JPC

MAI 1986

NUMERO 34

Le numéro 35 FF.

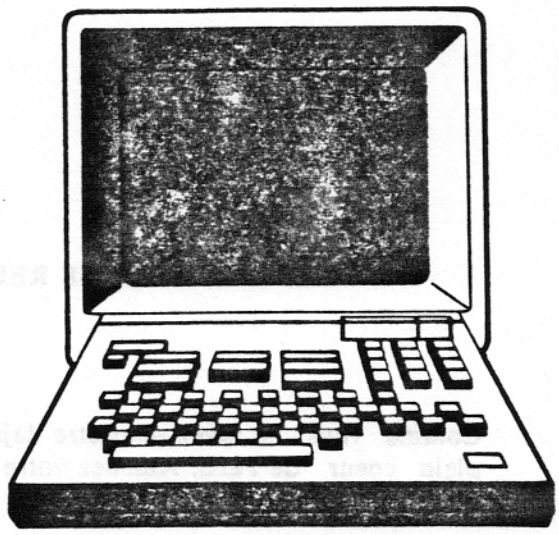
A PROPOS DU CLUB

ASSEMBLEUR

BASIC

FORTH

Edité par PPC-PC : 56, rue Jean-Jacques ROUSSEAU - 75001 PARIS



EDITORIAL

Chers amis,

Nous sommes envahis par les Nibbles. Vous les avez probablement rencontrés au détour d'une ligne ou d'une colonne, ces petits êtres se révèlent très encombrants, deux fois plus nombreux que les Octets, ils ne nous laissent plus.

Si vous parvenez à maitriser quelques uns d'entre-eux, lisez attentivement les conseils de la rubrique "AH ! Vous écrivez" et faites-nous part de vos découvertes.

Ce mois-ci un article HARD pour (enfin) favoriser la communication par HPIL.

Des excuses pour le dernier numéro dans lequel le poisson d'Avril a été trop bien reçu puisque beaucoup d'entre vous ont cru à l'arrivée d'un nouveau HP72c. Espérons qu'un jour ce ne soit pas un bobard.

Le mois prochain, c'est bientôt pour celui qui réalise la maquette, nous publierons la liste officielle des points d'entrée dans le système d'exploitation telle qu'HP vient de la redéfinir.

Puis ce sera les vacances.....

J.J. DHENIN

PPC PARIS SE REUNIT UNE FOIS PAR MOIS

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

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

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

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

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

La date de la prochaine réunion :
Samedi 7 juin 1986

Pierre DAVID
(P#37, SIG#1)

AH ! VOUS ECRIVEZ

Vous vous sentez en verve, mais vous ne savez pas sous quelle forme l'équipe de rédaction souhaite recevoir votre prose (ou vos vers) ! 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, parce que vous habitez dans les Cévennes, qu'il n'y a pas un seul autre membre du club à 100 km à la ronde, et aucun revendeur HP pour vous permettre de faire votre copie alors et alors seulement, faites-le sur papier.

COMMENT FORMATER VOTRE TEXTE

Que ce soit sur une feuille de papier ou sur support magnétique, respectez le format JIPSIEN (50 caractères par ligne approximativement).

Pourquoi ? Imaginez les difficultés rencontrées par l'équipe de rédaction pour visualiser des lignes de 96 caractères, voire davantage !

Utilisez des fichiers textes dans la mesure du possible. Si vous ne possédez pas de module Text Editor, et dans ce cas seulement, rédigez votre texte dans un BASIC sous forme de remarques. Mais soyez conscients que le travail d'élimination des numéros de lignes prend du temps. D'ailleurs nous pouvons vous fournir un programme éditeur de texte (pour l'instant moitié BASIC, moité LEX, prochainement entièrement en ASSEMBLEUR).

Préparez votre texte pour que le travail de formattage soit le plus court possible. Pour cela, utilisez les 2 symboles "^" et "\" décrits dans le formateur JIPSIEN :

"^" est utilisé pour centrer une ligne :

Ex : ^TITRE

"\" [CHR$(92)] est utilisé pour délimiter les paragraphes :

Ex :
\Début du 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 votre paragraphe.\

Sautez une ligne entre chaque paragraphes, deux lignes entre le titre et votre article et deux lignes entre votre article et votre signature.

Ne soulignez pas les titres et ne laissez pas les caractères de formattage du module "Editeur de Texte".

Ne prenez pas FRALEX pour mettre vos accents. Cela nous obligerait à passer votre texte au peigne fin pour remplacer chaque caractère spécial par son équivalent en ROMAN8 (police de caractères de la THINKJET et de la LASERJET). Nous pouvons vous fournir un CHARSET ROMAN8 et un jeu de touches assignées très pratique.

Relisez bien votre article. Nous passons une centaine d'heures à chaque journal pour modifier l'orthographe ou la syntaxe de vos articles. Il est préférable de répartir ce temps-homme entre plusieurs personnes (les auteurs notamment).

Vous aimez commencer par lire un article décrivant le but d'un programme, BASIC ou autre, son mode d'emploi et disposer d'un exemple. N'hésitez donc pas à présenter votre travail comme vous aimeriez le trouver dans JPC.

Vérifiez votre programme. Eh oui ! Il est fréquent qu'après la dernière mise au point, dans l'euphorie d'avoir enfin réduit le dernier bogue, et dans la hâte de le mettre sous pli, vous recopiez une version antérieure et il nous faut quelques heures pour débuger un programme

que nous n'avons pas écrit. C'est coton !

Pour les BASICS n'envoyez que des versions sous forme de textes (utilisez TRANSFORM INTO TEXT) cela nous évite de chercher à quoi correspondent les XWORD de votre programme. Prenez même la précaution de commencer par une remarque précisant les LEX utilisés par votre programme.

Vous trouverez dans ce numéro l'ensemble des logiciels nécessaire pour vos créations :

CHARLEX	Caractères ROMAN8
TEXTEK	Assignation des caract. accentués
FORMALEX	
FORMAT	Justifie le texte
ASSRED	Formateur pour source assembleur

Jean-Jacques DHENIN

COURRIER DU COEUR

Alain HERREMAN, 2 rue du parc Montsouris 75014 Paris (Tel : 45-89-09-41) . Vends un Port-Extender ET une boucle HP-IL le tout 500 !!!

Philippe GUEZ, 56,rue JJ ROUSSEAU 75001 PARIS : Vends 2 modules 4K pour HP71. 500f pièce

JP BONDU : Vends 3 modules 4K pour HP71 300f pièce
1 HP71 + HPIL pour 4300f les deux.

L'ANTI-CHRONIQUE DU MENTEUR

Je viens de rencontrer un ancêtre qui se lamantait de ne pas avoir eu de HP41. Qu'aurait-il fait de mieux en la possédant ? Je vous laisse réfléchir.

Quoi de neuf dans le château ? Et pourquoi...

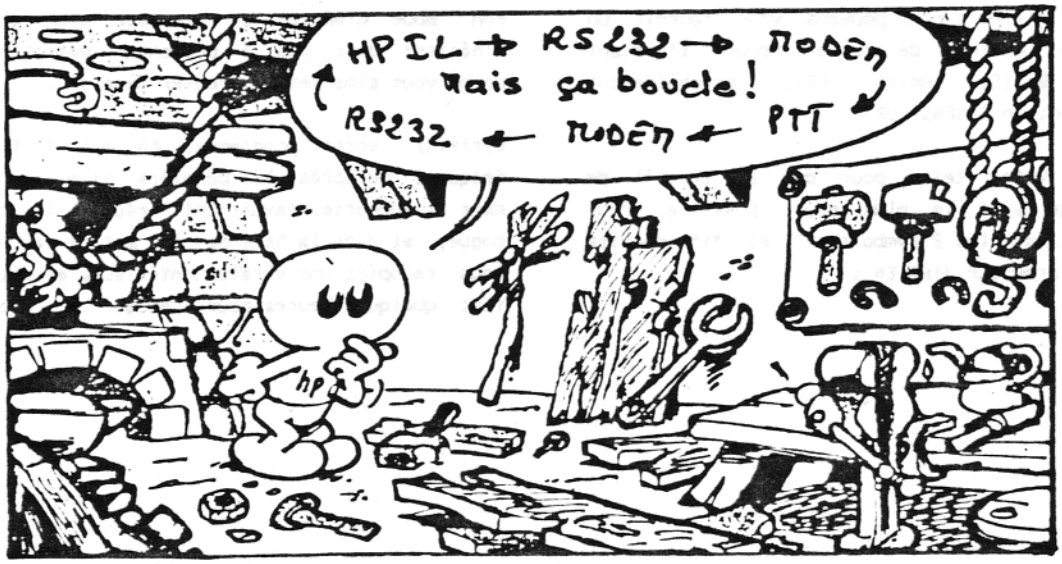
Contrairement à ce que le Menteur annonçait le mois dernier la HP41CV plafonne toujours à 130$!

Il a bien ri l'animal. Et chacun de se gausser jonquillement en reluquant sa pauvre machine si cher payée !

Venons en au sujet de ce matin : What's new in the U.S. ?
Pour ce faire j'ai pris la redoute u.s. i.e. Educalc.

- Une pochette en cuir pour 41 (page 2) $21.9599
- Un module CCD (et le JJD alors!) $149.95(99)
- Un module HP-IL bas prix pour la même becanne $99.99 (le prix d'une poupée)
- Un MLDL (toujours) $399.95
- Pig que ça 32 ou 64 Ko dans la 41 (ou plutôt dessous) $384.95/643.95 = ...
- Le tubo 41 pour 71 (qui en veut encore?) $918.0000000
- Pour le 71? lui même $399.95
- Module 32Ko $149.95 disponibles !

Alain FARGE



La fonction COPY : Bizarre, bizarre le HP-71 et son lecteur

Un fichier sans type

Un jour, peut-être, vous voudrez lire un fichier nommé TOTO :

```
COPY TOTO:CARD
```

mais vous aurez appuyé deux fois sur la touche [ENDLINE], par mégarde. Le méssage:

```
Pull Card
```

s'affiche. Que faire? Par exemple [INIT 1]. On reprend la procédure de lecture. Mais ici, le HP. annonce : Insufficient Memory.

Ah? MEM affirme qu'il ne reste qu'une centaine d'octets. Cataloguez TOTO. Surprenant, non ?

```
TOTO                    15998    date/heure.....
```

Un fichier sans type nommé TOTO, qui a récupéré la totalité de la mémoire disponible.

Une bogue du lecteur plutôt désagréable

Lorsque la machine attend une carte et et qu'une initialisation 1er niveau se produit, le HP. considère avoir lu la carte.

Mais alors, si ces simples mots sont éxécutés:

```
COPY :CARD   ?
```

Je pense que Titan créera un fichier sans type avec un nom composé de 8 espaces. Ceci est d'autant plus absurde que l'on perd la mémoire vive sans pouvoir la récupérer. En effet, les fonctions de fichiers n'autorisent pas des noms de 8 caractères.

Autrement dit, la commande avortée COPY:CARD provoque l'évaporation de la mémoire.

Un HP distrait, soyez vigilant

Le plus amusant, c'est que les fichiers sans type résistent aux POKE . En cas d'évaporation de la mémoire il n'y a guère qu'un Memory Lost.(1)

Un fichier sans nom et sans type est difficile à manipuler. En somme, une règle supplémentaire de la fonction COPY : éviter de provoquer [INIT 1], lors de la lecture d'une carte.

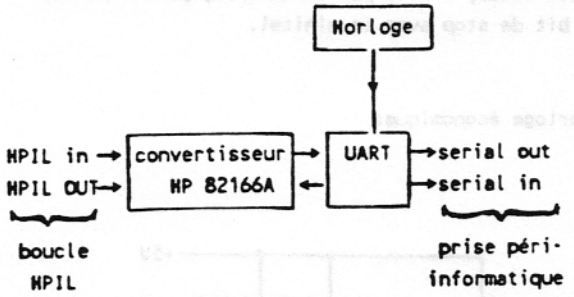
Dans le style: vous avez le module FORTH. Faites un MEM() du port où il est : on y trouve 1999 octets de libres. Frustrant, non ?

Bill X.

(1) NdR : Avec le module Forth il n'y plus de POKE qui tienne ! Mais ceci est une autre histoire.

INTERFACE HPIL «=» MINITEL

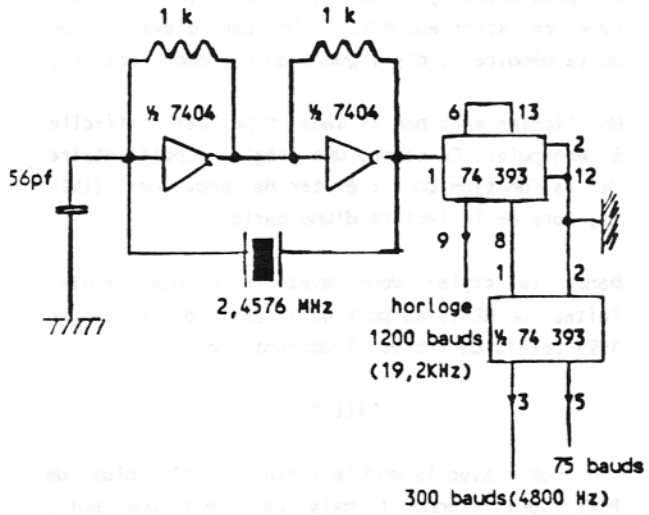
A) SYNOPTIQUE:



UART = AY - 3 - 1015 de General Instruments

B) SCHEMAS:

* Convertisseur: acheté tout monté ou en kit.
* Horloge



pattes 17 et 40 du AY-3-1015 (module UART)

Alimentation des circuits TTL:

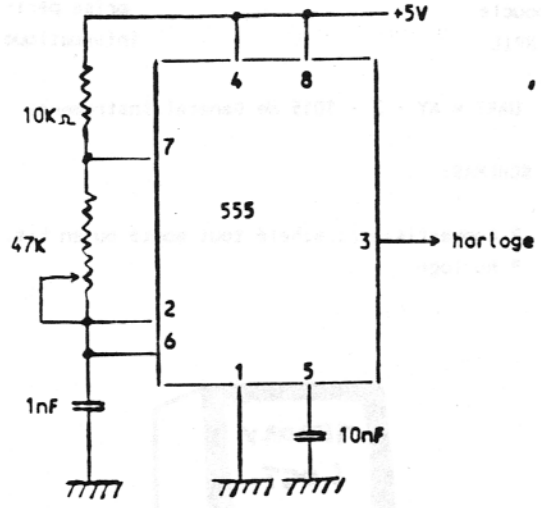
7404: masse=7, +5V=14

74393: masse=7, +5V=14

(Voir plus loin un schéma d'horloge "économique" sans quartz, mais qui nécessite un réglage.)

Ce module assure les communications séries à 1200 bauds, 7 bits par caractères, parité paire, 1 bit de stop avec le minitel.

Horloge économique:



Régler l'ajustage pour avoir 19200 Hz en sortie

C) REGISTRES DU CONVERTISSEUR :

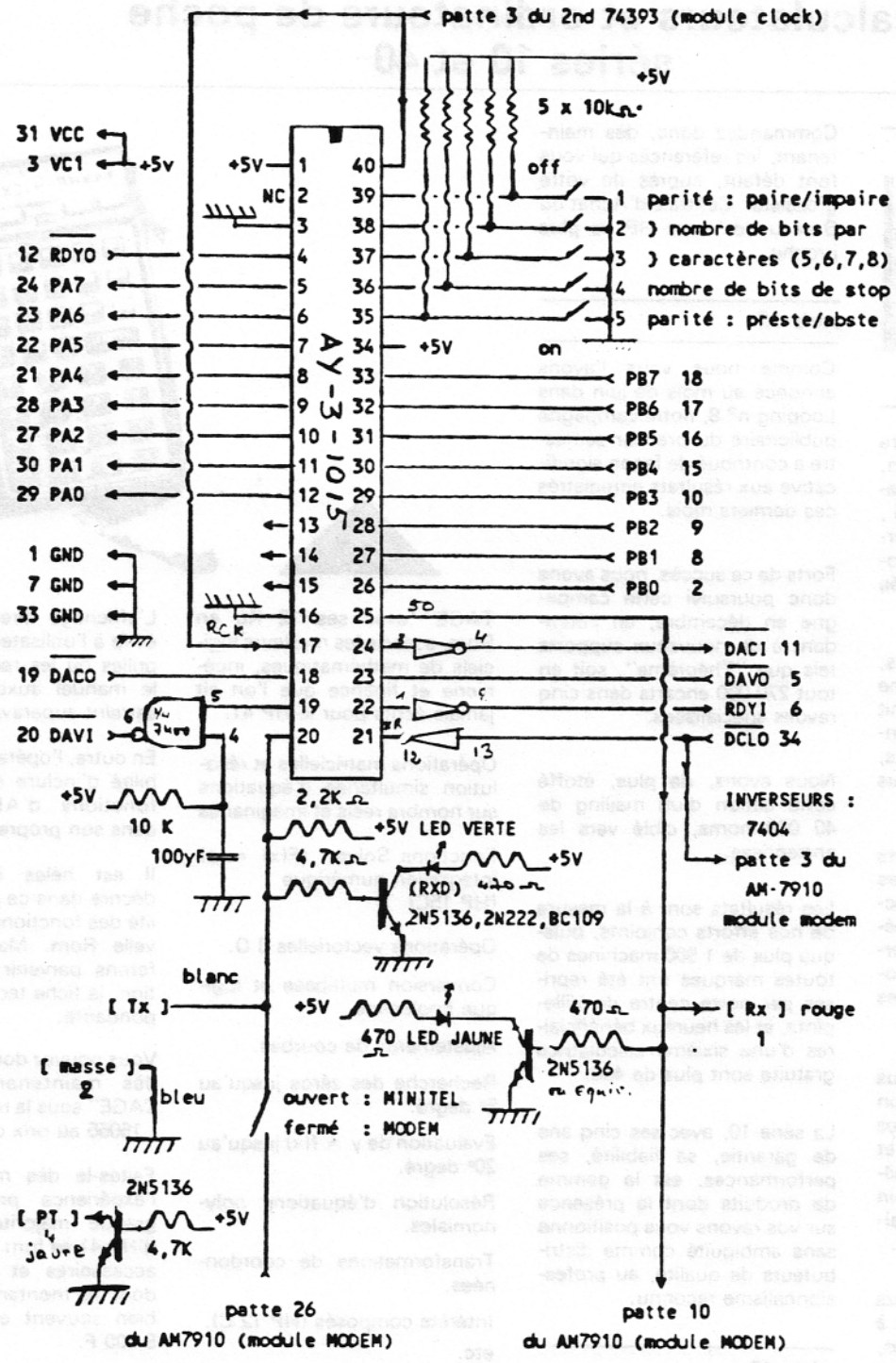
- Il convient de modifier leur valeur par défaut, en particulier pour se mettre en mode uni-directionnel.
- avec une HP41 (et le module X-I/O), le petit programme suivant fera l'affaire:

```
NLOOP
ADROFF
1          (à changer si le convertisseur
LAD         a une autre adresse)
0
DDL
130        (130 mieux que 64 : la présence
OUTXB          de DATA en provenance du RTC
16             génére un SREQ
OUTXB
218
OUTXB
UNL
ADRON
```

- Toutes les données envoyées au convertisseur apparaitront sur l'écran du minitel, toutes les touches tapées au clavier du minitel pourront être lues dans le convertisseur.

Lionel

MODULE UART



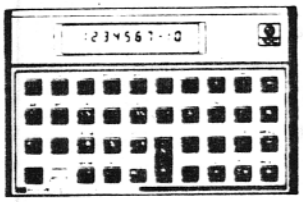
(1) on peut remplacer les 3 inverseurs 7404 ( ) par des portes NAND 7400 ( )

(2) on peut remplacer le convertisseur (82166) par l'interface GPIO (82165)

(3) alimentation 7400: masse=7, +5V=14

matériels

calculateurs et ordinateurs de poche séries 10 et 40



Si cette année promet d'être un excellent cru pour le vin, nous pouvons, dès maintenant, vous affirmer que 1985, en ce qui concerne les performances de cette ligne de produits, s'est révélée une année aussi exceptionnelle.

En effet, pour la première fois, la France est, ce deuxième semestre 85, première au hit parade des ventes européennes, et ce succès, c'est à vous, chers distributeurs, que nous le devons.

L'analyse de nos résultats prouve encore plus, que les meilleures ventes sont effectuées par ceux dont la compétence est la plus forte en termes de connaissance des produits et de disponibilité des machines.

Beaucoup d'entre vous nous ont fait part de la satisfaction que leur avaient apportée les présentoirs d'aide à la vente, et nous ne saurions trop conseiller à ceux qui auraient besoin d'exemplaires supplémentaires, de nous les commander.

De plus, les nombreux posters que nous éditons, rappellent à votre clientèle que vous êtes, dans votre ville, un des points de vente agréés Hewlett-Packard. N'hésitez pas à les afficher à votre meilleure place et à profiter aussi de cette excellente accroche.

De même, une documentation de qualité, que vous pourrez immédiatement distribuer sur simple demande, sera un plus auquel votre clientèle sera particulièrement sensible.

Commandez donc, dès maintenant, les références qui vous font défaut, auprès de votre grossiste : Centrale d'Achat ou Bureau de Vente HP, le plus proche.

série 10

Comme nous vous l'avons annoncé au mois de juin dans Looping n° 8, notre campagne publicitaire du premier semestre a contribué de façon significative aux résultats enregistrés ces derniers mois.

Forts de ce succès, nous avons donc poursuivi cette campagne en décembre, en l'étendant à de nouveaux supports tels que "Théorème", soit en tout 270 000 encarts dans cinq revues spécialisées.

Nous avons, de plus, étoffé cette action d'un mailing de 40 000 noms, ciblé vers les entreprises.

Les résultats sont à la mesure de nos efforts conjoints, puisque plus de 1 500 machines de toutes marques ont été reprises par notre centre de Villepinte, et les heureux bénéficiaires d'une sixième calculatrice gratuite sont plus de 400.

La série 10, avec ses cinq ans de garantie, sa fiabilité, ses performances, est la gamme de produits dont la présence sur vos rayons vous positionne sans ambiguïté comme distributeurs de qualité, au professionnalisme reconnu.

série 40

Depuis le 1er janvier, vous pouvez commander un nouveau module qui vient étoffer la liste déjà longue des extensions et accessoires que le HP 41 est encore le seul calculateur de poche à proposer sur le marché.

Ce module appelé "ADVANTAGE" avec ses 12 Ko en Rom, associe les meilleurs logiciels de mathématiques, ingénierie et finance que l'on ait jamais écrits pour le HP 41.

Opérations matricielles et résolution simultanée d'équations sur nombre réels et imaginaires

Fonctions Solve : (F(x) = o) intégration numérique (HP 15C)

Opérations vectorielles 3 D.

Conversion multibase et logique booléenne.

Ajustements de courbes.

Recherche des zéros jusqu'au 5e degré.

Evaluation de y = f(x) jusqu'au 20e degré.

Résolution d'équations polynomiales.

Transformations de coordonnées.

Intérêts composés (HP 12 C).

etc.

A la différence des autres modules, la plupart des programmes "ADVANTAGE" sont écrits en langage machine, donc d'exécution plus rapide.

