JPC

FEVRIER 1988
NUMERO 51

Le numéro 40 F

A PROPOS DU CLUB

P. David	Editorial	*1*
J.J. Dhénin	Compte-rendu des Assemblées Générales	*2*
	Compte-rendu d'activités - PPC Paris 1987	*2*
	Courrier du coeur	*3*

DUR ET MOU

P. David	Module infra-rouge pour HP-41	*6*
J. Taillandier	Nouveaux calculateurs HP	*6*
P. David	Nouveaux produits CMT	*6*
P. Courbis	34 Ko sur HP-28C	*7*

HP28

P. Courbis	Le tour du HP-28C	*10*
P. Courbis	PEEK et POKE pour HP-28C	*24*

HP41

M. Markov	Utilitaires de gestion de disques (acte III)	*28*

HP75

J.Y. Hervé	KEYLEX pour HP-75	*32*

HP71

J. Baudier	L'assembleur du HP-71 (acte VI)	*34*
F. Duret-Lamouroux	Modélistes, à vos tours !	*39*

EDITORIAL

Chers Membres,

Tout d'abord, permettez-moi de vous remercier au nom du Bureau pour la confiance que vous nous avez accordé en nous élisant lors de notre Assemblée Générale.

Ce début d'année est riche en faits nouveaux.

La présentation du Journal évolue. Vous vous en êtes peut-être déjà rendu compte, le journal est imprimé en format A3, puis agrafé au centre et plié. J'espère que cette présentation saura vous plaire.

Nous inaugurons ce mois-ci « l'année beige ». Les 10 journaux de cette année auront tous des couvertures de cette couleur.

La présentation intérieure change également : vous trouverez régulièrement les informations générales sur votre Club favori dans les pages centrales.

D'autre part, Hewlett-Packard et Corvallis MicroTechnology annoncent des nouveautés importantes. Nous pensons qu'elles vous intéresseront.

En vous souhaitant beaucoup de bonnes choses pour cette année, je vous laisse apprécier votre nouveau JPC, cuvée 1988.

Pierre David (37)

COMPTE-RENDU
DES ASSEMBLEES GENERALES

Nos Assemblées Générales se sont déroulées le samedi 16 janvier 1988, de 14 à 16 heures.

35 membres étaient présents, et 76 étaient représentés par les pouvoirs que vous nous avez envoyés. C'est dire que nous avons largement dépassé le quorum nécessaire.

L'Assemblée Générale Extraordinaire a adopté les statuts proposés en tenant compte de deux modifications. Ainsi, les articles 2 et 17 sont ils à présent :

Article 2 :

Cette Association a pour but de réunir des amateurs et utilisateurs de l'informatique ainsi que d'assurer la mise en commun des connaissances informatiques de chacun au profit de tous.

Article 17 :

Une Assemblée Générale Extraordinaire peut être convoquée dans les mêmes conditions qu'à l'Article 16 ou sur demande d'au moins un quart des membres actifs.
Son objet ne peut être alors que de décider :
- une modification des Statuts ;
- une radiation ;
- une fusion avec toute association de même objet ;
- la dissolution.

Le quorum d'une telle Assemblée est fixé à 20 % des membres actifs, présents ou représentés, et les décisions doivent y être prises à la majorité des 2/3. Si le quorum n'est pas atteint, une nouvelle Assemblée Générale Extraordinaire est convoquée qui délibérera sans contrainte.

Les statuts ont été adoptés à l'unanimité.

L'Assemblée Générale s'est ensuite ouverte par la présentation du rapport moral (voir article suivant) par Pierre David.

Notre Trésorier, Janick Taillandier, nous a ensuite présenté le rapport financier. La situation du Club s'est nettement améliorée. Sauf incident majeur, il n'y aura pas d'augmentation de la cotisation cette année.

L'Assemblée Générale a ensuite approuvé à l'unanimité le rapport financier.

Le nouveau Bureau a été finalement élu (toujours à l'unanimité !) et comprend :

- Président : Pierre David (37)
- Trésorier : Janick Taillandier (246)
- Secrétaire : Jean-Jacques Dhénin (177)
- Secrétaire : Olivier Arbey (118)
- Secrétaire : Jacques Baudier (192)
- Secrétaire : Eric Gengoux (108)

Nous remercions chaleureusement les membres qui se sont déplacés ainsi que ceux qui nous ont envoyés leur pouvoir.

Jean-Jacques Dhénin (177)

COMPTE-RENDU D'ACTIVITES
PPC PARIS - 1987

L'année 1987 a été une très bonne année pour notre Club. De nombreuses modifications ont été apportées au fonctionnement du Club parmi lesquelles certaines sont fondamentales.

Commençons par le Journal : vous avez pu apprécier la régularité de parution du Journal et la qualité de sa présentation. Ceux qui sont venus aux réunions ont toujours trouvé sur place leur exemplaire et pour les autres, il a été expédié le soir même. Ceci signifie d'une part une planification et un respect des contraintes que nous nous sommes fixées, mais aussi un gros travail, quelquefois en peu de temps. Pour parvenir à cette amélioration, il a fallu développer de nouveaux programmes.

Comme vous le savez maintenant, la gestion du fichier des adhérents est à présent informatisée. Cela simplifie certaines tâches administratives, et notamment l'expédition du Journal puisque les étiquettes sont imprimées automatiquement et non plus écrites à la main comme dans les temps zéroïques. Bien entendu, ce fichier a été déclaré à la Commission Nationale de l'Informatique et des Libertés.

Nous avons obtenu en mai une boîte postale. Depuis, le courrier est traité régulièrement chaque semaine, ce qui a renforcé le lien qui unit le Bureau aux membres du Club.

Dans la foulée, les demandes de renseignements (par exemple les cartes placées dans les boîtes de HP-41) sont satisfaites par un document de deux pages beaucoup moins coûteux et plus explicite que l'envoi d'un ancien numéro comme dans les temps zéroïques.

Le projet d'une Rom est considérablement avancé. A ce jour plus de cent mots-clef sont fixés. Le manuel existe en français et en anglais. Grâce aux Lex que vous nous avez envoyés, il a été possible de réaliser un module de 26 Ko, représentant 505 Ko et 25000 lignes de source. Actuellement, une vingtaine de personnes bénéficient déjà de ce module et nous espérons que vous serez de plus en plus nombreux.

De nombreux Clubs étrangers sont intéressés par la distribution de JPC Rom.

Tout cela fait un ensemble de bonnes raisons pour expliquer notre passage de 130 à 240 membres actifs.

Cette bonne santé du Club contraste avec la disparition de clubs similaires au nôtre. Ainsi, CHHU, le club de Richard Nelson aux Etats-Unis, PPC aux Etats-Unix, PPC en Suisse alémanique, et enfin le club de Toulouse. Nous sommes à présent le seul club en France.

Bien que Jean-Daniel (responsable du club de Toulouse) ait préconisé à ses membres de nous rejoindre, peu d'entre eux l'ont à présent fait.

Par ailleurs, nos amis Danois ont organisé une excellente conférence (voir JPC 47) à Copenhague cet été. Elle démontre si besoin est la vitalité de leur Club. Nous étions malheureusement trop peu nombreux à représenter PPC Paris.

Hewlett-Packard a sorti au début de cette année le HP-28C. C'est la première machine d'une série qui devrait redonner de l'élan au Club. Nous-mêmes avons publié un banc d'essai très complet, et nous continuerons à le faire tant que possible. De même, certains d'entre nous étaient présents au Sicob sur le stand HP même. A cette occasion, HP a publié un tiré à part du banc d'essai, largement distribué.

Nous avons également travaillé à la refonte des statuts qui commençaient à dater des temps zéroïques.

Nous avons étendu le service du Club à la vente à prix avantageux de matériel. Ainsi, des interfaces vidéo, des lecteurs de cartes, etc. vous sont proposés. Il s'agit de matériel neuf cédé par Hewlett-Packard pour un bon prix.

Nous sommes toujours en possession de la bibliothèque européenne des utilisateurs de Genève (U.P.L.E.). Celle-ci contient plusieurs centaines de programmes pour HP-41 et date des temps zéroïques. Nous espérons trouver cette année des volontaires pour l'exploiter.

Philippe Assouline s'est proposé pour gérer et diffuser la bibliothèque de programmes pour HP-71 parus dans JPC. Espérons que la mise en place se fera rapidement pour la satisfaction de tous.

Nous avons suspendu récemment la ligne téléphonique du Club. Le peu d'appels reçus ne justifiait pas le coût de l'abonnement.

Deux projets sont déjà en route pour cette nouvelle année. Premièrement, le journal est édité en format A3, plié et agraffé au centre. Deuxièmement, nous expérimentons actuellement la création d'un serveur PPC sur minitel (3615, code SER*PPC) pour une communication plus rapide et plus directe.

Nous ne voyons pas se profiler à l'horizon de projet aussi important que JPC Rom. Dans ce domaine, l'avenir vous appartient.

Le Bureau

COURRIER DU COEUR

Philippe Guez
56 rue J.J. Rousseau
75001 Paris
Tél : (1) 45 06 25 03 (H.B.)

Vend :
Imprimante ThinkJet HP-2225B : 3500 F, interface vidéo 32 colonnes HP-82163B : 500 F, lecteur de cartes magnétiques pour HP-71B : 1250 F, cartes magnétiques pour HP-71B : 1 F pièce, convertisseur HP-82166A : 1250 F, module HP-IL pour HP-41 : 1250 F, modules Ram 4 Ko pour HP-71B : 500 F pièce, module Games, Auto-duplication, et Math pour HP-41 : 150 F pièce, machine à écrire Praxis 35 interfaçable RS232 : 3000 F.

Janick Taillandier
335 rue Lecourbe
75015 Paris

Vend :
Lecteur de disquettes HP-9114A : 3000 F, imprimante ThinkJet HP-2225B : 2500 F.

Un module 32 Ko Ram + 32 Ko Eprom HHP pour HP-71 : 3000 F.

Jean-Jacques Dhénin
35 rue Boileau
92120 Montrouge
Tél : (1) 42 53 91 60

Vend :
HP-75 + accessoires et docs d'origine.

Pierre Picheret
Calmon Aiguefonde
81200 Mazamet

Vend :
Imprimante ThinkJet HP-2225B : 2500 F, imprimante HP-82905B : 1500 F.

Daniel Jacob
4 rue des Blanchisseurs
92370 Chaville

Vend :
HP-41C + Modules quadruple, time, X-Fonctions + Imprimante HP-82143A + lecteur de cartes + documentation (manuels, *Tips and Routines*, *Synthetic Programming*) + programmes nombreux avec documentation : le tout pour 2500 F.

MLDL pour pratiquer l'assembleur sur HP-41 (24 Kmots Rom adressables + 8 Kmots Ram) avec logiciel (en assembleur) sur Eprom (MLDL Eprom de chez Eramco) + Manuels + listings internes de la HP-41 + feuillets d'initiation au Micro-code. Le tout pour 1000 F.

Eric Gengoux
8 rue de Furstenberg
75006 Paris
Tél : (1) 46 33 65 79 (soir)

Vend :
Lecteur de cassettes HP-82161A (1983), révisé par HP : 1700 F.
Imprimante thermique HP-82162A (1983) : 1200 F.
Mini table traçante HP-IL Sicape (convertisseur HP-IL intégré) (1982) : 1800 F.
Machine à écrire Brother EP44 (1983) (8 Ko, interface RS232) utilisable comme clavier d'un HP-71 ou HP-75, avec logiciel pour ces machines : 1100 F.

Jean Reibel
9 square Victor Fleming
92350 Le Plessis Robinson

Cherche :
Modules Paname, PPC Rom, Graphique, HP-IL development et interface HP-IL/HP-IB.

Laurent Istria
18 rue Pierre et Marie Curie
75005 Paris
Tél : (1) 46 33 71 57

Vend :
HP-28C : 1500 F.

Jean-Jacques Moreau
64 avenue de la Paix
93150 Le Blanc-Mesnil
Tél : (1) 48 67 33 04

Vend :
HP-71 : 3000 F, lecteur de disquettes HP-9114A : 3000 F, module HP-IL pour HP-71 : 1000 F, module Forth / Assembleur pour HP-71 : 1000 F, modules Maths : 600 F, excellent état.

DUR ET MOU

P. David — Module infra-rouge pour HP-41 *6*
J. Taillandier — Nouveaux calculateurs HP *6*
P. David — Nouveaux produits CMT *6*
P. Courbis — 34 Ko sur HP-28C *7*

MODULE INFRA-ROUGE POUR HP-41

Hewlett-Packard produit un nouveau module pour HP-41 : il s'agit d'un module infra-rouge pour utiliser l'imprimante thermique HP-82240 des HP-18C et HP-28C.

