lui, se rapproche de MS-DOS. L'affichage comprend 8 lignes de 21 caractères, ou 64 × 128 pixels en mode graphique. La mémoire peut aller jusqu'au méga-octets, et deux ports RS-232 sont fournis en standard. Là où ça devient intéressant, c'est que cette machine comprend un émulateur de HP-41, le *41M*. C'est une super HP-41, plus rapide (5 à 10 fois), avec plus de mémoire, un plus grand afficheur, et des caractéristiques supplémentaires. Espérons que nous reparlerons bientôt de ce petit bijoux !

Ensuite, VM Electronics, une firme danoise, annonçait un nouveau module pour le HP-41 : le module HEPAX. C'est un module apportant 8 ou 16 Ko de mémoire additionnelle, s'utilisant comme le module XFonctions. Une différence majeure, cependant : les programmes peuvent être exécutés directement dans la mémoire du module, sans avoir besoin de les recopier en mémoire principale au préalable. En plus, une mémoire morte de 16 K mots contient des fonctions de gestion de cette mémoire, aussi bien que des utilitaires généraux tels que désassemblage, édition de la mémoire en hexadécimal, etc. Le prix est de 260$ pour le module 8 Ko, et 295$ pour le 16 Ko.

Le dernier orateur avant le déjeuner était la firme anglaise Zengrange, qui présentait le module ZepROM. C'est un module Eprom pour le HP-41, comparable à celui de CMT. La contenance est de 8, 12 ou 16 Kmots, paginés si besoin est.

Après le repas, typiquement danois, W&W software, la firme allemande, nous a parlé de son module 64 K ram pour la HP-41, tenant dans un boitier de lecteur de cartes.

La parole a été ensuite donnée aux utilisateurs. Stefano Tendon nous a parlé de *Turbo Assembler 71*. Il s'agit d'un assembleur écrit en Forth et en assembleur, beaucoup plus rapide, permettant d'écrire beaucoup plus facilement des primitives Forth. En fait, il ne s'agit pas d'un assembleur, mais plutôt d'un nouveau langage. Les opérations de base sont toujours les opérations du micro-processeur, mais complétées par des structures de contrôle évoluées telles que *if ... endif* ou *begin ... until*, et des *procedures*, *subroutines* et *macros*. Bref, il s'agit d'un très beau produit, et nous l'attendons avec impatience !

Ensuite, Kim Holm et Wlodek Mier-Jedrzejowicz ont présenté les résultats de leurs recherches sur la structure interne du HP-28C. Vous trouverez dans ce JPC une présentation détaillée de ces découvertes par Wlodek.

Après une pause, les conférences ont repris en deux séries parallèles. D'un coté, Wlodek nous confiait quelques règles pour mieux programmer le HP-41. Dans un autre amphithéatre, Kim Grau parlait des bases de données sur HP-75. Après, Torsten Hagemann parlait de la programmation multi-tâche sur HP-41. Simultanément, Sherman Lowell expliquait de la communication entre nos ordinateurs préférés et les machines MS-DOS.

La conférence était finie pour la journée. Il y eû alors une réunion des présidents de clubs, qui nous a permis de faire un recensement des clubs PPC représentés.

La soirée s'est ensuite achevée dans les jardins de Tivoli, à Copenhague.

Le lendemain matin, Stefano Tendon nous a parlé de son éditeur de textes pleine page pour HP-71. VE71 (c'est son nom) est vraiment très puissant, et il a décidé de le mettre dans le domaine public. C'est une excellente initiative, et nous ne pouvons que le remercier.

Ce fut ensuite à notre tour de parler. Janick Taillandier a présenté JPCLEX, son contenu, son histoire. Nous en avons profité pour montrer le manuel de plus de 200 pages, ainsi que la traduction anglaise qu'il a réalisé. Apparemment, cela a beaucoup impressionné les auditeurs.

Steen Petersen, après une pause, a présenté le micro-code du HP-41 aux débutants. Il a expliqué ce que c'est, l'intérêt, les applications, le matériel nécessaire et la manière de programmer, et enfin les erreurs fréquentes quand on développe en assembleur.

La dernière conférence était consacrée à un pauvre HP-28C, honteusement torturé par Kim Holm ! Non content d'y avoir ajouté 4 Ko de mémoire, il l'a accéléré. Non mais, vous vous rendez compte ? Quel scandale... Espérons que nous pourrons bientôt publier des schémas complets. !

Le repas, lui aussi typiquement danois, a clos les présentations. L'après-midi était consacré aux discussions particulières, aux échanges et aux réunions de petits groupes.

Ainsi s'est déroulée cette conférence internationale, la première depuis deux ans. Souhaitons que les prochaines soient moins espacées.

Une bonne nouvelle pour tous ceux qui viennent aux réunions mensuelles : les résumés des conférences peuvent être consultés.

Pierre David (37)

CHOC EN RETOUR

Michael Markov nous a signalé une «amélioration» possible dans les exemples de l'article sur FINPUT paru dans JPC 43 (page 16). Au lieu de mettre :

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

il aurait été plus habile d'utiliser l'astuce suivante :

```
121 DIM S$[82]
122 S$=""
123 S$[82]=" "
```

La ligne 121 dimensionne la variable S$ à 82 caractères, la ligne 122 l'initialise à la chaîne vide, et la ligne 123 met le 82ème caractère à blanc, ce qui a pour effet de mettre les 81 premiers à blanc également. Cette astuce prend moins de place, est plus élégante et est plus rapide.

Comme quoi faire trop d'assembleur fait quelquefois oublier les petits trucs utiles...

Pierre David (37)
Janick Taillandier (246)

COURRIER DES LECTEURS

Jack Elhay
9/494-498 Mitcham Road
Mitcham, Victoria, 3132
Australie

Chers amis de PPC Paris,

C'est en lisant JPC que l'envie d'écrire des fichiers Lex m'est venue. Avant ma venue à PPPM, le Club de Melbourne en Australie, personne ne pouvait comprendre le contenu de votre Journal. Quelques uns assemblaient vos fichiers Lex, mais devaient essayer de deviner la syntaxe.

Je suis arrivé en Australie quand j'étais petit, il y a environ 40 ans. A l'époque, je parlais le Français, appris au Lycée Français du Caire. Bien que je ne l'ai pas parlé pendant la majeure partie de ma vie, je n'ai pas tout oublié ! J'ai donc préparé des résumés des JPC que nous recevions à Melbourne, et ils sont parus dans notre Journal (*Technical Notes*) sous le titre «The French Connection». Cela me fit réapprendre le français, et cela était d'autant plus difficile que je n'y connaissais rien en assembleur.

Vos articles sur l'assembleur ont été une source constante d'inspiration. En particulier, *La décomparse* dans JPC 28 (Octobre 1985) nous a aidé considérablement pour les mots-clefs. Je voudrais partager avec vous les fruits de cette fertilisation. Aussi, je vous envoie quelques fichiers Lex et programmes pour HP-41.

J'aimerais vraiment comprendre comment manipuler des variables en assembleur. Apparemment, Jean-Jacques Dhénin le sait ! Il y en a certainement d'autres parmi vous qui le savent aussi. Le fichier Lex VECTLEX n'est rien moins que brillant. Je crois aussi que SWAP doit identifier des variables pour les échanger, si j'ai correctement deviné son fonctionnement. Maintenant, si quelqu'un parmi vous pouvait donner un peu de son temps pour écrire un petit article expliquant comment accéder aux variables, comment leur affecter une valeur, je lui serais reconnaissant.

Une autre chose : j'ai souvent besoin d'un tri dans mes programmes de calcul. J'ai besoin de trier des tableaux à une dimension, du plus petit élément jusqu'au plus grand. Voici ce que je fais actuellement :

```
20 SUB SORT(X(),Y()) @ M=UBND(X,1) @ MAT Y=X @
   FOR I=1 TO M @ FOR J=I TO M
30 ON (Y(J)>=Y(I))+1 GOTO 'A','B' @ 'A':A=Y(J) @
   Y(J)=Y(I) @ Y(I)=A @ 'B': NEXT J @ NEXT I
```

Bien que le ON...GOTO soit meilleur que IF...THEN..., ce serait encore meilleur si nous pouvions écrire :

```
MAT Y=X @ SORT Y
```

n'est-ce pas ? Et ce serait aussi plus rapide. Je demande donc à Jean-Jacques : «pourriez-vous me faire la faveur d'écrire ce mot-clef et de le documenter complètement ?».

Nous avons un bon groupe, ici à Melbourne, mais il semble que ce novice (moi) était le premier à écrire un fichier Lex original. Les autres qui étaient intéressés utilisaient soit des fichiers de CHHU, des disquettes d'échange de Michael Markov ou de JPC, soit les modifiaient. Par exemple, Graham Fraser améliora FILE? pour avoir un meilleur contrôle d'erreur, et fit la même chose que MSG$. Maintenant, ils commencent à écrire des choses originales. C'est, pour moi, très excitant, n'ayant commencé à écrire des fichiers Lex que depuis décembre 1986.

De nouveau, je vous remercie. Vous pouvez répondre en français, s'il n'y a pas d'argot ou de mot trop difficile. Ne riez pas, s'il vous plaît, j'essaye. Mais je suis sûr que si j'écris plus que ces quelques mots, je vais faire des bêtises...

Jack Elhay

NDLR : merci pour ces nouvelles. Ce témoignage nous va droit au coeur. Merci pour l'effort, et bonne continuation...

François Duret-Lamouroux
5, chemin de la Tuilerie
78240 Chambourcy

Chers amis,

Je tiens à remercier Jacques Baudier pour son article publié dans JPC du mois de juin relatif à la programmation en assembleur du HP-71.

A l'instar, j'imagine, d'autres membres du Club, je suis tout à fait novice, n'ayant un HP-71 que depuis le début de cette année et n'ayant précédemment fait qu'un peu de programmation sur HP-67 et HP-41. J'ai donc encore beaucoup à apprendre et la littérature du commerce est souvent d'un faible secours compte-tenu des particularités du HP-71, spécialement en ce qui concerne l'assembleur pour lequel je n'ai jusqu'à présent rien trouvé.

J'ai certes tiré grand profit des réunions du Club ainsi que de la lecture de JPC. Cependant, le haut niveau des « locomotives » du Club fait qu'il est parfois difficile de les suivre, même de loin !

Il me paraît donc tout à fait opportun qu'il soit publié une série d'articles donnant les bases élémentaires sans lesquelles ceux qui sont dans mon cas risquent de rester des spectateurs admirant des acrobates professionnels ou semi-professionnels ! Ce qui semble à l'opposé des objectifs d'un club où les plus avancés peuvent aider les nouveaux venus à se mettre à un niveau suffisant pour apporter ensuite à leur tour leur contribution à l'enrichissement commun.

Ceci ne doit pas être perçu comme une critique de l'action du Club qui, je le répète est tout à fait positive. Je veux seulement exprimer, à la lumière de ma courte expérience personnelle, qu'il est difficile, sans acquérir certaines bases, de faire autre chose que «pomper» le travail des autres. Or, s'il est normal qu'un nouvel arrivant se comporte en assisté, il devrait progressivement être en mesure de produire quelque chose d'utile aux autres.

Merci d'avance à ceux qui écriront de longs articles d'initiation comme Jacques Baudier a entrepris de le faire. A l'image de ce qui a été fait avec JPCLEX, ces articles pourraient être ultérieurement regroupés dans un manuel du débutant.

Bonne programmation !

François Duret-Lamouroux (329)

Dominique Marcaillou
29, rue du Moulin
31320 Castanet Tolosan

Le Corbeau de JPC

Voilà mon premier article, qui représente une sommation de tous les bogues parus dans le JPC (je vais m'attirer les foudres de la rédaction !).

Tous ces bogues se rapportent aux codes du Lex du coin des Lex :

Numéro 31 :

- ENDUPLEX : la fonction `ENDUP` provoque un message d'erreur lors de son utilisation. De plus, lors d'un passage au SCANALEX de Jean-Pierre Bondu, cette charmante fonction le plante en plein travail, dommage...

- FINDLEX : on attendait le message d'erreur `Not Found`. Cette fois-ci, on aura uniquement `Err:`.

Numéro 26 :

- REPEAT : qu'est-ce que l'on entend dans l'article de Jean-Jacques Moreau par «des conflits peuvent éventuellement avoir lieu entre le Lex REPEAT et d'autres Lex» ? Un Memory Lost ne doit pas être très loin du mot «conflit».

Quand au programme de dérivées symboliques de JPC 44, c'est toute une histoire... Quelle joie de voir un tel programme pour un élève de Terminale, mais quelle déception de savoir que l'ordinateur réinvente la notion de dérivée...

Premièrement, une erreur visible : la dérivée de SQRT(X) donne 1/.5*SQRT(X). Bizarre, bizarre ! Il me semblait que c'était 1/2*SQR(X). On regarde la ligne 910, et on remplace `.5` par `2`. On redérive des fonctions avec SRQ(X), et ça marche ! On souffle un peu... puis on dérive COS(X)^3 et boum ! Par magie, on a un résultat un peu tronqué : 3*COS(X)^2. Mais où est passé -SIN(X)*... Même procédure, même résultat pour TAN(X)^3, SIN(X)^3. Il ne faut pas oublier que la dérivée de $u(x)^n$ est $nu'(x)u(x)^{n-1}$. Espérons que ce programme n'est pas un iceberg, avec une partie de bogues visibles et d'autres...

Deuxièmement, après avoir déblatéré sur ces bogues Basic, il faut remarquer que concentrer un dérivateur en 4000 octets n'est pas chose facile, et surtout avec une rapidité à faire palir les divers *MuMath* sur Apple. Je remarquerai que faire un dérivateur est quelque chose qui ressemble à une mission impossible vues mes recherches à ce sujet. Vivement l'acte II de cet article avec la correction des bogues et un petit émulateur de HP-28C sur cette si bonne machine.

Dominique Marcaillou (315)

COURRIER DU COEUR

PPC-Paris
B.P. 604
75028 Paris Cedex 01

Vend :
Interfaces vidéo HP82163B (32 colonnes) neuves (dans leur boîte, avec documentation) : 600 F seulement.

Lecteurs de cartes magnétiques HP82400A neufs pour HP-71 (dans leur boîte, avec documentation et 5 cartes magnétiques) : 500 F seulement.

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

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

Vend :
Imprimante 80 colonnes HP82905B + papier + étiquettes : 2000 F.

Convertisseur HP82166A : 1000 F.

Patrick Biani
28, rue Francis Jammes
13730 St Victoret

Vend :
HP-71B + interface vidéo HP82163B + moniteur vidéo HP82913A + lecteur de disquettes HP9114A avec alimentation secteur (Lambda Electronic 8AV6) + interface HP-IL HP82401A : 9500 F HT (ancienneté : 3 ans et demi).
Table traçante HP7470A opt. 003 : 5000 F HT.

Vincent Herlicq
Tél : (1) 46 40 15 33

Vend :
Module Paname pour HP-41 : 600 F, HP-71 + HPIL + Ram : 4500 F, module Forth / Assembleur + debugger (+ IDS) : 800 F, lecteur de disquettes HP9114A : 3500 F, ThinkJet : 3500 F, interface vidéo 32 colonnes : 500 F.

NOUVEAUX PRODUITS

Quatre manuels d'application devraient bientôt être disponibles pour le HP-28C. Trois sont en anglais, et un est en cours de traduction. Il s'agit de :

- *Vectors and Matrices*, la traduction devrait être bientôt en vente.
- *Algebra and College Math*,
- *Calculus*,
- *Probability and Statistics*.