L'utilisation est de type convivial et pratiquement tous les programmes bénéficient des clefs reconfigurables au clavier.

L'affichage direct des menus évite à l'utilisateur l'emploi des grilles ou les recherches dans le manuel auxquelles il était astreint auparavant.

En outre, l'opérateur a la possibilité d'inclure en totalité des fonctions d'ADVANTAGE", dans son propre programme.

Il est hélas impossible de décrire dans ce journal la totalité des fonctions de cette nouvelle Rom. Mais nous vous ferons parvenir dès sa parution, la fiche technique correspondante.

Vous pouvez donc commander dès maintenant "ADVANTAGE" sous la référence 00041 - 15055 au prix de 554 F HT.

Faites-le dès maintenant car l'expérience prouve que la grande majorité des ventes d'HP 41 se font avec modules, accessoires et périphériques, dont le montant total atteint bien souvent entre 5 000 et 8 000 F.

Ceux qui vendent et fidélisent leur clientèle, sont ceux qui présentent et sont à même de proposer une gamme complète.

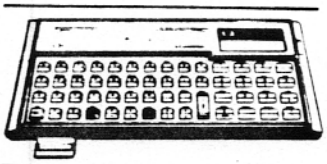
Bonnes ventes !

Philippe Chaillot
Responsable du Programme Calculateurs

matériels

série 70

informations synthétiques sur le HP 71B



produits nouveaux non HP

La Société COSERM,
18, rue Morvan - 94633 Rungis-
Tél. : (1) 46.86.64.75
M. Chauvier importe les mémoires RAM de fabrication HHP :

Il y a cinq types de produits qui s'implantent à la place du lecteur de cartes magnétiques :
- Support Module EPROM
- Support Module EPROM + 32 Ko RAM
- Module RAM 32 Ko
- Module RAM 64 Ko
- Module RAM 96 Ko.

Rappelons que le HP 71B complètement équipé peut atteindre 128 Ko de mémoire RAM utilisateur ...

La Société KRISTAL
Clos Zisrt, 38240 Meylan
Tél. : 76.90.38.18

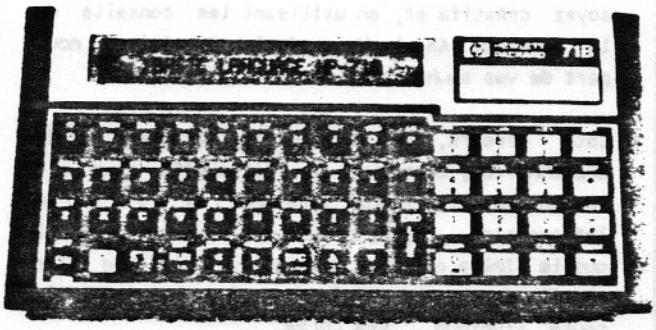
commercialise une interface HPIL --→ Minitel.

Cette interface permet au poste Minitel de jouer le rôle de clavier et d'écran (25 lignes, 40 colonnes) du HP 71B équipé de l'interface HPIL. Le clavier du HP 71 reste actif en même temps que celui du Minitel, utilisation du Minitel en écran du HP 41 et du HP 75, utilisation du modem du Minitel pour transférer et recevoir de l'information d'un centre serveur (stockage du bottin par exemple ...). Transmission de caractères ASCII et de fichiers binaires ...

La Société ZENGRANGE
Limited - Greenfield Road
Leeds
L 598DB West Yorkshire
England
Tel. : 0532.48.90.48.

commercialise un lecteur de codes-barres qui a plusieurs fonctionnalités spécifiques : lecture codes-barres, Codabar (USD - 4) ; code 11 (USD - 8) ; code 39 (USD - 3) ; EAN - 13 ; EAN - 8 ; UPC - A ; UPC - E ; UPC - E (1) ; 2/5 industriel ; 2/5 entrelacé.

Il est pourvu d'un système de décodage automatique : il répond à la première lecture par le nom du code, à la seconde par la valeur, après avoir décodé le type, toutes les lectures suivantes du même code se font à la première lecture, vérification du digit de parité, analyse du bar code et impression du code sur l'imprimante Thinkjet, ce module équipé d'un crayon lecteur HP, occupe un logement de la face avant du HP 71B.



rappel

Le module Traitement de Texte 82485 AF, devrait être vendu avec chaque HP 71 BF, toutes ses instructions permettent de faciliter la programmation. En plus de ses fonctions de traitement de texte, formatage, etc..., il permet la gestion de fichier très simple à réaliser, grâce à cinq instructions paramétrables :

INSERT - REPLACE - DELETE - SEARCH - FILES2R

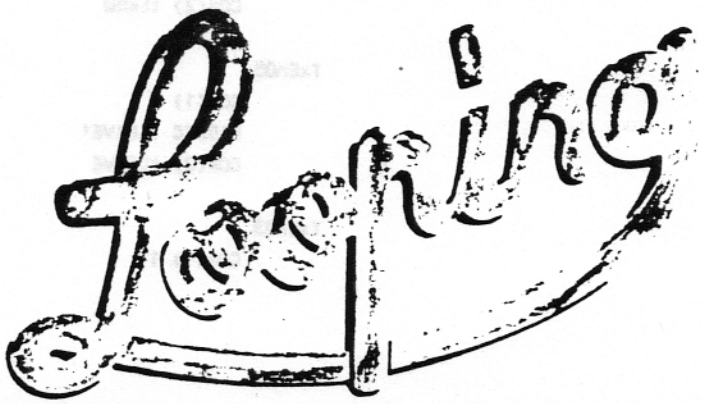
Ces instructions permettent la création, modification et suppression d'enregistrement de façon très simple.

Ce module prend tout son intérêt également lorsque le HP 71B est associé à l'interface MINITEL et à l'imprimante THINKJET.

Robert Bayle
Chef des Produits Portables
Informatique Personnelle

Edité par :
HEWLETT-PACKARD FRANCE
Parc d'activité du Bois Briard
2, avenue du Lac
91040 EVRY CEDEX

Tirage : 2.500 exemplaires



JANV/FEV 1986

BULLETIN
MENSUEL
DE L'INFORMATIQUE
PERSONNELLE

N° 12

CHOCS EN RETOUR

Le dernier numéro, daté d'AVRIL, contenait gag et erreurs. Côté gag, le poisson devait être plutôt bien noyé puisque beaucoup d'entre les lecteurs ont mordu à l'hameçon. Pourtant la description de ce prétendu 72C était tellement invraissemblable que nous n'avions pas supposé être crédibles.

Puisque ce bobard est passé pour une information nous devons tirer quelques conclusions : notre attente colective est grande d'un nouveau contrôleur de boucle plus rapide, disposant d'un écran multilignes et doté d'un système d'exploitation très évolué. Probablement pourrions-nous, dans ces colonnes, faire part de nos rêves dans ce domaine. N'hésitez donc pas, soyez créatifs et, en utilisant les conseils de la rubrique "AH ! Vous écrivez", faites nous part de vos souhaits.

Côtè erreurs, la version de struclex était ancienne et, bien que J. TAILLANDIER nous ait communiqué la bonne version, il y a eu interversion. Bref, nous ne savions plus à quelle [mère ou père] version il fallait se fier en fin de ponte. Voici donc l'authentique, la seule, l'unique ...MEA CULPA

Pour ALLER LES GOTO !, il manquait le source du FORTH; et pour la construction de la table d'équivalence, il manquait le source de LINKLEX.

En conséquence voici l'erratum

J.J. DHENIN

```
      LEX   'STRUC1'

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

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

*
*MAIN  TABLE
*

      CON(3) (TxEn01)-(TxTbSt)
      REL(5) ENDW
      CON(1) 13

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

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

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

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

*
*TEXT  TABLE
*

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

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

TxEn03
      CON(1) 11
```

```
       NIBASC 'REPEAT'
       CON(2) tREPT

TxEn04
       CON(1) 9
       NIBASC 'UNTIL'

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

       NIBHEX 1FF

LEXID  EQU    #E1
tXWORD EQU    #EF
WHILEt EQU    9
REPTt  EQU    10
eNOWHL EQU    3
eNORPT EQU    4
eLEAVE EQU    5
eNOENW EQU    6
eINVLD EQU    236
oFTYPh EQU    #10
fBASIC EQU    #0E214
eSYSER EQU    #17
bSTMT  EQU    #801

STMTD0 EQU    #2F891
STMTD1 EQU    #2F896
PRGMEN EQU    #2F567
TRACEM EQU    #2F7B0
CURRST EQU    #2F55D

WRDSCN EQU    #02C2A
FIXP   EQU    #02A6E
TKSCN7 EQU    #08A99
FIXDC  EQU    #05493
REST*  EQU    #03035
EOLXC* EQU    #052EC
OUTNBC EQU    #05423
POPUPD EQU    #08F3E
EXPEXC EQU    #0F186
POP1N+ EQU    #0BD91
PSHGSB EQU    #08F13
NXTSTM EQU    #08A48
BSERR  EQU    #0939A
MFERR  EQU    #09393
RUNRT1 EQU    #074E7

STOPDC EQU    #05303
TRFROM EQU    #0FE59
TRTO+  EQU    #0FE7B
I/OFND EQU    #118BA
EOLSCN EQU    #08AA7

ermsgs CON(2) eNOWHL
       CON(2) eNOENW

bNOWHL CON(2) (fNOWHL)-(bNOWHL)
       CON(2) eNOWHL
       CON(1) 4
       NIBASC 'WHILE'
       CON(1) #C
fNOWHL
bNORPT CON(2) (fNORPT)-(bNORPT)
       CON(2) eNORPT
       CON(1) 8
       NIBASC 'no REPEA'
       NIBASC 'T'
       CON(1) #C
fNORPT

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

bNOENW CON(2) (fNOENW)-(bNOENW)
       CON(2) eNOENW
       CON(1) 11
       CON(1) 11
       NIBASC 'no END W'
       NIBASC 'HILE'
       CON(1) #C
fNOENW

       NIBHEX FF

*
* END WHILE
*

       REL(5) ENDWd
       REL(5) ENDWp
ENDW   CD0EX
```

```
        D0=(5) (=STMTD1)
        DAT0=C A
        CD0EX
        GOSUB  popupd
        GOC    ENDW01
        P=     15
        LC(1)  WHILEt
        P=     0
        ?C=D   S        compare avec notre type
        GOYES  ENDW02   Ok
ENDW01  P=     0
        LC(2)  eNOWHL
myerr   P=     2
        LC(2)  LEXID
        P=     0
        GOVLNG =BSERR
ENDW02  ?ST=1  13
        GOYES  ENDW04   depuis un programme
        P=     0
        LC(3)  bSTMT    depuis le clavier
        GOSBVL =I/OFND
        B=A    A        longueur du buffer
        CD1EX           C[A] début du buffer
        ?D<C   A        addr < début du buffer
        GOYES  ENDW01
        C=C+B  A        C[A] à fin du buffer
        ?D>=C  A
        GOYES  ENDW01
ENDW04  C=D    A        adresse de l'expression
        D0=(5) (STMTD0)
        DAT0=C A        STMTD0 ^ avant expression
        CD0EX
        GOSUB  eval
        GOC    ENDW10
        D0=(5) (STMTD1)
        C=DAT0 A
        CD0EX
        GOTO   runrt1

ENDW10  CD0EX
        R2=C            sauve D0 aprés l'expression
          *              dans R2
        D0=(5) (=STMTD0) push addresse sur GOSUB
          *              stack
        A=DAT0 A        .
        P=     15       .
        LC(1)  WHILEt   .
ENDW11  GOSUB  pshgsb   .
        C=R2            restaure D0 après
          *              expression
        CD0EX
ENDW15  ?ST=0  15       en mode TRACE
        GOYES  runrt1   non
        GOSUB  trflck   trace active ?
        GOC    runrt1   non
        CD0EX
        R2=C
        GOSBVL =TRFROM  exécute la partie FROM de
          *              trace
        C=R2
        D0=C
        GOSBVL =TRTO+   partie TO de trace
        A=R2
        D0=A            restaure D0 et fin
runrt1  GOVLNG =RUNRT1

trflck  ST=0   10       pompé sans vergogne
          *              (#0FE18)
        ?ST=0  15
        RTNYES
        ?ST=0  13
        RTNYES
        ?ST=1  10
        RTNYES
        D1=(5) TRACEM
        P=     0
        LCHEX  2
        A=DAT1 B
        A=A&C  P
        A=A-1  P
        D1=(5) =CURRST
        C=DAT1 A
        D1=C
        D1=D1+ (oFTYPh)-1
        A=DAT1 A
        ASR    A
        LC(5)  =fBASIC
        ?A=C   A
        GOYES  clrcy
        RTNSC
clrcy   RTNCC

* WHILE <expression>
*

        REL(5) WHILEd
        REL(5) WHILEp
WHILE   GOSUB  saveD0   sauve D0 dans STMTD0
        CD0EX
        GOSUB  eval
```

```
        GONC   WHIL20   condition fausse
        CD0EX
        R2=C
        D0=(5) (STMTD0) adresse du début
          *              d'expression
        A=DAT0 A
        P=     15
        LC(1)  WHILEt
        GOSUB  pshgsb
        C=R2            restaure D0
        CD0EX
nxtstm GOVLNG =NXTSTM

* La condition est fausse, on doit trouver le END
          *             WHILE
* correspondant à notre WHILE. On utilise un
          *             compteur
* dans R2[A] incrémenté par un WHILE
* et decrementé par END WHILE.

WHIL20 C=0    A         prepare compteur
       C=C+1  A         R2[A] incrementé par notre
         *               WHILE
       R2=C
       ?ST=0  13
       GOYES  WHIL05
       D1=(5) PRGMEN    fin du programme
       C=DAT1 A
       GOTO   WHIL06
WHIL05 P=     0
       LC(3)  bSTMT     clavier: cherche la fin du
         *               buffer
       GOSBVL =I/OFND
       CD1EX
       C=C+A  A
WHIL06 D=C    A         fin de recherche dans D[A]
WHIL21 P=     0
       LC(2)  tXWORD
       GOSBVL =TKSCN7
       GOC    WHIL22    trouvé
       LC(2)  eNOENW
       GOTO   myerr     'no END WHILE'
WHIL22 D0=D0+ 2
       LC(4)  (tWHILE)*#100+(LEXID)
       P=     3
       A=DAT0 WP
       ?A#C   WP
       GOYES  WHIL23    ce n'est pas un WHILE
       C=R2
       C=C+1  A         compte un WHILE
       R2=C
       GOTO   WHIL30
WHIL23 P=     2
       LC(2)  tENDW
       P=     3
       ?A#C   WP
       GOYES  WHIL30    ce n'est pas END WHILE
       C=R2
       C=C-1  A         compte un END WHILE
       ?C=0   A         terminé ?
       GOYES  WHIL40    oui !
       R2=C
WHIL30 GOSUB  eol       va à tEOL ou t@
       GOTO   WHIL21    retourne chercher

WHIL40 D0=D0+ 4
       GOTO   ENDW15    TRACE s'il y a lieu

* Evalue l'expression pointée par D0, retour avec
       *                 carry clear
* si 0
* D0 ^ fin d'expression au retour

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

* va en fin de ligne si D0 ^ tXWORD

eol    D0=D0- 4         retour à l'octet longueur
       A=0    A
       A=DAT0 B         longueur de ligne
       B=A    A
       ADOEX
       A=A+B  A         A[A] ^ tEOL ou t@
       D0=A
       RTN

pshgsb GOVLNG =PSHGSB
popupd GOVLNG =POPUPD
```

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

*
* WHILE <expression>
*

       REL(5) WHILEd  parse & decompile = WHILE
       REL(5) WHILEp
UNTIL  CD0EX
       D0=(5) (=STMTD1)
       DAT0=C A
       GOSUB  popupd
       GOC    UNTI01
       P=     15
       LC(1)  REPTt
       P=     0
       ?C=D   S       compare avec notre type
       GOYES  UNTI02  Ok
UNTI01 LC(2)  eNORPT
       GOTO   myerr
UNTI02 ?ST=1  13
       GOYES  UNTI04  depuis un programme
       P=     0
       LC(3)  bSTMT   depuis le clavier, trouve
         *             le buffer
       GOSBVL =I/OFND
       B=A    A       longueur du buffer
       CD1EX          D1 buffer start
       ?D<C   A       addr < buffer start
       GOYES  UNTI01
       C=C+B  A       C[A] end of buffer
       ?D>=C  A
       GOYES  UNTI01
UNTI04 C=D    A       adresse de l'expression
       D0=(5) (STMTD0)
       DAT0=C A       sauve adresse après REPEAT
       D0=D0+ 5       D0 ^ STMTD1
       C=DAT0 A
       CD0EX          D0 ^ expression
       GOSUB  eval
       GONC   UNTI10
       GOTO   runrt1
UNTI10 D0=(5) (STMTD0) adresse après REPEAT
       A=DAT0 A
       R2=A
       P=     15
       LC(1)  REPTt
       GOTO   ENDW11  trace s'il y a lieu

*
* REPEAT
*

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

*
* LEAVE
*

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

* On utilise R2[A] comme compteur, incremente par
         *              REPEAT ou WHILE
* decremente par UNTIL ou END WHILE

LEAV02 C=0    A        initialise compteur
       C=C+1  A
       R2=C
       ?ST=0  13       depuis le clavier ?
       GOYES  LEAV05   oui
       D1=(5) PRGMEN   fin du programme
       C=DAT1 A
       GOTO   LEAV06
LEAV05 P=     0        depuis le clavier, cherche
         *              le buffer
       LC(3)  bSTMT
       GOSBVL =I/OFND
```

```
       CD1EX
       C=C+A  A
LEAV06 D=C    A        adr. de fin d'analyse dans
         *              D[A]
       D0=(5) (=STMTD0)
       C=DAT0 A
       CD0EX           restitue D0
LEAV21 P=     0
       LC(2)  tXWORD
       GOSBVL =TKSCN7
       GOC    LEAV22   si trouvé
       LC(2)  eLEAVE
       GOTO   myerr
LEAV22 D0=D0+ 2
       LC(4)  (tENDW)*#100+(LEXID)
       P=     3
       A=DAT0 WP
       ?A=C   WP
       GOYES  LEAV30   c'est END WHILE
       P=     2
       LC(2)  tUNTIL
       P=     3
       ?A=C   WP
       GOYES  LEAV30   c'est UNTIL
       P=     2
       LC(2)  tREPT
       P=     3
       ?A=C   WP
       GOYES  LEAV33   c'est REPEAT
       P=     2
       LC(2)  tWHILE
       P=     3
       ?A=C   WP
       GOYES  LEAV33   c'est WHILE
LEAV32 GOSUB  eol      retourne chercher
       GOTO   LEAV21
LEAV33 C=R2
       C=C+1  A
       R2=C            met a jour le compteur
       GOTO   LEAV32
LEAV30 C=R2
       C=C-1  A
       R2=C            met a jour le compteur
       ?C#0   A
       GOYES  LEAV32   pas termine
       GOSUB  eol      D0 ^ tEOL ou t@
       GOTO   ENDW15   TRACE s'il y a lieu

REPTd  GOVLNG =STOPDC
REPTp  RTNCC
```

```
ENDWp  GOSBVL =WRDSCN
       CON(2) =tXWORD  essayer WHILE après END
       CON(2) =LEXID
       CON(2) tWHILE
       REL(3) ENDWp1   si ok
       CON(2) D0
       P=     0        sinon on essaie d'autres
         *              END
       GOVLNG =REST*
ENDWp1 D0=D0- 6         si END WHILE on ne stocke
         *              pas tWHILE
       RTNCC
ENDWd  LCASC  'ELIHW' display WHILE
       P=     9
       GOSBVL =OUTNBC
       GOSBVL =EOLXC*  on doit être à EOL

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

ADDENDUM à LINK

Dans le but d'augmenter la puissance du programme LINK j'ai rajouté la pseudo-op INCLUDE. Syntaxe : INCLUDE <NomFich>. Lorsque LINK éxamine votre source il insère à la place de INCLUDE le fichier TEXT spécifié. Ainsi les programmeurs qui se sont construits des sous-routines peuvent directement les appeler. <NomFich> NE DOIT PAS DEPASSER 8 CARACTERES. Si la syntaxe est incorrecte l'éditeur de texte est immédiatement appelé afin que la correction s'effectue 'dans la foulée'. Si le fichier n'est pas en RAM il est recherché sur la première mémoire de masse. A ce stade une recherche infructueuse engendre un message d'erreur.

S'il n'y a pas assez de RAM pour insérer ce fichier, les lignes chargées sont effacées et le source est donc laissé PARFAITEMNT INTACT. Si tout s'est correctement déroulé l'op INCLUDE est effacée.

Un exemple :

.

```
        GOVLNG NXTSTM
INCLUDE JUMPER
LABEL   D1=C
        .
```

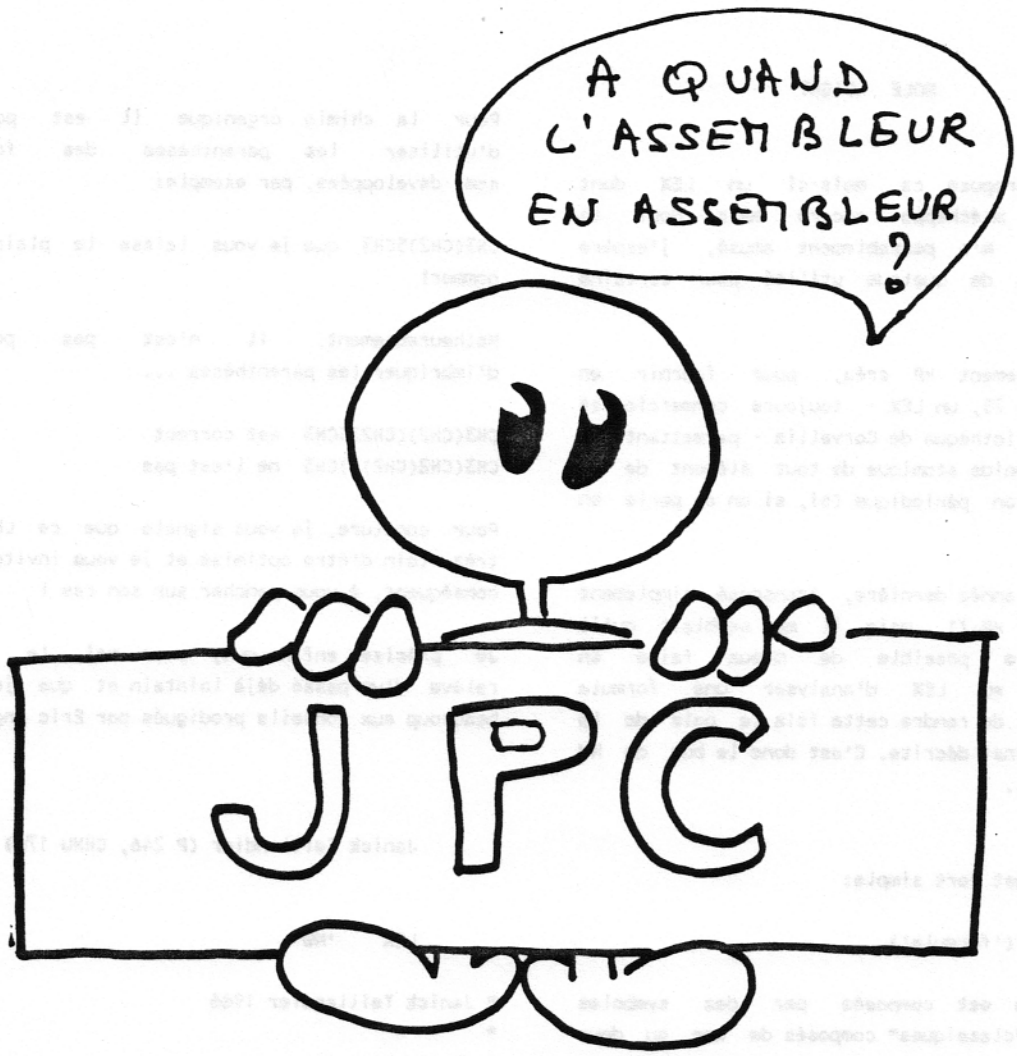
JUMPER sera inclus avant LABEL. Voilà ! Bons pré-assemblages avant l'assembleur en Langage Machine.