L'ensemble module infra-rouge HP-82242 + imprimante HP-82240 est vendu aux Etats-Unis $65 + $135 (ou $55 + $106 chez EduCALC). C'est à présent le moyen le plus économique pour imprimer sur HP-41.

Pierre David (37)

NOUVEAUX CALCULATEURS HP

Hewlett-Packard a annoncé début janvier 4 nouvelles machines dans la série des HP-18C et HP-28C maintenant bien connues.

HP-28S

Le HP-28S (S pour Scientific) est très semblable au HP-28C, mais il comporte (enfin) 32 Ko de mémoire, et de nouvelles fonctions, en particulier pour le stockage et le rappel de pages graphiques, ainsi que des menus arborescents.

Le prix annoncé est de $ 235, c'est à dire le prix actuel du HP-28C.

HP-27S

Disposant de deux lignes d'affichage seulement, ce nouveau calculateur scientifique ne possède pas de fonction graphique, mais 6,7 Ko de mémoire. Le jeu de fonctions est un sous-ensemble de celui du HP-28C, enrichi toutefois de fonctions de calculs financiers. La présentation diffère du reste de la gamme dans la mesure où ce calculateur est en un seul volet au lieu de deux rabattables comme sur les HP-28C et HP-18C actuels.

Le HP-27S se situe donc en bas de gamme de la série scientique des nouveaux calculateurs.

Le prix annoncé est de $ 110.

HP-19B

C'est un calculateur financier (B pour Business) qui reprendrait les fonctions du HP-18C et ajouterait le graphique, et porterait la mémoire à 6,5 Ko.

Le prix annoncé est de $ 175.

HP-17B

C'est un calculateur financier simplifié, également en un seul volet.

Le prix annoncé est de $ 110.

Ces machines ne seront pas disponibles en Europe avant au mieux le deuxième trimestre 1988. Nous vous tiendrons bien sûr informés de leur sortie.

Je tiens à remercier Vincent Delorme et Eric Gengoux qui m'ont très rapidement communiqué ces informations.

Janick Taillandier (246)

NOUVEAUX PRODUITS CMT

La firme américaine CMT (Corvallis MicroTechnology, Inc) vient d'annoncer trois nouveaux produits.

MC-II

Je vous en parlais dans JPC 47 (page 3), CMT a annoncé officiellement la sortie de MC-II, une nouvelle machine portable. Il s'agit d'un ordinateur basé sur le processeur 80C88, disposant de 640 Ko au maximum, 8 lignes de 21 caractères.

MC-II possède un émulateur de HP-41 permettant d'utiliser les programmes écrits pour cette machine.

D'autre part, MC-II se programme en langage C sur compatible PC, les programmes étant envoyés ensuite vers MC-II par une liaison RS232 et le logiciel Kermit (fourni avec MC-II).

Le prix comprenant MC-II, 128 Ko de mémoire, le système d'exploitation et l'émulateur 41M en Rom est de $ 495 (offre spéciale de lancement) et passera après le 31 mars à $ 750.

Les extensions prévues comprennent un interface HP-IL (disponible au deuxième trimestre 1988), un langage Basic (deuxième trimestre), un mode base de données et un mode horloge.

D'autres modèles baptisés MC-I, MC-IV et MC-V seront annoncés en mars et juin 1988.

Multicase et écran a cristaux liquides

Le *Multicase* est un boîtier protecteur pour le HP-71. Il comprend un écran à cristaux liquides de 8 lignes de 40 caractères, un Ram disc HP-IL de 128 Ko (extensible à 512 Ko), des sorties HP-IL et RS-232, ainsi qu'un radiateur pour réchauffer le boîtier en cas de travail en extérieur à basse température. Le prix est de $ 925, sans HP-71.

Le point le plus intéressant de cette annonce est l'afficheur à cristaux liquides, puisque celui-ci sera disponible isolément (en version HP-IL) au deuxième trimestre pour le prix de $ 295.

Cet afficheur supporte un mode graphique accessible avec les mêmes séquences d'échappement que l'imprimante ThinkJet, et une résolution de 64×240 points.

Module mémoire pour HP-71

Complétant sa gamme, CMT annonce un nouveau module de mémoire vive pour le HP-71. Il s'agit d'un module 64 Ko (avec sauvegarde par batterie intégrée) enfichable dans les ports avant.

Le prix est fixé à $ 295.

L'adresse de ce fabricant très dynamique est :
Corvallis MicroTechnology, Inc
895 N.W. Grant Avenue
Corvallis, OR 97330
U.S.A.

Pierre David (37)

34 K OCTETS SUR HP-28C

La société Software Operations and Systems, Co propose la modification de HP-28C en super HP-28C (avec 34 Ko ou 6 Ko de mémoire). J'ai fait transformer ma machine et voici le résultat :
- mémoire 34461 octets (1693 + 32 × 1024)
- au dos de la machine une plaque de plastique de 2 centimètres sur 3 recouvre l'incision pratiquée pour placer le module.

Les avantages :
- possibilité d'avoir de nombreux programmes simultanément,
- possibilité d'éditer ceux-ci sans problème (`No Room to Enter...`),
- gain en vitesse nul ou presque.

Les inconvénients :
- si la pile est très remplie, la fonction `MEM` est très lente, le garbage collector aussi,
- la fonction `PREV` ne fonctionne plus correctement.

Tout ceci coûte $300 pour 32 Ko et comprend module, montage et transport par messagerie (4 jours) ou $295 si le transport est fait par la Poste (3 semaines ou plus) ; $155 pour 4 Ko par messagerie ($150 sinon).

Le délai est d'environ 1 mois et demi après réception de la machine.

Voici l'adresse de l'entreprise :

Software Operations and Systems, Co
1850 East 17th Street
Suite 102
Santa Ana, CA 92701
USA

Téléphone : 714 558-1806

Paul Courbis (392)


CMT MC-II
CMOS SYSTEM
CMT CORVALLIS MICROTECHNOLOGY
ON


ON
RUN
END
LINE

HP28

P. Courbis — Le tour du HP-28C *10*

P. Courbis — PEEK et POKE pour HP-28C *24*

LE TOUR DU HP-28 EN 80 PAGES ... OU PRESQUE

L'article ci-dessous est le résultat de plusieurs mois de travail de Paul Courbis sur la structure interne du HP-28C. Paul a fait l'effort de rédiger et de présenter dans JPC *l'état actuel de ses connaissances. Cela semblera peut-être un peu difficile d'abord. Mais pour expérimenter vous-mêmes, Paul a également fourni PEEK et POKE.*

La publication de ce travail et sa disponibilité constituent un enrichissement pour tous.

Félicitons tous Paul pour ce travail considérable, et souhaitons que beaucoup parmi vous continue dans la voie ainsi tracée.

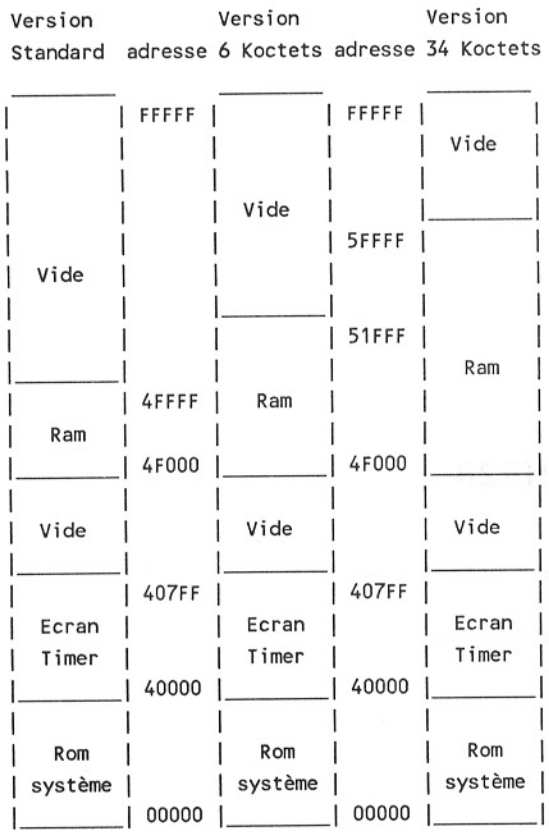
Le but de cet article est d'expliquer, ou plutôt de tenter d'expliquer, la manière de programmer en assembleur le HP-28C (c'est à dire en langage machine). Pour cela, une bonne connaissance de la structure interne de la machine est nécessaire et la première partie de cet article est consacrée à cela.

Tout d'abord, le microprocesseur du HP-28C est quasiment le même que celui du HP-71, ce qui permet de connaître les codes des différentes instructions du processeur. Il faut cependant savoir qu'une instruction nouvelle vient compléter celles présentes sur le HP-71 :

`PC=(A)` de code 808C. Cette instruction lit un groupe de 5 quartets situé à l'adresse contenue dans A et charge ces 5 quartets (c'est à dire une adresse) dans PC (Program Counter).

L'instruction RPL qui a permis l'accès à la programmation en assembleur est bien entendu `SYSEVAL` dont le rôle est exactement le même que `PC=(A)` : `SYSEVAL` prend l'entier binaire au niveau 1 de la pile, le tronque pour n'en garder que 5 quartets ; lit 5 quartets à cette adresse et les place dans le registre PC.

La structure de la mémoire est la suivante :



Le contenu de la mémoire est constitué de 4 types de groupes de quartets :

- ceux concernant l'écran ou le timer qui sont des zones mémoire d'entrée / sortie, quartets d'un type particulier,
- des données : suites de quartets représentant quelque chose. Par exemple, la valeur de correction de la routine d'horloge,
- des routines en langage machine,
- des objets RPL (comme ceux que l'utilisateur crée).

Il est à noter que ce genre de groupe de quartets se rencontre aussi dans la Rom ; en particulier, les instructions RPL sont des objets au même titre que ceux créés par l'utilisateur.

LES OBJETS

Pour bien comprendre le fonctionnement du HP-28C, il est nécessaire de bien connaître les différents types d'objets. Ils sont au nombre de 19 dont 9 peuvent être créés directement par l'utilisateur.

Ils commencent tous par un groupe de 5 quartets, le *prologue* qui indique leur type et, éventuellement, par des renseignements sur leur longueur, dimension...

Le tableau suivant résume les différents types d'objet ainsi que leur prologue. Ces objets sont ensuite détaillés.

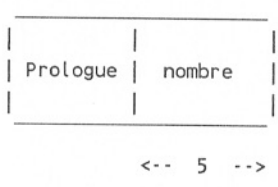
Objet	Prologue
Short integer (*)	02911
Real	02933
Extended Real (*)	02955
Complex	02977
Extended Complex (*)	0299D
Byte (*)	029BF
Premier objet inconnu (*)	029E1
Array	02A0A
Second objet inconnu (*)	02A2C
String	02A4E
Binary integer	02A70
List	02A96
Ram/Rom Pair (*)	02AB8
Algebraic	02ADA
Program	02C67
Assembly Code (*)	02C96
Global Name	02D12
Local Name (*)	02D37
Rom Pointer (*)	02D5C

(*) Ces objets sont impossibles à créer par l'utilisateur et apparaissent tous comme `System Object` (sauf Local Name).

Short integer

Prologue : 02911_{16}

Structure :

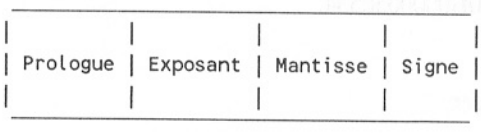


Exemple : 1192054321 est le short integer 12345.

Real

Prologue : 02933_{16}

Structure :

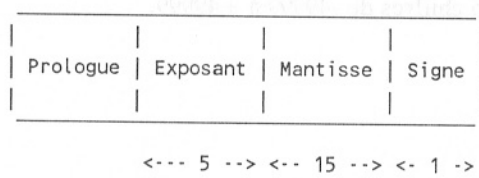


L'exposant et la mantisse sont en décimal codé binaire (DCB). L'exposant varie de 000 à 499 (positif) et de 999 à 501 (négatif). La mantisse contient 12 chiffres. Le signe est le signe de la mantisse : 0 si positif, 9 si négatif. Par exemple, $\pi \times 10^5$ est représenté, en valeur numérique, par : 339205009535629514130.