En outre, un petit fascicule montrant des exemples de programmation est livré depuis peu avec les machines aux Etats-Unis. Nous vous tiendrons informés s'il est traduit en français.

La firme américaine CMT (Corvallis Micro Technology) diffuse un nouveau périphérique HP-IL : il s'agit d'un *ramdisc*, c'est à dire d'une unité de mémoire accessible comme une disquette ou cassette, mais beaucoup plus rapide.

Cette mémoire est non volatile : elle garde les données même quand l'appareil est éteint. Il fonctionne sur batterie, et l'autonomie a l'air plutôt grande.

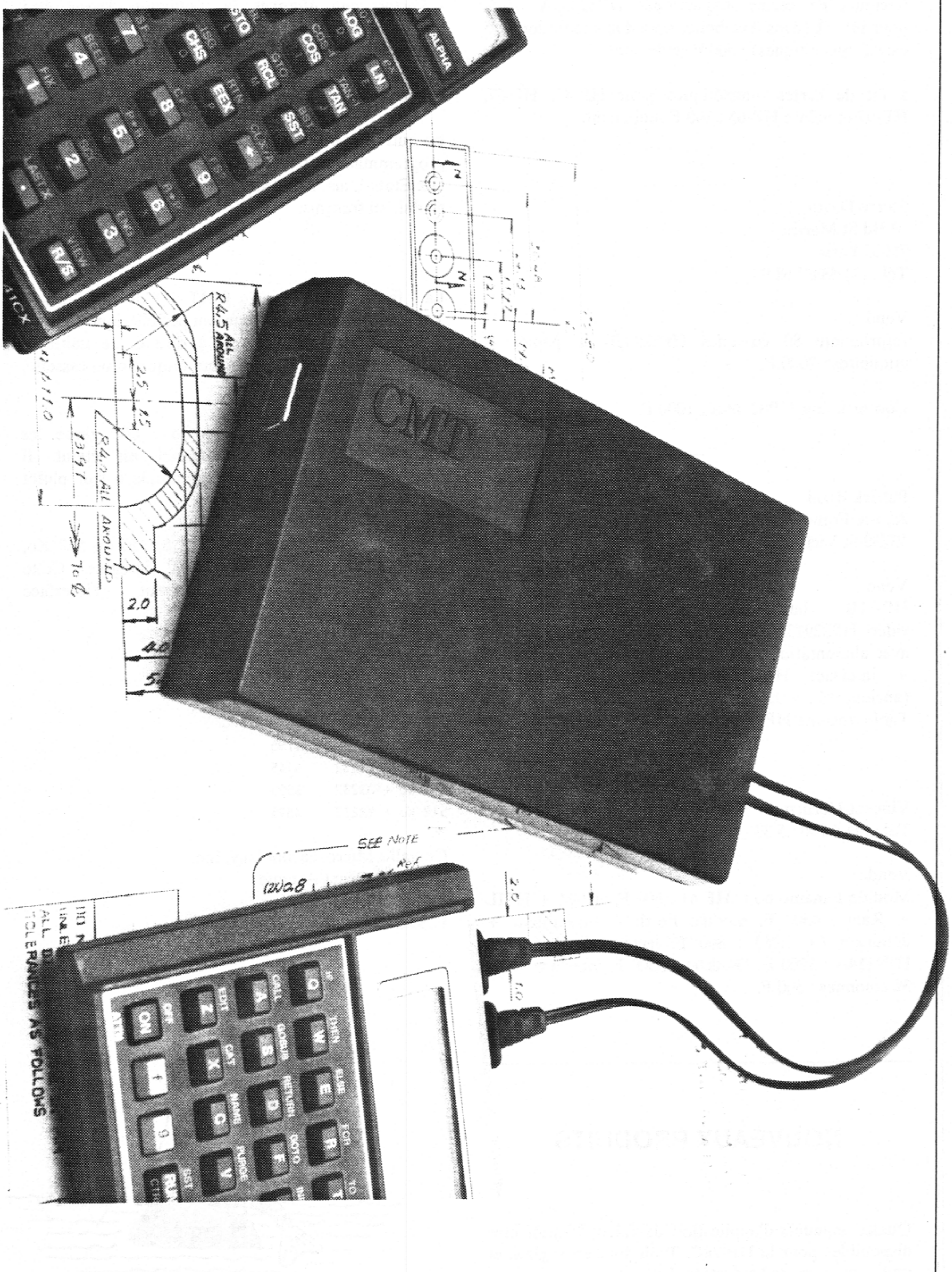
Ces unités sont disponibles en 128, 256 ou 512 Ko, avec ou sans interface RS-232 intégrée. Cette interface émule complètement l'interface HPIL/RS232 d'HP.

Les prix sont de :

128 Ko	$345
256 Ko	$495
512 Ko	$795
128 Ko + RS232	$445
256 Ko + RS232	$595
512 Ko + RS232	$895

Corvallis Micro Technology, Inc.
895 N.W. Grant Avenue
Corvallis, OR 97330
U.S.A.

Pierre David (37)


CMT

HP-28

W. Mier-Jedrzejowicz	Programmation Synthétique	*10*
W. Mier-Jedrzejowicz	La fonction PEEK	*14*
H.P.	Les anomalies du HP-28C	*14*

PROGRAMMATION SYNTHETIQUE SUR HP-28C

L'article ci-dessous nous a été envoyé par Wlodek Mier-Jedrzejowicz d'Angleterre. C'est le résultat des recherches entreprises par HPCC (le Club anglais) pour dévoiler la structure interne du HP-28C.

Cet article, qui explique en détail ces recherches, les moyens et le résultat est un excellent exemple de la coopération qui règne entre les Clubs PPC à travers le monde.

Tout ceci doit être considéré comme NOMAS. *Ce terme, inventé par Richard Nelson aux Etats-Unis, signifie NOt MAnufacturer Supported. Autrement dit, Hewlett-Packard n'a rien à voir là dedans, et ne répondra à aucune question sur le sujet.*

A vous de profiter de ces découvertes. Et pourquoi le prochain article sur le sujet ne serait-il pas de vous ?

Le jour où la HP-28C a été annoncée, j'ai envoyé quelques copies d'un article intitulé *NOMAS programming on the HP-28C*. Mon souhait était que se développe rapidement un échange d'idées et de découvertes semblable à celui survenu dans le *PPC Journal* après l'annonce du HP-41. J'ai eu la chance de pouvoir disposer d'un HP-28C à des fins d'évaluation pendant une semaine. Pendant ce temps, j'ai pu faire quelques découvertes dont je faisais part dans l'article afin d'éviter aux autres de devoir redécouvrir les mêmes choses. Malheureusement, les événements ne se sont pas passés ainsi, principalement en raison de la faible activité des clubs américains. Que pouvons nous faire alors ?

Pour commencer cet article a été publié dans *DATAFILE*, le Journal du Club anglais HPCC. Depuis, j'y ai fait paraître une rubrique régulière consacrée au HP-28C. Elle a pour but de diffuser l'information que j'obtiens et de donner à ceux qui ne possèdent pas la machine une idée sur ce qui se passe.

Le présent article se propose de vous expliquer comment analyser la structure mémoire des objets manipulés par le HP-28C.

Dans mon premier article je proposais, entre autres, qu'un groupe d'individus se partage l'espace de la Rom du HP-28C et que chacun étudie sa zone en utilisant SYSEVAL. Seul Ian Maw a répondu a cet appel et nous avons décidé de nous intéresser aux adresses hautes de la Rom. Ian a étudié les adresses #3F800 à #3FFFF et moi les adresses #3F000 à #3F7FF. Ian a très intelligement découvert que l'adresse # 261031 (décimal) fonctionne comme STO mais sans vérification particulière. Ainsi vous pouvez utiliser un objet se trouvant au niveau 1 de la pile pour en faire un nom de variable puis replacer ce nom dans la ligne de commandes en tant que chaîne de caractères et analyser son contenu grâce à NUM.

Il m'a été possible en utilisant cette technique de trouver les adresses de toutes les commandes du HP-28C. J'ai également écrit un ensemble de programmmes HP-71 afin de pouvoir trier et éditer cette liste par adresses croissantes. Ceci conduit à une découverte très intéressante : presque toutes les commandes sont séparées de la suivante par $5n+1$ quartets. Comme l'unité centrale *Saturn* utilise des adresses codées sur 5 quartets, ceci signifie que chaque commande est constituée par une liste d'appels à des routines de plus bas niveau. Ceci est très semblable à ce qui se passe en Forth et est confirmé par la remarque de Bill Wickes précisant que les commandes de la machine sont écrites dans un sur-ensemble du langage connu par l'utilisateur.

A un moment ou à un autre, le HP-28C doit atteindre un niveau où des instructions machines sont exécutées. Ceci doit être indiqué d'une manière particulière, soit par une adresse de commande qui correspond à l'exécution de langage machine, soit par un code particulier ou peut-être même les deux. Ceci pourrait expliquer pourquoi chaque commande a pour longueur $5n+1$, un quartet supplémentaire semble être stocké devant chaque commande.

Avoir découvert que le HP-28C fonctionne essentiellement grâce à des adresses de 5 quartets permet d'aller plus loin. Par exemple, beaucoup d'objets (sinon tous) sur la pile sont en fait représentés sous forme d'adresses. Ainsi DUP, OVER et d'autres commandes ne font que déposer 5 quartets sur la pile correspondant à l'adresse en mémoire de l'objet manipulé. Par exemple, un nombre réel est codé sur 16 quartets : le signe, la mantisse sur 12 quartets et l'exposant sur 3 quartets. Le tout est précédé par les 5 quartets de la routine qui l'identifie comme un réel, soit 21 quartets globalement si le nombre se trouve dans un programme. Cependant, si un nombre réel est stocké sur la pile, il utilise 32 quartets (en supposant que LAST, UNDO et COMMAND ne sont pas actifs). Je suppose que le nombre est stocké avec une adresse sur 5 quartets plus un pour préciser qu'il ne s'agit pas d'une variable nommée mais d'un objet sur la pile, ceci étant completé par l'adresse en mémoire du nombre ; nous retrouvons bien les 32 quartets. Les nombres complexes sont stockés de la même manière mais nécessitent 16 quartets supplémentaires pour la partie imaginaire.

Beaucoup d'opérations du HP-28C sont basées sur l'utilisation d'adresses sur 5 quartets. Ces adresses elles-mêmes ne sont pas des objets, mais elles pointent sur des objets, des instructions ou d'autres adresses. Si l'adresse ne pointe pas sur un objet ou une commande, elle est visualisée comme System Object. Les adresses sont des entités tellement importantes que je propose que les utilisateurs de HP-28C leur donnent un nom.

Je pourrais vous donner plus de détails, mais vous êtes sans doute curieux de savoir comment faire ces découvertes vous-même. Le reste de cet article va donc vous expliquer comment j'ai utilisé la découverte du STO synthétique par Ian Maw pour analyser la structure des objets du HP-28C.

1- Mettez l'objet à étudier au niveau 1 : en fait comme je l'ai dit plus haut, il n'y a probablement que l'adresse de l'objet qui soit poussée sur la pile, mais STO remplacera le nom par la valeur donc ceci n'a pas d'importance.

2- Assurez-vous de la présence d'au moins un objet au niveau 2. Ceci peut se faire simplement en appuyant sur [ENTER] pour dupliquer l'objet que vous voulez étudier.

3- Exécutez le programme « #261031 SYSEVAL » (le HP-28C étant en mode décimal). C'est équivalent à un STO à ceci près qu'il ne teste pas si l'objet au niveau 1 est un nom valide. Vous pouvez le vérifier en stockant un nom valide au niveau 1 et en exécutant le programme : le nom sera ajouté au menu USER de la même manière que si STO avait été effectué. S'il y a un autre type d'objet au niveau 1, celui-ci est traité comme un nom et ajouté au menu USER. Le nom ainsi produit n'a, en général, pas de sens mais vous pouvez verifier de quel objet il s'agit en appuyant sur la touche et en examinant ce qui se passe.

4- Appuyez sur ["], puis sur la touche à laquelle vous venez d'assigner le nom : ceci amène le nom que vous avez créé dans la ligne de commandes.

5- Vous venez de convertir l'objet à étudier en une chaîne de caractères dans la ligne de commande. Vous pouvez utiliser la fonction NUM du menu STRING pour extraire un par un les octets de cette chaîne et ainsi analyser le contenu de l'objet octet par octet. Ignorez le caractère espace inséré par le HP-28C entre le " et le premier caractère de la chaîne. Le dernier caractère est également un espace et peut être utilisé pour reconnaître la fin de la chaîne (voir point 10). Malheureusement, il y a quelques problèmes.

6- La fonction NUM ne vous donne la valeur que du premier octet de la chaîne. Vous devez conserver une copie du reste de la chaîne pour pouvoir l'étudier. J'utilise le programme suivant :

```
«
  DUP
  2 999 SUB
  SWAP
  NUM HEX R→B
»
```

pour donner la valeur hexadécimale du premier octet au niveau 1 et le reste de la chaîne au niveau 2.

7- Ensuite, les nombres et les adresses sont stockés en mémoire à l'envers. Ainsi, le nombre 1.2345443424E104 est représenté par la chaîne hexadécimale 0123454443424104 (soit le signe, suivi des 12 chiffres de la mantisse et des trois chiffres de l'exposant). Si vous mettez deux copies de ce nombre sur la pile et si vous exécutez #261031 SYSEVAL (en décimal), vous verrez le nom ABCD ajouté au menu USER. Le dernier octet (04) est interpreté comme la longueur du nom et les 4 octets précédents (44434241) sont utilisés comme nom. 41 représente A, 42 represente B, et ainsi de suite. C'est le code ASCII également utilisé par la HP-41 ou le HP-71. En d'autres termes, le nombre est effectivement stocké comme 04, 41, 42, 43, 44, 45, 23, 01. Le tout est précédé de l'adresse de la routine définissant les nombres réels, comme je l'ai déjà indiqué, mais elle est ignorée par le STO synthétique. Les adresses sont également stockées à l'envers. Par exemple, l'adresse de SYSEVAL est #1A582 et si vous examinez un programme contenant cette fonction, vous trouverez les octets 2X, 58, 1A ou 82, A5, X1. La première forme représente 1A582 précédé d'un quartet quelconque X le tout stocké à l'envers. La deuxième forme est similaire, mais le quartet est ajouté après l'adresse. Tout ceci ne rend pas le décodage facile mais vous pouvez toujours écrire un petit programme pour effectuer l'inversion.

8- Nous avons vu dans ce qui précède que les 5 premiers quartets d'un objet au niveau 1 de la pile sont perdus et que l'octet suivant est utilisé comme longueur de la chaîne. Pour étudier la totalité de l'objet, mettez le dans une liste puis stockez la liste avec #261031. Ainsi les éléments perdus font partie de la liste et non de l'objet. L'octet utilisé pour définir la longueur de la chaîne sera 96 en hexadécimal, ce qui vous permet d'étudier des objets importants. Les trois premiers quartets du nom constitueront le reste de la définition de fonction, ils seront suivis par les octets constituant l'objet.

9- Si la chaîne de nom que vous avez créé contient au moins un caractère ", alors seuls les caractères précédant le premier " seront utilisés. Une manière de résoudre ce problème est d'appuyer sur la touche située à la droite de la touche SHIFT pour rentrer dans le menu d'édition, et de remplacer les " (code 22 hexadécimal) par des ' (code 27 hexadécimal) par exemple.