Jean-Piere BONDU (PPC 33, SIG 3)

```
        LEX     'LINKLEX'
        ID      #E1
        MSG     0
        POLL    0

        ENTRY   LABEL$
        CHAR    #F
        ENTRY   PMN
        CHAR    #F
        KEY     'LABEL$'
        TOKEN   76
        KEY     'MNEMO'
        TOKEN   77
        ENDTXT

ADHEAD  EQU     #181B7
D=AVMS  EQU     #1A460
DRANGE  EQU     #1B076
EXPR    EQU     #0F23C
FNRTN4  EQU     #0F238
HDFLT   EQU     #1B31B
REV$    EQU     #1B38E
REVPOP  EQU     #0BD31

*****************************************************
*
* Nom:     MNEMO    ;   Type: fonction
*
* Syntaxe:  MNEMO(chaine)
*
* But: 'chaine' est l'enregistrement d'un fichier
*      source, MNEMO renvoie la position de la
*      mnémonique en tenant compte des cas
*      particuliers suivants.
*
* Sortie:   P= MNEMO(chaine)
*      P=0:   chaine nulle (LEN=0)
*             chaine blanche
*             chaine débutant par '*'
*             POS(1er caractère # espace) > 24
*      P#0:   P ^ mnémo si enregistrement valide
*             P=LEN(chaine) si chaine='label...'
*****************************************************

        NIBHEX 411      1 chaine
PMN     GOSBVL REVPOP   MS <- chaine retournée
        CD1EX           D1 @ 1er car (low mem)
        D1=C            C[A]= D1
        C=C+A  A        et A[A]= longueur
        RSTK=C          RSTK @ sommet Math Stack
        ASRB            Nb quartets -> octets
        R1=A            R1= LEN
        ST=1   0        * Cherche 1er caractère
        GOSUB  PMN10    * <> espace
        GOC    exit0    Tous blancs/chaine nulle
        B=C    B        B[B]= 1er caractère
        LCASC  '*'
        ?B=C   B        Ligne de remarque ?
        GOYES  exit0    oui
        C=R1            C[A]= LEN
        C=C-A  A
        B=C    A        B[A]= position
        C=0    A
        LC(2)  24
        ?B>C   A        Position > 24 ?
        GOYES  exit0    C'est une ligne de Rem
        LC(2)  3
        ?B<C   A        Position < 3 ?
        GOYES  PMN20    C'est un label

* Routine de sortie/retour au basic
* Entrée:
*    RSTK @ sommet Math Stack
*    B[A] = position
*
Exit    C=RSTK
        D1=C
        D1=D1- 16
        A=B    A        * Conversion B[A] hexa
        GOSBVL HDFLT    * en décimal flottant
        C=A    W        * en C[W]
        GOVLNG FNRTN4   Renvoie C[W]
exit0   B=0    A
        GOC    Exit     B.E.T.

* On cherche le prochain espace
*
PMN20   ST=0   0
        GOSUB  PMN10
```

```
        GONC   PMN30     Trouvé
* Sinon on renvoie la longueur de la chaine +1
* (= longueur du label [+espaces] +1)
*
exitln  A=R1
        A=A+1  A
        B=A    A         B[A]= LEN +1
        GONC   Exit      B.E.T.

* On cherche le prochain caractère <> espace
* (= début mnémonique)
*
PMN30   ST=1   0
        GOSUB  PMN10
        GOC    exitln    Pas trouvé
        C=R1             C[A]= LEN
        C=C-A  A
        B=C    A         B[A]= position
        GONC   Exit      B.E.T.

**********************************
* Recherche du 1er caractère
*     = blanc  ( ST=0 0 )
*     # blanc  ( ST=1 0 )
*
* Entrée:
*     D1  = début recherche
*     A[A]= longueur chaine
*
* Sortie:
*     Carry SET --> Not Found
*     Carry CLR:
*        D1  @ caractère suivant
*        A[A]= Len - position
*        C[B]= caractère
**********************************

PMN10   LCASC  ' '
        B=C    B         B[B]= ' '

PMN15   A=A-1  A         Fin chaine ?
        RTNC             oui: retour CS
        C=DAT1 B         C[B]= caractère
        D1=D1+ 2
        ?ST=1  0         Teste si # blanc ?
        GOYES  tst#      oui
tst=    ?B#C   B         blanc ?
        GOYES  PMN15
        RTN              oui: retour CC

tst#    ?B=C   B         # blanc ?
        GOYES  PMN15
        RTN              oui: retour CC

**********************************
* Nom:   LABEL$    ; Type: fonction
*
* Syntaxe: LABEL$(string$)
*
* But: vérifie que string$ est une étiquette
* valide
*      d'un programme assembleur. Les conditions
*      à remplir sont les suivantes.
*
* Entrée: une chaine, appelée LABEL$
*
* Sortie/algorithme:
*      Si LABEL$[1,1]='='
*         LABEL$=LABEL$[2]
*      Si len(LABEL$)>6
*         ou len(LABEL$)<2
*         ou LABEL$[1,1]='#'
*         ou LABEL$ commence par 1 chiffre
*           --> renvoie chaine nulle
*      Si len(LABEL$)=2 et LABEL$='XS'
*           --> renvoie chaine nulle
*      Renvoie LABEL$
*
* Exemples:
*      "TEST145"  -->  "" car len>6
*      "LBLtst"   -->  "LBLtst"
*      "=NXTSTM"  -->  "NXTSTM"
*       car '=' en 1er position est ignoré
*      "#FFC"     -->  "": c'est une adresse
*      "#TST"     -->  assimilé cas précédent
*      "1token"   -->  "": commence par 1 chiffre
*
**********************************

        NIBHEX 411       1 chaine
LABEL$  GOSBVL REVPOP    cf MNEMO
        B=0    W
        B=A    A
        BSRB             B[A]= LEN en octets
        C=0    A
        C=C+1  A
        ?B<=C  A         LEN<2 ?
        GOYES  null$
        A=DAT1 A         A[3-0]= LABEL$[1,2]
        LCASC  '='
```

```
        ?A#C   B        LABEL$[1,1]='=' ?
        GOYES  LBL10    non

* On éjecte '=': LABEL$=LABEL$[2]
*
        B=B-1  A        LEN= LEN-1
        D1=D1+ 2        D1 à 2e caractère
        A=DAT1 A

* B[A]= LEN ; D1 à 1er caractère valide
* A[3-0]= LABEL$[1,2]
*
LBL10   LC(2)  2        C[A]= 00002
        ?B<C   A        LEN<2 ?
        GOYES  null$
        LC(1)  6
        ?B>C   A        LEN>6 ?
        GOYES  null$
        LCASC  '#'
        ?A=C   B        LABEL$= '#....' ?
        GOYES  null$
        GOSBVL DRANGE   Commence par un chiffre ?
        GONC   null$    oui
        LC(2)  2
        ?B#C   B        LEN#2 ?
        GOYES  LBL30    oui

* LEN=2: teste si LABEL$='XS'
*
        LCASC  'SX'
        P=     3
        ?A#C   WP       Différent ?
        GOYES  LBL30    oui

* Renvoie une chaine nulle. Pour cela on
* positionne le pointeur de début de chaine D1
* en fin de chaine & on indique une longueur nulle
*
null$   CD1EX
        C=C+B  A        D1=D1+LEN (quartets)
        C=C+B  A
        CD1EX           D1 à sommet MS
        B=0    A        LEN=0

* Renvoie la chaine pointée par D1, dont la
* longueur est en B[B] en octets
*
LBL30   P=     0
        B=B+B  B        octets -> quartets
        GOSBVL D=AVMS   D[A]=AVMS ; C=D1
        C=C+B  A        C= pointeur+longueur
        R1=C            R1 à sommet MS (hi mem)
         *              D1 à début chaine (lw mm)
        ST=1   0        Retour après ADHEAD
        GOSBVL ADHEAD   Construit en-tête chaine
        GOSBVL REVS     Retourne chaine
        GOVLNG EXPR     Renvoie la chaine pointée
         *              par D1 sur la MS

        END
```



ASSEMBLEUR

MOLE WEIGHT

Je vous propose ce mois-ci un LEX dont l'intéret m'échappe encore mais dont la réalisation m'a passablement amusé, j'espère qu'il sera de quelque utilité pour certains parmi vous.

Au commencement HP créa, pour fournir en logiciel le 75, un LEX - toujours commercialisé par la bibliothèque de Corvallis - permettant de donner le poids atomique de tout élément de la classification périodique (si, si on en parle en chimie ...).

J'avais, l'année dernière, transposé simplement ce LEX pour HP-71, mais il me semblait qu'il devait être possible de mieux faire en permettant au LEX d'analyser une formule chimique et de rendre cette fois le pois de la molécule ainsi décrite. C'est donc le but de MW présenté ici.

La syntaxe est fort simple:

```
MOLEWT('formule')
```

où formule est composée par des symboles chimiques "classiques" composés de une ou deux lettres; la première lettre est obligatoirement en majuscule et la seconde, si elle existe, est en minuscule:

CO = carbone + oxygène
Co = cobalt

bien entendu chaque atome peut être affecté d'un facteur de répétition ce qui permet de rentrer des formules telles que:

H2O ou C6H6 ou CH3COOH équivalemt à CH3CO2H

une imperfection de ce LEX fait que le facteur de répétition, pour chaque occurence, est limité à (10^12)-1. Si ceci vous gêne vous pouvez répeter l'atome autant de fois qu'il sera nécessaire (je serai dans ce cas curieux de voir la formule ...).

Pour la chimie organique il est possible d'utiliser les parenthèses des formules semi-developpées, par exemple:

CH3(CH2)5CH3 que je vous laisse le plaisir de nommer!

Malheureusement, il n'est pas possible d'imbriquer les parenthèses ...

CH3(CH2)(CH2)4CH3 est correct
CH3(CH2(CH2)4)CH3 ne l'est pas

Pour conclure, je vous signale que ce LEX est très loin d'être optimisé et je vous invite, par conséquent, à vous pencher sur son cas !

Je précise enfin que, pour moi, la chimie relève d'un passé déjà lointain et que je dois beaucoup aux conseils prodigués par Eric Angelini.

Janick Taillandier (P 246, CHHU 176)

```
        LEX     'MW'

* Janick Taillandier 1986
*
* Change history
* 1.3 version w/o left & right parenthesis
        *               86/03/15
* B: as 1.3 + parenthesis ie (CH3)2CCOOH 86/03/30
*

        ID      #5C
        MSG     0
        POLL    poll

        ENTRY   mass
        CHAR    #F
        KEY     'MOLEWT'
        TOKEN   1
        ENDTXT


in(     EQU     0
FUNCDO  EQU     #2F888
```

```
RANGE  EQU    #1B07C
ARGERR EQU    #0BF19
REVPOP EQU    #0BD31
DRANGE EQU    #1B076
STSCR  EQU    #0E92C
RCSCR  EQU    #0E954
SPLITA EQU    #0C6BF
SPLITC EQU    #0C940
AD2-15 EQU    #0C363
MP2-12 EQU    #0C432
FLOAT  EQU    #1B322
uRES12 EQU    #0C994
FNRTN1 EQU    #0F216
MP2-15 EQU    #0C43A

       EJECT

poll   ?B=0   B
       GOYES  hVER$0
       RTNSXM              no other polls
hVER$0 C=R3                handle VER$
       D1=C
       A=R2
       D1=D1- (VER$en)-(VER$st)-2
       CD1EX
       ?A>C   A
       GOYES  hVER$1
       D1=C
       R3=C
VER$st LCASC  ' MW:B'
VER$en DAT1=C (VER$en)-(VER$st)-2
hVER$1 RTNSXM




       NIBHEX 411
mass   CD0EX
       D0=(5) (FUNCD0)
       DAT0=C A           save D0 in FUNCD0
       ST=0   in(
       A=0    W
       B=0    W
       GOSUB  stscr
       GOSBVL =REVPOP     pop & reverse string
       C=0    W           for CSRB
       C=A    A           get length in nibbles
       CSRB               length in bytes
       ?C#0   A
       GOYES  mass01
argerr GOVLNG ARGERR
mass01 C=C-1  A           count 1st char
       R0=C               length in R0
mass11 A=DAT1 B
       GOSUB  Arange
       GONC   mass12
       LCASC  '('
       ?C=A   B
       GOYES  mass13
       LCASC  ')'
       ?C=A   B
       GOYES  mass14
       GOTO   argerr      if not in A..Z

mass13 GOSUB  LP
       D1=D1+ 2
       GOTO   mass11

mass14 P=     1
       GOSUB  mass40
       GOSUB  mass+
       GOTO   RP

mass12 R1=A               save char in R1
       C=R0               restore count
       ?C#0   A
       GOYES  mass20      not yet finished
       P=     1
       GOTO   mass90      find component and exit

mass20 R0=C               save count
       D1=D1+ 2
       A=DAT1 B           read next char
       GOSBVL =DRANGE
       GONC   mass21      it is a digit
       GOSUB  Arange
       GONC   mass25      it is in A..Z
       LCASC  'za'
       GOSUB  range
       GONC   mass26      it is in a..z
       LCASC  '('
       ?A=C   B
       GOYES  mass25
       LCASC  ')'
       ?A=C   B
       GOYES  mass28
       GOTO   argerr      otherwise 'Invalid Arg'

mass21 P=     1
```

```
        GOTO   mass34   process digit after
          *             processing mass

mass25  P=     1
        GOSUB  mass40
        GOSUB  mass+
        GOSUB  cnt-1
        GOTO   mass11

mass26  C=R1           uppercase + lowecase
          *             component
        CSL    W
        CSL    W
        C=A    B
        R1=C           new character appended
        GOSUB  cnt-1   update count
        ?C#0   A
        GOYES  mass31
        P=     3       # of nibbles in comp name
        GOTO   mass90

mass28  P=     1
        GOSUB  mass40
        GOSUB  mass+
        GOSUB  cnt-1
        GOTO   RP

mass31  D1=D1+ 2
        A=DAT1 B       read new char
        GOSUB  drange
        GONC   mass32  in 0..9
        GOSUB  Arange
        GONC   mass33
        LCASC  '('
        ?A=C   B
        GOYES  mass3a
        LCASC  ')'
        ?A=C   B
        GOYES  mass3b
        GOTO   argerr  otherwise 'Invalid Arg'

mass3a  P=     3
        GOSUB  mass40
        GOSUB  mass+
        GOSUB  LP
        D1=D1+ 2
        GOTO   mass11

mass3b  P=     3
        GOSUB  mass40
        GOSUB  mass+
        GOSUB  cnt-1
        GOTO   RP

mass33  P=     3
        GOSUB  mass40  return mass in C[W]
        GOSUB  mass+   update total
        GOSUB  cnt-1
        GOTO   mass11  process new elt

mass32  P=     3
mass34  GOSUB  mass40
        R2=C           save mass in R2
        GOSUB  dig
        ?C=0   A
        GOYES  mass37  end of string
        GOSUB  Arange
        GONC   mass38
        LCASC  '('
        ?A=C   B
        GOYES  mass35
        LCASC  ')'
        ?A=C   B
        GOYES  mass36
        GOTO   argerr

* dig<12: test if number of digits as counted in
          *            D[A]
*         is <= 12.
*         RTN if ok, otherwise error exit

dig<12  C=0    A
        LCHEX  C       12
        ?D<=C  A
        RTNYES
        GOTO   argerr

mass35  GOSUB  dig<12
        GOSUB  massad
        GOSUB  LP
        D1=D1+ 2
        GOTO   mass11

mass36  GOSUB  dig<12
        GOSUB  massad
        GOTO   RP

mass38  GOSUB  dig<12
        GOSUB  massad
        GOTO   mass11  process new component
```

```
mass37 GOSUB  dig<12
       GOSUB  massad
       GOTO   mass80   exit...

mass90 GOSUB  mass40
       GOSUB  mass+    add it to total
*                      fall into mass80

* Exit

mass80 ?ST=0  in(
       GOYES  mass81
       GOTO   argerr   we have no ) before end
mass81 GOSUB  rcscr    result in C/D
       D0=(5) (FUNCD0)
       A=DAT0 A
       D0=A            restore D0
       D1=D1+ 2        D1 @ end of string
       BCEX   W        from XYEX
       CDEX   W
       BCEX   W
       ACEX   W        C/D to A/B
       GOSBVL =uRES12  pack result
       GOVLNG =FNRTN1

RP     ?ST=1  in(      already found ( ?
       GOYES  RP01
       GOTO   argerr   no, then error
RP01   ST=0   in(      no more in ()
       C=R0
       ?C#0   A
       GOYES  RP02     still chars after )
       GOSUB  add
       GOTO   mass80
RP02   D1=D1+ 2
       A=DAT1 B        what is after )
       GOSUB  drange
       GONC   RP03     in 0..9
       GOSUB  Arange
       GONC   RP04     in A..Z
       LCASC  '('
       ?A=C   B
       GOYES  RP04     we have '..)(..'
       GOTO   argerr   otherwise error

RP04   GOSUB  add
       GOSUB  cnt-1
       GOTO   mass11

RP03   GOSUB  dig       read number
       ?C=0   A
       GOYES  RP051     nothing after last digit

       GOSUB  endRP     process (
       GOTO   mass11    go to next component

RP051  GOSUB  endRP
       GOTO   mass80

LP     ?ST=0  in(
       GOYES  LP01      not already inside ()
       GOTO   argerr    otherwise error
LP01   ST=1   in(
       A=0    W
       B=0    W
       GOSUB  stscr
       GOSUB  cnt-1
       ?C#0   A
       RTNYES
       GOTO   argerr    they have written '...('

* endRP: sub-prgm for byte savings only ....

endRP  GOSUB  dig<12
       SETDEC
       A=B    W
       GOSBVL =FLOAT    result in A
       GOSBVL =SPLITA
       GOSUB  rcscr
       GOSBVL =MP2-15
       GOSUB  stscr
       GOSUB  add       add returns in HEX mode
       RTN

* add: add 2 15-dig numbers on top of scratch
*      scratch math stack and push result on stack

add    GOSUB  rcscr     pop 1st element on top of
         *               stack
       BCEX   W         from XYEX
       CDEX   W
       BCEX   W
       ACEX   W
       GOSUB  rcscr     pop second element
       SETDEC
       GOSBVL =AD2-15   add them
       SETHEX
       GOSUB  stscr
       RTN
```

```
* dig: process string of digit
* in:  D1 @ first digit
*      # of characters left in R0
*
* out: C=0 A end of string
*      C#0 A non digit char in A[B]
*      both cases number (DEC) in B[W]

dig    B=0    W        prepare image of count
       D=0    A        prepare digit counter
       GOSUB  cnt-1
       A=DAT1 B        read char
dig01  P=     0        for B=A P
       BSL    W
       B=A    P
       D=D+1  A        update digit count
       C=R0
       ?C=0   A
       RTNYES
       GOSUB  cnt-1
       D1=D1+ 2        point to new character
       A=DAT1 B
       GOSUB  drange   use C[A]
       GONC   dig01    character in 0..9
       RTN

drange GOVLNG =DRANGE

Arange LCASC  'ZA'
range  GOVLNG =RANGE

stscr  GOSBVL =STSCR
       P=     0
       RTN

rcscr  GOSBVL =RCSCR
       P=     0
       RTN

* cnt-1

cnt-1  C=R0
       C=C-1  A
       R0=C
       RTN

* massad: multiply mass in R2 by count in B[W]
*         and add to running total on top of
*              scratch stack

massad SETDEC
       A=B    W
       GOSBVL =FLOAT
       C=R2
       GOSBVL =MP2-12
       GOTO   mass+1

* mass+ : add mass in C[W] to running total
*         on top of math stack.
* use: A,B,C,D,D0,P

mass+  SETDEC
       A=C    W        copy mass in A[W] for
         *              addition
       GOSBVL =SPLITA  in 15-digit format in A[W]
mass+1 GOSBVL =RCSCR   recall top of scratch in
         *              C[W]
       GOSBVL =AD2-15  add them
       GOSBVL =STSCR   push them again
       P=     0
       SETHEX
       RTN

* mass40: get mass from component name
* entry: name in R1[WP] (P=1 or P=3)
* exit: mass in C[W]
* use: A[W],C[W],D0

mass40 A=R1            recover name
       ?P=    1
       GOYES  srch10
       GOTO   srch20
srch10 GOSUB  srch11
       LIST   OFF
       NIBASC 'B'
       NIBHEX 100000000001801
       NIBASC 'C'
       NIBHEX 100000000011021
       NIBASC 'F'
       NIBHEX 100000030489981
       NIBASC 'H'
       NIBHEX 000000000097001
       NIBASC 'K'
       NIBHEX 100000000389093
       NIBASC 'N'
       NIBHEX 100000000760041
       NIBASC 'O'
       NIBHEX 100000000499951
       NIBASC 'P'
```

```
        NIBHEX 100000006737903
        NIBASC 'S'
        NIBHEX 100000000006023
        NIBASC 'U'
        NIBHEX 200000000920832
        NIBASC 'V'
        NIBHEX 100000000414905
        NIBASC 'W'
        NIBHEX 200000000058381
        NIBASC 'Y'
        NIBHEX 100000000950988
        NIBHEX 00
        LIST   ON
srch11  C=RSTK          get address
        D0=C
srch12  C=DAT0 2        get name
        ?A=C   WP
        GOYES  srch13
        ?C=0   WP
        GOYES  srch14
        D0=D0+ 2        past name
        D0=D0+ 15       past numeric value
        GOTO   srch12
srch14  GOTO   argerr
srch13  P=     0
        D0=D0+ 2
        C=0    W        prepare register
        C=DAT0 15       read mass
        P=     0
        RTN

* 2 letters components

srch20  GOSUB  srch21
        LIST   OFF
        NIBASC 'cA'
        NIBHEX 200000000000722
        NIBASC 'gA'
        NIBHEX 200000000868701
        NIBASC 'lA'
        NIBHEX 100000004518962
        NIBASC 'mA'
        NIBHEX 200000000000342
        NIBASC 'rA'
        NIBHEX 100000000084993
        NIBASC 'sA'
        NIBHEX 100000000612947
        NIBASC 'tA'
        NIBHEX 200000000000012
        NIBASC 'uA'
        NIBHEX 200000005669691
        NIBASC 'aB'
        NIBHEX 200000000033731
        NIBASC 'eB'
        NIBHEX 000000000812109
        NIBASC 'iB'
        NIBHEX 200000004089802
        NIBASC 'kB'
        NIBHEX 200000000000742
        NIBASC 'rB'
        NIBHEX 100000000040997
        NIBASC 'aC'
        NIBHEX 100000000008004
        NIBASC 'dC'
        NIBHEX 200000000014211
        NIBASC 'eC'
        NIBHEX 200000000021041
        NIBASC 'fC'
        NIBHEX 200000000000152
        NIBASC 'lC'
        NIBHEX 100000000035453
        NIBASC 'mC'
        NIBHEX 200000000000742
        NIBASC 'oC'
        NIBHEX 100000000233985
        NIBASC 'rC'
        NIBHEX 100000000069915
        NIBASC 'sC'
        NIBHEX 200000004509231
        NIBASC 'uC'
        NIBHEX 100000000064536
        NIBASC 'yD'
        NIBHEX 200000000005261
        NIBASC 'rE'
        NIBHEX 200000000062761
        NIBASC 'sE'
        NIBHEX 200000000000452
        NIBASC 'uE'
        NIBHEX 200000000069151
        NIBASC 'eF'
        NIBHEX 100000000074855
        NIBASC 'mF'
        NIBHEX 200000000000752
        NIBASC 'rF'
        NIBHEX 200000000000322
        NIBASC 'aG'
        NIBHEX 100000000002796
        NIBASC 'dG'
        NIBHEX 200000000052751
        NIBASC 'eG'
```

```
NIBHEX 100000000009527
NIBASC 'eH'
NIBHEX 000000000062004
NIBASC 'fH'
NIBHEX 200000000094871
NIBASC 'gH'
NIBHEX 200000000095002
NIBASC 'oH'
NIBHEX 200000004039461
NIBASC ' I'
NIBHEX 200000005409621
NIBASC 'nI'
NIBHEX 200000000028411
NIBASC 'rI'
NIBHEX 200000000022291
NIBASC 'rK'
NIBHEX 100000000000838
NIBASC 'aL'
NIBHEX 200000005509831
NIBASC 'iL'
NIBHEX 000000000001496
NIBASC 'rL'
NIBHEX 200000000000062
NIBASC 'uL'
NIBHEX 200000000079471
NIBASC 'dM'
NIBHEX 200000000000852
NIBASC 'gM'
NIBHEX 100000000050342
NIBASC 'nM'
NIBHEX 100000000083945
NIBASC 'oM'
NIBHEX 100000000004959
NIBASC 'aN'
NIBHEX 100000007798922
NIBASC 'bN'
NIBHEX 100000000460929
NIBASC 'dN'
NIBHEX 200000000042441
NIBASC 'eN'
NIBHEX 100000000097102
NIBASC 'iN'
NIBHEX 100000000000785
NIBASC 'oN'
NIBHEX 200000000000552
NIBASC 'pN'
NIBHEX 200000002840732
NIBASC 'sO'
NIBHEX 200000000002091
NIBASC 'aP'
NIBHEX 200000009530132
NIBASC 'bP'
NIBHEX 200000000002702
NIBASC 'dP'
NIBHEX 200000000004601
NIBASC 'mP'
NIBHEX 200000000000541
NIBASC 'oP'
NIBHEX 200000000000902
NIBASC 'rP'
NIBHEX 200000007709041
NIBASC 'tP'
NIBHEX 200000000090591
NIBASC 'uP'
NIBHEX 200000000000442
NIBASC 'aR'
NIBHEX 200000004520622
NIBASC 'bR'
NIBHEX 100000000876458
NIBASC 'eR'
NIBHEX 200000000702681
NIBASC 'hR'
NIBHEX 200000005509201
NIBASC 'nR'
NIBHEX 200000000000222
NIBASC 'uR'
NIBHEX 200000000070101
NIBASC 'bS'
NIBHEX 200000000057121
NIBASC 'cS'
NIBHEX 100000000955944
NIBASC 'eS'
NIBHEX 100000000006987
NIBASC 'iS'
NIBHEX 100000000558082
NIBASC 'mS'
NIBHEX 200000000004051
NIBASC 'nS'
NIBHEX 200000000096811
NIBASC 'rS'
NIBHEX 100000000002678
NIBASC 'aT'
NIBHEX 200000009749081
NIBASC 'bT'
NIBHEX 200000004529851
NIBASC 'cT'
NIBHEX 100000000000079
NIBASC 'eT'
NIBHEX 200000000006721
NIBASC 'hT'
```

```
        NIBHEX 200000001830232
        NIBASC 'iT'
        NIBHEX 100000000000974
        NIBASC 'lT'
        NIBHEX 200000000073402
        NIBASC 'mT'
        NIBHEX 200000002439861
        NIBASC 'eX'
        NIBHEX 200000000003131
        NIBASC 'bY'
        NIBHEX 200000000040371
        NIBASC 'nZ'
        NIBHEX 100000000008356
        NIBASC 'rZ'
        NIBHEX 100000000002219
        NIBHEX 0000     end of table
        LIST   ON
srch21  C=RSTK
        DO=C
srch22  C=DAT0 4        get name
        ?A=C   WP
        GOYES  srch23
        ?C=0   WP
        GOYES  srch24
        DO=DO+ 4
        DO=DO+ 15       past numeric value
        GOTO   srch22
srch24  GOTO   srch14   not in table
srch23  P=     0
        DO=DO+ 4        @ numeric value
        C=0    W
        C=DAT0 15
        P=     0
        RTN
        END
```