Extended Real

Prologue : 02955_{16}

Structure :

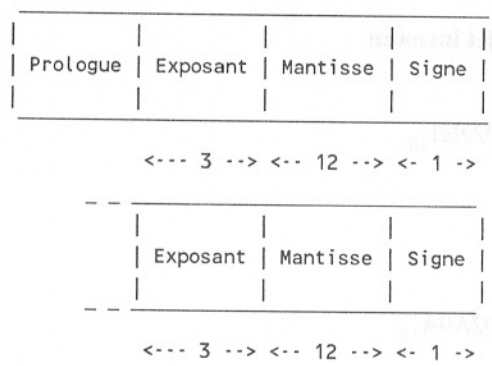


Même remarques que pour Real et exemple similaire avec 3 décimales de plus et un exposant compris entre -49999 et +49999.

Complex

Prologue : 02977_{16}

Structure :

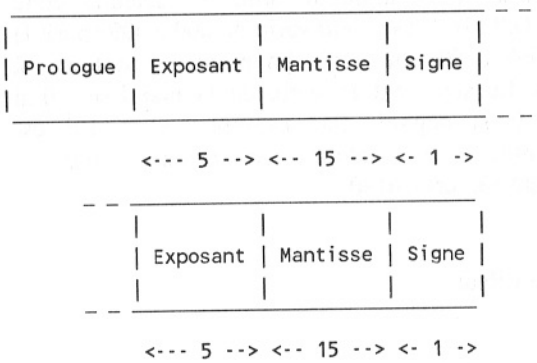


On trouve d'abord la partie réelle puis la partie imaginaire, sinon même structure que Real. Par exemple, `(10,20)` est représenté par : 77920100000000000001010000000000000020.

Extended Complex

Prologue : $0299D_{16}$

Structure :

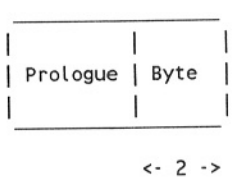


Même remarques que pour Real, exemple similaire à Complex avec 2 × 3 décimales en plus et des exposants à 5 chiffres de -49999 à +49999.

Byte

Prologue : $029BF_{16}$

Structure :



Par exemple, octet 01 : FB92010.

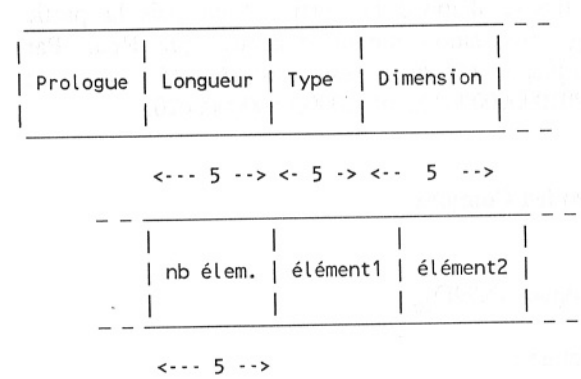
Premier objet inconnu

Prologue : $029E1_{16}$

Array

Prologue : $02A0A_{16}$

Structure d'un vecteur :



La *longueur* est le nombre de quartets à ajouter au pointeur RPL (D0) lorsqu'il pointe sur cette longueur pour qu'il pointe sur la fin de l'objet, c'est à dire la longueur + 5 (longueur de la longueur). Le *type* est soit réel soit complexe et est représenté par le prologue Real ou Complex. *Dimension* vaut 00001 dans le cas du vecteur (une seule dimension). *Nombre d'éléments* est le nombre de colonnes de la matrice 1 × n qu'est le vecteur.

La structure d'une matrice est quasiment la même. Cependant, *dimension* vaut 00002 et on a 2 × 5 quartets pour le nombre d'éléments (nombre de lignes / nombre de colonnes). Par exemple, `[[1 2] [3 4]]` est représenté par :

A0A20	prologue
95000	longueur
33920	type réel
20000	dimensions
20000	nombre de lignes
20000	nombre de colonnes
0000000000000010	premier réel
0000000000000020	deuxième réel
0000000000000030	troisième réel
0000000000000040	quatrième réel

Second objet inconnu

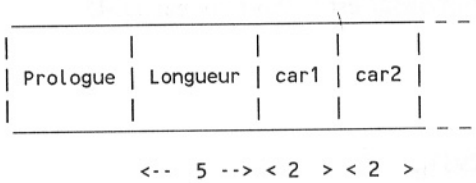
Prologue : $02A2C_{16}$

Cet objet, ainsi que le précédent inconnu, sont impossibles à charger par un `STO` ; ils provoquent un message `Range Exception`.

String

Prologue : $02A4E_{16}$

Structure :

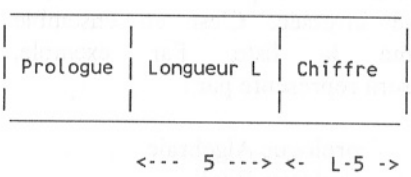


Longueur est définie comme pour Array et vaut également 2 × `SIZE(String)` + 5. Par exemple, `"PPC"` vaut : E4A20B0000050534.

Binary integer

Prologue : $02A70_{16}$

Structure :

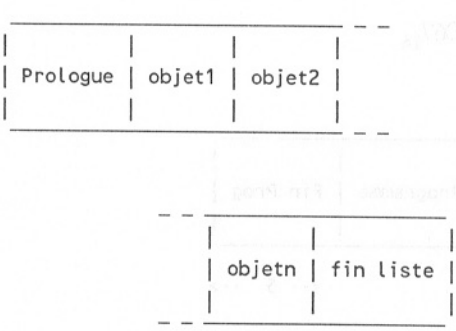


Habituellement L = 15_{16} pour les nombres entrés par l'utilisateur. Cependant, la valeur peut être différente ce qui permet un gain de place pour le stockage de petits entiers : #0 peut être codé 07A20600000. Par exemple, #123456789ABCDEF0 est représenté par : 07A20510000FEDCBA987654321.

List

Prologue : $02A96_{16}$

Structure :



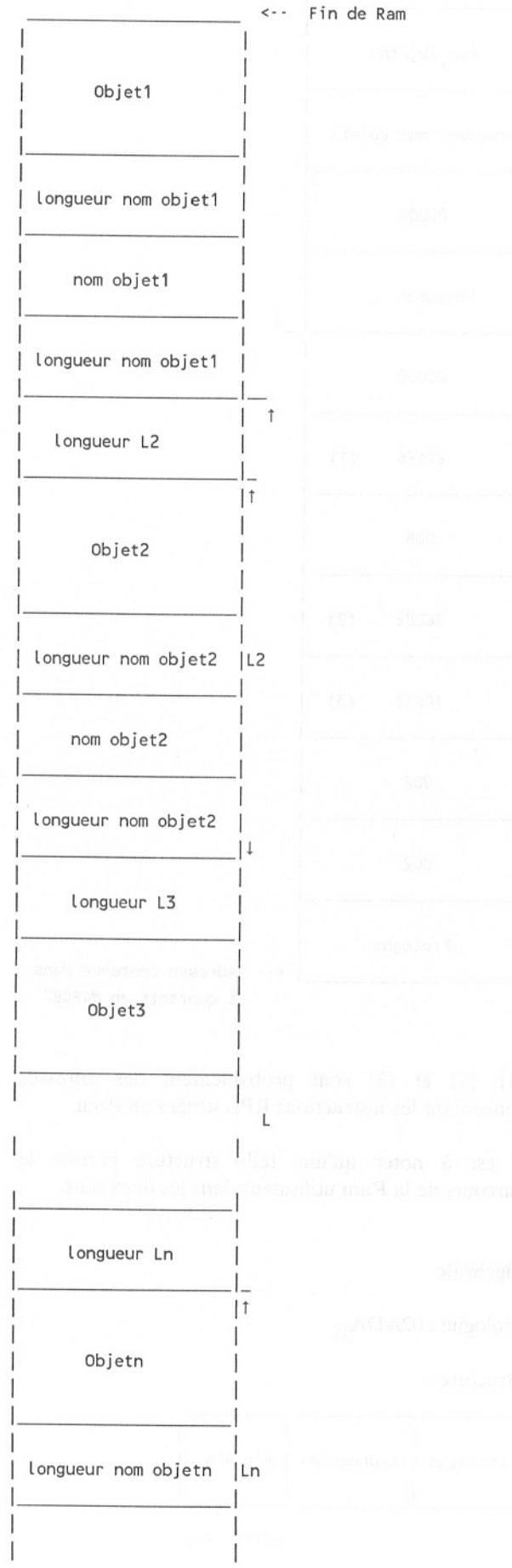
Fin de liste est le groupe de quartets 02F90. Par exemple, { #123456789ABCDEF0 "PPC" } est représenté par :

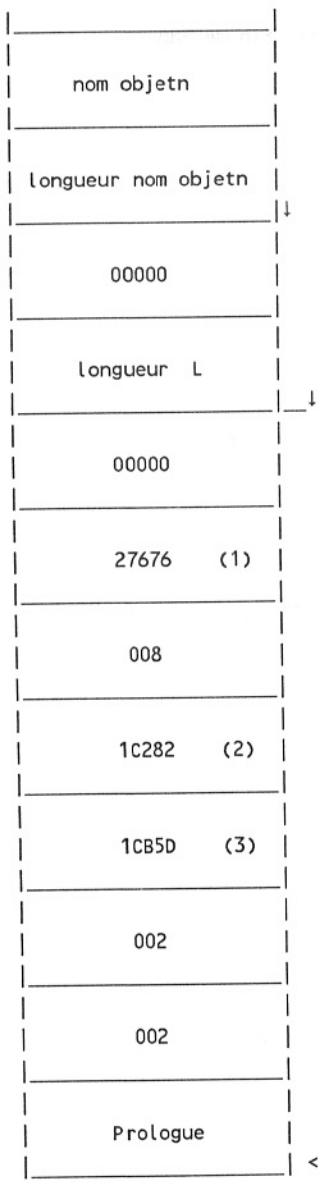
69A20	prologue List
07A20	prologue Binary integer
510000FEDCBA987654321	l'entier binaire
E4A20	prologue String
B0000050534	la chaîne
09F20	fin de liste

Ram/Rom Pair

Prologue : $02AB8_{16}$

Cet objet constitue en fait la Ram USER et sa structure est mal connue. Son explication est indissociable de celle de la structure de la partie de la Ram contenant les objets utilisateur. En voici la structure :





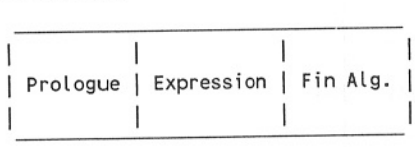
(1), (2) et (3) sont probablement des adresses concernant les instructions RPL situées en Rom.

Il est à noter qu'une telle structure permet le parcours de la Ram utilisateur dans les deux sens.

Algebraic

Prologue : $02ADA_{16}$

Structure :



Fin d'algèbre vaut 02F90. L'expression contient la suite des opérations à effectuer pour faire le calcul, en notation polonaise inversée. C'est un ensemble composite comme la liste. Par exemple, 1.33 × 7.42 + √74 sera représenté par :

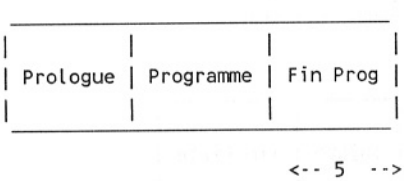
ADA20	prologue Algebraic
33920	prologue Real
0000000000003310	réel 1.33
33920	prologue Real
0000000000002470	réel 7.42
D9F81	+
33920	prologue Real
1000000000000470	réel 74
DD591	√
D9F81	+
09F20	fin d'Algebraic

Une telle structure permet de ne pas avoir à stocker de parenthèses : les calculs se font par l'intermédiaire de la pile.

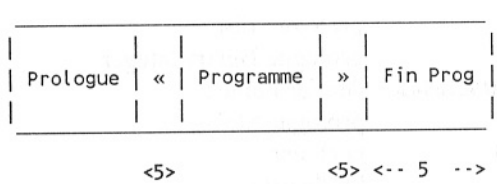
Program

Prologue : $02C67_{16}$

Structure :



Fin de programme vaut 02F90. Le programme est une suite d'objets : c'est donc un objet composite. Pour les programmes créés par l'utilisateur, la structure est plus complexe car elle est de la forme :



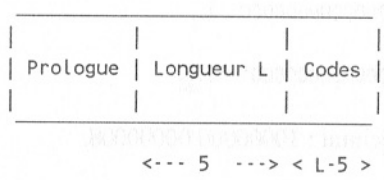
C'est encore un objet composite. Par exemple : « "PPC" #123456789ABCDEF0 » est représenté par :

76C20	prologue
A0F72	«
E4A20	prologue String
B0000050534	PPC
07A20	prologue Binary Integer
510000FEDCBA987654321	#123456789ABCDEF0
F1F72	»
09F20	fin de programme

Assembly Code

Prologue : $02C96_{16}$

Structure :

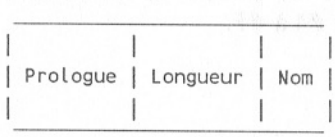


Codes représente la suite des codes des instructions du microprocesseur.