10- Le nom peut également contenir des caractères nuls et fin de ligne (codes 00 et 0A hexadécimal). Les manuels ne sont pas très clairs à propos du caractère nul : ils indiquent simplement que vous ne pouvez pas éditer une chaîne contenant ces caractères. La raison est qu'il est utilisé pour marquer la fin de chaîne. On peut obtenir de telles chaînes avec la méthode décrite précedemment. Si vous appuyez sur [ENTER], le caractère nul sera compris comme une fin de chaîne (comme pour "). Pour vous débarasser des caractères nuls, fin de ligne et ", procédez comme suit :
a- allez dans le menu d'édition (voir point 9 plus haut),
b- appuyez sur [ ][<] pour vous rendre au début de la ligne,
c- appuyez sur [ ][>] pour aller à l'endroit où le HP-28C pense qu'il y a une fin de ligne. C'est soit le premier nul, soit un caractère fin de ligne, soit la vraie fin de la ligne. Si le caractère immédiatement à gauche du curseur est un espace, vous êtes probablement sur la vraie fin de chaîne, sinon le curseur s'est arrêté sur un caractère nul ou fin de ligne. En général c'est un nul : notez sa position et remplacez-le par un autre caractère (j'utilise Z pour zéro). Si le caractère est un seul nul, alors dès que le Z l'aura remplacé, de nouveaux caractères aparaîtront à sa droite. Cependant, il peut y avoir plusieurs nuls et vous pouvez avoir à répéter le processus [ ][>] Z plusieurs fois. Au bout d'un certain nombre de répétitions, vous verrez apparaître de nouveaux caractères ou l'espace qui marque la fin de la chaîne.
d- Si après avoir répété plusieurs fois le processus décrit ci-dessus, vous ne voyez pas apparaître de nouveaux caractères à la droite du curseur, vous êtes sans doute arrivé sur une fin de ligne. Vous pouvez le savoir en pressant [ENTER]. S'il y a une fin de ligne dans la chaîne, elle sera visualisée par un petit carré plein (le symbole des caractères non visualisables), suivi de Z, et d'autres caractères. Le caractère fin de ligne est difficile à traiter : une approche est de créer une nouvelle version de la chaîne que vous êtes en train d'analyser avec une nouvelle commande avant le caractère de fin de ligne. Comme cette commande fera 5 quartets, les 0 et A se retrouveront dans deux octets différents. Une autre méthode est de trouver la position du caractère de fin de ligne comme décrit plus haut puis simplement d'appuyer sur [ENTER] en acceptant le fait que le prochain nul ou " sera traité comme fin de chaîne.
e- Je repète le point c (et d si nécessaire) jusqu'à ce que je sois sûr d'avoir atteint la fin de la chaîne ou que suffisamment de nuls aient été enlevés pour permettre d'étudier la partie intéressante de l'objet. C'est une question de bon sens : les chaînes peuvent être longues et le HP-28C a peu de mémoire si bien qu'il n'est pas toujours possible d'analyser des chaînes entières, en particulier si vous utilisez la mémoire à d'autres fins.
f- Appuyez sur [ ][<] pour retourner au début de la chaîne et utilisez [>] pour remplacer les " par des ' en notant la position du caractère.
g- Appuyez sur [ENTER] pour déposer la chaîne au niveau 1 où elle sera analysée.

Maintenant, vous pouvez utiliser le programme présenté au point 6 pour analyser la chaîne ou écrire un autre programme convenant mieux à vos besoins. N'oubliez pas de prendre en compte les Z et ' utilisés pour remplacer les nuls et les ".

Bien, tout ceci est très utile mais je n'ai pas encore trouvé de PEEK et de POKE qui seraient indispensables. Au moins, le langage machine du HP-28C est le même que celui du HP-71 ce qui devrait nous aider à interpréter ce que nous lirons. Dans le même temps, il devrait être possible d'écrire des programmes pour faire PEEK et POKE en créant de longues chaînes de caractères contenant les octets du programme. Il suffira alors de connaître l'adresse de ces chaînes et d'utiliser SYSEVAL pour les évaluer, il faudra également connaitre le code qui dit au HP-28C qu'une chaîne de quartets est une routine en langage machine et non l'adresse d'une routine. Certaines opérations exécutées par SYSEVAL retournent le contenu de la mémoire sur la pile comme des nombres binaires, il devrait être possible de trouver un SYSEVAL qui retourne le valeur du compteur programme : ceci serait très utile.

Vous trouverez après cet article la liste des points d'entrée promise au début. Toutes les fonctions semblent être dans une zone comprise entre #17000 et #1C000 sauf les structures de programme (telles que IF, WHILE, NEXT) qui sont comprises entre #27CB4 et #28340. J'ai ajouté quelques points d'entrée qui ne correspondent pas à des commandes mais qui se trouvent dans les mêmes zones tels que : «, » ou ' notez que → suivi de « est différent de → suivi de '. J'ai trouvé les adresses des commandes qui définissent différents types d'objets mais je n'en ai pas encore fait la liste. Toutes les adresses sont données en hexadécimal.

Voilà, amusez vous bien avec tout ceci, et n'hésitez pas à faire part de vos découvertes.

adr.	nom com.	dist. à la com. suiv.
177F1	DUP	26
17825	SWAP	26
17859	DROP2	26
1788D	OVER	26
178C6	DROPN	26
178FF	PICK	26
17933	ROLLD	26
1796C	→LIST	36
179EB	RE	96
17AB5	SUB	86
17B43	C→R	66
17C03	POS	36
17C41	STR→	36
17C89	CHR	36
17D62	→ARRY	121
17E36	RDM	106
17F7D	IDN	151
18065	PUT	86
1830A	GET	86
188AE	SAME	41
18950	OR	121
18A1F	XOR	121
18B0C	≠	121
18BF9	>	116
18CE1	≥	116
18DD3	NEG	76
18E6B	CONJ	76
18EE5	MAXR	46
18F41	e	46
18F9D	+	186
190FD	*	206
19271	^	226
1939F	ARG	86
1951D	R→P	126
195DD	√	111
196C0	SIN	66
19744	TAN	66
197C8	COSH	66
1984C	ASIN	121
1993E	ATAN	66
199C2	ACOSH	116
19AAF	EXP	66
19B60	LOG	111
19C11	LNP1	56
19C81	FACT	56
19CF1	FP	56
19D61	CEIL	56
19DD1	MAX	76
19E69	MOD	76
19EED	D→R	56
19F53	→HMS	36
19F9B	HMS+	36
19FE3	RNRM	36
1A02B	DET	36
1780B	DUP2	26
1783F	DROP	26
17873	ROT	26
178A7	DEPTH	31
178E0	DUPN	31
17919	ROLL	26
1794D	CLEAR	31
17990	R→C	91
17A4B	IM	106
17B0B	LIST→	56
17B85	SIZE	126
17C27	→STR	26
17C65	NUM	36
17CAD	TYPE	181
17DDB	ARRY→	91
17EA0	CON	221
18014	TRN	81
180BB	PUTI	591
18360	GETI	1358
188D7	AND	121
189C9	NOT	86
18A98	==	116
18B85	<	116
18C6D	≤	116
18D55	=	126
18E1F	ABS	76
18EB7	π	46
18F13	MINR	46
18F6F	i	46
19057	-	166
191CB	/	166
19353	INV	76
193F5	P→R	296
1959B	SIGN	66
1964C	SQ	116
19702	COS	66
19786	SINH	66
1980A	TANH	66
198C5	ACOS	121
19980	ASINH	66
19A36	ATANH	121
19AF1	LN	111
19BCF	ALOG	66
19C49	EXPM	56
19CB9	IP	56
19D29	FLOOR	56
19D99	XPON	56
19E1D	MIN	76
19EB5	MANT	56
19F25	R→D	46
19F77	HMS→	36
19FBF	HMS-	36
1A007	CNRM	36
1A04F	DOT	36
1A073	CROSS	36
1A0C5	%	76
1A15D	%CH	66
1A1B9	RDZ	36
1A242	STO	116
1A32A	MEM	56
1A3A4	CLUSR	31
1A3DD	ABORT	26
1A411	ERRM	26
1A463	IFTE	196
1A582	SYSEVAL	90
1A60A	RND	46
1A65C	→NUM	31
1A761	WAIT	36
1A79F	KEY	26
1A7DD	SF	61
1A857	FS?	61
1A8D6	DEG	36
1A91E	FIX	56
1A98E	ENG	56
1A9E0	FS?C	81
1AA87	BIN	26
1AABB	HEX	26
1AAEF	STWS	36
1AB32	RCLF	31
1ABED	STOΣ	56
1AC3F	RCLΣ	26
1AC9B	Σ-	26
1ACCF	CORR	26
1AD03	MAXΣ	26
1AD37	MINΣ	26
1AD6B	TOT	26
1AD9F	LR	26
1ADDD	UTPC	36
1AE25	UTPF	36
1AE6D	SCLΣ	26
1AEA1	COLΣ	36
1AF5C	SNEG	151
1B076	STO+	281
1B2AD	STO/	286
1B683	EXGET	36
1B6DF	OBSUB	36
1B727	FORM	56
1B797	EXPAN	56
1B7F3	QUAD	36
1B845	TAYLR	46
1B914	RCEQ	31
1B952	ROOT	166
1BB70	ASR	36
1BBB8	RLB	36
1BC00	RRB	36
1BC48	SLB	36
1BC90	SRB	36
1BCD8	B→R	36
1BD3E	INDEP	46
1BD90	PMAX	36
1BDD8	CENTR	36
1BE20	*H	36
1A097	RSD	46
1A111	%T	76
1A19F	RAND	26
1A1DD	RCL	101
1A2B6	PURGE	116
1A362	ORDER	66
1A3C3	KILL	26
1A3F7	ERRN	26
1A42B	EVAL	56
1A527	IFT	91
1A5DC	DISP	46
1A638	BEEP	36
1A67B	LAST	230
1A785	CLLCD	26
1A7B9	CLMF	36
1A81A	CF	61
1A894	FC?	66
1A8FA	RAD	36
1A956	SCI	56
1A9C6	STD	26
1AA31	FC?C	86
1AAA1	DEC	26
1AAD5	OCT	26
1AB13	RCWS	31
1AB51	STOF	156
1AC25	CLΣ	26
1AC59	Σ+	66
1ACB5	NΣ	26
1ACE9	COV	26
1AD1D	MEAN	26
1AD51	SDEV	26
1AD85	VAR	26
1ADB9	PREDV	36
1AE01	UTPN	36
1AE49	UTPT	36
1AE87	DRWΣ	26
1AEC5	SINV	151
1AFF3	SCONJ	131
1B18F	STO-	286
1B3CB	STO*	696
1B6A7	EXSUB	56
1B703	OBGET	36
1B75F	COLCT	56
1B7CF	ISOL	36
1B817	SHOW	46
1B873	∂	161
1B933	STEQ	31
1B9F8	∫	376
1BB94	RL	36
1BBDC	RR	36
1BC24	SL	36
1BC6C	SR	36
1BCB4	R→B	36
1BCFC	CONVERT	66
1BD6C	PMIN	36
1BDB4	AXES	36
1BDFC	RES	36
1BE44	*W	36

1BE68 DRAW	31	1BE87 PIXEL	36
1BEAB DRAX	31	1BECA PR1	26
1BEE4 PRSTC	26	1BEFE PRST	26
1BF18 CR	26	1BF32 PRUSR	26
1BF4C PRVAR	36	1BF70 PRMD	26
1BF8A PRLCD	48426	27CB4 IF	21
27CC9 THEN(IF)	96	27D29 ELSE	31
27D48 END(IF)	21	27D5D pas → avant «	26
27D77 → avant «	311	27EAE → avant '	66
27EF0 fin →	26	27F0A «	21
27F1F »	21	27F34 ' début nom	26
27F4E ' fin nom	21	27F63 WHILE	31
27F82 REPEAT	66	27FC4 END(WHILE)	31
27FE3 DO	31	28002 UNTIL	21
28017 END(DO)	66	28059 START	71
280A0 FOR	91	280FB NEXT	302
28229 STEP	59	28264 IFERR	199
2832B THEN(IFERR)	21	28340 HALT	

Notes :

1- Cette table montre que les commandes sont stockées par groupes logiques.

2- La plupart des commandes sont séparées de $5n+1$ quartets.

3- La distance à la commande suivante peut-être une indication sur la complexité de la commande ou sur la présence de code entre les deux commandes.

4- J'ai inclus quelques entités qui ne sont pas vraiment des commandes mais qui font partie de la structure des programmes telles que « ou ». D'autres élements tels que les entiers de -9 à 9 ou les majuscules auraient pu être inclus.

5- Cette liste a été préparée par Wlodek Mier-Jedrzejowicz avec certaines adresses trouvées par Bruce Bailey. Faites part de vos découvertes !

Wlodek Mier-Jedrzejowicz

PEEK SUR HP-28

Lors de la Conférence Internationale de Copenhague, Wlodek Mier-Jedrzejowicz nous a présenté le tout dernier outil qu'il a réalisé pour l'étude de la structure interne du HP-28C. Il s'agit d'un programme prenant sur la pile une adresse hexadécimale et retournant le contenu des 8 octets situés à partir de cette adresse.

Le fonctionnement est assez complexe et il faut souhaiter que Wlodek le documente complètement. Dans les grandes lignes, il s'agit de créer un programme artificiellement dans la partie haute de la mémoire de la machine et de demander son exécution par un SYSEVAL approprié.

Le programme est donc le suivant (vous devez être en mode hexadécimal pour le rentrer) :

```
«
  RCWS SWAP 28 STWS SLB 64 STWS
  #B00E6FB5100000F1 +
  #E63013314FF7B SWAP
  #E6F9514FFF0F1
  #C808961241131311 4
  →LIST 'X1' DUP PURGE STO
  "DUMMYSTR" 'X2' DUP
  PURGE STO #4FF76
  SYSEVAL STWS 'X2'
»
```

Vous pouvez nommer ce programme PEEK. Un exemple amusant d'utilisation est :

```
#3FE7A
[PEEK] --> 1: "Version "
#3FE8A
[PEEK] --> 1: "1BBCopyr"
#3FE9A
[PEEK] --> 1: "ight(c) "
#3FEAA
[PEEK] --> 1: "Hewlett-"
#3FEBA
[PEEK] --> 1: "Packard "
#3FECA
[PEEK] --> 1: "1986"
```

En d'autres termes, entre les adresses #3FE7A et #3FED2 on trouve le message "Version 1BBCopyright(c) Hewlett-Packard 1986". Il ne vous reste plus qu'à explorer la machine et n'oubliez pas de faire part de vos découvertes.

LES ANOMALIES DU HP-28C

Voici la liste des anomalies du HP-28C connues à ce jour. Cette liste nous a été communiquée par Hewlett-Packard, que nous remercions ici vivement.

Ne vous alarmez pas ! Dans la quasi-totalité des cas, il s'agit de défauts mineurs qui n'affectent en rien l'utilisation habituelle de cette prodigieuse machine.

La liste des problèmes les plus importants est maintenant fournie avec les machines.

Vous n'avez donc pas besoin d'appeler le Support Technique si vous vous trouvez confronté à un de ces petits problèmes...