LE VER$ DANS LES POLL

L'interception de Poll est un miracle quotidien que le 71 accomplit, la plupart du temps, dans l'indifférence générale. Je dis la plupart du temps pour que l'auteur de KBEEP ne trouve pas prétexte à sévir de nouveau!. Pour l'amateur (exclusif) de BASIC il s'agit même certainement d'un point sans importance. Et bien les choses vont changer! HUGH ...

Dans un peu moins de 5 minutes vous saurez tous (j'ai bien dit TOUS, alors pas la peine de filer à l'anglaise et revenez vous asseoir, non mais!), vous saurez donc tous modifier un Lex quelles que soient vos connaissances en assembleur. En dehors de la joie immense que cela vous procurera (si si ...), il est même possible que ça vous serve pour vos programmes. En bref le pied, le flash, la panacée, un petit bout de paradis pour 35 Fr par mois chez tous les bons libraires. Dites merci dans le micro. Afin de récompenser les braves qui sont encore en ma compagnie, je vais même vous dire de quoi il s'agit.

Avec l'été les Lex fleurissent dans JPC, bien sûr vous avez trouvé parmi ceux-ci la fonction dont vous n'osiez même pas rêver il y a un mois. Avec le programme MLEX vous vous empressez de la compiler aux autres dans le Lex utilitaire que vous vous êtes taillé sur mesure (veinard). Seulement voilà : comment vos programmes -BASIC- vont-ils savoir si ce Lex est, ou non, en machine? Très simple et plus fiable que ADDR$, lavant plus blanc et pour moins cher, il ... pardon je m'égare ! Reprenons. Nous allons intercepter (à vos postes) le Poll (mission de routine) pVER$. Cela va nous permettre de renvoyer dans la chaîne VER$ un intitulé identifiant sans ambigüité notre Lex. Voici la manière de procéder.

En tête de chaque Lex vous rajouterez la séquence:

```
        POLL   POLHND
         .
         .
POLHND  ?B=0   B        VER$ poll ?
        GOYES  pXC10
        GONC   pEND
pXC10   C=R3
        D1=C
        A=R2
        D1=D1- (VER$en)-(VER$st)-2
        *      Longueur de l'intitulé du Lex
        CD1EX
```

```
        ?A>C   A        reste assez de mémoire?
        GOYES  pEND     non
        D1=C            pointeur OK pour
                        écriture mémoire.
        R3=C
VER$st  LCASC  ' NOM:A '
VER$en  DAT1=C (VER$en)-(VER$at)-2
pEND    RTNSXM          retour

* En entrée:  B[A]= #poll ; R3[A]= Stack Pointer
*             R2[A]= (AVMEMS)
```

Remplacez ' NOM:A ' par l'intitulé désiré, sans dépasser 5 caractères (nous ne sommes pas les seuls à intercepter ce poll), et sans oublier un espace avant et après. Dans un programme vous utiliserez la fonction:

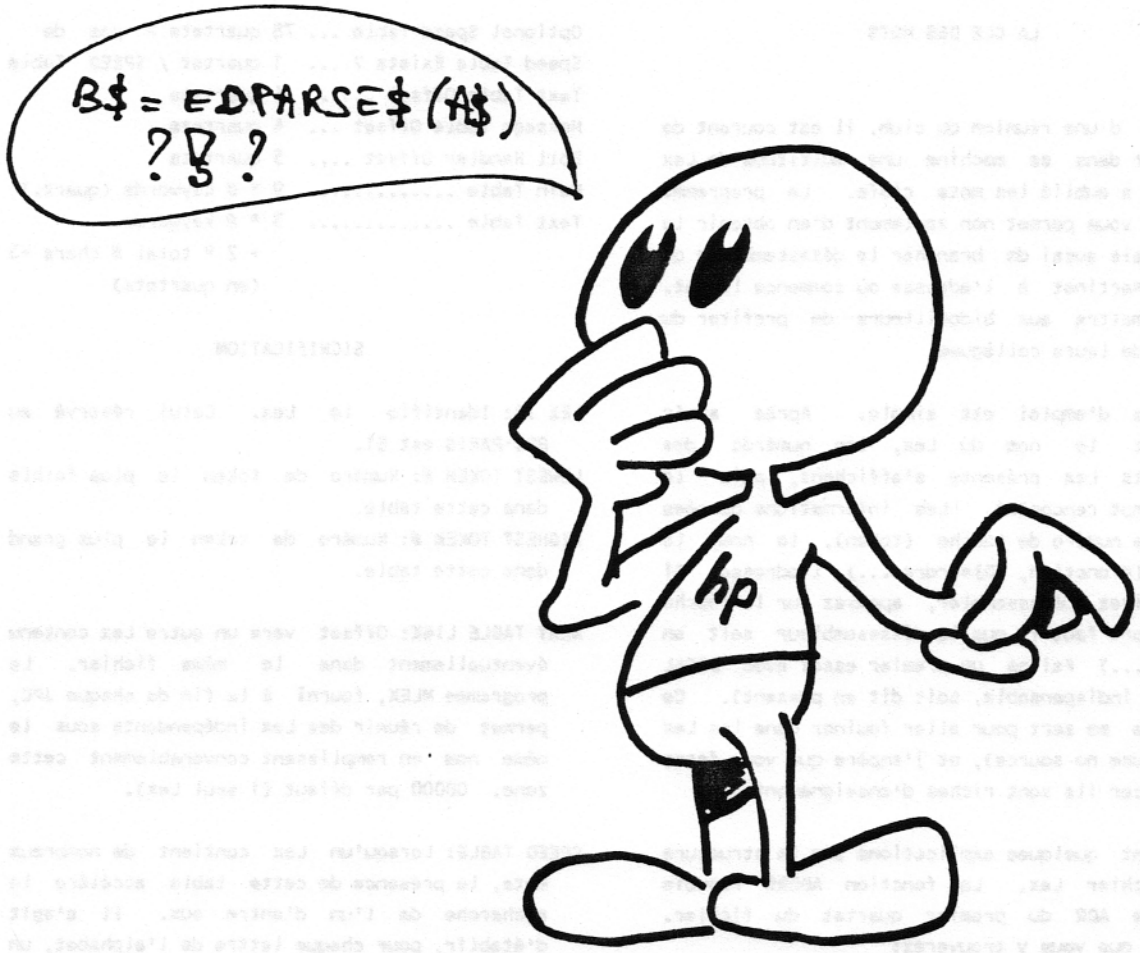
```
IF POS(VER$,' NOM:A ') THEN ...
```

en reprenant l'intitulé de notre exemple. N'abusez pas trop du procédé, mais regoupez plutôt vos routines au sein d'un même Lex.

Heureuse programmation!

Jean-Pierre BONDU (PPC 33, SIG 3)

```
DEF KEY 'fW',CHR$(197);
DEF KEY 'fE',CHR$(193);
DEF KEY 'fR',CHR$(201);
DEF KEY 'fY',CHR$(203);
DEF KEY 'fU',CHR$(195);
DEF KEY 'fI',CHR$(209);
DEF KEY 'fO',CHR$(194);
DEF KEY 'f7',CHR$(155);
DEF KEY 'f8',CHR$(124);
DEF KEY 'f9',CHR$(95);
DEF KEY 'f/',CHR$(92);
DEF KEY 'fA',CHR$(192);
DEF KEY 'fS',CHR$(200);
DEF KEY 'fD',CHR$(205);
DEF KEY 'fJ',CHR$(207);
DEF KEY 'fK',CHR$(221);
DEF KEY 'fL',CHR$(206);
DEF KEY 'f*',CHR$(160);
DEF KEY 'fC',CHR$(181);
DEF KEY 'f1',CHR$(96);
DEF KEY 'f-',CHR$(126);
```



BASIC

LA CLE DES MOTS

A la fin d'une réunion du club, il est courant de retrouver dans sa machine une multitude de Lex dont on a oublié les mots clefs. Le programme SCANALEX vous permet non seulement d'en obtenir la liste, mais aussi de brancher le désassembleur de Michel Martinet à l'adresse où commence le mot. Cela permettra aux bidouilleurs de profiter du travail de leurs collègues.

Le mode d'emploi est simple. Après avoir introduit le nom du Lex, les numéros des différents Lex présents s'affichent, puis le premier mot rencontré. Les informations données sont: le numéro de touche (token), le nom, le type ([F]=fonction, [D]=ordre ...), l'adresse. Si vous désirez désassembler, appuyez sur la touche [=] (encore faut-il que le désassembleur soit en mémoire!...) Faites un premier essai avec DESAL (qui est indispensable, soit dit en passant). Ce programme me sert pour aller fouiner dans les Lex HP (because no source), et j'espère que vous ferez de même car ils sont riches d'enseignements.

Maintenant quelques explications sur la structure d'un fichier Lex. La fonction ADDR$ renvoie l'adresse ADR du premier quartet du fichier. Voici ce que vous y trouverez:

FILE HEADER

File Name	16 quartets
File Type	4 quartets
Flags	1 quartet
Copy Code	1 quartet
Creation Time	4 quartets
Creation Date	6 quartets
File Chain Lenght	5 quartets
----------	-------
TOTAL	37 quartets

Dans un Lex on trouve ensuite:

LEX IDentity	2 quartets
Lowest Token #	2 quartets
Highest Token #	2 quartets
Next LEX Table Link	5 quartets
Speed Table Exists ? ...	1 quartet \ 1 quartet si
Optional Speed Table ...	78 quartets > pas de
Speed Table Exists ? ...	1 quartet / SPEED Table
Text Table Offset	4 quartets
Message Table Offset ...	4 quartets
Poll Handler Offset	5 quartets
Main Table	9 * # keywords (quart.)
Text Table	3 * # keywords + 2 * total # chars +3 (en quartets)

SIGNIFICATION

LEX ID: Identifie le Lex. Celui réservé au PPC-PARIS est E1.

LOWEST TOKEN #: Numéro de token le plus faible dans cette table.

HIGHEST TOKEN #: Numéro de token le plus grand dans cette table.

NEXT TABLE LINK: Offset vers un autre Lex contenu éventuellement dans le même fichier. Le programme MLEX, fourni à la fin de chaque JPC, permet de réunir des Lex indépendants sous le même nom en remplissant convenablement cette zone. 00000 par défaut (1 seul Lex).

SPEED TABLE: Lorsqu'un Lex contient de nombreux mots, la présence de cette table accélère la recherche de l'un d'entre eux. Il s'agit d'établir, pour chaque lettre de l'alphabet, un offset qui pointera vers le premier mot commencant par cette lettre. Un offset occupe 3 quartets, il y a 26 lettres dans l'alphabet (j'ai vérifié !), soit un encombrement de 3*26=78 quartets. Ok ?

TEXT TABLE OFFSET: Pointe vers le second quartet de la Text Table. Le premier indique la longueur de la chaine en quartets -1.

MESSAGE TABLE OFFSET: Pointe vers la zone contenant les messages. Voir FINDLEX de Jean-Jacques MOREAU dans le JPC triple 31-32-33 de Février/Mars/Avril. 0000 par défaut.

POLL HANDLER OFFSET: Du même tonneau que Message Offset.

MAIN TABLE: Contient les informations nécéssaires à l'éxécution et à la décompilation de chaque mot. Les entrées sont rangées par ordre de token croissant. Chaque entrée est longue de 9

quartets, décomposés comme suit:

TEXT TABLE OFFSET(3): Ajouté à l'adresse de la Text Table, il pointe le quartet de longueur de l'intitulé du mot. L'intitulé suit immédiatement ce quartet. Par exemple le mot "ABC" sera codé: 5142434. 5 donne la longueur en quartets-1, 14 doit se lire 41 hexa = 65 = "A", et ainsi de suite. En faisant le calcul indiqué on pointerait sur le 5 ('nibble count' en anglais).

EXECUTION ADDRESS(5): Offset par rapport à l'adresse courante donnant le début du code éxécutable.

CHARACTERIZATION NIBBLE(1): Type du mot suivant le canevas suivant:

- bit 0: légal depuis le clavier.
- bit 1: inutilisé
- bit 2: légal après THEN/ELSE
- bit 3: programmable

TEXT TABLE: Contient l'intitulé des tokens rangés par ordre alphabétique.

SIZE OF TEXT-1(1): C'est le 'nibble count' vu plus haut.

TEXT: 2-8 octets. Un mot doit faire au moins 2 caractères pour ne pas entrer en conflit avec une variable.

TOKEN # (2): Token du mot.

J'espère que ce large extrait des IDS I vous sera profitable car il m'a demandé pas mal de temps ... au niveau de la mise en page.
Bonnes Vacances!

Jean-Pierre BONDU (PC 33, SIG 3)

VECTEURS SANS MODULE

Voici un programme sur les vecteurs pour ceux qui ne possèderaient pas le module math (?)...

Ce programme n'est pas de moi, car pris dans "Users'Library Solutions", avec toutefois quelques modifications.

OPERATIONS SUR UN VECTEUR

N : norme
M : multiplie le vecteur par un scalaire (nombre)
U : calcule le vecteur unitaire d'un vecteur, c'est à dire le vecteur ayant même sens, même direction, dont la norme est égale à 1.
C : conversion RECTANG/POLAIRE et vice-versa

OPERATION SUR DEUX VECTEURS

A : additionne deux vecteurs
S : soustrait deux vecteurs
X : multiplie deux vecteurs
I : angle de deux vecteurs
D : produit scalaire

AUTRE

F : permet de changer de format E/S
A : fin

REFERENCES :

| Salas, S. and Hille, E.; "Calculus", XEROS
| Hudson, R.; "The engineer's MANUAL", Wiley
| and sons, Inc.

Tony GUILLOUX

LA PENSEE DU MAITRE SHADOCK

Le Master Mind tout le monde connait, et je me demande si tout le monde ne commence pas à s'en lasser? Mais comme il faut bien se remuer un peu les méninges, je vous propose un nouveau jeu.

Ce jeu est une variante du Master Mind (toujours lui) que l'on doit à Mr George Noël. En voici les régles :

vous devez devinez un entier, par exemple de 4 chiffres, que notre machine a "choisi", vous devez alors proposer un nombre, et le 71 vous répondra par un nombre de la forme 'x,y' où 'x' désigne le nombre de chiffres du nombre qu'a choisi le 71 qui sont stictement supérieur au chiffre de même rang du nombre que vous avez proposé et 'y' désigne le nombre de ces chiffres qui sont strictement inférieur. Un exemple éclaircira tout!

Le nombre que vous devez trouver est 4621 et vous proposez 1979, la réponse de la machine sera :

1,3
car 1 de vos chiffre est trop petit et les 3 autres trop grands.

Si vous désirez voire les coups précédents, appuyez sur la flèche qui descend (#51), pour revenir appuyez sur la flèche qui monte (#50). Essayez au début avec 2 chiffres, puis avec 3, le maximum est à neuf, mais alors là, il vaut mieux être accompagné!

Le nombre d'essais autorisés est fixé à 3*N, N étant le nombre de chiffres dans le nombre à trouver.

Alain Herreman

ECIRTAM

A quand remonte le dernier programme d'inversion de matrices paru dans JPC ? A cette question je crois pouvoir répondre : février 1985 ! Eh ! oui, cela fait bien plus d'un an.

Je me permettrai donc de rappeler ce vieil adage, qu'un ancien de JPC me remémora :

"JPC, journal toujours à la pointe, se doit d'avoir, dans ses célèbres colonnes où tout les styles se côtoient, un programme d'inversion de matrices à hautes performances" (relisez bien cette phrase elle frise la poésie).

Gauss ! Tout ceux qui ont les bras leviers pour avoir pensé à la méthode du pivot peuvent se rasseoir, ils ont tout faux !

L'algorithme que je vous propose n'est peut être pas connu de tous, il est pourtant très simple.

Soient A la matrice à inverser et I la matrice identité de même dimension que A. Si vous faites des opérations sur les colonnes de A de manière à obtenir la matrice indentité, en faisant ces mêmes opérations sur la matrice I, quand A sera la matrice identité, I sera la matrice inverse de A. Ce n'est pas magique, il existe même une démonstration ...

Il ne me reste plus qu'à vous dire que pour inverser une matrice 10*10 ce programme prend 1 minute. Je dois avouer que j'ai vraiment l'impression qu'avec ce programme j'ai aidé à l'amélioration des conditions de travail du taupin français .

Alain Herreman

PERI FEERIE

PERIFLAG est un groupe d'utilitaires (à vous de leur trouver une utilité...) portant sur les flags et les périphériques.

PERITAB charge un tableau de 128 entiers avec les adresses des périphériques ayant les ID 0 à 127. Ainsi, l'adresse du premier lecteur de cassettes sera A(16).

PERITEST attribue un flag à chacun des "device word" (tape, printer...). Ce flag est allumé s'il existe sur la boucle un appareil de type correspondant, éteint sinon. Le paramètre L (pouvant être choisi entre 0 et 52) est la limite inférieure du groupe de 11 flags utilisés. Après éxécution de PERITESY, on pourra vérifier la présence d'un type d'appareil simplement en testant le flag correspondant.

XF et FX sont équivalents à X<>F (extension de fonctions de la HP-41). FX sauve l'état des flags 0 à 7 dans une variable X, et XF restitue cet état.

RFLAG permet aux flags 0 a 4 de "refléter" l'état de tout autre groupe de 5 flags, défini par sa limite inférieure.

Jean-Michel Kefaloucos (PPC#140)

S C R A B B L E

Voici un programme qui vous servira à tenir le décompte des points de chaque joueur durant une partie de Scrabble.

Après avoir fait RUN, le programme demande le nombre de joueurs et propose une valeur par défaut (modifiable en ligne 140), puis le nom de ces joueurs (5 caractères maxi) à rentrer dans l'ordre dans lequel ils joueront.