Global Name

Prologue : $02D12_{16}$

Structure :



Longueur est le nombre de caractères du nom. Par exemple, `'ABC'` est représenté par : 21D2030142434.

Local Name

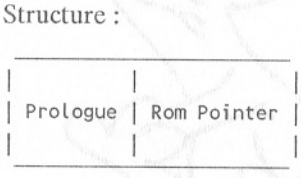
Prologue : $02D37_{16}$

Structure identique à un nom global, seul le prologue change.

Rom Pointer

Prologue : $02D5C_{16}$

Structure :



Rôle inconnu.

A ces 19 objets, il convient d'ajouter :
- les instructions RPL,
- les instructions internes du HP-28C, par exemple 1C285 qui vérifie la présence d'un élément dans la pile, le stocke dans LAST, si LAST est validé, ou affiche `Too Few Arguments` si la pile est vide. La liste de ces instructions ne peut encore être envisagée.
- les lettres de A à Z,
- les nombres de -9 à +9
- diverses constantes.

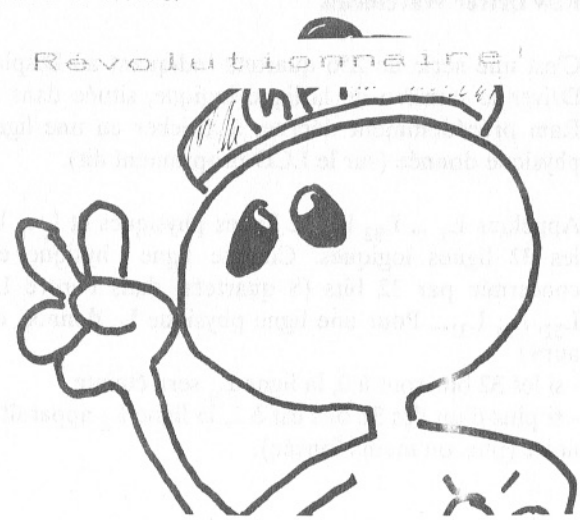
tous codés sur 5 quartets.

Une liste de certaines instructions RPL a été donnée dans JPC 47, en voici d'autres :

```
A : 2789F     U : 279CB     5 : 1E353
B : 278AE     V : 279DA     6 : 1E368
C : 278BD     W : 279E9     7 : 1E37D
D : 278CC     X : 279F8     8 : 1E392
E : 278DB     Y : 27A07     9 : 1E3A7
F : 278EA     Z : 27A16   π : 1E479 (approx)
G : 278F9    -9 : 1E464   π : 1E48E (étendu)
H : 27908    -8 : 1E44F  MAXR : 1E4A8
I : 27917    -7 : 1E43A -MAXR : 1E4BD
J : 27926    -6 : 1E425  MINR : 1E4D2
K : 27935    -5 : 1E410 -MINR : 1E4E7
L : 27944    -4 : 1E3FB     0 : 1E4FC (ext. réel)
M : 27953    -3 : 1E3E6     1 : 1E516 (ext. réel)
N : 27962    -2 : 1E3D1     2 : 1E530 (ext. réel)
O : 27971    -1 : 1E3BC     3 : 1E54A (ext. réel)
P : 27980     0 : 1E2EA     4 : 1E564 (ext. réel)
Q : 2798F     1 : 1E2FF     5 : 1E57E (ext. réel)
R : 2799E     2 : 1E314
S : 279AD     3 : 1E329
T : 279BC     4 : 1E33E
```

Ces objets sont tous considérés comme des instructions et sont en fait représentés par leur adresse d'exécution (utilisable par `SYSEVAL`).

Après ce petit (!) prologue, passons à présent au contenu de la mémoire vive ; la Rom (128 Koctets) étant encore loin d'être connue, il serait illusoire de vouloir la décrire.



ZONE D'ENTREE / SORTIE

407FF	Valeur du Timer	8 quartets
407F8	???	246 quartets
40701	Contraste	1 quartet
	???	33 quartets
406DF	Row Driver Waveforms	256 quartets
405E0		
405DF	Partie droite écran	480 quartets
40400		
	???	288 quartets
402DF	Partie gauche écran	616 quartets
40078		
	???	120 quartets
40000		

L'écran

Chaque pixel est représenté par un bit, soit donc 4 pixels par quartet. L'écran est codé colonne par colonne (les 8 premiers quartets représentent donc la première colonne), du haut vers le bas (le bit 0 du premier quartet représente donc le pixel en haut à gauche).

Row Driver Waveforms

C'est une série de 256 quartets indiquant au Display Driver le numéro de la ligne logique, située dans la Ram précédemment décrite, à afficher en une ligne physique donnée (sur le LCD proprement dit).

Appelons L_1 ... L_{32} les 32 lignes physiques et l_1 ... l_{32} les 32 lignes logiques. Chaque ligne physique est concernée par 32 bits (8 quartets) dans l'ordre L_1, L_{32}, L_2, L_{31}... Pour une ligne physique L_n donnée, on aura :
- si les 32 bits sont à 0, la ligne L_n sera éteinte.
- si plus d'un des 32 bits est à 1, la ligne L_n apparaîtra noire (plus ou moins foncée).
- si le $m^{ème}$ bit et lui seulement est à 1, la ligne l_m sera affichée en L_n.

L'état normal est donc (L_1 et L_{32} seulement) en bits :

10000000000000000000000000000000 L_1

00000000000000000000000000000001 L_{32}

soit donc, en hexadécimal : 10000000 00000008.

Il est à noter qu'un arrêt système après une modification de ces 256 octets remet le Row Driver Waveforms à la valeur standard.

Le contraste

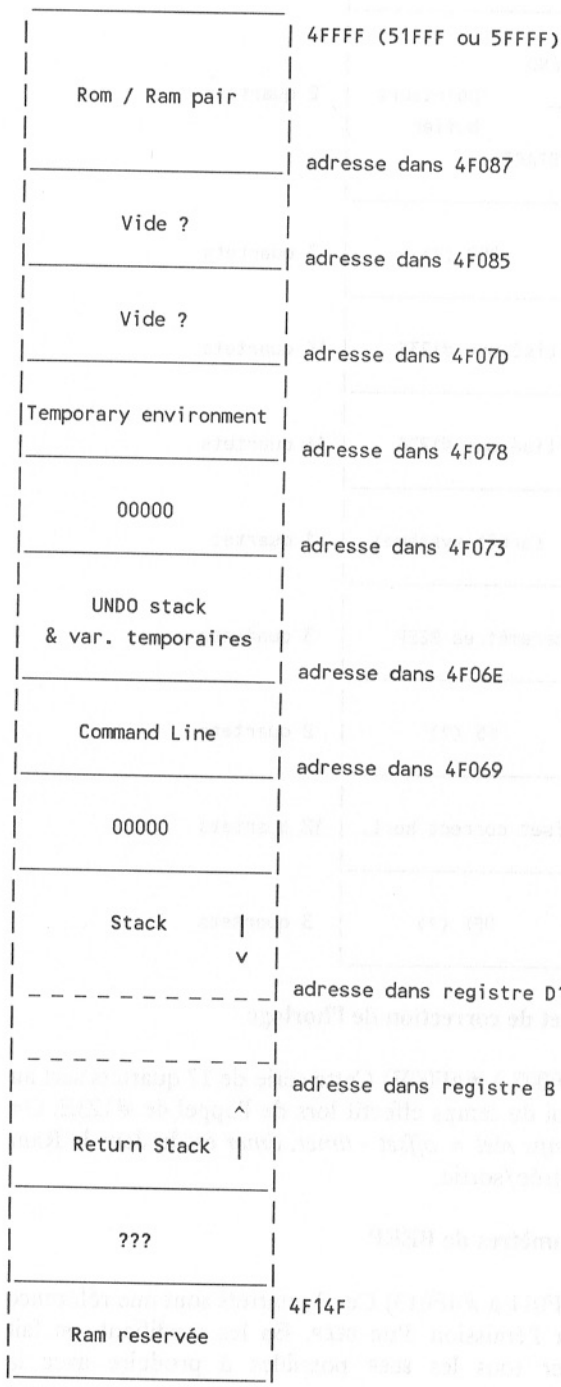
C'est un quartet dont la valeur va de #0, contraste le plus faible, à #F, contraste le plus fort. Il est à noter que les touches [ON][+] et [ON][-] ne permettent d'accéder qu'aux valeurs #4 à #F.

Le timer

C'est un groupe de 8 quartets dont la valeur va décroissant de #077FFF à #000000. La valeur exacte obtenue par #123E SYSEVAL est calculée en faisant la différence entre un offset de correction situé en Ram et la valeur du timer.

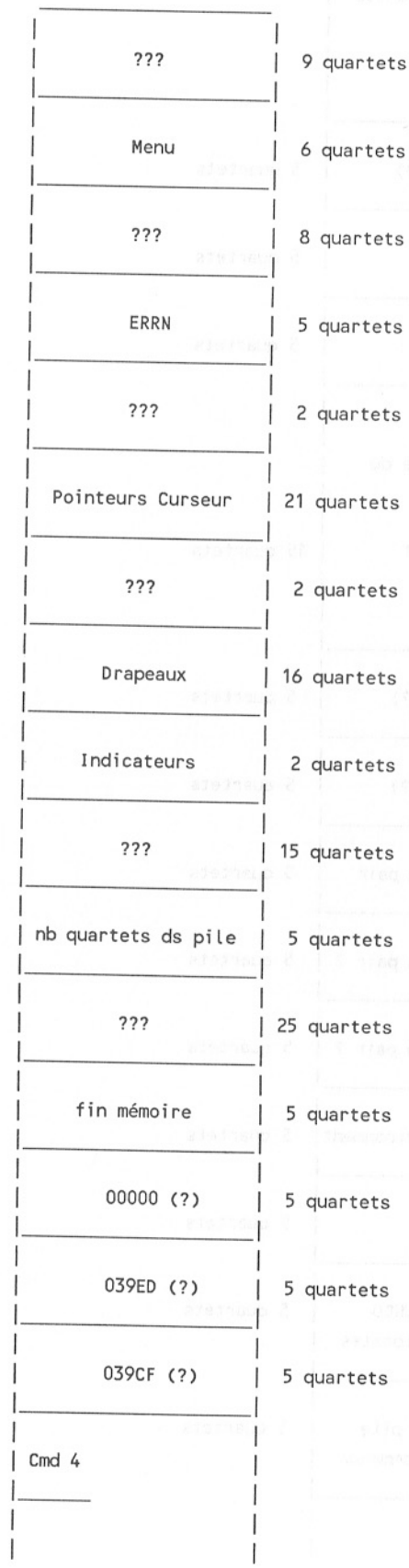
LA MEMOIRE

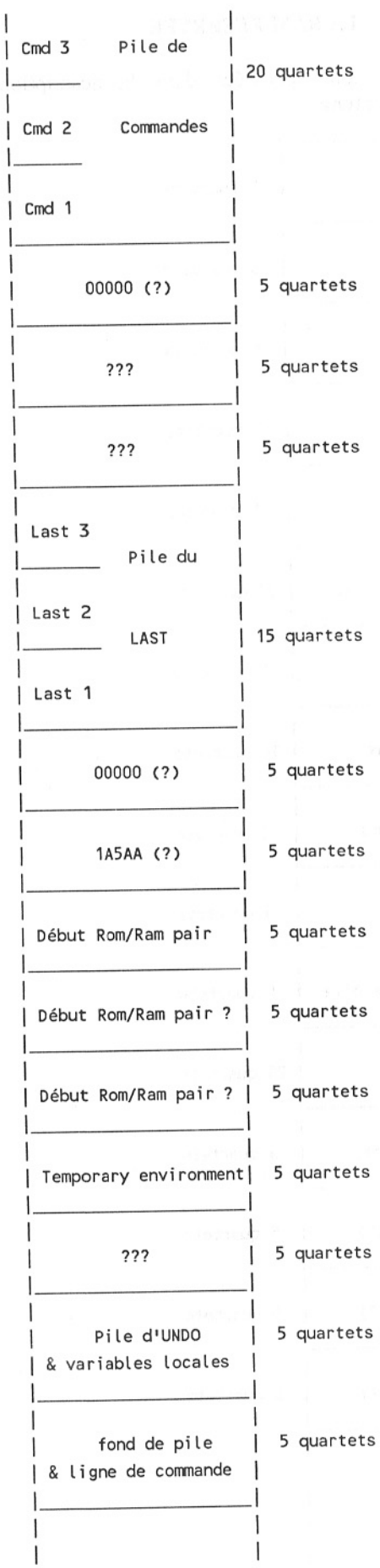
En version standard, elle va de 4F000 à 4FFFF, en version 6 K-octets de 4F000 à 51FFF et en version 34 K-octets de 4F000 à 5FFFF.