Utilisation de SOLVR

Le fait d'utiliser l'algorithme de résolution d'équations avec un programme qui n'augmente pas la « profondeur » de la pile opérationnelle (c'est à dire le nombre d'objets de la pile) peut corrompre celle-ci ; cette situation sera marquée par l'apparition du message `System Object` sur la pile. Par exemple :

```
[SOLV]
« A DROP
[ENTER]
[STEQ] [SOLVR]
[] [A]
```

donne sur la pile :

```
Non-real Result
2:        System Object
1:        System Object
```

Si ceci se produit effectuer immédiatement un «System Halt» de la manière suivante :

Appuyez sur `[ON]` et maintenez la pression, appuyez sur `[^]` puis relachez `[ON]`.

Il ne faut pas faire `[DROP]` ou `[CLEAR]`. Si vous essayez de faire disparaître `System Object` ou d'effacer la pile, vous risquez de provoquer une remise à zéro du calculateur.

Suppression d'une ligne vide en ligne de commandes

Pour supprimer une ligne vide en ligne de commandes, placez le curseur sur la ligne vide et appuyez sur `[DEL]`.

N'appuyez pas sur `[][INS]` (qui efface tous les caractères à gauche du curseur) lorsque le curseur est placé sur une ligne vide. Ceci corrompt la ligne de commande et toutes les opérations subséquentes ; cette situation peut aboutir à une remise à zéro du calculateur. Si vous corrompez la ligne de commande, appuyez immédiatement sur `[ON]` pour supprimer la ligne de commande ou sur `[ENTER]` pour l'évaluer.

Division faisant intervenir des entiers binaires

Le résultat de la division est incorrect lorsque le dividende (au niveau 2) est un entier binaire de 64 bits dont le bit de poids le plus fort est à 1 (c'est à dire, lorsque # n ≥ 8000000000000000_{16}.

```
[BINARY]
[HEX] 64 [STWS]
#0 [ENTER] 1 [-]  --> # FFFFFFFFFFFFFFFF
#F [/] ------------> # 1FFFFFFFFFFFFFFF
```

Le résultat devrait être : # 111111111111111. Il n'y a pas de solution de remplacement.

Comparaison d'égalité entre entiers binaires

Les entiers binaires de tailles différentes sont toujours considérés comme inégaux. Les entiers binaires que vous saisissez au clavier ont 64 bits ; ceux qui sont obtenus par calcul sont tronqués en fonction de la taille de mot en vigueur. Avant de tester l'égalité de deux entiers binaires, assurez vous que chacun est le résultat d'une opération effectuée avec la la taille de mot courante.

```
[BINARY] 16 [STWS]
#0 [STACK] [DUP] [DUP]
[+] [TEST] [NEXT] [==] ---> 0
```

Le résultat devrait être 1 (vrai). On a comparé un entier de 64 bits et le même entier sur 16 bits, car `[+]` retourne un resultat sur 16 bits.

Le petit programme suivant tronque deux entiers binaires en fonction de la taille de mot courante et test leur égalité :

```
« 0 + SWAP 0 + == »
```

SOLVR et flags 57 ou 58

Si les indicateurs binaires 57 ou 58 sont armés, la fonction SOLVR peut renvoyer une erreur de dépassement de capacité pendant le processus d'itération. Par exemple :

```
[TEST] 57 [SF] 58 [SF]
'X/X-2' [ENTER] [SOLV] [STEQ] [SOLVR]
1E490 [X]
[ ] [X] --->  Overflow
              2:        1.00
              1:        2.00
```

SOLVR peut donner "Sign Reversal" à tort.

SOLVR peut renvoyer le message "Sign Reversal" sans trouver les points voisins où la fonction change de signe. Il peut y avoir une discontinuité entre les valeurs donnant des signes inverses. Par exemple :

```
[SOLV] '(-27)^N=-3'
[STEQ] [SOLVR]
0 [N]
[ ][N] ---> N:0
            Sign Reversal
            1:            0
```

Unités définies par l'utilisateur

Des unités utilisateur qui sont des multiples de degrés Fahrenheit ou Celsius ne sont pas converties correctement. Par exemple :

```
{ 10 "°F" } [ENTER] 'F10' [STO]
1 [ENTER] 'F10' [ENTER] '°F' [ENTER]
[CONVERT] -------> 2:       -449.67
                   1:          '°F'
```

Le résultat correct est 10 °F.

Définition de l'année

L'unitée *année* (yr) compte 365 jours. Elle aurait dû compter environ 365.25 jours pour rester cohérent avec la définition de l'*année lumière* (lyr).

Fonction RND

La fonction RND ne fonctionne pas correctement en mode FIX. Par exemple :

```
.7 [ENTER]
[MODE] 0 [FIX] -----> 1
[REAL] [RND] -------> 0
```

Le *Manuel de Référence* (page 272) décrit le fonctionnement de RND en mode FIX. La description correspond à ce que fait le calculateur, mais ce n'est pas correct.

Comparaisons de chaînes

La comparaison de chaînes ne donne pas de résultat correct si les chaînes sont de même longueur. Par exemple :

```
"AAA" [ENTER]
"AAB" [ENTER]
[<] [ENTER] -----> 0
```

Comme décrit page 255 du *Manuel de Référence*, cet exemple devrait retourner 1 (vrai).

EDIT / CHS

La séquence de touches suivante donne un résultat inattendu quand on appuye sur [CHS] :

```
[[2 3]] [ENTER]
[EDIT]
[>] [>] [>] [>] [>]  amène le curseur sur le 3
4 [CHS]  ----> [[2 4-]]
```

Au lieu de donner -4, la séquence `4 [CHS]` donne à l'affichage : `[[2 4-]]`.

EEX

Suivant la position du curseur, l'utilisation de [EEX] pendant que l'on écrit une expression algébrique peut provoquer de résultats inattendus. Utiliser plutôt la lettre E dans ces cas si l'on veut garder la notation exponentielle.

HP-41

J. Elhay
R. Dine

Probabilités *18*
Les nombres premiers *19*

PROBABILITES

Voici trois petits programmes que certains pourront trouver utiles. Tous les trois sont employés dans le domaine des probabilités discrètes.

Ils évaluent des termes ou des sommes partielles de distributions de Poisson, binomiale ou hypergéométrique.

Loi de Poisson

Faire XEQ "POISSD". Le programme affiche le message "READY".

Les 5 touches [A] à [E] ont dès lors la signification suivante :

[A] : évalue $P(X=x)$, après avoir demandé les valeurs de λ et x. $P(X=x) = (e^{\lambda} * \lambda^x)/x!$

[B] : évalue $P(X=x)$ pour un nouvel x et le même λ.

[C] : évalue $P(X \leq x)$.

[D] : évalue $P(X \leq x)$ pour un nouvel x et le même λ.

[E] : évalue $P(X>x)$.

Loi binomiale

Le programme BIMOD a un fonctionnement similaire, excepté qu'il demande le nombre d'échantillons n, la proportion p et finalement x.

Loi hypergéométrique

Le programme HYPGD demande la population N, le nombre de défauts D, la taille de l'échantillon n et finalement x.

Ces programes sont relativement rapides, car j'utilise le schéma de Horner pour les distributions cumulatives. D'autre part, il y a demande explicite de toutes les valeurs pour les touches [A], [C] et [E], alors que les touches [B] et [D] ne requièrent que x.

Bonnes statistiques...

Jack Elhay

```
01*LBL "POISSD"
SF 27  FIX 4  "READY"  AVIEW  STOP

07*LBL A
XEQ 20

09*LBL B
XEQ 21  RCL 01  RCL 03  Y^X  RCL 02  *  RCL 03
FACT  /  GTO 01

20*LBL 20
"LAMBDA ?"  PROMPT  STO 01  CHS  E^X  STO 02  RTN

28*LBL 21
"X= ?"  PROMPT  STO 03  RTN

33*LBL C
XEQ 20

35*LBL D
XEQ 21  RCL 01  ENTER^  ENTER^  ENTER^

41*LBL 02
RCL 03  /  1  +  *  DSE 03  GTO 02  RCL 01  /
RCL 02  *  FS?C 00  RTN  GTO 01

56*LBL E
SF 00  XEQ C  1  X<>Y  -

62*LBL 01
"P="  ARCL X  AVIEW  RTN  END
```

```
01*LBL "BINOMD"
SF 27  FIX 4  "READY"  AVIEW  STOP

07*LBL 20
"N= ?"  PROMPT  STO 00  1  +  STO 01  "P= ?"
PROMPT  STO 02  1  X<>Y  -  STO 03  RCL 02  X<>Y
/  STO 04  RCL 03  RCL 00  Y^X  STO 05  RTN

30*LBL C
XEQ 20

32*LBL D
"X= ?"  PROMPT  STO 10  RCL 04  ENTER^  ENTER^
ENTER^  SF 01

41*LBL 00
RCL 01  RCL 10  -  RCL 10  /  FC?C 01  *  *  1  +
DSE 10  GTO 00  RCL 05  *  FS?C 00  RTN

58*LBL 02
"P="  ARCL X  AVIEW  RTN

63*LBL E
SF 00  XEQ C  1  X<>Y  -  GTO 02
```

```
70*LBL A
XEQ 20

72*LBL B
"X ?"  PROMPT  STO 10  RCL 00  X<>Y  XEQ 98
STO 09  RCL 02  RCL 10  Y^X  ST* 09  RCL 03
RCL 00  RCL 10  -  Y^X  ST* 09  RCL 09  GTO 02

92*LBL 98
X=0?  GTO 99  X<>Y  ENTER^  X<> Z  -  X=0?  GTO 99
LASTX  X<=Y?  X<>Y  RDN  1  X<> Z

107*LBL 01
ST* Z  X<>Y  ST/ Z  X<>Y  DSE X  DSE Y  GTO 01
X<> Z  RTN

117*LBL 99
1  RTN  END
```

```
01*LBL "HYPGD"
FIX 4  SF 27  "READY"  AVIEW  STOP

07*LBL 80
"POPN ?"  PROMPT  STO 03  STO 02  STO 07
"DEFECTIVES?"  PROMPT  STO 01  ST- 02  ST- 07
"SAMPLE SIZE?"  PROMPT  STO 04  ST- 07  RTN

23*LBL 70
"X= ?"  PROMPT  X=0?  GTO 75  STO 05  RCL 02
RCL 04  XEQ 20  STO 06  RCL 03  RCL 04  XEQ 20
ST/ 06  1  ST+ 01  ST+ 04  ENTER^  ENTER^  ENTER^

43*LBL 00
RCL 01  RCL 05  -  RCL 04  RCL 05  -  *  RCL 07
RCL 05  +  /  RCL 05  /  *  1  +  DSE 05  GTO 00
RCL 06  *  FS?C 00  RTN  RTN

67*LBL 02
"P="  ARCL X  AVIEW  RTN

72*LBL C
XEQ 80  XEQ 70  GTO 02

76*LBL E
SF 00  XEQ 80  XEQ 70  1  X<>Y  -  GTO 02

84*LBL 20
X=0?  GTO 99  X<>Y  ENTER^  X<> Z  -  X=0?  GTO 99
LASTX  X<=Y?  X<>Y  RDN  1  X<> Z

99*LBL 01
ST* Z  X<>Y  ST/ Z  X<>Y  DSE X  DSE Y  GTO 01
X<> Z  RTN

109*LBL 99
1  RTN
```

```
112*LBL A
XEQ 80

114*LBL B
"X ?"  PROMPT

117*LBL 75
STO 05  RCL 01  RCL 05  XEQ 20  STO 09  RCL 02
RCL 04  RCL 05  -  XEQ 20  ST* 09  RCL 03  RCL 04
XEQ 20  ST/ 09  RCL 09  FS?C 00  RTN  GTO 02  END
```

LES NOMBRES PREMIERS

Le but est de savoir si un nombre est premier et de connaître les nombres premiers consécutifs.

UTILISATION

PREM

A l'appel de "NB IMP", entrer le nombre impair étudié *n*. Le programme divise ce nombre par les nombres impairs consécutifs de 3 à SQR(*n*). S'il n'y a eu aucun reste nul, le nombre *n* est déclaré premier et affiché, et le programme continue avec *n*+2 etc. Les nombres premiers successifs sont affichés jusqu'à ce que l'on arrête par [R/S]. Les résultats sont imprimés si une imprimante est branchée.

PREM1 et PREM2

Le programme PREM est simple, mais devient lent si *n* est grand. Si *n* est aux environs de 1000000, il faut une moyenne de 200 secondes entre chaque résultat. Ne pourrait-on diviser par les nombres premiers consécutifs au lieu des nombres entiers impairs ? Au lieu de faire environ 500 divisions, on en ferait environ 168 (nombre de nombres premiers de 3 à 1000), soit le tiers. Encore faut-il, me direz-vous, avoir ces 168 nombres ? Vous les avez : le programme PREM1 insère ces nombres premiers dans les registres R10 à R65 (il faut environ 16 minutes).

Ensuite, le programme PREM2 réalise le calcul par division avec les nombres premiers successif de 3 à SQR(*n*). Les résultats sont affichés et éventuellement imprimés comme pour PREM.

CONCLUSION

On obtient les résultats suivants.

A partir du nombre entré à la demande du programme, voisin de 1000000 (par exemple 988033 qui est premier), on trouve les temps suivant :

	PREM	PREM2	BASIC (APPLE IIe)
988033	170 sec.	118 sec.	7 sec.
988051	390 sec.	298 sec.	14 sec.
988061	570 sec.	420 sec.	20 sec.

Nous n'avons pas beaucoup gagné en passant de PREM à PREM2. D'autant plus qu'il aura fallu entrer les données de R10 à R65. Mais nous nous sommes bien amusé et nous avons montré qu'on sait se servir d'une HP-41. Mais on n'est pas plus fier pour cela, car j'ai fait un programme basic de 7 lignes avec le même organigramme que PREM. Ses performances sont indiquées dans la dernière colonne ci-dessus : 28 fois plus vite que le programme PREM, 21 fois plus vite le programme PREM2. A noter que sur un Questar M (Bull) j'avais exécuté 10 fois plus vite seulement.