En réponse à la question "Score de ...", il est possible de répondre :
- le score de celui-ci !
- "+" qui affiche les scores de chacun avec leur moyennes
- RETURN pour finir la partie

"+" et RETURN ne peuvent être utilisés qu'après un tour complet.

A la fin de la partie, le programme demande la valeur des points restants de chaque joueur, les totalise et les ajoute au score du gagnant, comme cela est prévu dans les règles du jeu.

Les résultats, ainsi que les moyennes, sont alors affichés ou imprimés par ordre décroissant, selon qu'une imprimante est connectée sur la boucle ou non. Dans ce cas, le programme précise pour chaque joueur le score de son meilleur coup ainsi que le plus mauvais...

Christophe SIMON (P 160. T 516)

LE MECHANT pi.

Bonjour tout le monde, bonjour M Connan, voici la débilité du mois.

Je suppose que vous connaissez tous par coeur votre formule de Méchain, et que vous savez donc tous que PI/4=4*ATAN(1/5)-ATAN(1/239)...

De plus, vous savez tous que le développement de arc tg est:

```
ATAN(x)=x-x^3/3+x^5/5-x^7/7+...
                          +(-1)^n*x^(2*n+1)/(2*n+1)+...
```

On mélange tout çà, on agite, on laisse reposer une demi-journée.
Vous avez deviné où je veux en venir: un superbe programme qui calcule pi avec presque

autant de décimales que dans vos livres de math habituels (dans "les nombres et leurs mystères" d'André Warusfel, on trouve 1000 décimales).

Avec ces deux formules, on définie ainsi la suite U(n):

```
U(n)=U(n-1)+4*(-1)^n
        /(2*n+1)*(4*(1/5)^(2*n+1)-(1/239)^(2*n+1))
```

La suite U(n) converge ainsi vers pi.
Le programme est très simple (quand il est commenté); les variables sont:

A(0,) tableau contenant 4*(1/5)^(2*n+1)
A(1,) tableau contenant (1/239)^(2*n+1)
A(2,) tableau zone de travail
U() tableau contenant U(n)
X nb de décimales à calculer plus 1 (dimension des tableaux)
N indice 2*n+1
S signe (-1)^n
D diviseur pour 'DIV'
I indice du tableau à diviser ('DIV')
J indice d'opération
K sauvegarde momentanée (ligne 110)
R retenue d'opération
C variable réelle utilisée par 'DIV'

Pour lancer le programme faites RUN, entrez le nombre de décimales recherchées, et armez vous d'un peu de patience et d'un bazooka si votre 71 n'est pas d'accord avec Warusfel...

Ce programme calcule 50 décimales en moins de 23'40"05 (23'40"04 précisément). J'avoue avoir eu la flemme de mettre ce programme sur HP-15C pour plusieurs raisons.
La première étant que je n'ai pas d'HP-15C, la seconde étant que je n'ai pas envie d'attendre la fin de mes jours pour qu'une HP-15C me donne 50 décimales (si le 71 met 20 minutes, cela m'étonnerai que la 15 mette moins d'une journée, la troisième étant que la mémoire de base du 71 le limite à une centaine de décimales, et que je connais par coeur l'éventuelle dizaine de décimales que pourrait me donner la 15C !

Ceci étant dit, je me retire en vous laissant un dernier programme qui vous donne pi en multiprécision, qui lui est beaucoup plus rapide et très facilement adaptable sur HP-15C.

Vous le commenterez vous-même (les insultes sont à envoyer au journal qui me les transmettera, et les félicitations sont à m'adresser directement).

Serge Vaudenay.

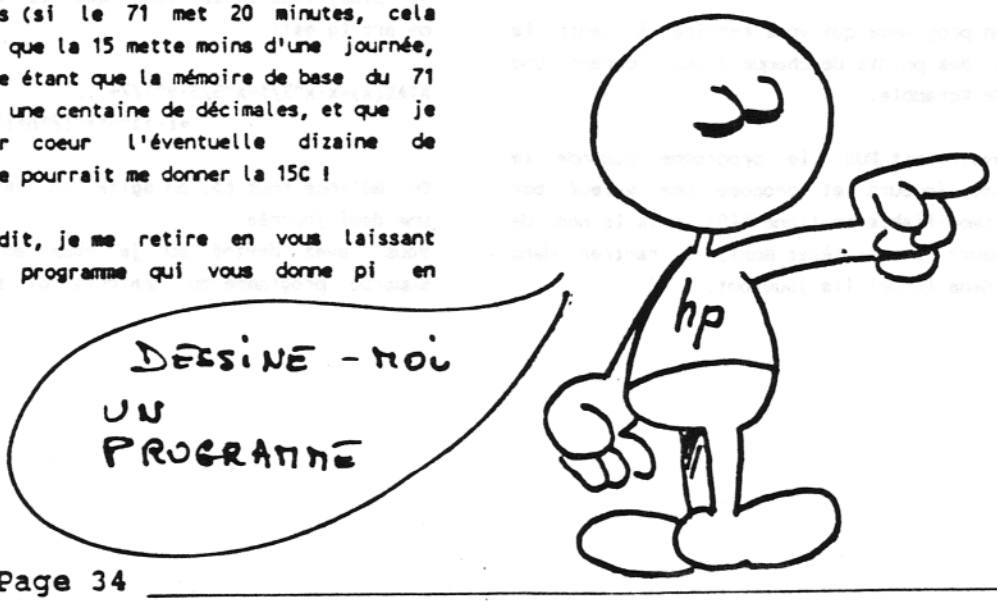
NdR : Pour une information complète relative à PI, on consultera avec bonheur l'excellent ouvrage réalisé par la revue Euréka. Plusieurs centaines de pages regroupent tout ce que les lecteurs connaissaient (et ils étaient nombreux).

A LA CONQUETE DU GRAPH

Il est toujours intéressant de lire des programmes que l'on n'a pas écrits, aussi en feuilletant l'excellent livre "Techniques de Basic" paru aux Editions Mémoire Vive de Guillo et Robertson, cela m'a donné l'idée d'écrire le programme GRAPH1.

Si vous aimez les chiffres, les diagrammes, les statistiques et autres tableaux récapitulatifs sur l'année, alors n'hésitez plus et jetez un coup d'oeil sur ce programme qui m'a permis de faire apparaitre le graphique que j'ai intitulé : "Niveau de la consommation mensuelle en kf", il ne reste plus qu'à en faire l'analyse...

Jean-Claude FOURES



"FORMAT" formateur de texte. Nécessite FORMALEX

```
================================================================
1000 SUB FORMAT(F$,N) @ DIM A$[256],B$[256+N]
1010 SFLAG -1 @ PURGE FTXT @ CFLAG -1 @ CREATE TEXT FTXT @ ASSIGN #1 TO F$ @ ASSIGN #2 TO FTXT
1020 P=0 @ ON ERROR GOTO 1050 @ READ #1,0;A$ @ OFF ERROR @ IF A$[1,1]="\" THEN 1060
1030 IF A$[1,1]="^" THEN A$=CENTER$(REDUCE$(A$[2]),N)
1040 DELETE #1,0 @ PRINT #2;A$ @ GOTO 1020
1050 PURGE F$ @ RENAME FTXT TO F$ @ END
1060 A$=A$[2] @ IF A$[1,1]="^" THEN A$=A$[2] @ P=CEIL(N/10)
1070 B$="" @ CFLAG 5 @ GOTO 1090
1080 READ #1,0;A$
1090 DELETE #1,0 @ A$=REDUCE$(A$)
1100 IF A$[LEN(A$)]="\" THEN SFLAG 5 @ A$=A$[1,LEN(A$)-1]
1110 IF NOT LEN(A$) THEN 1140 ELSE B$=B$&" "&A$
1120 A$="" @ IF LEN(B$)<=N-P THEN 1140 ELSE A$=B$[CESURE(B$,N-P)+2] @ B$=B$[1,CESURE(B$,N-P)]
1130 B$=SPACE$(P)&FORMAT$(B$,N-P) @ PRINT #2;B$ @ B$=A$ @ P=0 @ IF LEN(B$)>N THEN 1120
1140 IF FLAG(5) THEN PRINT #2;SPACE$(P)&REDUCE$(B$) @ GOTO 1020 ELSE 1080
```

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

"SCANALEX" pour scruter les LEX. Nécessite DESAL

```
   - PEEK$,HTA$,REV$ (=DESAL) + HPILROM
  10 INPUT "Lex File ";F$ @ CALL SCANALEX(F$)

================================================================
1000 SUB SCANALEX(F$)
1010 ON ERROR GOTO 1260
1020 A=HTD(ADDR$(F$))+37 @ B=A @ OFF ERROR
   - A et B @ LEX ID#
1030 N$=PEEK$(DTH$(A-21),4)
   - N$ = File Type
1040 IF DTH$(BINAND(HTD(N$),HTD('802E')))#'0802E' THEN DISP MSG$(63) @ END
   - LEX File Type =  '802E'  ,  '902E'  ,  'A02E'  ,  'B02E'
     suivant état     NORMAL  ,  SECURE  ,  PRIVATE ,  S & P
1050 DELAY 0 @ DISP @ DISP F$;TAB(12);"LEX#: ";

================================================================
1060 'CNTLEX': DISP REV$(PEEK$(DTH$(B),2));" ";
   - Affiche l'ID du LEX pointé par B
1070 B=B+6 @ Z=HTD(REV$(PEEK$(DTH$(B),5)))
   - Z= Next LEX Table Link : un autre LEX est-il linké ?
1080 IF Z THEN B=B+Z @ GOTO 'CNTLEX' ELSE WAIT 1 @ DISP
   - oui: on pointe ce LEX (B=B+offset) et on recommence (CNTLEX)

================================================================
1090 'MAINLOOP': DISP @ DISP REV$(PEEK$(DTH$(A),2)) @ WAIT .5
   - Tous les Lex ID ont été affichés, on va maintenant
     lister les mots contenus dans chaque Lex.
1100 B=A+12 @ IF PEEK$(DTH$(A+11),1)='0' THEN B=B+79
   - Si la Speed Table éxiste (longue de 78+1 quartets), on la dépasse.
1110 T=B+HTD(REV$(PEEK$(DTH$(B),4)))
   - B pointe le Text Table Offset
     T pointe la Text Table (= intitulé des tokens).
```

```
1120 M=B+13
   - M pointe la Main Table

============================================================
1130 'KEYLIST': FOR I=0 TO 255
1140 N$=PEEK$(DTH$(T-1),19)
   - N$[1,1]=     longueur de l'intitulé -1 en quartets =x-1
     N$[2,x+1]=   intitulé en ASCII (de 2 à 8 caractères = 4 à 16 quartets)
     N$[x+2,x+3]= Token# du mot
1150 IF N$[1,3]='1FF' THEN 'NXTLEX' ! Fin de la Text Table
1160 Z=HTD(N$[1,1])+1 @ T=T+Z+3
   - Z= longueur en quartets de l'intitulé
     T @ intitulé suivant dans la Txt Table
1170 N$=STR$(HTD(REV$(N$[Z+2,Z+3])))&" "&HTA$(N$[2,Z+1]) ! N$= token# + nom
1180 Z=M+9*I+3
   - 9*I= offset que l'on ajoute à l'ADR de Main Table (M)
     Z @ token dans la Main Table
1190 Z$=REV$(PEEK$(DTH$(Z),6))
   - Z$[1,1] = type du mot ; Z$[2,6]= offset to execution adress
1200 DISP N$;TAB(13);'['&Z$[1,1]&']';TAB(17);'#';DTH$(Z+HTD(Z$[2]))
   - Token   Nom   [Type]   #Point d'entrée
1210 K$=KEYWAIT$
1220 IF K$='=' THEN CALL DESAS(DTH$(Z+HTD(Z$[2]))) @ END
1230 NEXT I ! Token suivant

============================================================
1240 'NXTLEX': A=A+6 @ Z=HTD(REV$(PEEK$(DTH$(A),5)))
   - cf ligne 1060
1250 IF Z THEN A=A+Z @ GOTO 'MAINLOOP'
1260 END SUB



****************************************************************************************************

"VECTO" pour se passer du module MATH


  10 RESET @ RADIANS @ DIM A$[4],B$[11],D$[2],E$[4] @ B$='ASXIDNMUCFQ' @ D$='YN'
  20 ON TIMER #1,.1 GOTO 30 @ BYE
  30 OFF TIMER #1 @ CHARSET CHARSET$&CHR$(0)&CHR$(255)&CHR$(0)&CHR$(255)
  40 DISP 'Entree R/C ?';
  50 E$=KEYWAIT$ @ IF NOT POS('CR',E$) THEN 50
  60 DISP @ CFLAG 1 @ IF E$='C' THEN SFLAG 1
  70 F$='x=' @ G$='y=' @ IF FLAG(1) THEN F$='r=' @ G$='='
  80 DISP 'Sortie R/C ?';
  90 E$=KEYWAIT$ @ IF NOT POS('CR',E$) THEN 90
 100 DISP @ CFLAG 2 @ IF E$='C' THEN SFLAG 2
 110 H$='x=' @ I$='y=' @ IF FLAG(2) THEN H$='r=' @ I$=''
 120 DISP 'A S X I D N M U C F Q'
 130 E$=KEYWAIT$ @ A=POS(B$,E$) @ IF NOT A THEN 130
 140 IF A=11 THEN 'Q'
 150 IF A=10 THEN CFLAG 1,2,3 @ GOTO 40
 160 IF A#9 THEN 190
 170 IF NOT FLAG(1) EXOR FLAG(2) THEN DISP 'I/O Incorrect' @ BEEP @ WAIT 1 ELSE 190
 180 GOTO 40
 190 IF A<6 THEN 230
 200 IF FLAG(3) THEN 280
 210 DISP F$; @ INPUT '';C @ IF FLAG(1) AND C<0 THEN DISP MSG$(11) @ BEEP @ WAIT 1 @ GOTO 200
 220 DISP G$; @ INPUT '';F @ INPUT 'z=';I @ GOTO 280
```

```
230 IF FLAG(3) THEN 260
240 DISP 'Vct 1:'&F$; @ INPUT '';C @ IF FLAG(1) AND C<0 THEN DISP MSG$(11) @ WAIT 1 @ GOTO 230
250 DISP 'Vct 1:'&G$; @ INPUT '';F @ INPUT 'Vct 1:z=';I
260 DISP 'Vct 2:'&F$; @ INPUT '';D @ IF FLAG(1) AND D<0 THEN DISP MSG$(11) @ WAIT 1 @ GOTO 260
270 DISP 'Vct 2:'&G$; @ INPUT '';G @ INPUT 'Vct 2:z=';J @ IF FLAG(1) THEN CALL C2R(D,G)
280 IF FLAG(1) THEN CALL C2R(C,F)
290 GOSUB E$
300 DISP @ IF IP((A--1)/3)=1 OR A=9 THEN CFLAG 3 @ GOTO 120
310 DISP 'Resul. Utile ?';
320 E$=KEYWAIT$ @ IF NOT POS(D$,E$) THEN 320 ELSE DISP
330 IF POS('N',E$) THEN CFLAG 3 @ GOTO 120
340 IF NOT POS('Y',E$) THEN 300
350 IF FLAG(2) THEN CALL C2R(B,E)
360 C=B @ F=E @ I=H @ SFLAG 3 @ GOTO 120

!===============================================================
370 'A': B=C+D @ E=F+G @ H=I+J @ GOSUB 'G' @ RETURN

!===============================================================
380 'S': B=C-D @ E=F-G @ H=I-J @ GOSUB 'G' @ RETURN

!===============================================================
390 'D': Q=C*D+F*G+I*J @ DISP 'U.trans(V)=';Q @ GOSUB 'W' @ RETURN

!===============================================================
400 'X': B=F*J-G*I @ E=I*D-J*C @ H=C*G-D*F @ GOSUB 'G' @ RETURN

!===============================================================
410 'M': INPUT 'Scl=';S @ B=C*S @ E=F*S @ H=I*S @ GOSUB 'G' @ RETURN

!===============================================================
420 'I': SFLAG 4 @ GOSUB 'N' @ CFLAG 4 @ P=SQR((D-C)^2+(G-F)^2+(J-I)^2)
430 L=(N*N+O*O-P*P)/(2*N*O) @ K=.00000000002
440 IF FP(FP(ABS(L))+K)<.000000000045 THEN L=IP(L+SGN(L)*K)
450 M=ACOS(L) @ DISP '=';M @ GOSUB 'W' @ RETURN

!===============================================================
460 'N': N=SQR(C*C+F*F+I*I) @ O=SQR(D*D+G*G+J*J)
470 IF NOT FLAG(4) THEN DISP 'Norme(V)=';N @ GOTO 'W'
480 RETURN

!===============================================================
490 'U': SFLAG 4 @ GOSUB 'N' @ CFLAG 4 @ B=C/N @ E=F/N @ H=I/N @ GOSUB 'G' @ RETURN

!===============================================================
500 'W': A$=KEYWAIT$ @ RETURN

!===============================================================
510 'C': B=C @ E=F @ H=I

!===============================================================
520 'G': IF FLAG(2) THEN CALL R2C(B,E)
530 DISP H$;B @ GOSUB 'W' @ RETURN

!===============================================================
540 'Q': RESET @ PUT '#43'

!===============================================================
550 SUB R2C(R,T) @ I=R @ J=T @ R=SQR(I*I+J*J) @ T=ANGLE(I,J)
```

```
560 SUB C2R(I,J) @ R=I @ T=J @ I=R*COS(T) @ J=R*SIN(T)
```

****

"NEW MASTER" pour faire d'un neuf !!!

```
 10 DELAY 0,.3 @ DESTROY ALL @ INTEGER N,E
 20 USER ON @ DEF KEY "#50","" @ DEF KEY "#51",""
 30 INPUT "Nbre de chiffres : ";N @ IF N<=1 OR N>9 THEN 30
 40 A=INT(10^N*RND) @ IF A<10^(N-1) THEN 40
 50 DIM B(2,3*N)
 60 E=E+1 @ IF E=3*N+1 THEN DISP "Une belote?" @ BEEP 400,2 @ WAIT 3 @ GOTO 'FIN'
 70 DISP "Essai n";STR$(E);": "; @ INPUT "";B$ @ IF LEN(B$)>N THEN 70
 80 IF LEN(B$)<2 THEN 180 ELSE B(1,E)=VAL(B$)
 90 IF B(1,E)<10^(N-1) THEN 70
100 A1=A @ B1=B(1,E) @ DISP "Essai n"[(N>5)*7,8];STR$(E);": ";STR$(B(1,E));" : ";
110 FOR I=1 TO N
120 C=FP(A1/10) @ D=FP(B1/10) @ A1=INT(A1/10) @ B1=INT(B1/10)
130 IF D>C THEN B(2,E)=B(2,E)+.1
140 IF D<C THEN B(2,E)=B(2,E)+1
150 NEXT I
160 IF B(2,E)=0 THEN DISP @ BEEP @ DISP "Vous avez trouve en";E;"coups!" @ WAIT 3 @ GOTO 'FIN'
170 DISP STR$(B(2,E)) @ WAIT 2 @ GOTO 60
180 IF E#1 THEN I=E-1 ELSE 70
190 DISP "Essai n"[(N>5)*7,8];STR$(I);":";B(1,I);": ";STR$(B(2,I))
200 IF KEYDOWN('#51') THEN 220
210 IF KEYDOWN('#50') THEN 230 ELSE 190
220 IF I=1 THEN 190 ELSE I=I-1 @ GOTO 190
230 IF I=E-1 THEN 70 ELSE I=I+1 @ GOTO 190
```

==========

```
240 'FIN': DISP "On continue [Y/N]"
250 IF KEYDOWN('#34') THEN DEF KEY "#50" @ DEF KEY "#51" @ END
260 IF KEYDOWN('#6') THEN DISP @ GOTO 10 ELSE 250
```

****

"ECIRTAM" tout simplement renversant

```
 10 DELAY 0 @ OPTION BASE 1 @ DESTROY C,B
 20 INPUT "Dim. de la matrice: ";N @ DIM A(N,N),B(N,N)
 30 FOR I=1 TO N @ FOR J=1 TO N
 40 DISP "A[";STR$(I);",";STR$(J);"]="; @ INPUT "";A(I,J)
 50 NEXT J @ NEXT I @ DISP "   Je calcule ..."
 60 FOR I=1 TO N @ B(I,I)=1 @ NEXT I
 70 FOR K=1 TO N
 80 IF ABS(A(K,K))>.0000000001 THEN 130
 90 FOR J=K+1 TO N @ IF ABS(A(K,J))>.0000000001 THEN 110
100 NEXT J @ BEEP @ DISP "Matrice non inversible" @ END
110 C=A(K,J) @ FOR I=1 TO N @ B(I,K)=(B(I,K)+B(I,J))/C @ IF I>K THEN A(I,K)=(A(I,K)+A(I,J))/C
120 NEXT I @ GOTO 160
130 C=A(K,K) @ FOR I=1 TO N
140 B(I,K)=B(I,K)/C @ IF I>K THEN A(I,K)=A(I,K)/C
```

```
150 NEXT I
160 FOR J=1 TO N @ IF J=K THEN 210
170 C=A(K,J) @ FOR I=1 TO N
180 B(I,J)=B(I,J)-C*B(I,K)
190 IF I>K THEN A(I,J)=A(I,J)-C*A(I,K)
200 NEXT I
210 NEXT J @ NEXT K
220 FOR I=1 TO N @ FOR J=1 TO N
230 DISP "InvA[";STR$(I);",";STR$(J);"]=";STR$(B(I,J)); @ INPUT '';A$
240 NEXT J @ NEXT I
```

****

"PERIFLAG" a plus d'un tour dans sa boucle

```
==================================================================
  10 'PERITAB':
  20 OPTION BASE 0
  30 DIM A(128)
  40 FOR I=0 TO 127
  50 A(I)=DEVADDR("%"&STR$(I))
  60 NEXT I

==================================================================
1000 SUB PERITEST(L)
   - L = Limite inferieure des flags utilises
1020 IF L<0 OR L>52 THEN DISP 'Erreur: 0 <= L <= 52' @ END

==================================================================
1030 DATA HP71,MASSMEM,TAPE,PRINTER,DISPLAY,INTRFCE,GPIO,MODEM,RS232,HPIB,INSTRMT,GRAPHIC

==================================================================
1040 N=11
1050 ON ERROR GOSUB 'ERREUR'
1060 FOR I=L TO L+N
1070 READ P$
1080 IF DEVADDR(P$)#-1 THEN SFLAG I ELSE CFLAG I
1090 NEXT I
1100 OFF ERROR
1110 END

==================================================================
1120 'ERREUR':
   - Cas ou la boucle est interrompue
1140 IF ERRN#255035 THEN GOTO 1180
1150 FOR I=L TO L+N
1160 CFLAG I
1170 NEXT I
1180 RETURN

==================================================================
2000 SUB FX(X)
2010 DESTROY X
2020 FOR I=1 TO 7
2030 IF FLAG(I) THEN X=X+2^I
2040 NEXT I
```

```
2050 DISP X

==============================================================
2060 SUB XF(X)
2070 FOR I=7 TO 0 STEP -1
2080 IF X DIV 2^I=1 THEN SFLAG I @ X=X-2^I ELSE CFLAG I
2090 NEXT I

==============================================================
3000 SUB RFLAG(L)
3010 IF L<-64 OR L>59 THEN DISP 'Erreur: ' @ END
3020 FOR F=0 TO 4
3030 DISP USING '^';FLAG(F,FLAG(F+L))
3040 NEXT F
3050 END SUB
```