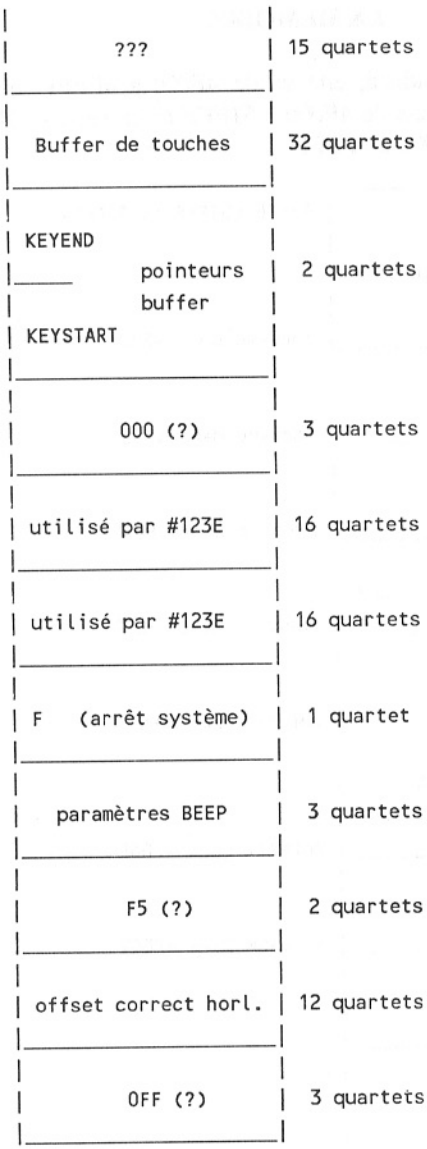


LA RAM RESERVEE

Les adresses sont spécifiées dans la description détaillée du contenu.







Offset de correction de l'horloge

(#4F003 à #4F00E) Cette série de 12 quartets sert au calcul du temps effectif lors de l'appel de #123E. On a *temps réel = offset - timer*, *timer* est lu dans la Ram d'entrée/sortie.

Paramètres de BEEP

(#4F011 à #4F013) Ces 3 quartets sont une référence pour l'émission d'un BEEP. En les modifiant, on fait varier tous les BEEP possibles à produire avec la machine.

Arrêt système et auto-tests

(#4F014) La valeur de ce quartet est normalement #F. En le mettant à 0, l'arrêt système [ON][^] ainsi que les deux auto-tests n'ont plus aucun effet. A noter que l'extinction de la machine réarme cet indicateur et permet à nouveau l'arrêt système.

Stockage temporaire

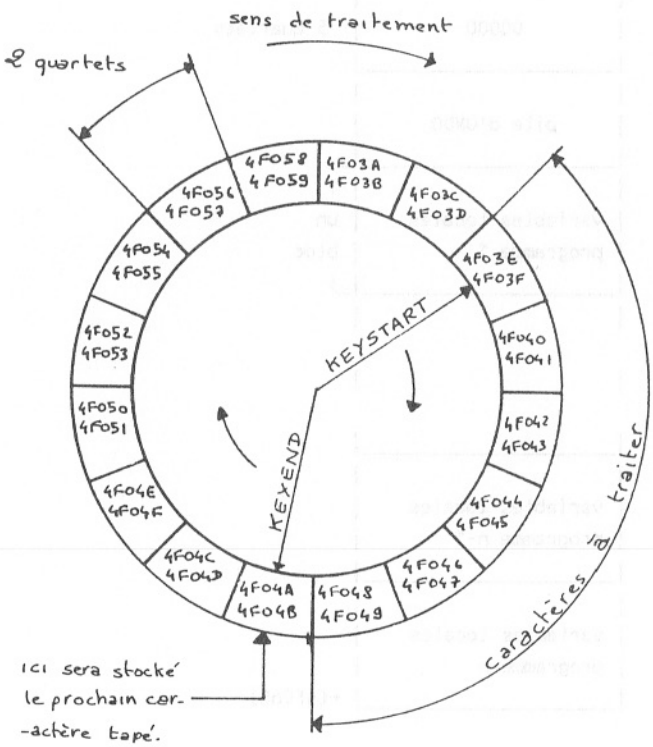
(#4F015 à #4F024 et #4F025 à #4F034). Ces deux zones sont utilisées par #0123E pour le stockage temporaire de certains registres. Elles sont sans doute utilisées de la même façon par d'autres routines.

Le buffer

(#4F038) contient KEYSTART : pointeur de début des touches à traiter.

(#4F039) contient KEYEND : pointeur de fin des touches à traiter.

(#4F03A à #4F059) contient les codes clavier des touches. Les touches restant à traiter vont de #4F03A + 2 × KEYSTART à #4F03A + 2 × KEYEND (non compris). En fait, la meilleure façon de représenter le buffer est de le comparer à une horloge dont les aiguilles seraient KEYSTART et KEYEND :

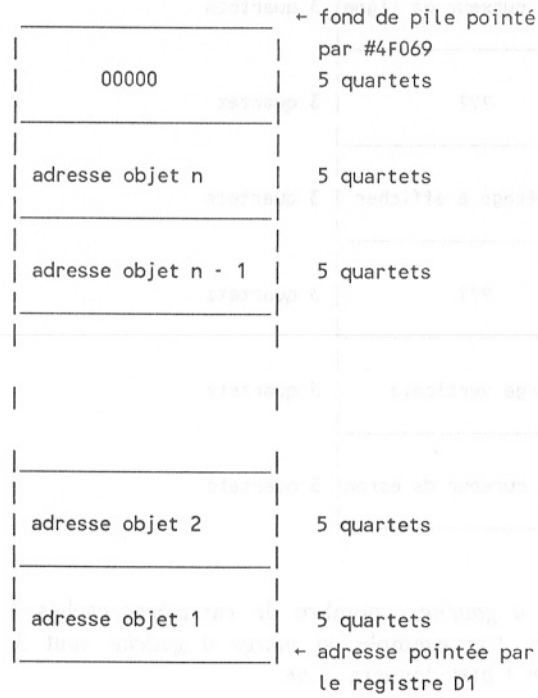


Attention : les codes stockés dans le buffer sont les codes clavier et non les codes ASCII. En voici la liste :

Touche	Code	Touche	Code	Touche	Code
#	18	'	2D	(	14
*	1A	+	02	,	01
-	0E	.	04	/	26
0	07	1	13	2	10
3	0D	4	1F	5	1C
6	19	7	2B	8	28
9	25	=	0B	A	41
B	42	C	48	D	47
E	46	F	44	G	35
H	36	I	3C	J	3B
K	3A	L	38	M	29
N	2A	O	30	P	2F
Q	2E	R	2C	S	1D
T	1E	U	24	V	23
W	22	X	20	Y	11
Z	12	[	16	{	17
«	0C	Space	06	α	08
LC	0A	EVAL	15	STO	21
EEX	31	DROP	32	Effac	33
CHS	34	ENTER	37	SOLV	3D
USER	3E	NEXT	3F	TRIG	40
Curs	43	↓	49	←	4A
→	4B	↑	4C	DEL	4F
INS	51	ON	7F	Shift	80

La pile

Les objets ne sont pas dans la pile, c'est leur adresse (sur 5 quartets) qui est stockée :

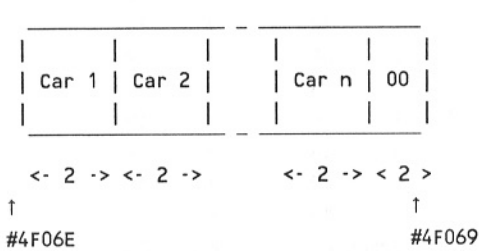


Ainsi pour obtenir l'adresse de l'objet situé au niveau 1 de la pile, suffit il de lire 5 quartets à partir de l'adresse contenue dans le registre D1.

Si, par exemple, le niveau 1 contient "PPC", à l'adresse ainsi lue, on trouvera E4A20B0000050534.

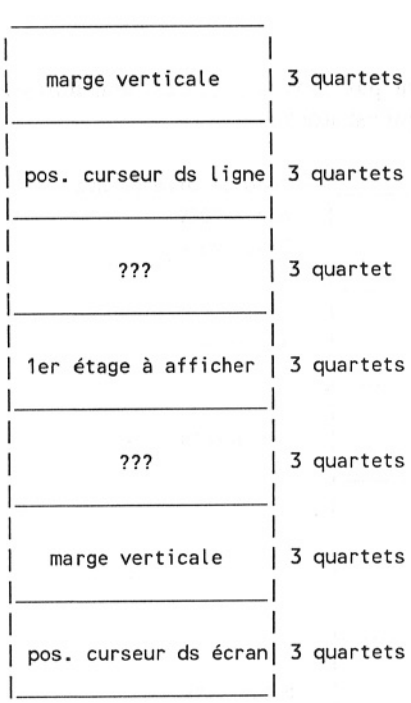
La ligne de commandes

Elle se situe entre l'adresse pointée par #4F069 et celle pointée par #4F06E (non inclus). La structure est la suivante :



Car i est le code ASCII du i^{ème} caractère de la ligne de commande. *n* vaut au minimum 23 si le nombre réel de caractères entrés est inférieur, ils sont completés à 23 par l'adjonction du ou plusieurs caractères de code 00.

Des informations sur le curseur sont stockées de #4F11D à #4F132 :



marge à gauche : nombre de caractères cachés à gauche. Par exemple, si *marge à gauche* vaut 3, `"12345" 1 DISP` donnera `...56`.

position curseur dans la ligne de commande i : position du caractère sous le curseur dans la ligne de commande.

premier étage de la pile à afficher : premier étage visible de la pile .

marge verticale : numéro de la ligne à partir de laquelle il faut afficher. Par exemple, si *marge verticale* vaut 3, on aura :

```
"1
2
3
4" 1 DISP
```

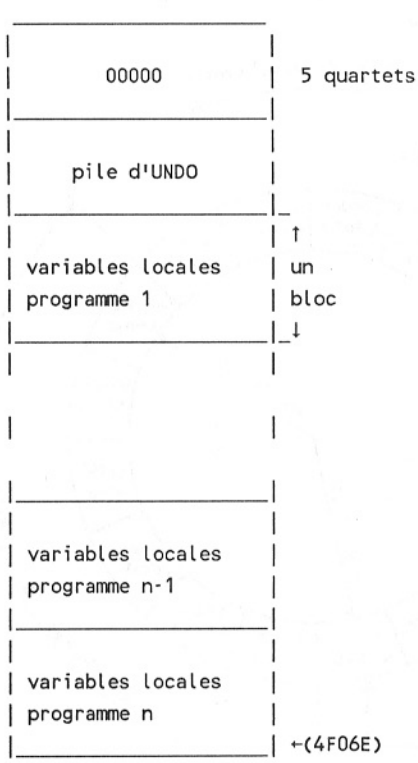
donnera :

```
3
4
```

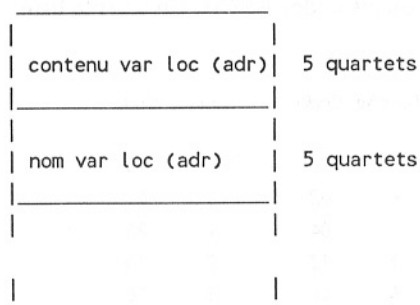
position horizontale dans écran : de 0 à 16_{16}.

La pile d'UNDO et des variables locales

Son adresse se trouve en #4F06E sur 5 quartets. La structure est la suivante :



Chaque bloc ayant la structure :



AH ! VOUS ECRIVEZ

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

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

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

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

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

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

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

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

N'oubliez pas de mettre les accents. Utilisez le jeu de caractères Roman8. Les possesseurs de HP71 utiliseront les redéfinitions de touches ci-dessous, ainsi que le fichier CHARLEX listé dans le coin des Lhex de JPC 50.