René Dine (293)

NDLR : sur HP-71, si on utilise JPCLEX et en particulier la fonction PRIM de Guy Toublanc, on obtient les résultats suivants :
988033 .52 sec.
988051 .50 sec.
988061 .50 sec.
Soit entre 13 et 40 fois mieux que sur un Apple IIe. L'honneur de HP est sauf !

```
01*LBL "PREM"
FIX 0  "NB IMP?"  PROMPT  STO 00

06*LBL 04
RCL 00  SQRT  INT  3000,02  +  1000  /  STO 01

15*LBL 01
RCL 01  INT  RCL 00  X<>Y  MOD  X=0?  GTO 03
ISG 01  GTO 01  RCL 00  VIEW 00

27*LBL 03
2  ST+ 00  GTO 04  END


01*LBL "PREM1"
FIX 0  3005007  STO 10  11,065  STO 02  9  STO 00

09*LBL 04
RCL 00  SQRT  INT  3000,02  +  1000  /  STO 01

18*LBL 01
RCL 01  INT  RCL 00  X<>Y  MOD  X=0?  GTO 03
ISG 01  GTO 01  RCL 00  VIEW 00  FS? 01  GTO 05  1
E6  *  STO IND 02  SF 01  GTO 03

37*LBL 05
FS?C 02  GTO 06  1  E3  *  ST+ IND 02  SF 02
GTO 03

45*LBL 06
ST+ IND 02  CF 01  ISG 02

49*LBL 03
2  ST+ 00  GTO 04  END

01*LBL "PREM2"
FIX 0  "NB IMP ?"  PROMPT  STO 00

06*LBL 03
RCL 00  SQRT  STO 01  10  STO 02

12*LBL 02
RCL IND 02  1  E6  /  ENTER^  INT  XEQ A  1  E3  *
ENTER^  INT  XEQ A  1  E3  *  XEQ A  1  ST+ 02
GTO 02

30*LBL A
RCL 01  X<>Y  X>Y?  GTO 04  RCL 00  X<>Y  MOD
X=0?  GTO 05  RDN  RDN  FRC  RTN

44*LBL 04
RCL 00  VIEW X

47*LBL 05
2  ST+ 00  GTO 03  END
```

FORTH

X. Bille — Tracé de courbes (acte II) *22*

ASSEMBLEUR

P. David & J. Taillandier — Assembleur pour HP-71 *24*
J. Buitenhuis & F. van der Windt — Module de développement *24*
J. Baudier — L'assembleur du HP-71 (acte III) *26*
J. Elhay — Inversez vos drapeaux *29*

BASIC

E. Gengoux — UNPURGE *30*

E. Gengoux — Programme "UNPURGE" *31*
P. Nicodème — Programme "GAO" (voir JPC 46) *31*

LE COIN DES LHEX

39

TRACE DE COURBES
(ACTE II)

Dans JPC 42 était paru un programme de tracé de courbes sur l'afficheur du HP-71. Cette première version a montré d'énormes inconvénients d'utilisation, en particulier le manque de souplesse. Je présente aujourd'hui un programme moitié Forth, moitié Basic. La partie graphique possède de nombreux points communs avec la version antérieure.

PRINCIPES

Une matrice de 132*128 pixels est créée dans un coin de mémoire (dictionnaire Forth) ; chaque pixel est adressable individuellement grâce à ses coordonnées. Cela suppose l'institution d'une origine dans ce repère et la possibilité de définir la valeur d'un pixel.

Pour simplifier l'utilisation, on définit l'étendue des valeurs représentables dans la matrice. Le pas entre deux valeurs est calculé tel que le maximum de points soit atteint : 132 points en abscisse (axe X'oX), 128 points en ordonnée (axe Y'oY).

Au niveau programmation, un certain nombre de mots Forth permettent de simplifier au maximum les opérations de calcul d'axes, de placement d'un pixel, etc. On remarquera sans doute l'analogie qui existe entre mes mots de programme et ceux d'une calculatrice graphique du commerce !

Et l'affichage graphique ? Toujours comme sur la première version : l'écran est une fenêtre de 132 par 8 que l'on déplace selon les ordonnées, de haut en bas, en faisant référence à un tableau. La fenêtre est déplacée à l'aide des touches de curseur (gestion par Basic), le pas étant de 4 pixels. La composition de la fenêtre se fait en Forth et en Assembleur. Compliqué certes, mais très agréable car rapide.

Le temps de calcul ? Log x est tracé en 45 secondes environ. Ce temps peut s'allonger, en fonction de la complexité de la fonction.

CHARGEMENT

Le programme comprend trois parties en assembleur, Forth et Basic. Il faut augmenter le dictionnaire Forth de 2000 quartets, assembler la partie assembleur puis compiler la partie Forth. Le programme principal est en Basic.

UN BEL EXERCICE DE STYLE

Oui, ce n'est qu'un amusement, le temps de calcul est bien trop long. Cependant, si tout avait été écrit en code machine alors peut-être que...

Faites tracer cos x + cos 10x. Le résultat, pas brillant, humm ? Il manque une fonction d'interpolation : étant donné deux points, valeurs d'une fonction, tracer la droite qui les joint. Moi, je vous la laisse sans regret !

MODE D'EMPLOI

Le principe sur lequel repose le programme principal est que l'état des variables n'est pas modifié entre deux exécutions. Dans le cas contraire, vous serez amené à relancer le système et réinitialiser les variables : `RUN BOOT`. Un menu propose les choix suivants :

`OT` : pour sortir du programme (out).

`CL` : (clear) pour nettoyer la matrice graphique et redessiner les axes du repère.

`RN` : (range) définit les paramètres du repère, les plages de valeurs. Les options possibles sont :
- P (paramètrique) pour préparer la machine à tracer des courbes paramètriques, et
- F (function) pour tracer les fonctions du type $y=f(x)$.

Dans les deux cas, le programme connaissant les valeurs maximale et minimale, calcule la place des axes. Toute introduction incompatible est annulée, puis redemandée. Dans le cas des courbes paramètrées (x(t),y(t)), le pas de calcul en t sera fourni: T.

`GR` : (graphic) donner en face du `"="` la valeur littérale de la fonction à tracer. Par exemple :

```
F(X)=EXP(-.2*X)*SIN(3*X)
X(T)=T*COS(.7*T) ; Y(T)=T*SIN(.7*T)
```

Un son annonce la fin des cogitations. Le nombre de valeurs ayant provoqué une erreur est affiché.

`SH` : (show) les touches `[v]`, `[^]`, `[g][v]` et `[g][^]` déplacent la fenêtre-écran de 4 pixels vers le bas ou le haut. La touche espace (32) renvoie au menu.

Le petit «plus» du programme : si le drapeau 4 est levé au clavier (`SFLAG 4`), le pas de calcul est diminué de moitié (264 itérations). Double précision, mais aussi temps de résolution doublé. Eternel conflit vitesse / précision (si, si !).

Xavier Bille (203)

Partie Assembleur ; elle est identique à celle parue dans JPC 42. On s'y reportera pour de plus amples explications.

```
        FORTH
here    EQU    #2FB93
        WORD   'MIX'    (add1 add2 -- add3)
        ADDEX           mot qui fabrique une
        R0=A            chaine de 132 octets à
        D0=(5) here     partir de 2 chaines de
         *              132 octets
        A=DAT0 A
        LC(5)  #5A
        A=A+C  A
        D0=A
        R1=A
        C=DAT1 A
        C=C+1  A
        DAT1=C A
        LC(2)  #84
        D1=D1+ 5
deb     R2=C
        GOSUB  adres
        ASL    B
        D1=D1- 5
        GOSUB  adres
        DAT0=A B
        D0=D0+ 2
        D1=D1+ 5
        C=R2
        C=C-1  B
        ?C#0   B
        GOYES  deb
        C=R1
        DAT1=C A
        C=R0
        D0=C
        RTNCC
adres   C=DAT1 A
        CD1EX
        A=DAT1 1
        D1=D1+ 2
        CD1EX
        DAT1=C A
        RTN
```

Partie Forth :

```
( crée une structure matrice-graphique          )

: GRAPH  ( GRAPH name)
  CREATE 264 * NALLOT
  DOES> SWAP 264 * + ;

16 GRAPH TABLE
( TABLE en est justement une !                  )

( efface la matrice-écran                       )

: CLEAR  (  -- )
  0 TABLE 2112 0 FILL ; IMMEDIATE

FVARIABLE XRA
FVARIABLE YRA
FVARIABLE ORIX
FVARIABLE ORY

( xra [resp. yra], valeur d'un point x'ox       )
( [resp. y'oy]                                  )
( orix [resp. ory], origine des x [resp. y]     )

( trace des axes en fonction des paramètres du  )
( repère                                        )

: AXIS   (  -- )
  ORY RCL 0. X<=Y? RDN 128. X>Y? AND
  IF
    RDN FTOI 8 /MOD
    TABLE SWAP 2. ITOF Y^X
    132 FTOI FILL
  THEN
  ORIX RCL 0. X<=Y? RDN 132. X>Y? AND
  IF
    255 RDN FTOI 2*
    16 0
    DO
      2DUP I TABLE + C!
    LOOP
    2DROP
  THEN ;

( définit l'étendue des valeurs en fonction de  )
( données fournies par Basic                    )

: RANGE   (  -- )
  " A" BASICF XRA STO
  " -X1" BASICF F* ORIX STO
  " B" BASICF YRA STO
  " Y2" BASICF F* ORY STO ;

( gère la fenêtre-écran.                        )
( 128=16*8 pixels, le pas de déplacement étant  )
( de 4 pixels, on a 32 écrans possibles,        )
( soit 0 <= n <= 31.                            )

: COMPOSE   ( n -- )
  2 /MOD SWAP 0=
  IF
    TABLE
  ELSE
    DUP 1+ TABLE SWAP TABLE MIX
  THEN 264 ;

( Allume un pixel étant donné ses coordonnées   )
( réelles.                                      )
```

```
( fonctionne avec des valeurs Basic, X et Y.      )

: PLOT    ( -- )
  ORIX RCL " X" BASICF XRA RCL F* F+
  FTOI DUP
  131 > OVER 0< OR
  IF DROP
  ELSE
    ORY RCL " Y" BASICF YRA RCL F* F-
    FTOI DUP
    127 > OVER 0< OR
    IF  2DROP
    ELSE
      8 /MOD TABLE
      ROT 2* +
      SWAP 2. ITOF Y^X FTOI
      OVER C@ OR SWAP C!
    THEN
  THEN ;
```

ASSEMBLEUR POUR HP-71B

Vous vous rappelez peut-être de deux articles parus dans JPC 35 et JPC 36 (Juin, Juillet / Août 1986), décrivant un assembleur très rapide pour le HP-71, tournant sur un gros ordinateur HP.

Aujourd'hui, cet assembleur est disponible sur les ordinateurs compatibles, tournant sous MS-DOS. Le portage du programme n'a pas été chose facile, ni amusante, mais le résultat est là : vous pouvez maintenant avoir, **gratuitement**, un assembleur rapide pour le HP-71.

Rappelons ce qu'est ce système : c'est un assembleur, un éditeur de liens, et quelques utilitaires (générateur de table de messages, transfert...). Il vous permet d'écrire votre programme source, de l'assembler sur votre ordinateur, puis de transférer le Lex résultant dans le HP-71. L'intérêt de l'opération est que l'assemblage est particulièrement rapide. Sur un HP Vectra, il est entre 100 et 200 fois plus rapide que l'assembleur du module Forth / Assembleur.

De plus, il y a un éditeur de liens, ce qui vous ouvre la voie de la compilation séparée. Les avantages sont nombreux, et les gains énormes.

Comme nous l'avons mentionné plus haut, ce système est gratuit. Si vous êtes intéressé, envoyez-nous une disquette 5"1/4 ou 3"1/2 déjà formattée, ainsi que le nécessaire pour la réponse (emballage et timbres).

En espérant que ceci contribuera à l'essor de l'assembleur sur HP-71...

Pierre David (37)
Janick Taillandier (246)

MODULE DE DEVELOPPEMENT POUR LE HP-71B

Jan Buitenhuis et Fred van der Vindt, de Hollande, développent actuellement un module pour le HP-71. Baptisé Development Rom, *celui-ci contiendra, outre le désassembleur décrit ci-dessous, un assembleur, un éditeur de liens, un émulateur et un éditeur pleine page. Le désassembleur est pratiquement terminé et l'assembleur devrait être bien avancé à la fin de l'année. Ce module devrait être disponible commercialement l'an prochain.*

Le désassembleur est écrit entièrement en assembleur, ainsi que tous les autres composants du module. Il crée un nouveau type de fichier : SYM. Ces fichiers contiennent les noms des symboles et leur valeur, et sont utilisés par le désassembleur et l'assembleur.

Le fichier HP71SYM compris dans le module comprend tous les points d'entrée supportés du système d'exploitation. Nous avons d'ailleurs trouvé quelques erreurs de HP qui seront, bien entendu, corrigées. En attendant, les voici :

SPLITA est en #0C6BF et non #0C68F
STAB1 est en #0D3D9 et non #0D309

D'autres fichiers SYM contiendront les symboles locaux et leur valeur et seront créés par l'assembleur.

La syntaxe de la fonction de désassemblage est la suivante :

DISASSEMBLE$(*str1* [, *str2* [, *str3*]])

str1 : offset
str2 : base
str3 : fichier SYM

(Les paramètres ne peuvent pas être des fonctions utilisateur)

Le premier argument, *offset*, précise l'adresse de début de désassemblage sous forme d'un déplacement par rapport à *base*. Si cette adresse de base n'est pas précisée, elle vaut "00000". Dans ce cas, les adresses données par le désassembleur seront absolues. Si *base* est spécifiée, par exemple l'adresse de début d'un fichier Lex, alors les adresses données par le désassembleur seront relatives à *base*. Si *offset* est passé par référence, alors cet argument sera automatiquement incrémenté pour pointer sur l'instruction suivante. Si *offset* est passé par valeur, il ne sera pas modifié. Ceci peut se résumer ainsi :

```
DISASSEMBLE$(A$)       référence : A$ est incrémenté
DISASSEMBLE$((A$))     valeur : A$ n'est pas modifié
DISASSEMBLE$("00000") valeur : pas de changement
```

Toutes les adresses référencées par des instructions telles que GOTO, GOYES ou GOSBVL sont recherchées dans un fichier SYM. Si l'adresse est trouvée, le nom du symbole sera utilisé plutôt que sa valeur. Les références absolues (GOSBVL et GOVLNG) sont cherchées dans le fichier HP71SYM.

Les références relatives, c'est à dire toutes les autres, sont cherchées dans le fichier SYM donné comme troisième paramètre. S'il n'y a pas de troisième paramêtre et si *base* vaut 0, le fichier HP71SYM est utilisé.

DISASSEMBLE$ retourne une chaîne de la forme suivante :

```
00000 COLDST P=     0
```

DISASSEMBLE$ peut être utilisé dans un petit programme de la manière suivante :

```
10 INPUT "Adresse début : ","?";A$
20 INPUT "Adresse fin : ","?";E$
30 DISP DISASSEMBLE$(A$) @ IF A$<E$ THEN 30
40 END
```

Exécutons ce programme entre 08A48 et 08A70. Ceci donne :

```
08A48 NXTSTM ST=0   1
08A4B        D0=(5) #2F679
08A52        A=DAT0 A
08A55        D0=A
08A58        A=0    A
08A5A        A=DAT0 B
08A5D        CD0EX
08A60        A=A+C  A
08A62        D0=A
08A65        GOLONG RUNRTN
08A6B TKSCN+ D0=D0+ 2
08A6E        D0=D0+ 4
```

Il y a quelques différences entre la sortie de notre assembleur et ce que fait HP.

Nous utiliserons dans notre assembleur certaines mnémoniques :

```
?HST=0 %...    Test le bit de statut matériel
HST=0 %...     Remet à 0 les bits de statut
               (%... est un masque binaire)
ASLB           décale A à gauche d'un bit
BSLB           décale B à gauche d'un bit
CSLB           décale C à gauche d'un bit
DSLB           décale D à gauche d'un bit
```

Ces quatre dernières opérations existent mais ne sont pas documentées.

Ce désassembleur peut être utilisé pour créer un fichier source à partir du Lex pour modification et assemblage immédiat.

Quelques mots pour décrire l'assembleur et l'éditeur de liens :

- rapide : au moins 30 fois plus rapide que l'assembleur Forth.

- le source peut être divisé en plusieurs parties assemblées séparément. L'assembleur crée un fichier OBJ à partir de chaque source. Ces fichiers OBJ sont réunis par l'éditeur de liens en un fichier LEX ou BIN. Si vous changez un des sources, il suffit d'assembler seulement ce fichier, d'où un gain de temps énorme.

- L'assembleur connaît tous les points d'entrée du système : plus besoin d'EQU pour ceux-ci.