****

"SCRABBLE" un oeil qui compte les points

```
100 DESTROY ALL @ LC ON
110 INTEGER D0,D1,I,K,K0,K1,N,T
120 D0=.1 @ D1=.01
140 INPUT "Nb de joueurs ? ","4";N
150 DIM J$(N) @ INTEGER B(N),S(N),M(N,2),P(N+1)

170 FOR I=1 TO N
180 DISP "Nom du joueur";I;
190 INPUT J$(I)
200 IF J$(I)="" THEN 180
210 IF LEN(J$(I))>5 THEN BEEP @ GOTO 180
220 M(I,1)=INF @ M(I,2)=-INF
230 NEXT I

250 DISP "Appuyer sur une touche"
260 A$=KEYWAIT$
270 T=1 @ I=1
280 DISP "Score de ";J$(I);" ";
290 INPUT A$
300 IF A$="" THEN 'FIN'
310 IF A$="+" THEN 'SCORES'
320 IF STR$(VAL(A$))#A$ THEN BEEP @ GOTO 280
330 S(I)=S(I)+VAL(A$)
340 M(I,1)=MIN(VAL(A$),M(I,1))
350 M(I,2)=MAX(VAL(A$),M(I,2))
360 I=I+1 @ IF I>N THEN I=1 @ T=T+1
370 GOTO 280

==============================================================
390 'SCORES':
400 IF T=1 THEN 280
410 DELAY INF,D1
420 DISP T-(I=1);" Tour"; @ IF T-(I=1)>1 THEN DISP "s";
430 DISP
440 FOR K=1 TO N
450 DISP J$(K);" : ";S(K);"->";IP(10*S(K)/(T-(K>=I)))/10
460 NEXT K
```

```
470 DELAY D0,D1
480 A$=KEY$
490 GOTO 280

================================================================
510 'FIN':
520 DISP "Fin de la partie?(o/n)"
530 A$=KEYWAIT$
540 IF UPRC$(A$)#"O" THEN 280
550 T=T-(I=1) @ I=I-1+N*(I=1)
560 FOR K=1 TO N
570 IF K=I THEN 620
580 DISP "Reste de ";J$(K);" ";
590 INPUT B(K) @ B(K)=-ABS(B(K))
600 IF B(K)=0 THEN 580
610 B(I)=B(I)-B(K)
620 NEXT K

640 S(0)=-INF
650 FOR K=1 TO N
660 K0=1
670 IF S(P(K0))+B(P(K0))>=S(K)+B(K) THEN K0=K0+1 @ GOTO 670
680 FOR K1=N TO K0 STEP -1
690 P(K1+1)=P(K1)
700 NEXT K1
710 P(K0)=K
720 NEXT K

740 DISP "Imprimante ? (o/n)"
750 A$=KEYWAIT$
760 IF UPRC$(A$)="O" THEN 'IMP'
770 DELAY INF,D1
780 DISP T;" Tours"
790 FOR K=1 TO N
800 DISP J$(P(K));" :";S(P(K)); @ IF B(P(K))>0 THEN DISP "+"; ELSE DISP "-";
810 DISP ABS(B(P(K)));"->";S(P(K))+B(P(K))
820 NEXT K
830 DELAY D0,D1
840 A$=KEYWAIT$
850 GOTO 770

================================================================
870 'IMP': RESTORE IO
880 PRINT "Date: ";DATE$[7,8];DATE$[3,6];DATE$[1,2]
890 PRINT @ PRINT T;" Tours"
900 FOR K=1 TO N
910 IMAGE 5A," :",4D,X,$3D," = ",4D
920 IMAGE 5"-",3X,2D," < ",2D.D," < ",3D
930 PRINT
940 PRINT USING 910;J$(P(K)),S(P(K)),B(P(K)),S(P(K))+B(P(K))
950 PRINT USING 920;M(P(K),1),IP(10*S(P(K))/(T-(P(K)>I)))/10,M(P(K),2)
960 NEXT K
970 PRINT @ PRINT "----------------------" @ PRINT
980 K=0
990 FOR I=1 TO N
1000 K=K+S(I)
1010 NEXT I
1020 PRINT "Total : ";K @ PRINT @ PRINT @ PRINT @ PRINT @ PRINT
```

"PI" au troisième BEEP vous aurez la précision

```
  - Calcul de pi (formule de Méchain)
    Initialisation des variables
 20 DESTROY ALL @ INPUT 'Nb de décimales ? ';X @ DISP '=.,'; @ X=X+1
 30 OPTION BASE 0 @ INTEGER A(2,X),W(X),U(X),N,S,D,I,J,K,R @ REAL C
  - fnN1(N)= chiffre  des unités de N(premier chiffre en partant de la droite)
    fnN2(N)= chiffre  des dizaines de N(deuxième chiffre en partant de la droite)
    fnC(N,I)= (i-ème chiffre de N en partant de la droite)

============================================================
 40 DEF FNN1(N)=MOD(IP(N),10) @ DEF FNN2(N)=FNN1(N/10) @ DEF FNC(N,I)=FNN1(N/10^(I-1))

============================================================
  - N=1 @ Signe positif @ A(0,)=4/5 @ A(1,)=1/239
 50 N=1 @ S=1 @ A(0,0)=4 @ A(1,0)=1 @ D=5 @ GOSUB 'DIV' @ D=239 @ GOSUB 'DIV' @ GOTO 70
  - Boucle de calcul de U(n)
    division chiffre à chiffre: A(0,)=A(0,)/5^2
    division chiffre à chiffre: A(1,)=A(1,)/239^2

============================================================
 60 'BOUCLE': D=25 @ GOSUB 'DIV' @ D=57121 @ GOSUB 'DIV'
  - soustraction chiffre à chiffre: A(2,)=A(0,)-A(1,)
 70 R=0 @ FOR J=X TO 0 STEP -1
 80 A(2,J)=A(0,J)-A(1,J)-R @ IF A(2,J)<0 THEN A(2,J)=A(2,J)+10 @ R=1 ELSE R=0
 90 NEXT J
  - multiplication chiffre à chiffre: A(2,)=A(2,)*4
100 R=0 @ FOR J=X TO 0 STEP -1
110 K=A(2,J) @ A(2,J)=FNN1(K*4+R) @ R=FNN2(K*4+R)
120 NEXT J
  - division chiffre à chiffre: A(2,)=A(2,)/N
130 D=N @ GOSUB 'DIV'
  - somme chiffre à chiffre: U()=U()+S*A(2,)
140 R=0 @ FOR J=X TO 0 STEP -1
150 U(J)=U(J)+S*(A(2,J)+R) @ IF U(J)<0 OR U(J)>9 THEN U(J)=MOD(U(J),10) @ R=1 ELSE R=0
160 NEXT J
170 S=-S @ N=N+2 @ I=0
  - affichage d'un point indiquant l'évolution du programme
180 DISP '.';
190 IF (N-1)/2<X THEN 'BOUCLE'
  - fin: affichage de pi
200 DISP @ DISP '=3,'; @ FOR N=1 TO X-1 @ DISP STR$(U(N)); @ NEXT N @ DISP @ END
  - division chiffre à chiffre: A(I,)=A(I,)/D
    sortie du sous-programme en incrémentant I(pour préparer la prochaine division)

============================================================
210 'DIV': C=0 @ FOR J=0 TO X
220 C=C*10+A(I,J) @ A(I,J)=IP(C/D) @ C=MOD(C,D)
230 NEXT J
240 I=I+1 @ RETURN
```

****

"PI2" toujours pire

```
10 INPUT 'Nb de décimales ? ';X
15 IF X>200 THEN DISP 'Cf: palais de la découverte.' @ END ELSE DISP 'pi=3,'
20 DIM A$[50] @ FOR I=0 TO IP((X-1)/50) @ READ A$ @ DISP A$[0,X-50*I] @ NEXT I

============================================================
30 DATA '14159265358979323846264338327950288419716939937510'
40 DATA '58209749445923078164062862089986280348253421170679'
50 DATA '82148086513282306647093844609550582231725359408128'
60 DATA '48111745028410270193852110555964462294895493038196'



************************************************************************************************************

"GRAPH" voyez la consommation


  - PRESENTATION DE VALEURS CHIFFREES SOUS FORME GRAPHIQUE
    PGM=GRAPH1  /  HP71B et ThinkJet
    JCF Dec 1985
10 DESTROY ALL
20 FIX 2
30 DIM M$(12)[7],R(12),A$[50]
40 FOR I=1 TO 12
50 READ M$(I)
60 NEXT I
70 FOR I=1 TO 12
80 READ R(I)
90 NEXT I

============================================================
100 DATA JANV +,FEV  +,MARS +,AVR  +,MAI  +,JUIN +
110 DATA JUIL +,AOUT +,SEPT +,OCT  +,NOV  +,DEC  +
120 DATA 4.19,6.09,4.021,2.128,3.51,4.994
130 DATA 3.653,5.36,2.91,3.78,6.011,3.9999

============================================================
140 PRINT "MOIS          NIVEAU DE LA CONSOMMATION MENSUELLE EN KF "
150 PRINT
160 L=0 @ S=0
170 FOR I=1 TO 12 @ IF L<=R(I) THEN L=R(I)
180 NEXT I
190 K=40/L
200 FOR I=1 TO 12
210 N=INT(R(I)*K)
220 S=S+R(I)
230 GOSUB 'FORMAT'
240 PRINT M$(I);TAB(8);A$,R(I)
250 NEXT I
260 PRINT USING "5' ',55'+',4' ',4'='"
270 A=S/12
280 N=A*K-2 @ GOSUB 'FORMAT'
290 PRINT
300 PRINT "MOY. + ";TAB(8);A$,A
310 PRINT "TOTAL POUR L'ANNEE ";TAB(64);S
320 END

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

```
330 'FORMAT':
340 FOR J=1 TO N
350 A$[J]='*'
360 NEXT J
370 FOR J=N+1 TO 40
380 A$[J]=' '
390 NEXT J
400 RETURN
```

```
MOIS            NIVEAU DE LA CONSOMMATION MENSUELLE EN KF

JANV + ****************************                    4.19
FEV  + ****************************************        6.09
MARS + ***************************                     4.02
AVR  + ************                                    2.13
MAI  + ***********************                         3.51
JUIN + *********************************                4.99
JUIL + ***********************                         3.65
AOUT + ************************************            5.36
SEPT + ******************                              2.91
OCT  + ************************                        3.78
NOV  + ****************************************        6.01
DEC  + ***************************                     4.00
     +++++++++++++++++++++++++++++++++++++++++++++++++  ====

MOY. + ************************                        4.22
TOTAL POUR L'ANNEE                                     50.65
```

****

"ASSRED" formateur de source assembleur

```
 10 DIM A$[256],B$,F$,I,J,L,P
 20 L=50
 30 INPUT "Fichier: ";F$ @ P=NOT POS(F$,":")
 40 ASSIGN #1 TO F$
 50 SFLAG -1 @ PURGE RT @ CFLAG -1 @ CREATE TEXT RT @ ASSIGN #2 TO RT
 60 FOR I=1 TO FILESZR(F$)
 70 READ #1;A$ @ IF P THEN DELETE #1,0
 80 IF LEN(A$)<=L THEN PRINT #2;A$ @ GOTO 'SUITE'
 90 B$=A$[1,23] @ A$=A$[24]

=============================================================
100 'TQ': J=L-22
110 IF NUM(A$[J])#32 THEN J=J-1 @ GOTO 110
120 PRINT #2;B$&A$[1,J-1]
130 A$=A$[J+1] @ B$="        *             "
140 IF LEN(A$)>L-22 THEN 'TQ'
150 PRINT #2;B$&A$

=============================================================
160 'SUITE': NEXT I @ ASSIGN #1 TO * @ ASSIGN #2 TO *
170 IF P THEN PURGE F$ ELSE F$=F$[1,POS(F$,":")-1]
180 RENAME RT TO F$
```

LE COIN DES LHEX

En plus des LEX de la rubrique assembleur vous trouverez dans le coin CHARLEX et FORMALEX qui vous sont nécessaires pour "AH ! VOUS ECRIVEZ".

D'autre part, vous trouverez ici la version authentique, la seule, l'unique de STRUC1.

J.J. DHENIN

```
CHARLEX     ID 225

FORMALEX    CENTER$     225034
            CESURE      225035
            FORMAT$     225036
            REDUCE$     225037
            SPACE$      225038

STRUC1      END         225066
            LEAVE       225070
            REPEAT      225068
            UNTIL       225069
            WHILE       225067

LINKLEX     LABEL$      225076
            MNEMO       225077

MW          MOLEWT       80001
```

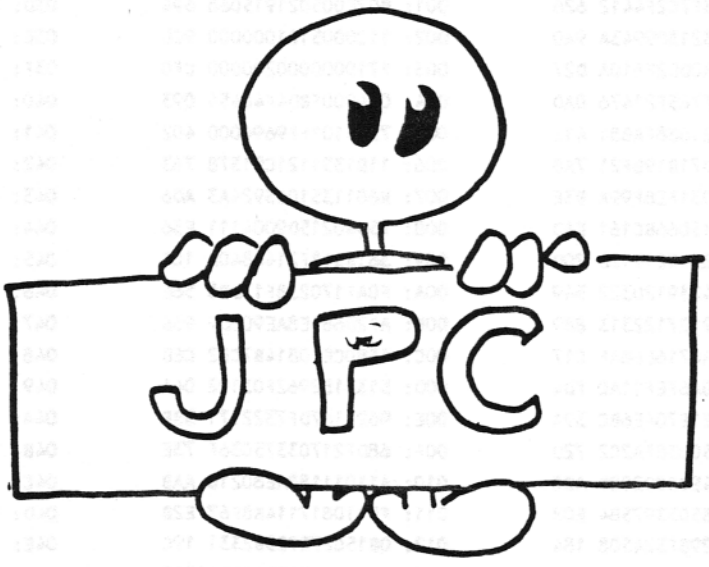
PROGRAMME MAKELEX

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

```
CHARLEX   ID#E1  624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E001021915068 6AE
002: 5E4001E000000000 9F2
003: FE00000000800001F D4C
004: F318F9614000328F 00F
005: 38F14A11DB10AD23 479
006: 07D532BFB8FD7911 82C
007: 11AD754D7A101743 BAF
008: 11014D1CB1500000 F1A
009: 71450375FF864834 297
00A: 5655581008355654 5EE
00B: 5810002455565870 938
00C: 0026555658700836 C8D
00D: 555658100836454S FE3
00E: 4A30000A49724000 336
00F: 0808094A2C180814 69F
010: A464242008355455 9F9
011: 581000054C714000 D3F
012: 0C3142404C700832 098
013: 41414A70002078A0 3F3
014: 2F30000000000000 71E
015: 0000000000000000 A2E
016: 0000000000000000 D3E
017: 0000000000000000 04E
018: 0000000000000000 35E
019: 0000000000000000 66E
01A: 0000000000000000 97E
01B: 0000000000000000 C8E
01C: 0000000000000000 F9E
01D: 0000000000000000 2AE
01E: 0000000000000000 5BE
01F: 0000000000000000 8CE
020: 0000000000000000 BDE
021: 000000000000080C F09
022: 1A28080008080A2C 273
023: 180008040E340800 58C
024: 08001E3018000000 8F6
025: 0000000000000000 C06
026: 0000000000000000 F16
027: 0000000000000000 226
028: 0201000000010200 53C
029: 0000000201020000 851
02A: 0001000100000002 865
02B: 0102010000000000 E79
02C: 0000000000000000 189
02D: 045E755142400101 4D5
02E: 0101010000000000 7E8
02F: 0000000000000000 AF8
030: 0000070507000000 E1B
031: 000000000834444C4 159
032: 4440007901112D70 489
033: 0500750509700000 803
034: 0070000000384540 B46
035: 4020014E322E3140 E9A
036: 084E794142400000 1EA
037: 000000000002E4559 528
038: 3200000000000000 83D
039: 0000000000000026 B55
03A: 5556587008365556 EB4
03B: 5810083645464830 205
03C: 0832414248700024 546
03D: 5655587008345655 8A3
03E: 5810083446454830 BF2
03F: 0C3042414C700024 F47
040: 5556587008355654 2A4
041: 5810083546444830 5F3
042: 0C3142404C700025 949
043: 5455587008355455 CA3
044: 5810083544454830 FF1
045: 0C3140414C700875 353
046: 14141870000A4972 6A4
047: 40000E3159454E30 A04
048: 0C7A0F7949400024 D7C
049: 5554587000084A71 008
04A: 40000C523A262D10 439
04B: 0424587458400875 790
04C: 1415187000094A70 AE0
04D: 4000083544454830 E24
04E: 0C3140414C300C74 18C
04F: 5655545000054C71 4E3
050: 40000            5DC

FORMALEX  ID#E1  454 octets

     0123456789ABCDEF sm

000: 64F425D414C45485 385
001: 802E002021915068 6D6
002: 093001E226200000 A14
003: FB30000000000000 D4F
004: 066100F110EA200F 0C2
005: 020EC100F130E110 42E
006: 0F2407D000FD3454 7AF
007: E445542542228345 B19
008: 435552554320647 E8C
009: 25D414454242D255 1F5
00A: 4445534544252B35 552
00B: 0514345442621FF0 887
00C: 48F83D801368F534 C4E
00D: B1137C2109134135 FAC
00E: D38508418423102A 312
00F: E5CC4E2851CC1811 68C
010: 4E965F08708EE785 A64
011: 05808408521C114D DCA
012: 51D86011861C0862 136
013: 70171CF018118FBC 4CD
014: 6314400017F13713 830
015: 51098F064A173004 B97
016: B231025908F40581 EFF
017: CC56F01411714F11 281
018: 98FB14811368408F 616
019: 064A18D7B1818D91 9A7
01A: FB0842274508F13D D35
01B: B0137135C21098F0 0A8
01C: 64A1118C642D8A88 43A
01D: 188E31E2AD00A81C 7FB
01E: 81C7B7F8508F7B18 8AA
01F: 18FE83B18DC32F08 F5E
020: FBC6315598A80917 2F8
021: F100018D04490842 66F
022: 27ADF769E119110E A10
023: 244EE24FD10ADA13 DBF
024: 416F41D13618495F 14E
025: 213288EFB119AF0D 503
026: A137135EA81C1188 88C
027: B6348A2A3E48AB33 C34
028: EAF8D810B8FE6CE0 022
029: 20108D8E57320110 381
02A: 118E2CD751011A13 707
02B: 56CCE31528D39390 A9C
02C: D731026620148171 DFB
02D: 1481619661FD4580 171
02E: 14C161CC57FCF8AF 541
02F: AD018422730FD88F 8E1
030: 83D8006AD0DA81C1 CA3
031: 37135C2C2D788497 039
032: E9E91353F02F312E 3DA
033: 2C2A383920A14896 76C
034: 261850CD4801715C AE6
035: E1104241108A0012 E49
036: 71C11488F68F6962 1F9
037: 710052F4818709CC 597
038: C10051C1188A160E 913
039: 5E5D4CCD81351CF8 CE1
03A: FB13B1AF680832F0 09C
03B: F                0E3

STRUC1    ID#E1  656 octets

     0123456789ABCDEF sm

000: 3545255534130202 341
001: 802E003021915068 693
002: 425001E246400000 9D4
003: FB30087000000000 D1E
004: 06D000D2309F100D 095
005: 6101A300D520C030 3F2
006: 00900FA300D554E4 784
007: 44249C4541465546 AEA
008: 4825540554144544 E44
009: 955E44594C454975 1CB
00A: 8494C454341FF306 551
00B: 001304758494C454 8AE
00C: C81408E6F6022554 C2F
00D: 05541445C3150ECE FB4
00E: 4C454146554CF160 331
00F: B8E6F60254E44402 6CA
010: 758494C454CFFD04 A76
011: 008E30013618698F DF0
012: 21441367D124E02F 164
013: 3092094371203130 4A3
014: 22311E208DA93908 81D
015: 7D1220321088FAB8 BA4
016: 11D813788700C988 F48
017: B9CD81B198F21441 2F3
018: 367181431186982F2 667
```

```
019: 1461366E4013610A 9C7
01A: 1B198F21422F3097 D46
01B: E8111A13686F927A 0D8
01C: 2042213610A8F95E 44F
01D: F011A1348FB7EF01 7F3
01E: 121308D7E47084A8 B72
01F: 6F0086D0087A001F EF7
020: 0B7F22030214B0E0 268
021: 6A0C1FD55F214713 602
022: 517E143F434412E0 979
023: 8A2400203403008F CD2
024: 2007801136798058 02D
025: 213610A1B198F214 39D
026: 22F30977D011A136 712
027: 8D84A80D2E610A86 AB4
028: D011F765F2147661 E33
029: 020321088FAB8111 19F
02A: 37C2D72031FE8F99 54A
02B: A804A03160658E16 8C1
02C: 1331E34231521916 C13
02D: E011AE610A6D1022 F97
02E: 312423916F011ACE 317
02F: 8AAD010A713069AF 688
030: 16365BE8F681F08F A65
031: 19DB0042E5A091CC E06
032: 011091C400302183 14A
033: D014AD8132C01300 4BD
034: 18D31F808DE3F801 85F
035: 361B198F214401FD BE9
036: 1003D1001361B698 F46
037: F21447FCF4E02F30 2F1
038: A20943A0314063BD 66C
039: 87D1220321088FAB 9F3
03A: 811D8137887DDC98 DA1
03B: BB6DDB1B198F2144 15B
03C: 1641461367A3F560 4C7
03D: 641E1B198F214210 83C
03E: 22F30A68CD701009 BC0
03F: 01001322F30A714F F25
040: 816136696E7E0009 29D
041: E000773F7C2F4412 626
042: F30A94321309943A 9A0
043: 031506A0DD2E610A D27
044: 86D011F765F21476 0AD
045: 61020321088FAB81 41E
046: 1137C2D71B198F21 7A8
047: 461362031FE8F99A B3E
048: 804A03150668C161 EAD
049: 331E242315219120 200
04A: 3223154239120322 549
04B: 3144239127122313 889
04C: 423912A0716E68AF C17
04D: 11AE610A6FEF11AC FD4
04E: E10A8AE1E704E6BC 39A
04F: C8D30350038FA2C2 72D
050: 0FE1E34E0000208D AB8
051: 5303018503397584 E08
052: 94C454298F324508 184
053: FCE2508DE6A208D3 542
054: 9450F 65F