Jean-Jacques Dhénin (177)

```
DEF KEY 'fW', CHR$(197);      (é)
DEF KEY 'fE', CHR$(193);      (ê)
DEF KEY 'fR', CHR$(201);      (è)
DEF KEY 'fY', CHR$(203);      (ù)
DEF KEY 'fU', CHR$(195);      (û)
DEF KEY 'fI', CHR$(209);      (î)
DEF KEY 'fO', CHR$(194);      (ô)
DEF KEY 'f/', CHR$(92);       (\)
DEF KEY 'fA', CHR$(192);      (â)
DEF KEY 'fS', CHR$(200);      (à)
DEF KEY 'fD', CHR$(205);      (ë)
DEF KEY 'fJ', CHR$(207);      (ü)
DEF KEY 'fK', CHR$(221);      (ï)
DEF KEY 'f*', CHR$(124);      (|)
DEF KEY 'fC', CHR$(181);      (ç)
```

PPC PARIS SE REUNIT
UNE FOIS PAR MOIS

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

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

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

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

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

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

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

Les dates des prochaines réunions sont :
Samedi 20 février 1988
Samedi 5 mars 1988
Samedi 16 avril 1988
Samedi 7 mai 1988
Samedi 4 juin 1988

Pierre David (37)

NOUS EN AVONS

La coopérative du Club dispose :

- de lecteurs de cartes magnétiques pour HP-71, neufs, dans leur boîte d'origine, avec 5 cartes magnétiques, pour 500 F (port compris),

- des anciens numéros de JPC, au prix de 40 F + 7,40 F de frais d'affranchissement,

- d'une année complète de numéros de JPC (février à décembre / janvier) pour 300 F (offre spéciale) port compris,

- des I.D.S. du module Forth / Assembleur (listing interne commenté par HP) pour 250 F (port compris),

- des VASM pour HP-41 (listings des Roms internes commenté par HP) pour 300 F (port compris).

- de manuels de service du HP-41 au prix de 75 F (port compris).

- de manuels de service du HP-75 au prix de 75 F (port compris).

Si vous souhaitez d'autres renseignements, n'hésitez pas à nous contacter.

VOUS EN VOULEZ

Nom :
Prénom :
No de membre :
Adresse :

Commande :

	Qté		Prix Unitaire		Prix Total
lecteur de cartes pour HP-71		x	500	F	
anciens numéros de JPC		x	47,40	F	
année complète de JPC		x	300	F	
I.D.S. du module Forth		x	250	F	
V.A.S.M.		x	300	F	
Manuel de service pour HP-41		x	75	F	
Manuel de service pour HP-75		x	75	F	
			Total		F

Préciser éventuellement les
numéros de JPC commandés :

PPC PARIS CHAPTER

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

BULLETIN D'ADHESION

Nom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Prénom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| Date de naissance |_|_|/|_|_|/|_|_|

Adresse |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_

Code Postal |_|_|_|_|_| Ville |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Téléphone Domicile |_|_|_|_|_|_|_|_|_|_|_|_|_| Bureau |_|_|_|_|_|_|_|_|_|_|_|_|_|

Profession ______________________________

Intérêts ______________________________

____

Matériel HP en votre possession ______________________________

____

Autre matériel informatique ______________________________

____

Comment avez-vous connu PPC Paris ? ______________________________

____

Que recherchez-vous au sein de PPC Paris ? ______________________________

____

____

La Loi No 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés, garantit à toute personne justifiant de son identité un droit d'accès et de rectification auprès des services ou organismes chargés de mettre en oeuvre des traitements informatiques comportant des informations nominatives le concernant.

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

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

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

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

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

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

contenu var loc (adr)	5 quartets
nom var loc (adr)	5 quartets
longueur du bloc	5 quartets
identificateur	5 quartets

nom variable locale pointe sur un nom vide ('') dans le cas où le bloc est relatif à la pile d'UNDO.

identificateur vaut 00002 pour la pile d'UNDO et 00000 pour un bloc de variables locales.

Dans le cas de la pile d'UNDO, le premier nombre après un nom vide est le nombre d'éléments dans la pile.

Si on effectue des boucles FOR ... NEXT ou START ... NEXT, des variables locales sont créées contenant la valeur du compteur ; son nom est ''noname' dans le cas de la boucle START (impossible à entrer au clavier). Une variable ''stop' contient la valeur de fin de boucle (impossible à entrer au clavier). Une boucle est considérée dans ce cas comme un sous-programme. Au retour d'un sous-programme, les variables locales le concernant sont détruites.

Si UNDO n'est pas activé, le bloc UNDO n'existe pas.

Temporary environment

Adresse en #4F078 à #4F07C. C'est une zone qui renseigne le HP-28C sur le menu à exécuter.

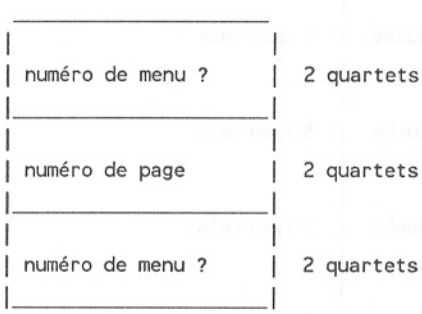
00000	5 quartets
0805A 08371 en mode édition	5 quartets
080BE	5 quartets
adresse 6ème touche	5 quartets
adresse 5ème touche	5 quartets
adresse 4ème touche	5 quartets
adresse 3ème touche	5 quartets
adresse 2ème touche	5 quartets
adresse 1ère touche	5 quartets
039D9	5 quartets
039D9	5 quartets
adresse disp 6	5 quartets
adresse disp 5	5 quartets
adresse disp 4	5 quartets
adresse disp 3	5 quartets
adresse disp 2	5 quartets
adresse disp 1	5 quartets
08C (?)	3 quartets

adresse disp n pointe sur une routine déterminant le nom à placer dans la ligne de commande en mode ALPHA. Si le menu n'est pas le menu USER, le nom en question sera affiché dans le menu.

Pour le menu USER les *adresses disp* sont dans l'ordre :
0E059, 0E07C, 0E0B3, 0E0FE, 0E117, 0E130
et d'exécution :
0DDBB, 0DDD9, 0DDF3, 0DE15, 0DEB3, 0DE51
Ces adresses ont pour rôle :
- pour l'affichage de placer dans la pile le nom du n^{ème} programme (chaîne de caractères).

- pour les secondes de déclencher l'exécution du n^{ème} programme du menu.

Ces routines utilisent certainement les six quartets concernant aussi le menu et situés de #4F141 à #4F146.



Numéro de page est le numéro de l'objet à partir duquel il faut commencer le menu (0,6,C,...)
Numéro de menu semble être le numéro de menu à afficher. Voici la table relevée :

```
Array  : 12  Real    : 0D  Ctrl   : 0E  User : 01
Binary : 0B  Stack   : 08  Branch : 03  Mode : 06
Cmplx  : 13  Store   : 10  Test   : 0F  Logs : 04
String : 14  Algebra : 09  Trig   : 07  Stat : 05
List   : 0C  Print   : 11  Solv   : 0A  Plot : 15
                           Solvr  : 02 (Sous menu)
```

A laquelle on peut ajouter :

```
16 : Menu FORM1  (Colct Expan Level Exget [←] [→])
17 : Menu FORM2  (DNEG DINV *1 /1 ^1 +1 -1)
18 : Menu FORM3  (-() ← → ←M M→ ←A A→)
19 : Menu FORM4  (AF)
1A : Menu FORM5  (1/() ← → ←D D→ ←A A→)
1B : Menu FORM6  (-() L() ←M M→)
1C : Menu FORM7  (1/() E() ←D D→ ←A A→)
1D : Menu FORM8  (1/() E^ D→)
1E : Menu FORM9  (-() L* D→)
1F : Menu FORM10 (→())
00 : pas de menu
```

Rom/Ram pair

Adresse en #4F087 à #4F08B. Elle a déjà été étudiée précédemment (voir la rubrique sur les différents types d'objets).

La pile du LAST

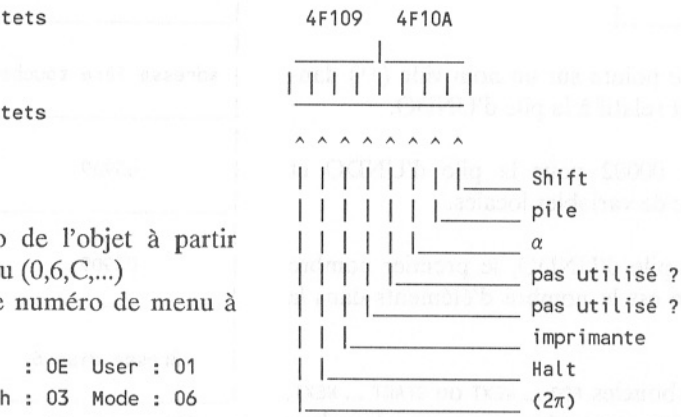
Adresse de #4F096 à 4F0A4. Elle contient les adresses de 1 à 3 objets (si il y a moins de 3 objets les adresses des objets non présents seront 00000). Ce sont les adresses des objets pris par la dernière fonction utilisée. Si LAST est invalidé, les 3 adresses seront 00000.

Pile de commandes

Adresse de #4F0B4 à #4F0C7. C'est une liste de 4 adresses, éventuellement à 00000, pointant sur des chaînes de caractères contenant les caractères d'une ligne de commande.

Stockage des valeurs des indicateurs

2 quartets de #4F109 à #4F10A.



Attention : changer ces valeurs ne conduit pas à un effet immédiat lors de l'exécution d'un programme : c'est seulement à la fin de ce programme que les indicateurs seront réactualisés en fonction de ces valeurs.

Les drapeaux sont de #4F10B à #4F11A. Ce sont 64 bits, chaque bit représentant un drapeau (Flag 1 : Bit 0 de #4F10B, Flag 64 : Bit 3 de #4F11A).

Adresses diverses

En #4F0D7 est inscrite l'adresse de fin de mémoire (#50000 en standard).

En #4F0F5 est inscrit le nombre de quartets dans la pile, c'est à dire 5 fois le nombre d'éléments dans la pile (5×DEPTH).

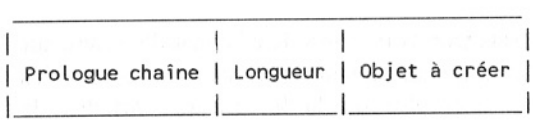
En #4F134 à #4F138 se trouve le numéro de la dernière erreur commise.

PROGRAMMATION EN ASSEMBLEUR

Après ce petit préliminaire, voici la deuxième partie de cet article : comment programmer en langage machine sur le HP-28C. Tout d'abord voici l'explication des programmes ASS et →LEX parus dans JPC numéro 50.

Le but recherché était de permettre la création de n'importe quel objet. L'idée retenue a finalement été un processus en deux étapes :

1- Mise en place de la suite de quartets constituant l'objet à créer sous forme de chaîne de manière à obtenir :



2- Cet objet étant placé dans la pile, la seconde étape consiste à décaler l'adresse contenue dans la pile de manière à ce que l'objet pointé soit l'objet créé et non plus la chaîne.

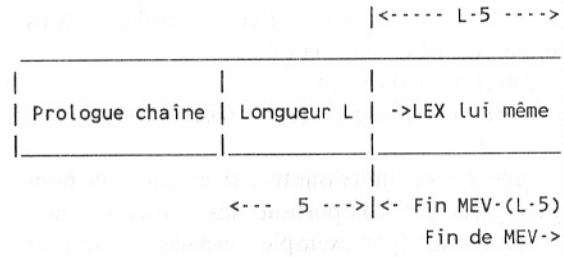
La première étape est réalisée par le programme ASS. La liste des codes hexadécimaux étant placée dans une chaîne, le programme prend 2 codes consécutifs et crée le CHR correspondant de manière à ce que les codes présents dans la Ram soient ceux désirés. Comme le microprocesseur l'attend, les codes ASCII sont stockés dans l'ordre (quartet de poids faible, quartets de poids fort), il faut inverser les 2 chiffres hexadécimaux avant de faire le CHR. Voici le listing de ASS :

```
« → LM                  On place la chaîne contenant les
                        codes hexadécimaux dans la
                        variable temporaire LM.
  « HEX                 Passage en mode hexadécimal.
    ""                  Chaîne vide : y seront ajoutés les
                        CHR représentant les codes.
   1'LM SIZE FOR X      Boucle pour prendre tous les
                        codes hexadécimaux.
    "#"                 préparation de la conversion de
                        deux nombres hexa sous forme de
                        chaîne en un nombre binaire.
     LM X DUP2          On prend deux chiffres hexa
     1 + DUP            et on les inverse. Exemple :
     SUB 3 ROLLD        si on entre "1234" pour X=1
     DUP SUB +          on obtient "21"
     + STR→             On ajoute "#" et on convertit en
                        un nombre binaire (ex "21" -> #21)
     B→R CHR            On prend le CHR correspondant,
                        dans l'exemple CHR(33) ; en
                        mémoire seront stockés, et
                        dans cet ordre, les codes 1 et 2
     +                  On ajoute CHR à la chaîne résultat
                        ("" au départ).
   2 STEP               Couple de caractères suivant.
» »
```

Le second programme, c'est →LEX, qui à partir de la chaîne contenant l'objet place l'objet dans la pile : ce programme ne pouvait être réalisé qu'en langage machine... Cycle infernal me direz vous ! Heureusement non. Là encore, on procède en plusieurs étapes :

- On transforme le listing des codes en un code exécutable grâce à ASS.