- L'assembleur crée un fichier SYM contenant tous les labels locaux. Ce fichier peut-être utilisé par le désassembleur et l'émulateur.

- Les erreurs sont données à la fin du listage. Si une erreur fatale se produit, le fichier OBJ ne sera pas créé mais l'assemblage continuera pour détecter toutes les erreurs.

- Si la pseudo-opération OBJ ne figure pas en tête du source l'objet ne sera pas créé. L'assembleur analysera le fichier source et donnera toutes les erreurs de syntaxe.

- La pseudo-opération LIST *fichier* permet de définir le fichier de listage.

- Les sources peuvent être localisés sur des mémoires externes (disques...). Le listage peut-être envoyé dans un fichier (interne ou externe) ou sur un appareil HP-IL.

- Le fichier OBJ est crée en mémoire principale.

- La génération automatique d'une *speed table* est possible grace à la pseudo-opération SPEED.

- Une pseudo-opération SUB fournit un moyen simple d'implémenter des sous-programmes dans un fichier Bin.

- Pas de FORTH (c'est un avantage).

- Pas de WORD (c'est un autre avantage).

- Pas de WORDI (c'est encore un autre avantage déterminant).

Toutes les suggestions sont les bienvenues. Nous travaillons maintenant à l'assembleur - ne tardez pas trop - qui ne devrait pas être beaucoup plus long à faire que le désassembleur l'a été. Nous pouvons utiliser pour l'assembleur beaucoup de code et de tables développés pour le désassembleur.

Jan Buitenhuis (296)
Fred van der Windt

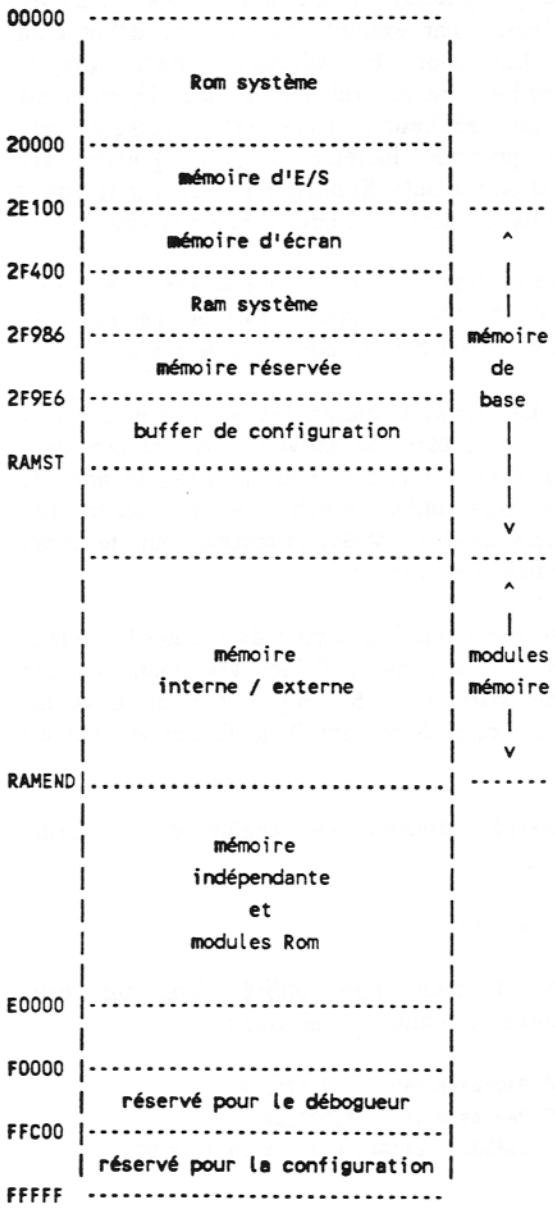
L'ASSEMBLEUR DU HP-71
(ACTE III)

Voici donc la suite de notre initiation à l'assembleur du HP-71. La dernière fois, nous avions commencé à manipuler les registres du microprocesseur et à utiliser quelques unes de ses instructions.

Aujourd'hui, nous allons continuer notre progression en accédant à la mémoire. Je rappelle que le compteur d'adresse du HP-71 a une longueur de 20 bits c'est à dire qu'il peut adresser 2^{20} cases mémoire, soit un méga quartets, puisque l'unité de mesure en ce qui concerne notre ordinateur est le quartet (ou *nibble*). On a donc à notre disposition 512 Ko.

Avant de commencer à voir quelles sont les instructions qui permettent d'avoir accès à cette mémoire, nous allons décrire l'organisation de la mémoire. Pour cela, il suffit de lire le schéma ci-après :

Organisation de la mémoire du HP-71 :



Pour pouvoir utiliser la mémoire de notre machine, le jeu d'instructions du microprocesseur doit posséder deux types d'instructions : celles de lecture et celles d'écriture. Bien entendu, ces instructions sont disponibles sur le HP-71, et c'est ce que nous allons étudier maintenant :

D'une manière générale, ces instructions utilisent les pointeurs d'adresse D0 et D1 qui indiquent l'endroit où se fera la lecture ou l'écriture.

Instructions de lecture :

```
A=DAT0 fs
C=DAT0 fs
A=DAT1 fs
C=DAT1 fs
```

avec *fs* = A, P, WP, XS, X, S, M, B, W ou un nombre *n* compris entre 1 et 16.

Ces instructions chargent un certain nombre de quartets pointés par D0 ou D1 dans les registres A ou C. Elles permettent donc de remplir un champ d'un des accumulateurs en un seule opération. Si *fs* est un nombre *n* entre 1 et 16, l'instruction chargera *n* quartets à partir de l'adresse contenue dans D0 ou D1 et les mettra de droite à gauche en partant de la position 0. Pour comprendre comment cela fonctionne, rien de tel que des exemples :

Si le contenu des registres est :

```
A  = FE20589ECD5AA88D
C  = 15F2EAC58DDE36A0
     [<-- M --->]

D0 = 2E350 -> 8001400220041008
D1 = 00000 -> 2034EE100060F481
```

L'exécution de `C=DAT0 M` donne :

```
C = 14002200410086A0
    [<-- M --->]
```

et `A=DAT1 7` conduit à :

```
A = FE20589EC1EE4302
```

Instructions d'écriture :

```
DAT0=A fs
DAT0=C fs
DAT1=A fs
DAT1=C fs
```

avec *fs* = A, P, WP, XS, X, S, M, B, W ou un nombre *n* compris entre 1 et 16.

Ces instructions permettent d'écrire un champ (ou un nombre de quartets à partir de la position 0) de A ou C en mémoire à partir de l'adresse pointée par D0 ou D1.

Par exemple, si le contenu des registres est :

```
A  = 0123456789ABCDEF
C  = FFFFFFFFFFFFFFFF

D0 = 31500 -> 00000000000000000000
D1 = 316F0 -> 00000000000000000000
```

L'exécution de `DAT0=C W` donne :

```
D0 = 31500 -> FFFFFFFFFFFFFFFF0000
```

et `DAT1=A 3` conduit à :

```
D1 = 316F0 -> FED000000000000000000
```

Nous allons maintenant étudier les instructions de manipulation de registres D0 et D1.

Manipulation des registres D0 et D1

Il existe de neuf sortes d'instructions énumérées ci-dessous avec *r* = A ou C et *ss* = D0 ou D1.

```
rssEX        échange le champ A de r avec ss
rssSX        échange les quartets 0 à 3
             de r avec ceux de ss
ss=r         copie le champ A de r dans ss
ss=rS        idem mais pour les quartets 0 à 3
ss=ss+ n     ajoute n à ss (n < 16)
ss=ss- n     soustrait n à ss (n < 16)

ss=(2) nnnnn charge les 2 quartets de poids
             faible de ss avec ceux de nnnnn.
ss=(4) nnnnn idem mais pour 4 quartets.
ss=(5) nnnnn copie nnnnn dans ss.
```

Il faut signaler que les trois dernières instructions peuvent s'écrire `ss=HEX hh`, `ss=HEX hhhh`, `ss=HEX hhhhh` avec *h* compris entre 0 et F.

Exemple d'application

Comme application des accès mémoire, nous allons écrire un programme binaire (Bin) qui se contentera de noircir une zone de l'écran. Dans le volume I des *IDS* (pages 3.1 et suivantes), nous allons trouver les adresses des zones écran. Je dis *des* car HP a bien fait les choses en plaçant l'écran en trois parties dans la mémoire, comme suit :

```
zone 1 : colonnes  0 à  45 : de 2E104 à 2E15F
zone 2 : colonnes 46 à  93 : de 2E200 à 2E260
zone 3 : colonnes 94 à 131 : de 2E300 à 2E34C
```

Pour afficher quelque chose à l'écran il suffit de mettre à 1 les bits qui correspondent aux points que l'on veut allumer. Le programme suivant se contente de mettre à 1 tous les bits des 4 premières colonnes de la zone 2.

```
        BIN     'SCREEN'
        CHAIN   -1

DD2ST   EQU     #2E200    début de la zone 2
DD2END  EQU     #2E260    fin de cette zone
barre   EQU     4         nombre de barres à afficher
ENDBIN  EQU     #0764B

debut   D0=(5)  DD2ST     pointe la zone
        A=0     B         A[B] = 00
        A=A-1   B         A[B] = FF
        P=      0         chargement à partir de C[0]
        LC(5)   barre     compteur de barre

boucle  DAT0=A  B         remplit un octet
        D0=D0+  2         pointe le suivant
        C=C-1   A         décremente le compteur
        ?C#0    A         C[A] est-il non nul ?
        GOYES   boucle    si oui on recommence
fin     GOVLNG  ENDBIN    si non on arrête

        END
```

Je vous laisse assembler ce fichier... Pour l'essayer, il suffit de faire comme pour tout autre fichier de son type : `RUN SCREEN` ou `CALL SCREEN`. Vous pourrez voir alors très brièvement s'afficher le résultat attendu. Le seul problème est que cet affichage est pour le moins fugitif. Pour supprimer ce problème, il suffit de faire `CALL SCREEN @ WAIT 1`, par exemple, ou mieux de taper le programme qui suit :

```
10 CALL SCREEN
20 GOTO 10
```

Nous allons maintenant améliorer ce programme à trois niveaux :

- nous allons remplir la zone 2 totalement, et non simplement quelques barres,

- l'écriture dans la mémoire d'écran se fera 8 octets à la fois comme le permet le HP-71, et non un par un,

- et enfin nous ajouterons à la fin une boucle d'attente qui permettra au résultat de rester plus longtemps à l'affichage.

On obtient alors le source suivant :

```
        BIN     'SCREEN2'
        CHAIN   -1
DD2ST   EQU     #2E200
DD2END  EQU     #2E260
long    EQU     (DD2END)-(DD2ST) longueur de la zone
ENDBIN  EQU     #0764B
duree   EQU     136000    environ 5 secondes

debut   D0=(5)  DD2ST
        A=0     W
        A=A-1   W         ici A[W] = FF....FF
        P=      0
        LC(5)   long
        CSR     A         division par 16

boucle  DAT0=A  W         remplit 64 quartets
        D0=D0+  16
        C=C-1   A
        ?C#0    A
        GOYES   boucle

        LC(5)   duree
wait    C=C-1   A
        ?C#0    A
        GOYES   wait
fin     GOVLNG  ENDBIN
        END
```

Qu'y a-t'il de nouveau par rapport au précédent programme ?

La valeur *long* sera égale à (2E260 - 2E200) soit 60 en hexadécimal, ce qui représente 96 quartets (ou encore 48 octets soit 48 barres affichées). Le nombre de boucles à effectuer est égal à *long* / 16 puisque l'on écrit 16 quartets à la fois avec l'instruction `DAT0=C W`. Pour diviser ou pour multiplier le contenu d'un registre par une puissance de deux, rien de plus facile : il suffit de décaler tous les bits du registre respectivement à droite ou à gauche. Pour diviser C[A] par 16, il faut donc faire un décalage de 4 bits ($2^4 = 16$) à droite . Cela se fait grâce à l'instruction `CSR A` (pour *C Shift Right A*) qui décale C[A] d'un quartet à droite.

Etudions maintenant la seconde partie du programme. Le but est simplement de faire faire quelque chose au microprocesseur pour laisser l'affichage intact. La chose la plus simple qu'on puisse lui faire faire est compter. Le type de boucle suivant convient alors parfaitement :

```
        LC(5)   duree
label   C=C-1   A
        ?C#0    A
        GOYES   label
```

Quelle valeur doit-on alors assigner à *durée* ? Il suffit de se référer aux *IDS* (volume I) qui donnent le temps d'exécution de chaque instruction. On y aprend que la boucle ci-dessus dure 25 cycles d'horloge par tour, soit 37 microsecondes environ. Pour attendre 5 secondes, il faut donc environ compter jusqu'à 136000. Qui a dit que le HP-71 était lent ? De 0 à 136000 en 5 secondes, qui dit mieux ? Essayez donc en Basic, vous m'en direz des nouvelles...

Pour ceux qui s'y connaissent déjà en assembleur, je dirai que ce programme peut être optimisé au niveau vitesse et longueur du code. En effet, il faut savoir que la boucle ci-dessus peut être remplacée par la séquence suivante. Elle prend 17 cycles au lieu de 25, et 5 quartets au lieu de 7.

```
        LC(5) (duree)-1
label   C=C-1 A
        GONC  label
```

L'instruction GONC (pour *Go On No Carry*) effectue un saut à l'adresse indiquée si le drapeau *carry* est à 0. Cette *carry* est un bit que le microprocesseur positionne en fonction du résultat de l'opération qu'il vient d'effectuer. Par exemple, pour C=C-1 A, la *carry* passe à un si C passe de 00000 à FFFFF. On peut donc réécrire le programme comme suit :

```
        BIN     'SCREEN3'
        CHAIN   -1

DD2ST   EQU     #2E200
DD2END  EQU     #2E260
long    EQU     (DD2END)-(DD2ST) longueur de la zone
ENDBIN  EQU     #0764B
duree   EQU     200000   environ 5 secondes

debut   DO=(5) DD2ST
        A=0     W
        A=A-1   W        ici A[W] = FF....FF
        P=      0
        LC(5)   long
        CSR     A        division par 16

boucle  DAT0=A W
        DO=DO+ 16
        C=C-1   A
        GONC    boucle

        LC(5)   duree
wait    C=C-1   A
        GONC    wait

fin     GOVLNG ENDBIN
        END
```

Ce dernier programme vous permettre de gagner 2 octets par rapport à son prédécesseur. Fantastique non ?

Voici venu le moment de nous quitter, hélas !
Pour vous donner envie de lire mon prochain article, je vous annonce que nous verrons comment écrire notre premier Lex. D'ici là, n'abusez pas de HP-71...

Jacques Baudier (192)

INVERSEZ VOS DRAPEAUX

Ce fichier Lex vous fournit une fonction pour inverser l'état des drapeaux (ou flags, ou indicateurs) système. Avant, pour inverser l'état d'un flag, il fallait faire :

```
DEF KEY'%','N=VAL(DISP$)@N=FLAG(N,NOTFLAG(N))':
```

Les indicateurs inversables sont ceux compris entre -1 et -59. Ceci afin de limiter les risques de Memory Lost.

Cette fonction résulte plus d'un processus d'apprentissage que d'un besoin réel. C'était un de mes premiers essais en assembleur. Maintenant, je peux voir que cette fonction pourrait être améliorée. Par exemple, le complément en hexadécimal pourrait être fait avec :

```
        C=A     B
        A=-A-1  B
```

Et ainsi de suite. Il est possible qu'un jour cette fonction inverse l'état des indicateurs utilisateur (ceux qui sont positifs).