LINKLEX  ID#E1  208 octets

0123456789ABCDEF sm

000: C494E4B4C4548502 392
001: 802E004021915068 6E5
002: 5A1001EC4D400000 A51
003: F020000000000000 D79
004: 09E000FF0082000F 0F3
005: BC4142454C442C49 47D
006: D4E454D4F4D41FF4 842
007: 118F13DB0137135C BC3
008: 20681C1018507270 F16
009: 444AE531A2961831 290
00A: 19E2D5D231818816 61E
00B: 231308B522071351 968
00C: CFD48FB13B1AF68D D4C
00D: 832F0D142EB40712 0C5
00E: 05C0111E4D85EC85 45F
00F: 07D004EE119E2D55 7F8
010: AB3102AE5CC40014 B88
011: F17187090965DE01 F10
012: 9616E01411BF13DB 29A
013: 0AF1D881DD2E688D 66A
014: 0514331D3966A0CD 9EF
015: 17114331208B5333 D3E
016: 06881B2313296222 0A1
017: 8F670B1581312096 412
018: 5B133853523916E0 77F
019: 137C9C9137D120A6 B0B
01A: 58F064A1C9109850 E8D
01B: 8F7B1818FE83B18D 24B
01C: C32F0 36E

MW  ID#50  1542 octets

0123456789ABCDEF sm

000: D475020202020202 340
001: 802E005021915068 694
002: 11C0005101000000 9C0
003: F710000000200000 CF0
004: 0E4000FBD4F4C454 093
005: 7545101FF9694000 402
006: 11B1351121C91378 763
007: B68113510B3924A3 AD6
008: 75D40215D9004111 E36
009: 361BBB8F2144840A 1CC
00A: F0AF170238F13DB0 56E
00B: AF2D681E8AE908D9 93B
00C: 1FB0CE10814B7CE2 CED
00D: 5133182962F03192 04A
00E: 9621167DF7322171 3BB
00F: 68DF21703375036F 73E
010: A11011188AE80216 AAB
011: F51108171148B8F67 E2B
012: 0B15C272925B2331 19C
013: 6A77B82503318296 512
014: 2513192962C3607F 883
015: 216AA0217EC273A2 C0F
016: 7B72646F119BF2BF FC4
017: 2AE610974628AEA1 35F
018: 236CE0217E927372 6DF
019: 7B426911171143B73 A49
01A: 1258573125F33182 DA7
01B: 962F03192962B16E 12C
01C: FE23726277327041 49C
01D: 17165FE237D42722 81C
01E: 27AF168C0237B327 BA8
01F: 01278E161DE23792 F27
020: 210A7C718AA857BA 2CE
021: 15543182962D1319 62E
022: 296272669ED230C8 9B8
023: BF00688E7EEF79A1 D84
024: 7CC017161BE7BDF7 138
025: 6916A507FCF7A816 4E1
026: 96E73CF7E716B007 88B
027: 7B17C8186060604E C0B
028: 70511BBB8F214213 F8A
029: 0171AFDAFFAFDAFE 392
02A: 8F499C08D612F087 733
02B: 060690E8401188AE AB1
02C: A0799060BF17114B E39
02D: 71E052271E05F031 1AB
02E: 829626065DD70707 524
02F: CE065DD79808AAA0 8E2
030: 7F2064CD77206A6F C85
031: 8606067AD850AF0A 020
032: F173A075B08AE006 3B4
033: D8D73FE05AF48F22 77B
034: 3B18FFB6C076808F B33
035: A34C070707200017 E8D
036: 170AFDAFFAFDAFE7 29B
037: 160058F363C00477 606
038: 4001AF1D3725014B 97F
039: 20BF1A88E71188AA D2B
03A: 007930171147B7500 07A
03B: 5FD018D670B13314 401
03C: A58DC70B18FC29E0 7C1
03D: 20018F459E020011 B1D
03E: 18CE1080105AF48F EB6
03F: 223B111A8F234C06 233
040: F0005AFA8FFB6C08 5FD
041: F459E08F363C08FC 9B8
042: 29E0200401111891 D04
043: 6067117FD0241000 061
044: 0000000180134100 383
045: 0000000110216410 6A3
046: 0000030489981840 9E9
047: 0000000097001B4 D20
048: 1000000003893E 066
049: 4100000000760041 38D
04A: F410000000049995 6DC
04B: 1051000000067379 A13
04C: 0335100000000006 D35
04D: 0235520000000092 061
04E: 0832651000000004 38E
04F: 1490575200000000 68F
050: 0583819510000000 9F7
```

```
051: 0950988000713415 D43
052: E19125191AC01611 0B2
053: 6E68EF668820161A 461
054: F215EE2001728636 7E6
055: 1420000000000072 B06
056: 2761420000000086 E3A
057: 8701C61410000000 179
058: 4518962D61420000 4CD
059: 0000000342271410 7F5
05A: 0000000084993371 B31
05B: 4100000000612947 E63
05C: 4714200000000000 185
05D: 0125714200000005 4B0
05E: 6696911624200000 7F4
05F: 0000337315624000 B26
060: 0000008121099624 E60
061: 2000000040898028 1A3
062: 6242000000000007 4C8
063: 4227241000000000 7EE
064: 4099716341000000 B2A
065: 0000800446342000 E59
066: 0000001421156342 186
067: 0000000002104166 4AA
068: 3420000000000015 7C9
069: 2C63410000000003 AFF
06A: 5453063420000000 E43
06B: 0000742F63410000 184
06C: 0000233985273410 4C3
06D: 0000000069915373 7FE
06E: 4200000004509231 B2C
06F: 5734100000000064 E5A
070: 5369744200000000 192
071: 0052612754200000 4C4
072: 0000627613754200 7FF
073: 0000000004525754 B2F
074: 2000000000691515 E5C
075: 6641000000000748 190
076: 55D6642000000000 4D0
077: 0075227642000000 803
078: 0000032216741000 B2D
079: 0000000279646742 E6C
07A: 0000000005275156 198
07B: 7410000000000952 4C7
07C: 7568400000000006 7FB
07D: 2004668420000000 B2B
07E: 0094871768420000 E73
07F: 0000095002F68420 1BD
080: 0000004039461029 4F3
081: 4200000005409621 824
082: E694200000000028 B68
083: 4112794200000000 E96
084: 0222912784100000 1D6
085: 00000083816C4200 519
086: 0000055098319 6C4 86E
087: 0000000000014962 B94
088: 7C42000000000000 EC4
089: 6257C42000000000 201
08A: 7947146042000000 551
08B: 0000085276041000 896
08C: 00000050342E6041 BE8
08D: 000000000 83945F6 F31
08E: D410000000000495 26C
08F: 916E410000000779 58D
090: 892226E410000000 904
091: 046092946E420000 C57
092: 000004244156E410 F9B
093: 0000000097102 96E 2E2
094: 4100000000000785 608
095: F6E4200000000000 952
096: 55207E4200000002 C92
097: 84073237F4200000 FE0
098: 0000020911605200 30A
099: 0000095301322605 63E
09A: 2000000000027024 95F
09B: 6052000000000046 C86
09C: 01D6052000000000 FB8
09D: 00541F6052000000 2F5
09E: 0000090227052000 620
09F: 0000770904147052 95E
0A0: 0000000009059157 C92
0A1: 0520000000000044 FB1
0A2: 2162520000000452 2DE
0A3: 0622262510000000 608
0A4: 0876458562520000 952
0A5: 0000702681862520 C91
0A6: 0000005509201E62 FD4
0A7: 5200000000000222 2F1
0A8: 5725200000000070 61D
0A9: 1012635200000000 941
0AA: 0571213635100000 C73
0AB: 0009559445635100 FBB
0AC: 0000000069879635 300
0AD: 1000000005580820 641
0AE: 6352000000000040 965
0AF: 51E6352000000000 CA0
0B0: 9681127351000000 FDB
0B1: 0000267816452000 314
0B2: 0000974908126452 65D
0B3: 0000000452985136 998
0B4: 4510000000000007 CB9
0B5: 9564520000000000 FE8
0B6: 6721864520000000 321
0B7: 1830232964510000 65D
0B8: 000000097 4C64520 9A5
0B9: 0000000073402D64 CE3
0BA: 5200000002439861 01B
0BB: 5685200000000003 348
0BC: 1312695200000000 675
0BD: 040371E6A5100000 9C6
0BE: 00000835627A5100 D0C
0BF: 0000000022190000 02A
0C0: 0713415E39125191 383
0C1: AC016316E68EF681 72F
0C2: 920163AF215EE200 AAF
0C3: 1                AE1
```



PPC PARIS CHAPTER

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

BULLETIN D'ADHESION

NOM |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

PRENOM |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| DATE DE NAISSANCE |_|_|/|_|_|/|_|_|

ADRESSE |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

COMMUNE |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

CODE POSTAL |_|_|_|_|_| PAYS |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

TELEPHONE DOMICILE |_|_|/|_|_|_|_|_|_|_|_|_|_| BUREAU |_|_|/|_|_|_|_|_|_|_|_|_|_|

PROFESSION ______________________________

INTERETS ______________________________

____

MATERIEL HP EN VOTRE POSSESSION ______________________________

____

____

AUTRE MATERIEL MICRO-INFORMATIQUE ______________________________

____

____

COMMENT AVEZ-VOUS CONNU PPC PARIS CHAPTER ?

PUBLICITE ______________ MAGAZINE ______________

AUTRE CLUB ______________ HP ______________

RELATIONS, MEMBRES DU CLUB, AUTRES ______________________________

QUE RECHERCHEZ-VOUS AU SEIN DU PPC PARIS CHAPTER ? ______________________________

____

____

____

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

DATE |_|_|/|_|_|/19|_|_|

SIGNATURE, PRECEDEE DE LA MENTION "LU ET APPROUVE" ______________

LE MONTANT DE LA COTISATION AU PPC PARIS CHAPTER S'ELEVE A 300.00 FF.

ETUDIANTS: 250.00 FF. (JUSTIFICATIF INDISPENSABLE)

PAIEMENT EFFECTUE LE |_|_|/|_|_|/19|_|_| A L'ORDRE DE PPC PARIS CHAPTER.

PAR [ ] CHEQUE BANCAIRE N° ______________ BANQUE ______________

[ ] CHEQUE POSTAL 3 VOLETS N° ______________________________

[ ] MANDAT LETTRE

EVENTUELLEMENT: JE M'ABONNE A COMPTER DU |_|_|/|_|_|/19|_|_|

JOINDRE A VOTRE INSCRIPTION UNE PHOTO D'IDENTITE ET UNE ENVELOPPE TIMBREE A VOTRE ADRESSE.

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

©1986

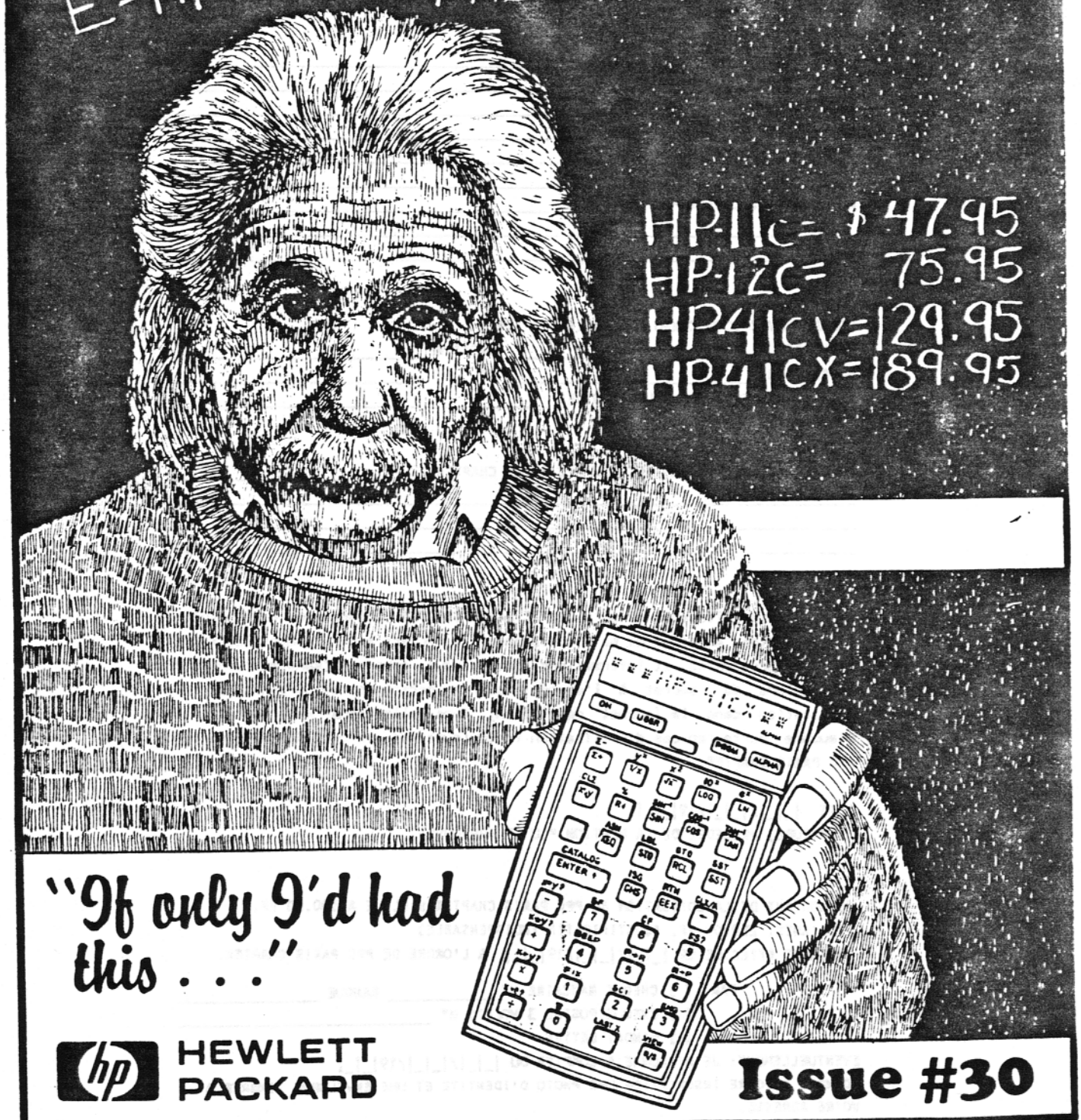
$1.00

EduCALC

CALCULATORS•COMPUTERS•BOOKS

$E=hp^2$

FREE HP-12C SOFTWARE

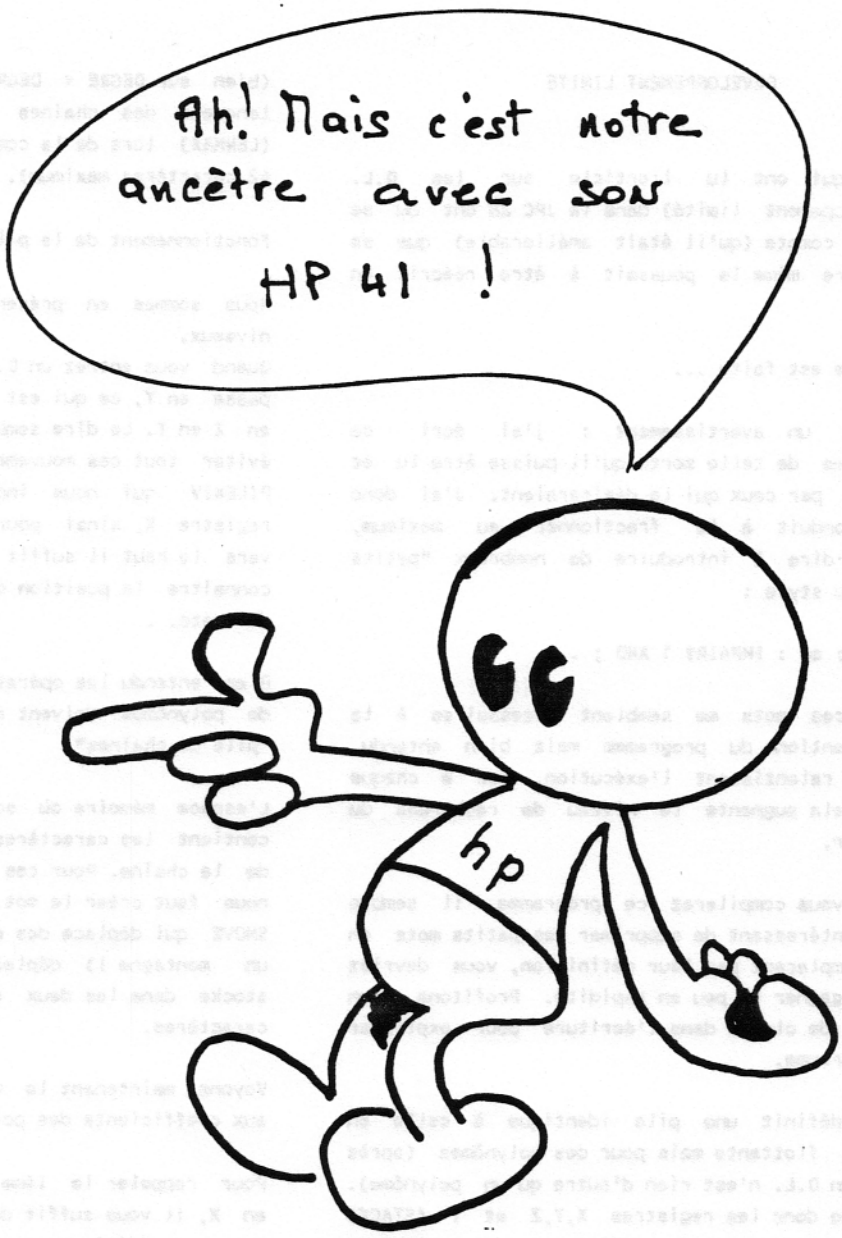


"If only I'd had this . . ."

HEWLETT PACKARD

Issue #30

27953 Cabot Road, Laguna Niguel, CA (714) 582-2637



FORTH

DEVELOPPEMENT LIMITE

Ceux qui ont lu l'article sur les D.L. (Développement limité) dans le JPC 28 ont du se rendre compte (qu'il était améliorable) que sa structure même le poussait à être réécrit en FORTH.

La chose est faite ...

D'abord un avertissement : j'ai écri ce programme de telle sorte qu'il puisse être lu et compris par ceux qui le désireraient. J'ai donc été conduit à le fractionner au maximum, c'est-à-dire à introduire de nombreux "petits mots" du style :

COEFF@ ; ou : IMPAIR? 1 AND ; ...

Tous ces mots me semblent nécessaires à la compréhention du programme mais bien entendu, ils en ralentissent l'exécution, car à chaque fois cela augmente le niveau de recherche du pointeur.

Quand vous compilerez ce programme, il semble donc intéressant de supprimer ces petits mots en les remplaçant par leur définition, vous devriez ainsi gagner un peu en rapidité. Profitons d'un effort de clarté dans l'écriture pour expliquer ce programme.

On redéfinit une pile identique à celle en virgule flottante mais pour des polynômes (après tout un D.L. n'est rien d'autre qu'un polynôme). On crée donc les registres X,Y,Z et T (STACK) qui contiendront les coéfficients des polynômes (les coéfficients étant en virgule flottante, il faut réserver une place de 8 octets par coéfficient). Il nous faut aussi créer une pile qui contiendra les chaines de caractères associées aux polynômes, elles seront stockées dans STACK$. Tout ces espaces mémoire doivent être alloués lors de la compilation, il nous faut donc définir un degré maximum (constante DEGREMAX), les D.L. ayant un ordre au plus égal à ce degré maximum, mais rien ne nous empêche en changeant la valeur de la variable DEGRE de réduire le nombre de coéfficients des polynômes (bien sur DEGRE < DEGREMAX). De même pour la longueur des chaines on limite leur longueur (LENMAX) lors de la compilation (dans le listing 42 caractères maximum).

Fonctionnement de la pile :

Nous sommes en présence d'une pile à quatre niveaux.
Quand vous entrez un D.L. en X ce qui est X passe en Y, ce qui est en Y en Z et ce qui est en Z en T. Le dire semble déjà long, alors pour éviter tout ces mouvements on crée une variable PILENIV qui nous indique où se trouve le registre X, ainsi pour faire tourner la pile vers le haut il suffit de décrémenter PILENIV, connaitre la position de X c'est connaitre celle de Y etc. .

Bien entendu les opérations faites sur la "pile de polynômes" doivent aussi êtres faites sur la "pile de chaines".

L'espace mémoire où sont ces chaines (STACK$) contient les caractères précédés de la longueur de la chaine. Pour ces transports de chaines il nous faut créer le mot SMOVELEN, qui en plus de SMOVE qui déplace des chaines (ne pas en faire un montagne !) déplace leur longueur et les stocke dans les deux octets qui précédent les caractères.

Voyons maintenant la manière dont on a accès aux coefficients des polynômes.

Pour rappeler le Iéme coefficient du polynôme en X, il vous suffit de faire X I COEFF@, et votre coefficient se trouvera en X de la pile en virgule flottante.

De même pour stocker un nombre qui se trouve dans le registre X de la pile en virgule flottante en Y (par exemple) de notre pile de polynôme il suffit de faire Y I COEFF!. Je pense qu'avec ce qui précède la définition de ces mots ne pose pas de problème.

Il devient très facile maintenant de créer les mots de manipulation de pile :

ENTER, RUP, RDN, CLX, CLST, X<>Y

(respectivement duplique le polynôme de X en Y, fait tourner la pile vers le haut, idem vers le bas, "clear" le registre X, "clear" la "stack", échange X avec Y).

Le mot REDIM nous permet de redimensionner les polynômes tant que cela est possible (vous ne pouvez pas dépasser DEGREMAX).

Il s'agit maintenant de rentrer tout les D.L. classiques (EXP, SIN, LOG(1+X), SH, COS, CH, ATAN ATH, ASIN, ASH).

Quand on rentre un D.L. connu on se rend bien compte que toute une série d'opérations se répétera (manipulation de chaines...), pour cette raison on crée le mot ?DEF ; ce qui diffère entre le D.L. de COS ou EXP (par exemple) c'est la définition des coéfficient et le nom de la fonction (!) (resp. COS, EXP).

Le mot COS déposera sa chaîne et le CFA de DEFCOS qui contient la définition des coefficients en fonction de la puissance de x. ? EF n'aura qu'à exécuter la définition de la fonction dont il connait le CFA et à traiter le nom du D.L. dont il a l'expression. Voilà, je pense que cette structure évite au maximum les répétitions inutiles.

Le mot POLY (ne risque pas d'être un gros mot !) permet de rentrer un polynôme quelconque avec le nom que vous désirez. Si par exemple vous voulez entrer le polynôme: 3.2+x, tapez POLY puis à la question "nom du poly." tapez 3,2+x, pour a(0) entrez 3.2, pour a(1) entrez 1 (inutile de taper 1. (1 avec un point)), tout les autres coéfficients étant nuls, pour a(2) tapez simplement END-LINE.
Pour les opérations sur les polynômes (+, *, -, +, comp) je n'ai fait que traduire en FORTH ce que j'avais écrit en BASIC dans le JPC 28, je n'en dirai donc pas plus.

Pour changer le signe du polynôme en X (CHS) une solution aurait été de faire une boucle avec :

X I COEFF@ CHS X I COEFF!

(où CHS cette fois change le signe du nombre qui est en X de la pile en virgule flottante). Mais bien plus rapide est de se ramener à la structure d'un nombre en virgule flottante en mémoire (cf. manuel FORTH/Assembleur). Pour changer le signe il suffit de changer le 9 (resp. 0) en 0 (resp. 9) de l'octet qui code le signe, TOGGLE fera cela trés bien.

Enfin pour mettre sous forme de fraction un coéfficient, on a au moins deux possibilités.

La première est d'utiliser comme dans la version BASIC (ne pas lire l'aversion BASIC!) le programme de Pierre David, on se servirait donc de BASICX.

Cette solution cette fois me semble peu élégante car elle mélange deux langages, ce qui ne sera pas possible sur toutes les machines.

La deuxième possibilité est de se servir du programme de mise en fraction en FORTH de Alain Herreman qui doit se trouver dans ce numéro même (l'auteur ayant bien voulu me le communiquer en avant première, je l'en remercie !). Ainsi pour avoir la fraction du 3ème coéfficient du polynôme qui se trouve en X, tapez 3 FRC .

Venons en pour finir à l'exemple nécessaire : si vous voulez le DL de TAN tapez, en étant dans le voc. VOC-DL, SIN COS / puis tapez . (dot) vous aurez les coéfficients en appuyant sur ENDLINE, si vous voulez la fraction du 4ème coefficient entrez:

3 FRC c'est tout .

Alain HERREMAN

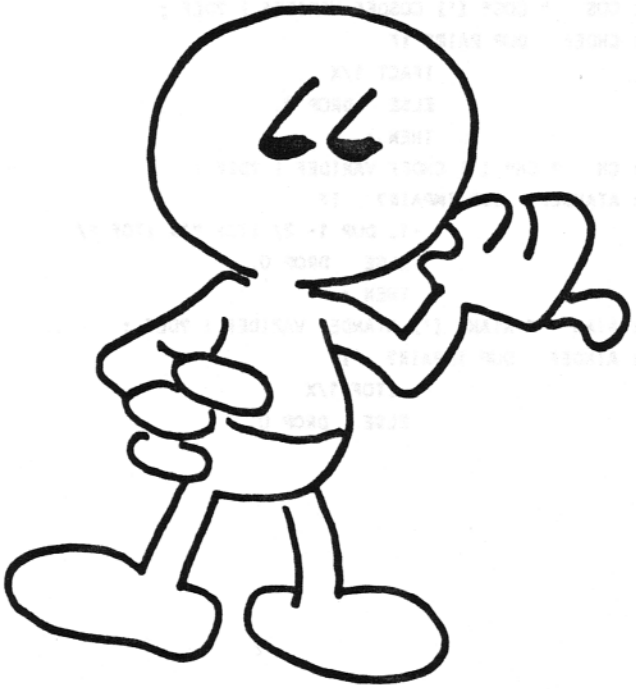
```
( opérations sur la pile en virgule flottante )
( seront notées (n1 -F- n2), n1 en X devient n2 )
DECIMAL
VOCABULARY VOC-DL   VOC-DL DEFINITIONS
   ( On crée un nouveau vocabulaire VOC-DL )
VARIABLE PILENIV   0 PILENIV ! ( position du )
   ( registre X de la pile de poly. dans STACK )
VARIABLE DEGRE   5 DEGRE !  ( DEGRE contient )
   ( l'ordre auquel doivent être fait les )
```