- On stocke →LEX (sous forme de chaîne) en début de menu USER. On connaît alors exactement l'endroit où il se trouve :



Ainsi en faisant (Fin de MEV-L+5) SYSEVAL, on pourra démarrer →LEX. Comme →LEX prend une chaîne dans la pile et retourne l'objet qu'elle contient, on place →LEX dans la pile avant de faire ce SYSEVAL. A ce stade, on a le vrai programme →LEX dans la pile : il ne reste plus qu'à le stocker dans →LEX ; on perd la chaîne qui n'est plus d'aucune utilité. Voici le listing de →LEX :

```
Code      Mnémonique       Commentaire
-------------------------------------------------

76C20     CON(5) #02C67    Objet : programme
5B3C1     CON(5) #1C3B5    Routine vérifiant la
                           présence d'au moins un
                           élément dans la pile.
69C20     CON(5) #02C96    Objet : Assembly code
      deb
E1000     CON(5) fin-deb   Longueur du Prgm
143       A=DAT1 A         A=^ Objet niv 1
133       AD1EX
179       D1=D1+ 10
133       AD1EX            A=^chaîne+10 =^Objet créé
141       DAT1=A A         Ecriture nouvelle adresse
                           dans la pile
142       A=DAT0 A         fin de programme :
164       D0=D0+ 5         cf explications plus bas
808C      PC=(A)
09F20     CON(5) #02F90    fin de l'objet programme
      fin
```

Il faut savoir que :
- D1 est l'adresse du sommet de la pile
- D0 est le pointeur RPL : à la fin de chaque programme assembleur, il faut toujours faire :

```
A=DATO A
D0=D0+ 5
PC=(A)
```

C'est en fait ceci qui permet l'exécution de l'objet suivant (ici 02F90, fin de structure). Une fois les codes entrés puis assemblés grâce à ASS, on procède comme expliqué plus haut. L'adresse du SYSEVAL à effectuer est :
#4FFCE (#50000 - #32) en version standard
#51FCE (#52000 - #32) en version 6 Ko
#5FFCE (#60000 - #32) en version 34 Ko
où #32 (50) est la longueur du programme →LEX.

Dorénavant pour créer un objet dans la pile aucun SYSEVAL n'est plus à faire. Par exemple, créons artificiellement "PPC" dans la pile :
Codes : E4A20B0000050534
Pour le créer : `"E4A20B0000050534" [ASS] [→LEX]`

A noter une chose intéressante : si on crée un nom (prologue : 02D12) comportant des caractères non acceptés au clavier (par exemple : espace, [,] ou tout autre caractère), il sera accepté pour un STO (la vérification de la bonne syntaxe d'un nom se fait à l'entrée et non lors du STO. Par exemple on peut créer un programme nommé 'A B C', entrez : `"21D20501402240234" [ASS] [→LEX].`

Il y aura alors 'A B C' dans la pile et le STO sera possible. De cette manière, on peut créer des programmes portant le même nom qu'une instruction RPL et ce programme sera prioritaire sur l'instruction mais seulement s'il est lancé du clavier et non exécuté par l'intermédaire du menu contenant l'instruction. Ainsi SYSEVAL et CLUSR peuvent être facilement recréées... Par exemple pour CLUSR :

```
<< "ET LE CODE ??? " 1 DISP 1000 .1 BEEP >>
"21D205034C4553525" [ASS] [→LEX] [STO]
```

Dorénavant, toute tentative pour faire un CLUSR ne pourra aboutir. Il est intéressant de remarquer qu'une fois le programme CLUSR créé, il est possible d'entrer directement le nom CLUSR sans passer par ASS. Pour revenir à un état normal : `'CLUSR' [PURGE].`

Dernier point : comment concevoir un Lex pour le HP-28C.

Tout dépend si vous désirez mixer des instructions RPL à de l'assembleur ou faire un programme assembleur seul. Dans le second cas, créer le programme assembleur en commençant par le prologue *Assembly code*. Dans le premier cas, réaliser une structure composite (prologue de programme) contenant les instructions RPL et un ou plusieurs Assembly codes.

Dans tout code assembleur :
- Sauvegarder les registres D0, D1, B et D si besoin est
- Terminer par la récupération éventuelle des dits registres puis par :

```
A=DATO
D0=D0+ 5
PC=(A)
```

Dans le cas de la structure composite programme, ne pas oublier le 02F90 terminal.

Je ne saurais trop vous conseiller l'achat d'un livre sur le HP-71B contenant les instructions avec leurs codes (HDS par exemple) ou la lecture des articles de Jacques Baudier dans JPC 46 page 25 et JPC 47 page 26.

A présent, c'est à vous ! Et envoyez vos réalisations au Journal...

En remerciant Pierre David et Janick Taillandier pour leur aide et leur patience, Graeme Cawsey pour sa liste des différents prologues et Sébastien Lalande pour son aide lors de l'analyse du Row Driver Waveform.

Paul Courbis (392)

NDLR : Il est bien entendu inutile de contacter HP à propos des informations contenues dans cet article. Il faut également noter que toutes les adresses données sont relatives au HP-28C version 1BB, elles peuvent être différentes sur les versions ultérieures.

PEEK ET POKE POUR LE HP-28

Après le PEEK de Wlodek Mier-Jedrzejowicz publié dans JPC 47 qui ne permettait que peu de facilités, il devenait nécessaire de créer un PEEK plus facile à utiliser. Grâce aux découvertes réalisées sur la structure des objets dans le HP-28C (voir l'article précédent) il est devenu possible de créer un PEEK fiable et commode.

Syntaxe :
Niveau 2 : adresse du PEEK
Niveau 1 : nombre de quartets à lire.

Ces deux nombres doivent être des entiers binaires. Attention : aucune vérification n'est faite sur le type des objets pour rendre le programme le plus court possible.

En sortie : une chaîne de caractères contenant les caractères 0, 1,..., 9, A,..., E, F représentant le contenu des quartets (nombres hexadécimaux).

Pour entrer le programme, entrez d'abord ASS et →LEX, si ce n'est pas déjà fait. Ensuite, entrez sous la forme d'une chaîne hexadécimale les nombres contenus dans la colonne *Code* du listing, en une seule ligne, sans espace, puis exécutez ASS et →LEX (voir article précédent et JPC 50 page 8). Stockez le résultat dans 'PEEK'. Ce programme est alors prêt à servir.

```
Code Label Mnemonique       Commentaires

76C20      CON(5) #02C67    Début de programme
5A3C1      CON(5) #1C3A5    Vérifie la présence
                            d'au moins deux
                            éléments sur la pile.
07A20      CON(5) #02A70    .
60000      CON(5) #00006    . #0
0          CON(1) #0        .
BF543      CON(5) #345FB    additionne 2 entiers
                            hexadécimaux on recrée
                            un nouveau nombre hexa
                            en mémoire.
5F020      CON(5) #020F5    SWAP
07A20      CON(5) #02A70    .
60000      CON(5) #00006    . #0
0          CON(1) #0        .
BF543      CON(5) #345FB    +
5F020      CON(5) #020F5    SWAP
E4A20      CON(5) #02A4E    "" amorce de la chaîne
50000      CON(5) #00005    qui contiendra le
                            résultat du PEEK
FB920 L0   CON(5) #029BF    Byte 00 (sera utilisé
00         CON(2) #00       pour créer un CHR)
69C20      CON(5) #02C96    Assembly code
C6000 deb1 CON(5) fin1-deb1 Longueur du code
132        ADOEX            R2 := D0
102        R2=A
133        AD1EX            R3 := D1
103        R3=A
133        AD1EX
17E        D1=D1+ 15        D1 = ^ adresse
143        A=DAT1 A
133        AD1EX
179        D1=D1+ 10
143        A=DAT1 A
132        ADOEX            D0 = ^ du PEEK
113        A=R3
133        AD1EX
143        A=DAT1 A
133        AD1EX            D1=^ BYTE (objet)
174        D1=D1+ 5         D1=^ BYTE (où écrire)
AE0        A=0    A
15A0       A=DAT0 1         On lit un quartet
3103       LCHEX  #30       .
A6A        A=A+C  B         . Contenu du
3193       LCHEX  #39       . quartet ->
9EA        ?A<=C  B         . code-caractère
90         GOYES  L11       . ("1","2"...,"F")
3170       LCHEX  #07       .
A6A        A=A+C  B         .
149   L11  DAT1=A B         code dans octet
113        A=R3             On récupère D1
133        AD1EX
112        A=R2             On récupère D0
132        ADOEX
142        A=DAT0 A         .
164        D0=D0+ 5         . fin de routine
808C       PC=(A)           .
FC630 fin1 CON(5) #036CF    Cette routine prend une
                            chaîne + un BYTE dans
                            la pile et retourne
                            Chaîne + CHR (BYTE)
69C20      CON(5) #02C96    Assembly code
E6000 deb2 CON(5) fin2-deb2 Longueur
133        AD1EX            Sauvegarde de D1
103        R3=A             dans R3
133        AD1EX
179        D1=D1+ 10
143        A=DAT1 A
133        AD1EX
179        D1=D1+ 10        D1=^ où est l'adresse
143        A=DAT1 A         du PEEK
133        AD1EX            D1= Adresse du PEEK
170        D1=D1+ 1         On l'incrémente
133        AD1EX
141        DAT1=A A         et on l'écrit
113        A=R3
133        AD1EX
174        D1=D1+ 5
143        A=DAT1 A
133        AD1EX
179        D1=D1+ 10        D1=^ où est la longueur
143        A=DAT1 A         A= longueur PEEK
133        AD1EX
1C0        D1=D1- 1         On décrémente longueur
133        AD1EX
141        DAT1=A A
8A8        ?A=0   A         Longueur nulle -> fini
11         GOYES  L21
132        ADOEX            On décrémente D0 pour
340F000    LCHEX  #000F0    continuer en L0
EA         A=A-C  A         lors du prochain
132        ADOEX            A=DAT0 D0=D0+ 5 PC=(A)
113   L21  A=R3
```

```
133          AD1EX              On récupère D1
142          A=DAT0 A           .
164          D0=D0+ 5           . fin de routine
808C         PC=(A)             .
69C20        CON(5) #02C96
72000 deb3   CON(5) fin3-deb3
133          AD1EX              . dans la pile
103          R3=A               . 3 = #adr
133          AD1EX              . 2 = #len
143          A=DAT1 A           . 1 = "résultat"
179          D1=D1+ 10          . devient
141          DAT1=A A           . 3 = "résultat"
113          A=R3               . 2 = #len
133          AD1EX              . 1 = "résultat"
142          A=DAT0 A
164          D0=D0+ 5           . après le DROP2
808C         PC=(A)             . 1 = "résultat"
A2120 fin3   CON(5) #0212A      DROP2
09F20        CON(5) #02F90      fin de structure
```

Après le PEEK voici le POKE :

Syntaxe :
Niveau 2 : adresse (entier binaire)
Niveau 1 : chaîne contenant les codes.

Exemple : #40078 "FFFFFFFF" POKE place 15 (#F en hexadécimal) dans les 8 quartets #40078, #40079,...,#4007F.