Bonnes inversions,

Jack Elhay

```
        LEX     'TFLAGS'
        ID      #5C
        MSG     0
        POLL    0
        ENTRY   IN
        CHAR    #F
        KEY     'TFLAG'  Syntaxe :  TFLAG (-n), 1<=n<=59
        TOKEN   12       Retourne l'ancienne valeur
POP1R   EQU     #0E8FD   dépile un réel
FLTDH   EQU     #1B223   convertit de flottant en hexa
HDFLT   EQU     #1B31B   inverse de la précédente
SFLAGT  EQU     #13608   routine utilisée par LC, USER
FNRTN1  EQU     #0F216
ARGERR  EQU     #0BF19
        ENDTXT

* Un paramètre numérique obligatoire
        NIBHEX 811
IN      GOSBVL POP1R     dépile un nombre réel
        A=0    S         efface le signe
        GOSBVL FLTDH     réel A(W) -> hexa A(A)
* A(A) = nombre dépilé. La fonction commence
        C=0    W
        LC(2)  #3C       #3C=60 en décimal
        ?C<=A  B         interdit les flags -60 à -64
        GOYES  ERROR
```

```
        C=0     A
        LC(3)   #100      No = complément à #100
        C=C-A   X
        GOSBVL  SFLAGT    Inverse. Erreur si flag user
* Fin de la fonction. Prépare la sortie
        A=0     W
        GONC    OUT       Cy = l'état antérieur du flag
        A=A+1   A         +1 si flag était à 1
OUT     GOSBVL  HDFLT     convertit hexa -> réel A(W)
        C=A     W         réel va dans C(W)
        GOVLNG  FNRTN1    retour...
ERROR   GOVLNG  ARGERR
        END               ouf !
```

UNPURGE

Voici un petit utilitaire pour les distraits, en Basic une fois n'est pas coutume, et qui illustre deux des mots-clés les plus puissants de JPCLEX (tu parles ! ils le sont tous !) : `RREC$` et `WREC`.

Vous venez d'exécuter, proprement et définitivement, une de vos oeuvres : la peine de mort n'est pas encore supprimée pour les fichiers, hélas ! Et vous vous avisez que, oh m... ! Ne touchez à rien, ne copiez rien sur votre disquette ou cassette, et lancez UNPURGE, qui vous ressuscitera votre défunt fichier...

Observez au passage combien il est reposant d'écrire `R$=RREC$(...))` plutôt que la litanie des `SEND LISTEN ... MTA ...` et autres protocoles dits de bas niveau...

J'insiste, pour finir, sur le fait que UNPURGE ne pourra *rien* pour les fichiers en Ram, ou si vous avez compacté, initialisé, ou copié sur votre support. Il se contente en effet de retrouver l'entrée du fichier détruit dans le catalogue du support, et de reconstituer l'octet *type de fichier* qui avait été transformé en nul.

Happy unPurging !

Eric Gengoux (108)

Programme "UNPURGE" (Retrouvez vos fichiers ! Necessite JPCLEX)

```
   - UNPURGE (utilise JPCLEX)
     (c) Eric Gengoux
  10 CALL UNPURGE @ SUB UNPURGE
  20 DIM R$[256],F0$[10],F1$[10],T$[2]
  30 INPUT "Unité ",":TAPE(1)";U$
  40 IF NOT DEVADDR(U$) THEN BEEP @ GOTO 30
  50 INPUT "Fich. purgé ? ";F0$ @ F0$=UPRC$(F0$)
  60 F0$=F0$&SPACE$(10-LEN(F0$))
  70 RESTORE 80 @ ON MENU(7,1) GOTO 'BAS','LEX','BIN','DTA','KEY','SDA','TXT'

================================================================
  80 DATA BASIC,LEX,BINAIRE,DATA,KEY,SDATA,TEXT

================================================================
  90 'BAS': T1$=HTA$("E214",1) @ GOTO 'CONT'

================================================================
 100 'LEX': T1$=HTA$("E208",1) @ GOTO 'CONT'

================================================================
 110 'BIN': T1$=HTA$("E204",1) @ GOTO 'CONT'

================================================================
 120 'DTA': T1$=HTA$("E0F0",1) @ GOTO 'CONT'

================================================================
 130 'KEY': T1$=HTA$("E20C",1) @ GOTO 'CONT'

================================================================
 140 'SDA': T1$=HTA$("E0D0",1) @ GOTO 'CONT'

================================================================
 150 'TXT': T1$=HTA$("0001",1) @ GOTO 'CONT'

================================================================
 160 'CONT': FOR I=2 TO MAXD(U$)/8+2
 170 R$=RREC$(I,U$)
 180 FOR J=0 TO 254 STEP 32
 190 F2$=R$[J+1,J+10] @ T$=R$[J+11,J+12]
 200 IF F2$#F0$ THEN GOTO 230
 210 IF T$#HTA$('0000') THEN EXIT J
 220 R$[J+11,J+12]=T1$ @ WREC R$,I,U$
 230 NEXT J
 240 NEXT I
 250 CAT F0$&U$
```

****

Programme "GAO" (Genealogie assistee par ordinateur, deuxieme partie, voir JPC 46)

```
   - ****************************************
     *                                      *
```

```
    *   GENEALOGIE                      *
    *                                   *
    *   NICODEME philippe               *
    *                                   *
    *   01/03/1986                      *
    *                                   *
    ***************************************

    ***************************************
    *          2ème PARTIE                *
    *programme principal, enregistré sous *
    *   le label GAO. Fonctionnement      *
    *indépendant de la première partie     *
    ***************************************

    ************************
    *   INITIALISATION     *
    ************************
 100 DESTROY ALL @ CFLAG ALL @ DELAY 1,1 @ USER @ DISP TAB(7);"GENEALOGIE"
  . ************************
    *   MENU PRINCIPAL     *
    ************************
 110 A=101 @ Z=105 @ P=A @ SFLAG 0 @ CALL L(A$,A,Z,P) @ CFLAG 0 @ IF P=101 THEN 110
 120 IF P<>105 THEN CALL A$ @ GOTO 110 ELSE USER @ DISP "au revoir"
  . ************************
    *   MENU FICHIER       *
    ************************
    *   menu principal     *
    ************************

=============================================================
 130 SUB FICHIER @ SFLAG 1
 140 A=168 @ Z=176 @ P=A @ CALL L(A$,A,Z,P) @ IF P=168 THEN 140
 150 IF P=175 THEN A=198 @ P=A @ Z=221 @ CALL L(A$,A,Z,P) @ GOTO 140
 160 IF P=176 THEN CFLAG 1 @ END
 170 ASSIGN #1 TO FIC @ RESTORE #1,0 @ READ #1;Q @ ASSIGN #1 TO * @ CALL A$(Q) @ GOTO 140
  . ************************
    *   MENU EDITER        *
    ************************
    *      fichier         *
    ************************

=============================================================
 180 SUB EDITER(Q) @ A=177 @ Z=183 @ P=A
 190 CALL L(A$,A,Z,P) @ IF P=177 THEN 190
 200 IF P=183 THEN END ELSE N$=CHR$(NUM(A$))&"I" @ CALL N$(Q) @ GOTO 190
  . ************************
    *  OPTION EDITER       *
    *  UNE FICHE           *
    ************************
    *      fichier         *
    ************************

=============================================================
 210 SUB UI(Q) @ CALL Q(E$,F$,P,Q)
  . ************************
    *MODULE DE RECHERCHE   *
    *TROUVE LE No D'UNE    *
```

```
     *FICHE A PARTIR DU   *
     *NOM ET DU PRENOM    *
     ************************

============================================================
 220 SUB Q(E$,F$,P,Q) @ ASSIGN #1 TO FIC @ IF P<>-1 THEN INPUT "nom? ";E$ @ INPUT "prénom? ";F$
 230 K=0 @ FOR I=1 TO Q-1 STEP 12 @ RESTORE #1,I @ READ #1;A$ @ RESTORE #1,I+1 @ READ #1;B$
 240 IF A$=E$ AND F$=B$ THEN K=I @ P=I
 250 NEXT I
 260 IF K=0 THEN DISP "inconnu" ELSE ASSIGN #1 TO * @ SFLAG 8 @ CALL B(K,Q)
   - ************************
     *  OPTION EDITER      *
     *  AVEC FAMILLE       *
     ************************
     *       fichier       *
     ************************

============================================================
 270 SUB AI(Q) @ P=0 @ PRINT @ PRINT "FAMILLE" @ CALL Q(E$,F$,P,Q) @ IF P=0 THEN END
 280 PRINT @ PRINT "PERE" @ P=P+6 @ CALL O(P,Q) @ PRINT @ PRINT "MERE" @ P=P+2 @ CALL O(P,Q)
 290 PRINT @ PRINT "CONJOINT" @ P=P+2 @ CALL O(P,Q)
 300 PRINT @ PRINT "ENFANTS" @ SFLAG 5 @ P=P-10 @ CALL O(P,Q) @ SFLAG 12 @ CALL O(P,Q)
   - ************************
     *LECTURE DE 2         *
     *ELEMENTS DU FICHIER  *
     *MAN (les éléments    *
     *sont pointés par P)  *
     ************************

============================================================
 310 SUB O(P,Q) @ ASSIGN #1 TO FIC @ RESTORE #1,P @ READ #1;C$(1) @ RESTORE #1,P+1
 320 READ #1;C$(2) @ D=1 @ J=1 @ ASSIGN #1 TO * @ IF FLAG(5,0) THEN D=7
 330 IF FLAG(12,0) THEN D=9
 340 CALL P(C$(),D,J,Q)
   - ************************
     *  OPTION EDITER      *
     *  LE FICHIER         *
     ************************
     *       fichier       *
     ************************

============================================================
 350 SUB LI(Q) @ PRINT @ PRINT "FICHIER" @ PRINT DATE$ @ PRINT TIME$ @ PRINT
 360 FOR I=1 TO Q-1 STEP 12 @ K=I @ SFLAG 8 @ CALL B(K,Q) @ NEXT I
   - ************************
     *  OPTION EDITER      *
     *  PAR CRITERES       *
     ************************
     *       fichier       *
     ************************

============================================================
 370 SUB PI(Q)
 380 INPUT "1 ou 2 critères ? ";N @ IF N<1 OR N>2 THEN 380
 390 IF N=2 THEN INPUT "liaison ET / OU ? ";A$
 400 IF A$="OU" THEN SFLAG 6
 410 FOR I=1 TO N @ A=184 @ P=A @ Z=197
 420 CALL L(A$,A,Z,P) @ C(I)=P-185 @ IF C(I)<1 THEN 420
```

```
430 INPUT "valeur ? ";C$(I) @ NEXT I @ D=C(1) @ J=C(2)-C(1) @ IF N=1 THEN C$(2)=C$(1) @ J=0
440 CALL P(C$(),D,J,Q)
  - **********************
    *MODULE DE RECHERCHE *
    *ET D'EDITION D'UNE  *
    *FICHE A PARTIR DE 1 *
    *OU 2 CRITERES       *
    **********************

================================================================
450 SUB P(C$(),D,J,Q) @ FOR I=D TO Q-1 STEP 12 @ ASSIGN #1 TO FIC @ L=L+1
460 RESTORE #1,I @ READ #1;A$ @ T=0 @ IF A$=C$(1) THEN T=1
470 RESTORE #1,I+J @ READ #1;A$ @ IF A$=C$(2) THEN T=T+2
480 ASSIGN #1 TO *
490 IF FLAG(6) AND T<>0 OR T=3 THEN SFLAG 8 @ K=L*12-11 @ CALL B(K,Q)
500 NEXT I @ CFLAG 6
  - **********************
    *  OPTION EDITER      *
    *  EN ARBRE           *
    **********************
    *      fichier       *
    **********************

================================================================
510 SUB EI(Q) @ PRINT @ PRINT "ARBRE" @ INPUT "(P)ère (M)ère ? ";A$ @ P=0
520 CALL Q(E$,F$,P,Q) @ IF P<1 THEN END
530 IF A$="P" THEN P=P+6 ELSE P=P+8
540 ASSIGN #1 TO FIC @ RESTORE #1,P @ READ #1;E$ @ RESTORE #1,P+1
550 READ #1;F$ @ ASSIGN #1 TO * @ P=-1 @ GOTO 520
  - **********************
    *  OPTION SAUVER      *
    **********************
    *      fichier       *
    **********************

================================================================
560 SUB SAUVER(Q) @ COPY FIC TO :TAPE(1)
  - **********************
    * OPTION CHARGER      *
    **********************
    *      fichier       *
    **********************

================================================================
570 SUB CHARGER(Q) @ PURGE FIC @ COPY FIC:TAPE(1)
  - **********************
    *  OPTION MODIFIER    *
    **********************
    *      fichier       *
    **********************

================================================================
580 SUB MODIFIER(Q)
590 INPUT "No ? ";N @ IF N>IP(Q/12)+1 THEN 590 ELSE R=Q @ Q=1+12*(N-1)
600 K=Q @ CALL B(K,Q) @ Q=R
  - **********************
    *  OPTION EFFACER     *
    **********************
```

```
*      fichier       *
************************

==============================================================
610 SUB EFFACER(Q) @ INPUT "confirmer ? O/N ";A$ @ IF A$<>"O" THEN END
620 ASSIGN #1 TO FIC @ RESTORE #1,0 @ PRINT #1;1 @ ASSIGN #1 TO *
  - ************************
    *       MODULE         *
    *  ENTREE / SORTIE     *
    *selon le flag 8       *
    *édition d'une fiche   *
    *entrée  d'une fiche   *
    ************************

==============================================================
630 SUB B(K,Q) @ DELAY 0,0 @ ASSIGN #1 TO MAN @ ASSIGN #2 TO FIC @ DIM E$[15]
640 C$="**********" @ IF FLAG(8) THEN PRINT @ PRINT C$ @ PRINT "FICHE";IP(K/12)+1 @ PRINT C$
650 ON ERROR GOTO 680
660 FOR I=186 TO 197 @ RESTORE #1,I @ RESTORE #2,K @ READ #1;A$
670 IF FLAG(8) THEN READ #2;B$ @ PRINT A$;TAB(11);B$ @ GOTO 690
680 DISP A$ @ WINDOW LEN(A$)+2 @ INPUT E$ @ PRINT #2;E$ @ WINDOW 1 @ Q=Q+1
690 K=K+1 @ NEXT I @ ASSIGN #1 TO * @ ASSIGN #2 TO * @ CFLAG 8
  - ************************
    *  OPTION AJOUTER      *
    ************************
    *      fichier         *
    ************************

==============================================================
700 SUB AJOUTER(Q) @ IF Q=840 THEN DISP "Eof" @ END ELSE K=Q @ SFLAG 20 @ CALL B(K,Q)
710 ASSIGN #1 TO FIC
720 RESTORE #1,0 @ PRINT #1;Q @ ASSIGN #1 TO * @ DISP (Q-1)/12;"fiches" @ ASSIGN #1 TO *
  - ************************
    *   MENU MOTES         *
    ************************
    *   menu principal     *
    ************************

==============================================================
730 SUB NOTES @ SFLAG 2
740 A=132 @ P=A @ Z=138 @ CALL L(A$,A,Z,P) @ IF P=132 THEN 740
750 IF P=138 THEN CFLAG 2 @ END
760 ASSIGN #1 TO MAN @ RESTORE #1,0 @ READ #1;T @ ASSIGN #1 TO * @ CALL A$(T) @ GOTO 740
  - ************************
    *   OPTION AIDE        *
    ************************
    *       notes          *
    ************************

==============================================================
770 SUB AIDE(T) @ A=139 @ P=A @ Z=154 @ CALL L(A$,A,Z,P)
  - ************************
    *   OPTION LECTURE     *
    ************************
    *       notes          *
    ************************

==============================================================
```

```
780 SUB LIRE(T) @ A=1 @ P=A @ Z=T-(T<>1) @ CALL L(A$,A,Z,P)
  - **********************
    *   OPTION ECRITURE  *
    **********************
    *       notes        *
    **********************

================================================================
790 SUB ECRIRE(T) @ ASSIGN #1 TO MAN @ DIM A$[20] @ ON ERROR GOTO 820 @ DELAY 1,1
800 INPUT A$ @ IF A$="F" THEN 830
810 IF T=100 THEN DISP "Eof" @ GOTO 830 ELSE RESTORE #1,T @ PRINT #1;A$ @ T=T+1 @ GOTO 800
820 DISP "trop grand" @ GOTO 800
830 RESTORE #1,0 @ PRINT #1;T @ ASSIGN #1 TO *
  - **********************
    *   OPTION DETRUIRE  *
    **********************
    *       notes        *
    **********************

================================================================
840 SUB DETRUIRE(T) @ INPUT "confirmer DETRUIRE ? ";A$ @ IF A$="N" THEN END
850 ASSIGN #1 TO MAN @ RESTORE #1,0 @ PRINT #1;1 @ RESTORE #1,1 @ PRINT #1;"" @ ASSIGN #1 TO *
  - **********************
    *   OPTION IMPRIMER  *
    **********************
    *       notes        *
    **********************

================================================================
860 SUB IMPRIMER(T) @ PRINT @ PRINT "BLOC-NOTES" @ PRINT DATE$ @ PRINT @ ASSIGN #1 TO MAN
870 FOR I=1 TO T-(T<>1) @ RESTORE #1,I @ READ #1;A$ @ PRINT A$ @ NEXT I @ ASSIGN #1 TO *
  - **********************
    *   MENU DATEUR      *
    **********************
    *   menu principal   *
    **********************

================================================================
880 SUB DATEUR @ SFLAG 3
890 A=106 @ P=A @ Z=111 @ CALL L(A$,A,Z,P) @ IF P=106 THEN 890
900 IF P=110 THEN A=112 @ Z=131 @ P=A @ CALL L(A$,A,Z,P) @ GOTO 890
910 IF P<>111 THEN N$=CHR$(NUM(A$)) @ CALL N$ @ GOTO 890 ELSE CFLAG 3
  - **********************
    *  OPTION CALENDRIER *
    **********************
    *      dateur        *
    **********************
    Adaptation d'un programme paru dans l'Ordinateur de Poche No 20 (page 27)

================================================================
920 SUB C @ DELAY 2,2 @ DISP "entrez la date" @ DISP "max : 10 NIVOSE AN 14"
930 INPUT "jour ? ";P @ IF P<1 OR P>30 THEN 930
940 DISP "mois en toutes lettres" @ INPUT "mois ? ";N$ @ K=0 @ ASSIGN #1 TO MAN
950 FOR I=155 TO 167 @ RESTORE #1,I @ READ #1;A$ @ IF N$=A$ THEN K=I-154
960 NEXT I @ ASSIGN #1 TO * @ IF K=0 THEN 940
970 INPUT "an ? ";R @ IF R<1 OR R>14 THEN 970
980 IF K=13 AND P>6 THEN DISP C$ @ GOTO 930
990 IF K=13 AND P=6 AND R<>3 AND R<>7 AND R<>11 THEN DISP C$ @ GOTO 930
```

```
1000 IF K=4 AND R=14 THEN 1010 ELSE 1020
1010 IF P>10 THEN DISP C$ @ GOTO 930
1020 IF K=13 THEN 1240
1030 IF K<6 THEN 1040 ELSE 1080
1040 IF (R#4)*(R<8) THEN X=0 @ Y=0 @ GOTO 1060
1050 IF (R=4)+(R>7)*(R<>12) THEN X=-1 @ Y=1 ELSE X=-2 @ Y=2
1060 IF K<3 THEN L=K+8+X @ GOTO 1150
1070 IF (K>2)*(K<6) THEN L=K+7+X @ GOTO 1150
1080 IF R<8 THEN X=0 @ Y=0
1090 IF (R=4)*(K=6) THEN Y=1 @ GOTO 1120
1100 IF R>=8 THEN X=-1 @ Y=1
1110 IF (R=12)*(K=6) THEN Y=2
1120 IF (K=6)+(K=7) THEN L=K+4+X @ GOTO 1150
1130 IF (K=8)+(K=9) THEN L=K+3+X @ GOTO 1150
1140 IF K>9 THEN L=K+2+X
1150 IF P<=L THEN M=K+8 ELSE M=K+9
1160 IF M>12 THEN M=M-12
1170 IF P>L THEN J=P-L @ GOTO 1250
1180 IF K<3 THEN J=P+21+Y @ GOTO 1250
1190 IF (K=3)+(K=7) THEN J=P+20+Y @ GOTO 1250
1200 IF K=4 THEN J=P+20+Y @ A=R+1791 @ GOTO 1260
1210 IF (K=5)+(K=8)+(K=9) THEN J=P+19+Y @ GOTO 1250
1220 IF (K=6)+(K>9)*(K<12) THEN J=P+18+Y ELSE J=P+17+Y
1230 GOTO 1250
1240 K=13 @ M=9 @ IF R<=7 THEN J=P+16 ELSE J=P+17
1250 IF K>=3 THEN A=R+1792 ELSE A=R+1791
1260 CALL E(F,J,M,A)
   - **********************
     * OPTION JOUR SEM.  *
     **********************
     *     dateur        *
     **********************

================================================================
1270 SUB J @ DELAY 1,1 @ DISP "entrez la date" @ CALL E(F,J,M,A)
   - **********************
     * OPTION Nb DE JOURS*
     **********************
     *     dateur        *
     **********************

================================================================
1280 SUB N @ DELAY 1,1 @ DISP "date 1" @ CALL E(F,J,M,A) @ D=F @ DELAY 1,1
1290 DISP "Date 2" @ J=0 @ CALL E(F,J,M,A) @ DISP ABS(F-D);" jours"
   - **********************
     * MODULE DE         *
     * CALCUL DU JOUR    *
     **********************

================================================================
1300 SUB E(F,J,M,A) @ IF J<>0 THEN 1340 ELSE DISP "valeurs numériques SVP"
1310 INPUT "jour ? ";J @ IF J<1 OR J>31 THEN 1310
1320 INPUT "mois ? ";M @ IF M<1 OR M>12 THEN 1320
1330 INPUT "annee   >1582  ";A @ IF A<1583 THEN 1330
1340 F=365*A+J+31*(M-1)+INT((A-1)/4)-INT(.75*INT((A-1)/100+1))
1350 IF M>2 THEN F=365*A+J+31*(M-1)-INT(.4*M+2.3)+INT(A/4)-INT(.75*INT(A/100+1))
1360 D=ABS(F+(INT(-F/7)+1)*7) @ IF D>6 THEN D=0
```

```
================================================================
1370 DATA SAMEDI,DIMANCHE,LUNDI,MARDI,MERCREDI,JEUDI,VENDREDI

================================================================
1380 FOR I=0 TO D @ READ A$ @ NEXT I @ DELAY 3,3 @ DISP A$;J;M;A
   - **********************
     * MODULE DE LECTURE *
     * lecture des menus *
     * du fichier MAN    *
     **********************

================================================================
1390 SUB L(A$,A,Z,P) @ ASSIGN #1 TO MAN
   - calcul du signe placé en fin de ligne (déplacements possibles)
1400 IF P=A THEN S=128 ELSE S=130-(P=Z)
   - lecture dans le fichier MAN de l'élément pointé par P
     affichage de cet élément et indication du sens de lecture ^ v
1410 RESTORE #1,P @ READ #1;A$ @ DISP A$;TAB(22);CHR$(S)
   - attente de la pression d'une touche
     éventuellement affichage de la date
1420 DELAY 0,0 @ E$=KEY$ @ IF E$="f0" THEN DELAY 1,1 @ DISP DATE$
   - éventuellement affichage de l'heure
1430 IF E$="fL" THEN DELAY 1,1 @ DISP TIME$
   - modification du pointeur selon la touche enfoncée
1440 IF E$="#51" THEN P=P+(P<>Z) ! pointeur + 1
1450 IF E$="#50" THEN P=P-(P<>A) ! pointeur - 1
   - si touche ENDLINE enfoncée fin du module sinon suite
1460 IF E$="#38" THEN ASSIGN #1 TO * ELSE 1400
   - ************************************
     *     FIN DE LA SECONDE PARTIE     *
     ************************************
```