```
   ( calculs )
10 CONSTANT DEGREMAX ( degré maximal des poly. )
CREATE STACK   DEGREMAX 1+ 64 * NALLOT ( espace )
     ( mémoire réservée à la pile des poly. )
42 CONSTANT LENMAX ( longueur maximale des )
    ( chaines caractérisant les poly. )
LENMAX 4 STRING-ARRAY STACK$ ( espace mémoire )
     ( contenant les chaines des poly. )
: BOUCLE   ( --- n1 ) ( valeur revenant souvent )
                      ( pour les boucles )
       DEGRE @ 1+ ;
: #NIVEAU   ( n1 --- n2 ) ( retourne l'adresse )
 ( du registre caractérisé par n1 ; 0 pour X etc )
       PILENIV @ + 4 /MOD DROP BOUCLE 16 * *
       STACK + ;
: X   ( --- n1 ) ( renvoit l'adresse du reg. X )
       0 #NIVEAU ;
: Y   ( --- n1 ) ( idem pour Y )
       1 #NIVEAU ;
: Z   ( --- n1 ) ( idem pour Z )
       2 #NIVEAU ;
: T   ( --- n1 ) ( idem pour T )
       3 #NIVEAU ;
: COEFF@   ( add n1 --- ) ( -F- n2 ) ( add est )
     ( l'adresse de X-Y-Z ou T, n1 est le )
 (numéro du coeff., n2 est le coeff. désiré )
: COEFF!   ( add n1 --- ) ( n2 -F- )
       ( stocke n2 dans le registre d'adresse )
       ( add au niéme emplaçement )
       16 * + STO ;
: SMOVELEN   ( str add --- ) ( voir article )
       OVER OVER 2- I SMOVE ;
: .DL ( affiche la chaine du poly. qui est en X )
       CR PILENIV @ 1+ STACK$ TYPE ;
: REDIM   ( n1 ---) ( change l'ordre des DL )
       DUP DEGREMAX >   IF
                     U. ." superieur a "
                      DEGREMAX U. EXIT
                      ELSE   DEGRE ! .DL
                      THEN ;
: #NIVEAU$   ( n1 --- str ) ( idem #NIVEAU mais )
       ( pour les chaines )
       PILENIV @ + 4 /MOD DROP 1+ STACK$ ;
: X$   ( ---str ) ( retourne la chaine du DL en )
                  ( X )
       0 #NIVEAU$ ;
: Y$   ( --- str ) ( idem pour Y )
       1 #NIVEAU$ ;
: Z$   ( --- str ) ( idem pour Z )
       2 #NIVEAU$ ;
: T$   ( --- str ) ( idem pour T )
       3 #NIVEAU$ ;
: BAS   ( décrémente PILENIV )
       PILENIV @ ?DUP   IF
                        1-
                        ELSE   3
                        THEN   PILENIV ! ;
: HAUT   ( incrémente la valeur de PILENIV )
        PILENIV @ 1+ 4 /MOD DROP   PILENIV ! ;
: RUP   ( fait tourner la pile vers le haut )
       BAS .DL ;
: RDN   ( fait tourner la pile vers le bas )
       HAUT .DL ;
: CLX   ( no comment )
       BOUCLE 0 DO
                T I COEFF@   X I COEFF!
                LOOP
       PILENIV @ 3 + 4 /MOD DROP   1+ STACK$
       PILENIV @ 1+ STACK$ DROP  SMOVELEN   RDN ;
: CLST   ( remet pile et chaines à zero )
       STACK BOUCLE 64 * 0 NFILL
       0 X$ DROP 2- C!
       0 Y$ DROP 2- C!
       0 Z$ DROP 2- C!
       0 T$ DROP 2- C! ;
: ENTER   ( duplique le poly. en X )
       X T BOUCLE 8 *   CMOVE
       X$ T$ DROP SMOVELEN   RUP ;
: IFACT   ( n1 --- ) ( -F- n2 ) ( n2 est )
                     ( FACT(n1) )
       DUP DUP 1 <> *   IF
                     DUP ITOF 1 DO
                         I ITOF F*
                         ELSE   DROP 1.
                         THEN ;
: .   ( affiche les coeff. du poly en X )
       BOUCLE 0 DO
                CR X I COEFF@   ." a(" I . ." )="
                F. KEY DROP
                LOOP
       .DL ;
: X<>Y ( échange le contenu de X avec celui de )
                                          ( Y )
       BOUCLE 0 DO
                X I COEFF@   Y I COEFF@
                X I COEFF!   X<>Y   Y I COEFF!
                LOOP
       X$ SWAP OVER PAD SMOVE Y$ X$ DROP SMOVELEN
       PAD SWAP Y$ DROP  SMOVELEN  .DL ;
VARIABLE DEPOT ( sauvegarde momentanée d'un nbr )
```

```
VARIABLE VARIDEF   ( contient le CFA de la def. )
      ( d'une fonction ; COS, SIN ... )
: ?DEF   ( voir article )
       BAS BOUCLE 0 DO
                    I VARIDEF @ EXECUTE
                    X I COEFF!
                    LOOP
       PILENIV @ 1+ STACK$ DROP   SMOVELEN   .
                     DL ;
: IMPAIR? ( n1 --- b ) ( renvoit 1 si n1 impair )
                              ( 0 sinon )
       1 AND ;
: PAIR?   ( n1 --- b ) ( renvoit -1 si n1 pair, )
                              ( 0 sinon )
       IMPAIR? 1- ;
: EXPDEF    IFACT 1/X ;
: EXP   " EXP" ['] EXPDEF VARIDEF ! ?DEF ;
: SINDEF   DUP IMPAIR?   IF
                  -1. DUP 1- 2/ ITOF Y^X IFACT F/
                   ELSE   DROP 0.
                   THEN ;
: SIN   " SIN" ['] SINDEF VARIDEF ! ?DEF ;
: LOGDEF   DUP IF
              DUP 1+ -1. ITOF Y^X ITOF F/
             ELSE   DROP 0.
             THEN ;
: LOG   " LOG" ['] LOGDEF VARIDEF ! ?DEF ;
: SHDEF   DUP IMPAIR? IF
                   IFACT 1/X
                   ELSE   DROP 0.
                   THEN ;
: SH   " SH" ['] SHDEF VARIDEF ! ?DEF ;
: COSDEF   DUP PAIR? IF
                  -1. DUP 2/ ITOF Y^X IFACT F/
                   ELSE   DROP 0.
                   THEN ;
: COS   " COS" ['] COSDEF VARIDEF ! ?DEF ;
: CHDEF   DUP PAIR? IF
                 IFACT 1/X
                 ELSE   DROP 0.
                 THEN ;
: CH   " CH" ['] CHDEF VARIDEF ! ?DEF ;
: ATANDEF   DUP IMPAIR?   IF
                  -1. DUP 1- 2/ ITOF Y^X ITOF F/
                   ELSE   DROP 0.
                   THEN ;
: ATAN   " ATAN" ['] ATANDEF VARIDEF ! ?DEF ;
: ATHDEF   DUP IMPAIR?   IF
                   ITOF 1/X
                   ELSE   DROP 0.
                   THEN ;
: ATH   " ATH" ['] ATHDEF VARIDEF ! ?DEF ;
: ASINDEF   DUP IMPAIR?   IF
                   DUP 1- DUP DUP IFACT 2. ITOF
                   Y^X F/ 2/ IFACT X^2 F/ ITOF
                     F/
                   ELSE   DROP 0.
                   THEN ;
: ASIN   " ASIN" ['] ASINDEF VARIDEF ! ?DEF ;
: ASHDEF   DUP IMPAIR?   IF
                  -1. DUP 1- 2/ ITOF Y^X DUP 1-
                  DUP DUP IFACT 2. ITOF Y^X F/
                    2/
                  IFACT X^2 F/ ITOF F/
                  ELSE   DROP 0
                  THEN ;
: ASH   " ASH" ['] ASHDEF VARIDEF ! ?DEF ;
: POLY   ( permet de rentrer un poly. quelconque )
      BAS   X BOUCLE 16 * 0 NFILL
      X$ DROP DUP CR ." Nom du poly : "
      EXPECT96 2- SPAN @ SWAP C!    DEPTH DEPOT !
      BOUCLE 0 DO
              ." a(" I U. ." )=" QUERY
             #TIB @   IF
                   INTERPRET DEPTH DEPOT @ -   IF

                      ITOF

                      THEN
                   X I COEFF!
                   ELSE   LEAVE
                   THEN
             LOOP
      .DL ;
CREATE SCRATCH1   DEGREMAX 1+ 16 * NALLOT
CREATE SCRATCH2   DEGREMAX 1+ 16 * NALLOT
CREATE SCRATCH3   DEGREMAX 1+ 16 * NALLOT
: /   ( divise le poly. en Y par celui en X )
      SCRATCH1 BOUCLE 16 * 0 NFILL
      -1 BEGIN
            1+ DUP X SWAP COEFF@   X=0? 1+
            UNTIL
      BOUCLE DEPOT @ DO
            Y I COEFF@   X DEPOT @ COEFF@
        F/
            SCRATCH1 I DEPOT @ - COEFF!
      0 DEGRE @ I - DO
           Y J I + COEFF@   X DEPOT @ I + COEFF@
          Y J COEFF@   F*   X DEPOT @ COEFF@ F/
           F-   Y J I + COEFF!
```

```
        -1 +LOOP
      LOOP
    BOUCLE 0 DO
            SCRATCH1 I COEFF@  Y I COEFF!
            LOOP
    " (" Y$ S>& " )/(" S<& X$ S<& " )" S<&
            2DROP
    CLX ;
: COMP  ( compose le poly. en Y par celui en X )
    Y 0 COEFF@  X=0? 1+  IF
             CR ." Y(0)#0" EXIT
            THEN
    SCRATCH2 BOUCLE 16 * 0 NFILL
    BOUCLE 0 DO
            Y I COEFF@  SCRATCH1 I COEFF!
            LOOP
     X 0 COEFF@  SCRATCH2 0 COEFF!
    BOUCLE 0 DO
           X 1 COEFF@  Y I COEFF@ F*
           SCRATCH2 I COEFF@ F+
           SCRATCH2 I COEFF!
           LOOP
    BOUCLE 2 DO
           BOUCLE 0 DO 0.
                I 1+ 0 DO
                     Y I COEFF@
                     SCRATCH1 J I - COEFF@
                     F* F+
                     LOOP
                  SCRATCH3 I COEFF!
                LOOP
           BOUCLE 0 DO
                  SCRATCH3 I COEFF@
                  Y I COEFF!
                  LOOP
           BOUCLE 0 DO
                  Y I COEFF@  X J COEFF@ F*
                  SCRATCH2 I COEFF@ F+
                  SCRATCH2 I COEFF!
                  LOOP
           LOOP
    BOUCLE 0 DO
             SCRATCH2 I COEFF@  Y I COEFF!
             LOOP
     X$ " [" S<& Y$ S>& " ]" S<& 2DROP CLX ;
: CHS  ( change le signe du poly. en X )
    X 15 +  BOUCLE 0 DO
            DUP 9 TOGGLE 16 +
            LOOP  DROP
    " -(" X$ S>& " )" S<& 2DROP .DL ;
```

```
: *  ( multiplie le poly. en X par celui en Y )
    BOUCLE 0 DO 0.
            I 1+ 0 DO
                 X I COEFF@  Y J I -
                COEFF@
                 F* F+
                 LOOP
            SCRATCH1 I COEFF!
            LOOP
    BOUCLE 0 DO
            SCRATCH1 I COEFF@  Y I COEFF!
            LOOP
    " (" X$ S>& " )*(" S<& Y$ S>& " )" S<&
             2DROP
     CLX ;
: +  ( additionne le poly. en X avec celui en Y )
    BOUCLE 0 DO
            X I COEFF@  Y I COEFF@  F+
            Y I COEFF!
            LOOP
    Y$ " +" S<& X$ S<&  2DROP CLX ;
: -  ( soustrait le poly. en X à celui en Y )
    BOUCLE 0 DO
            Y I COEFF@  X I COEFF@  F-
             Y I COEFF!
             LOOP
    Y$ " -(" S<& X$ S<& " )" S<& 2DROP CLX ;
: FRC  ( n1 --- )  ( donne la fraction du )
     ( coeff. d'ordre n1 du poly. en X )
    ( utilise FRC qui doit se trouver dans le )
    ( vocabulaire FORTH, voir article )
    DUP DEGRE @ > IF
                U. ." c'est trop grand." EXIT
                THEN
    X SWAP COEFF@  FRC  PAD EXPECT96  .DL ;
```



PARTIES DE FRACTIONS

Je connaissais deux programmes de mise en fraction d'un nombre décimal, l'un de Jean-Jacques Dhénin paru dans son livre sur la 41, l'autre, en BASIC, de Pierre David paru dans la célèbre revue JPC (no20).
Ces deux programmes sont très bons, mais les auteurs ne pouvaient pas nous expliquer leur fonctionnement !
Par conséquent, je vous propose un programme de mise en fraction en FORTH dont le fonctionnement est compréhensible par toute personne ayant suivi une classe de 6ème même moyenne.

Si on appelle X le nombre décimal que l'on désire mettre sous forme de fraction, notre problème se ramène à trouver les plus petits entiers A et B vérifiant X=A/B.
Maintenant, je ne pense troubler personne en écrivant A=BX, il ne nous reste plus qu'à faire varier B dans l'ensemble des entiers naturels (B=1,2,3,4,...) et dès que le produit BX est entier, on a notre couple (A,B). Voilà le plus compliqué est compris, maintenant il ne s'agit plus que de "remarques".

Première remarque: si vous espérez tomber sur un entier en faisant le produit BX, vous pouvez laisser votre programme tourner très longtemps. Nous considérerons donc qu'un nombre est entier si sa partie fractionnaire est inférieure à 0,000001 ou supérieure à 0,999999.

Deuxième remarque: si le résultat que l'on cherche est par exemple 3/1000, B devra varier de 1 jusqu'à 1000, ce qui peu paraitre un peu long. Donc quand X est inférieure à 1 (i.e. B>A) le plus rapide est de faire varier A plutôt que B (A=B/X), dans notre exemple on aura A=3, ce ce qui sera moins long que B=1000.
Ceci étant vu, il n'empêche que si la fraction que vous cherchez est 1000/1001, il faudra attendre que notre boucle s'effectue mille fois, d'où l'intéret d'une méthode magique...

Alain HERREMAN

```
: FRC
   X=0?   IF   ." 0" EXIT   THEN
     0 0. X<>Y X<Y?   IF   1-   THEN   FABS
     0 1. X>Y? X<>Y   IF   1+   1/X   THEN
  0 BEGIN
      1+ DUP   FENTER   ITOF F* FENTER FP FABS
     .000001 X>Y?   RDN .999999 X<Y?   +   RUP
     UNTIL
  RUP .1  F+ IP   ROT   IF   ." -"   THEN   SWAP
  IF   .   ." /"   F.
   ELSE   F.   ." /"   .
   THEN ;
```

TRI

Dans le numéro 30 de JPC je vous proposais un programme de tri dans la pile en FORTH, ce mois-ci je vous propose le même programme, toujours aussi inutile, mais bien plus rapide puisqu'il s'agit d'une primitive FORTH. Quand je dis plus rapide je veux dire qu'il trie 100 nombres en moins de 2 secondes! L'algorithme différe un peu de celui utilisé précédemment, mais un autre membre du club a publié un article qui explique le principe du tri selon cette méthode, je me contenterai donc de vous renvoyer à l'article de Jean-Claude Foures dans le JPC n26 page 63.

Alain Herreman

```
FORTH
WORD  'SORT'
CD1EX
R0=C
R1=C
CD1EX
LC(5)  #2FB11
CD0EX
A=DAT0 A
```

```
        CD0EX
        D1=D1+ 5
       CD1EX
        ?C>=A  A
        GOYES   FIN
        P=      0
LBL1    LC(5)   #2FB11
        CD0EX
        A=DAT0 A
        CD0EX
        AD1EX
LBL2    D1=D1- 5
        A=DAT1 A
        D1=D1+ 5
        C=DAT1 A
        ?C>=A  A
        GOYES   C>=A
        DAT1=A A
        D1=D1- 5
        DAT1=C A
        D1=D1+ 5
C>=A    D1=D1- 5
        C=R0
        D=C     A
        AD1EX
        C=A     A
        AD1EX
        ?C#D    A
        GOYES   LBL2
        LC(5)   #2FB11
        CD0EX
        A=DAT0 A
        CD0EX
        C=R0
        CD1EX
        D1=D1+ 5
        CD1EX
        R0=C
        ?C#A    A
        GOYES   LBL1
FIN     C=R1
         CD1EX
        RTNCC
```


KRISTAL S.A.
informatique

HEWLETT PACKARD

systèmes informatiques
interfaçage et électronique
applications techniques
instrumentation
logiciels
tél : 76 90 38 13

KRISTAL, chemin des Clos Zirot 38240 MEYLAN

SIMPLE COMME HPIL

L'HPIL ou interface Loop est un moyen simple, économique, fiable pour relier des périphériques autour d'un contrôleur.

En complément à la gamme de périphériques existants chez HEWLETT-PACKARD : imprimante, unité de disque , centrale d'acquisition, voltmètre, interfaces HPIL RS 232 ou HPIL HPIB.

KRISTAL SA propose une panoplie de cartes industrielles de faible coût.

1) Carte COMPTEUR intelligent K009

Comptage simultané sur 6 voies jusqu'à 99 999 pour une fréquence maximale de 1 Mégahertz. La temps de comptage est programmable de 10 millisecondes à quelques jours. Cette carte peut éventuellement interrompre le contrôleur quand le comptage est terminé.

2) Interface BCD-HPIL K010

Ce boîtier permet la connexion des machines HP 41 C, HP 71 et la série HP 80 en HPIL à tout système équipé d'une sortie BCD parallèle d'un maximum de 8 digits.
Très polyvalente, cette interface se configure par logiciel et dispose de 4 Digits en sortie, pour commande éventuelle de voyants, Leds ou Relais.

3) Interface HPIL-PLOTTER K011

Les périphériques disposant d'une entrée parallèle de type CENTRONICS : Imprimante, Traceur, pourront être intégrés dans la boucle HPIL grâce à ce produit.
Cette interface assure également la transformation des commandes HP-en langage plotter TANDY, CANON, SHARP...

4) MODEM pour réseau privé K012-K013

Ce modem conforme aux normes Européennes
.CCITT V 21 (300 bauds duplex intégral)
.CCITT V 23 (600 bauds semi duplex et 1200 bauds)
assure la détection de sonnerie, la réponse et la numérotation automatique. Il détecte les erreurs de transmission. Il est remarquable pour sa faible consommation 220 mA en fonctionnement et 5 mA en veille. Ce modem se présente sous forme de 2 cartes au format Europe 110 x 160 mm. Une des cartes est destinée au branchement sur le réseau commuté.

5) Interface MINITEL K016

Transforme votre HP 41 ou HP 71 et 75 en lui offrant un véritable clavier écran et un modem agréé PTT.

6) Coffret de conversion de données sur 12 bits

Petite Centrale d'Acquisition, elle est composée de :
. 7 entrées analogiques 0 à 10 volts
. 1 entrée thermocouple K
. 2 sorties analogiques de 0 à 10 volts
Les 7 relais du multiplexeur d'entrée sont utilisables en actuateur.

7) Carte parallèle K017

20 lignes d'entrée sortie bidirectionnelles dont 2 interruptibles (disponible fin décembre 85).
Coller un relais, lire la position d'un switch interrupteur "simple comme HPIL".

8) Clavier 16 touches et affichage 20 caractères

Pour la visualisation et la saisie de données en milieu industriel (disponible fin janvier 1985).

Détail des prix unitaires H.T

- Compteur K009	3 920 F
- BCD K010	6 300 F
- Plotter K011	4 500 F
- Modem en coffret K012	6 750 F
les 2 cartes nues sans Alimentation	4 000 F
- Minitel K016	2 500 F
- Parallèle K017	2 500 F environ
- Centrale d'acquisition	19 500 F

Ces prix sont des prix unitaires, des remises motivantes sont consenties suivant les garanties.

N.B : KRISTAL SA reste à votre disposition pour développer toute carte HPIL ou adapter le HPIL sur un de vos équipements existants.

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

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

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