LE COIN DES LHEX

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

Rappelons ce qu'est un fichier Lex : c'est un programme pour le HP-71, en assembleur, qui apporte de nouvelles fonctions. Celles-ci sont utilisables directement, ou dans des programmes Basic.

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

Il suffit de recopier le petit programme basic "MAKELEX" ci-dessous, de le lancer et de recopier les codes du fichier Lex désiré. Quand vous avez fini, les nouvelles fonctions sont accessibles, après avoir éteint et rallumé votre HP-71.

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

Vous trouverz donc le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !") et le Lex TFLAGS de Jack Elhay.

```
CHARLEX

TFLAGS    TFLAG       XFN 092012
```

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

```
CHARLEX  ID#E1  624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E008421328078 6B8
002: 5E4001E000000000 9FC
003: FE0000000800001F D56
004: F31BF961400032BF 0E9
005: 38F14A11DB10AD23 483
006: 07D532BFB8FD7911 836
007: 11AD754D7A101743 BB9
008: 11014D1CB15D0000 F24
009: 71450375FF864834 2A1
00A: 5655581008355654 5F8
00B: 5810002455565870 945
00C: 0026555658700836 C97
00D: 5556581008364545 FED
00E: 4A30000A49724000 340
00F: 0808094A2C180814 6A9
010: A464242008355455 A03
011: 581000054C714000 D49
012: 0C3142404C700832 0A5
013: 41414A70002078A0 3FD
014: 2F30000000000000 728
015: 0000000000000000 A38
016: 0000000000000000 D48
017: 0000000000000000 058
018: 0000000000000000 368
019: 0000000000000000 678
01A: 0000000000000000 988
01B: 0000000000000000 C98
01C: 0000000000000000 FA8
01D: 0000000000000000 2B8
01E: 0000000000000000 5C8
01F: 0000000000000000 8D8
020: 0000000000000000 BE8
021: 000000000000080C F13
022: 1A28080008080A2C 27D
023: 180008040E340800 5C6
024: 08001E3018000000 900
025: 0000000000000000 C10
026: 0000000000000000 F20
027: 0000000000000000 230
028: 0201000000010200 546
029: 0000000201020000 85B
02A: 0001000100000002 B6F
02B: 0102010000000000 E83
02C: 0000000000000000 193
02D: 045E755142400101 4DF
02E: 0101010000000000 7F2
02F: 0000000000000000 B02
030: 0000070507000000 E25
031: 000000000834444C4 163
032: 444000790111 2D70 4C3
033: 050D750509700000 80D
034: 0D70000000384540 B50
035: 4020014E322E3140 EA4
036: 084E794142400000 1F4
037: 00000000002E4559 532
038: 3200000000000000 847
039: 0000000000000026 B5F
03A: 5556587008365556 EBE
03B: 5810083645464830 20F
03C: 0832414248700024 550
03D: 5655587008345655 8AD
03E: 5810083446454830 BFC
03F: 0C3042414C700024 F51
040: 5556587008355654 2AE
041: 5810083546444830 5FD
041: 5810083546444830 5FD
042: 0C3142404C700025 953
043: 5455587008355455 CAD
044: 5810083544454830 FFB
045: 0C3140414C700875 35D
046: 14141870000A4972 6AE
047: 40000E3159454E30 A0E
048: 0C7A0F7949400024 D86
049: 5554587000084A71 0E2
04A: 40000C523A262D10 443
04B: 0424587458400875 79A
04C: 1415187000094A70 AEA
04D: 4000083544454830 E2E
04E: 0C3140414C300C74 196
04F: 5655545000054C71 4ED
050: 40000            5E6

TFLAGS   ID#5C  66 octets

     0123456789ABCDEF sm

000: 4564C41474350202 356
001: 802E009421328078 6B1
002: 88000C5C0C000000 A0F
003: F710000000000000 D3D
004: 091000F94564C414 0A5
005: 74C01FF8118FDF8E 46F
006: 0AC08F322B1AF231 808
007: C39EEC2D232001B3 BA7
008: 28F80631AF0540E4 F32
009: 8FB13B1AF68D612F 2F1
00A: 08D91FB0F        50E
```

PPC PARIS CHAPTER

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

BULLETIN D'ADHESION

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

Profession ______________________________

Intérêts ______________________________

____

Matériel HP en votre possession ______________________________

____

____

Autre matériel informatique ______________________________

____

____

Comment avez-vous connu PPC Paris ? ______________________________

____

Que recherchez-vous au sein de PPC Paris ? ______________________________

____

____

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

Date |_|_|/|_|_|/19|_|_|

Signature, précédée de la mention "Lu et Approuvé" ______________

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

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

Eventuellement: je m'abonne à compter du |0|1|/|_|_|/19|_|_|

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

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

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

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

ENGLISH SUMMARY

JPC 47 - SEPTEMBER 1987

This new JPC contains a lot of different materials. For example, a report about the Copenhagen International Conference in August can be found on page 3. This conference was very interesting, and the author of the article was impressed by the quality of the organization, and the contents of the various speeches. After this, Michael Markov, from USA, give us a small tip improving one of the examples used in the article about FINPUT in JPC 43. Next, we have a letter from an Australian friend, Jack Elhay. He explains how he began to write assembly code reading JPC. On page 5, François Duret-Lamouroux thanks Jacques Baudier for his series of assembly language introductory articles.

We have two articles from Wlodek Mier-Jedrzejowicz (from HPCC) dedicated to Synthetic Programming on the HP-28C. Wlodek is probably the most remarquable expert of this subject. Next, we list all the HP-28C known bugs.

The HP-41 column begins with an article from Jack Elhay providing three useful programs to compute various probability distributions. On page 19, René Dine gives us a routine to find prime numbers.

In the HP-71 corner, you will find a new version of the plotter program from Xavier Bille. This new version is written entirely in assembly and in Forth.

Pierre David and Janick Taillandier present us the MS-DOS version of their super-fast cross-assembler for the HP-71. It is about 100 to 200 times faster then the Forth / Assembler module, and adds many new features. Good news ! It is free.

Jan Buitenhuis and Fred van der Windt, from Holland, give us a report of the current status of their Devrom : it is an assembler, linker, disassembler, and emulator module for the HP-71. It sounds very interesting... Starting page 26, you will find the third part of Jacques Baudier's series teaching assembly language. For the first time, there is a serious and long series of articles on this subject. The small function by Jack Elhay, on page 29, is simple, and easy to understand.

Finally, a very useful utility from Eric Gengoux, on page 30, recovers accidentally purged files. It heavily uses JPCLEX functions.

Until next month,

Happy Programming and JPC reading...