Entrée du programme : taper les nombres de la première colonne sous forme d'une chaîne, en une seule ligne, sans espace puis [ASS] et [→LEX].

```
Code Label Mnemonique      Commentaires

76C20      CON(5) #02C67   Structure programme
5A3C1      CON(5) #1C3A5   2 éléments dans la pile
69C20      CON(5) #02C96   Assembly code
C9000 deb  CON(5) fin-deb  Longueur code
132        ADOEX
103        R3=A            On sauve D0
AFC        ABEX   W
102        R2=A            On sauve B
133        AD1EX
101        R1=A            On sauve D1
133        AD1EX
174        D1=D1+ 5        D1 pointe sur #adresse
                           dans la pile
143        A=DAT1 A
133        AD1EX
179        D1=D1+ 10
143        A=DAT1 A
132        ADOEX           D0= ^ du POKE
111        A=R1
133        AD1EX
143        A=DAT1 A
133        AD1EX
174        D1=D1+ 5
143        A=DAT1 A
3450000    LCHEX  #00005
DC         ABEX   A        B= longueur + 5
174        D1=D1+ 5        D1= ^ début chaîne
E1    L1   B=B-C  A
8A9        ?B=0   A        POKE fini ?
13         GOYES  L3
14B        A=DAT1 B
3103       LCHEX  #30      .
B6A        A=A-C  B        . code ASCII
3190       LCHEX  #09      . -> de 0 à F
9EA        ?A<=C  B        .
90         GOYES  L2       .
3170       LCHEX  #07      .
B6A        A=A-C  B        .
1580  L2   DAT0=A 1        On fait le POKE
160        D0=D0+ 1        . caractère suivant
171        D1=D1+ 2        .
3420000    LCHEX  #00002
6DCF       GOTO   L1
112   L3   A=R2
AFC        ABEX   W        On récupère B
113        A=R3
132        ADOEX           On récupère D0
111        A=R1
133        AD1EX           On récupère D1
142        A=DAT0 A        .
164        D0=D0+ 5        . fin de routine
808C       PC=(A)          .
A2120      CON(5) #0212A   DROP2
09F20      CON(5) #02F90   fin de structure
```

Remarque : le programme sauve les éléments présents dans la pile avec la routine #1C3A5. Par la suite, un appel est fait à la routine #0212A qui est un DROP2 différent du standard, parce qu'il ne sauvegarde pas dans LAST. Ainsi, la modification faite par PEEK dans la pile n'est pas répercutée dans LAST.

Paul Courbis (392)

HP41

M. Markov Utilitaires de gestion de disques (acte III) *28*

UTILITAIRES DE GESTION DE DISQUES (ACTE III)

Les fonctions WA et RA étendent les possibilités de WRTA et READA. Elles vous permettent de sauvegarder toute la mémoire du HP-41, y compris les mémoires étendues, sur un support de masse. Les informations sont stockées dans un gros fichier de 1360 registres de type WA,A. Moins de 1360 registres seront utilisés si votre HP-41C n'est pas équipé d'un module Quad.

Ces programmes permettent à l'utilisateur de tirer pleinement profit des mémoires étendues en fournissant un moyen simple et rapide de restaurer tous les fichiers, après un MEMORY LOST, en une seule opération. Le temps économisé en utilisant WA et RA est d'autant plus important que le nombre de fichiers est élevé.

Ce programme vous permet d'utiliser la mémoire étendue comme une unité de stockage de masse portable et très fiable en fournissant un moyen de sauvegarde commode.

HISTORIQUE

Je crois que les routines WA et RA ont été initialement publiées dans le journal du club de Melbourne *Technical Notes* (numéro 17 page 77). L'auteur était Michael Pocksteiner et le listing était daté du 25 novembre 1983. Cette première version de 108 octets ne requérait pas de module Rom.

Vers la fin de 1984, Gerhard Kruse m'a envoyé une version très courte de WA et RA qui n'utilisait pas la programmation synthétique mais les fonctions du module *CCD ROM* non encore disponible à l'époque.

J'ai réécrit le programme en utilisant le minimum de fonctions synthétiques et les fonctions de traitement de chaînes de caractères disponibles dans le module *Extended I/O*. Vous pouvez utiliser les fonctions équivalentes du module *HP-IL Development*. Cette version comprend des sécurités qui minimisent les risques de l'utilisateur dans la mesure où ce programme manipule le contenu du registre d'état c.

Les 1360 registres des fichiers WA,A utilisent 43 secteurs du support soit 9 % d'une cassette ou 2 % d'une disquette. L'espace en question n'est pas immédiatement réutilisable si vous purgez le fichier. Certains contrôleurs permettent de compacter le support, mais ceci doit être évité en raison de l'usure du support et de l'appareil engendrée par l'opération. Vous avez donc intérêt à contrôler le nombre de ces fichiers soigneusement. De plus, comme l'espace correspondant n'est pas récupéré, il n'est pas utile de purger ces fichiers à moins que vous ne les remplaciez par des fichiers de données de plus de 1376 registres.

En général, un ou deux fichiers WA,A devraient suffir à contenir votre environnement de travail : les deux fichiers étant protégés en écriture pour éviter un effacement accidentel. Un autre fichier, non protégé, est utile quand vous developpez de nouveaux programmes.

Environ 30 % du fichier ne contient pas d'information ou de l'information redondante. C'est le prix à payer pour avoir un temps de rechargement court. Si vous souhaitez optimiser l'espace, utilisez les fonctions standard.

MODE D'EMPLOI

Vous pouvez utiliser WA et RA exactement comme vous utilisez WRTA ou READA à partir du moment où le programme est en mémoire principale. Vous devez vous assurez que le modules HP-IL et *Extended I/O* sont bien présents, que la boucle HP-IL est convenablement branchée et que le nom du fichier est placé dans le registre ALPHA avant d'exécuter WA ou RA. De plus, si vous utilisez RA assurez vous que la configuration mémoire est bien la même que celle existante au moment du WA.

WA et RA doivent transférer à peu près quatre fois plus de données que READA et WRTA une fois la recherche dans le catalogue effectuée. Ces fonctions dureront donc environ quatre fois plus longtemps.

Vérifiez également que votre lecteur a une batterie suffisament chargée avant d'exécuter WA et RA : c'est une bonne idée d'avoir le chargeur branché. Enfin, évitez d'être en mode TRACE.

Attention ! N'interrompez pas WA et RA. Ces fonctions modifient temporairement le registre d'état c. Une interruption conduit de manière certaine au MEMORY LOST, à moins que vous ne restauriez le rideau à sa valeur d'origine.

Les erreurs HP-IL telles que NO DRIVE, NO MEDIUM, FILE NOT FOUND, NONEXISTENT (pas de module HP-IL), BAD MEDIUM, NO ROOM (mauvaise configuration mémoire avec RA) terminent le programme et provoquent un BEEP. Le programme restaure automatiquement le registre c. Ceci permet à l'utilisateur de corriger le problème et de recommencer.

Si vous interrompez ce programme alors que le témoin BUSY du lecteur est allumé, exécutez d'abord un CLRLOOP ; toutefois, vous risquez de devoir reformatter le support. Ensuite, XEQ 00 restaurera le contenu du registre c. Si vous avez de la chance vous pourrez éviter le MEMORY LOST.

Ce qui suit ne s'adresse qu'à ceux qui ont recours à la programmation synthétique pour déplacer le rideau (adresse du registre 00). Celui-ci doit être en mémoire principale quand vous exécutez WA. Si le rideau se trouve sur un des registres d'état ou en mémoire étendue, il en résultera sans doute une perte de place sur le support. Toutes sortes d'autres problèmes peuvent également survenir qui conduisent plus ou moins sûrement au MEMORY LOST.

AMELIORATIONS DE WA

WA peut être amélioré pour permettre l'utilisation de tout l'espace de stockage d'un disque (voir première partie de cet article dans JPC 48 page 8). Vous pouvez, bien sur, créer un fichier de données de 1360 registres et le purger. Cette procédure a un inconvénient : le temps passé pour remettre à zero le fichier.

La méthode la plus simple pour améliorer WA est la suivante :
- mettre en place un point d'entrée global après la ligne 71 du programme WRTP / CREATE ; un LBL "01" (label alphabétique et non numérique) est un bon choix.
- utiliser WA2 au lieu de WA (voir listings).

WA2 n'a pas toutes les sécurités inclues dans WA. La raison en est que WA2 ne peut faire la différence entre des erreurs système HP-IL et l'erreur invalide MEDM FULL. Ceci implique que vous devez faire d'autant plus attention à ne pas provoquer d'erreurs système. Si une vraie erreur survient, le programme au LBL 01 ne peut créer une entrée fictive, vous obtiendrez alors un message tel que TRANSMIT ERR. Si ceci survient, restaurez le registre c immédiatement en effectuant GTO "WA2" et XEQ 00.

UN PEU DE THEORIE

La fonction WA du module HP-IL doit fonctionner avec toutes les HP-41 ayant 64, 128 192, 256 ou 330 registres de mémoire principale en fonction du nombre de modules mémoire enfichés. Par conséquent, WRTA appelle le sous programme assembleur FNDEND à l'adresse 1730_{16} pour déterminer l'adresse du dernier registre à sauvegarder.

FNDEND calcule cette adresse en vérifiant l'existence de chaque registre en partant de l'adresse donnée pour R00. Cette routine accepte n'importe quelle adresse de rideau donnée dans le registre c et continue jusqu'a ce qu'un vide soit trouvé comme, par exemple, à l'adresse 200_{16}.

Comme la zone contenant l'adresse de R00 fait 12 bits, il est théoriquement possible d'avoir l'adresse du rideau en FFF_{16}, ou 4095, même si la mémoire principale se termine en $1FF_{16}$. Ainsi, en manipulant le contenu du registre c, on peut contraindre WA à faire de très grands fichiers WA.

L'espace mémoire va, en fait, de 000_{16} à $3FF_{16}$, soit 1024 registres. Si vous essayez de lire ou d'écrire dans un registre à une adresse plus grande que $3FF_{16}$, l'adresse réelle sera modulo 1024. Ainsi l'adresse 400_{16} est la même que 000_{16} et $5FF_{16}$ est la même que $1FF_{16}$. Cette propriété permet de copier toute la mémoire du HP-41.

Notez que le plus petit fichier WA,A que l'on peut utiliser pour sauvegarder la mémoire du module *Extended Functions* sur une HP-41CV / CX fait 1360 registres. 418 sont donc vides (pas de mémoire à cette adresse) ou contiennent de l'information redondante. En d'autres termes, 13 secteurs sur un total de 43, soit 30 % de la capacité du fichier, sont mal employés.

Un des points clef de ce programme est l'exécution automatique : quand vous exécutez RA, vous transférez le contrôle à la copie de RA que vous avez stockée au WA. Le programme utilise cette caractéristique pour restaurer le rideau sans intervention de l'utilisateur.

L'exécution automatique d'autres programmes devrait être possible car WRTA fait une copie exacte de la mémoire, y compris la pile d'adresses de retour. Le programme à exécuter devrait donc appeler WA comme sous programme. Ensuite, quand vous exécutez RA, vous devez retourner au programme appelant après avoir restauré le contenu du registre c. Tout programme utilisant cette possibilité doit être conçu pour éviter une activation inopinée de WA. Ceci se fait très bien par un GTO, par exemple :

```
... GTO 00, XEQ "WA", LBL 00, ....
```

Si vous voulez stocker le fichier, utilisez GTO. nnn pour vous positionner sur l'instruction XEQ "WA", puis appuyez sur [R/S].

ANALYSE DETAILLEE

Le sous-programme au LBL 00 déplace le rideau. Il est conçu pour utiliser un seul bit de l'adresse du registre R00 contenue dans c. En pratique, il augmente ou diminue l'adresse de 1024 registres en fonction des besoins. Le registre ALPHA est utilisé puis restauré. La pile est utilisée et non restaurée.

L'instruction X<=Y? de la ligne 21 détermine si l'adresse du registre R00 doit être incrémentée ou décrémentée. Les utilisateurs « avancés » doivent avoir conscience que le test d'égalité n'est utilisé que lorsque l'adresse initiale du rideau est stockée dans les registres d'état. Vous pouvez donc préférer X<Y? qui est légerement plus rapide.

Le reste de l'utilitaire, lignes 1 à 14, utilise un système complexe de flags destiné à minimiser le temps d'exécution, à mettre en place la fonction d'auto-exécution et à fournir une récupération d'erreurs HP-IL en douceur.

Quand vous exécutez RA, le flag 11 est effacé. Cependant, après avoir exécuté READA à la ligne 12, vous reprenez en ligne 11 avec le flag 11 levé, comme laissé par WA. Ainsi, vous n'entrez pas dans une boucle infinie. Le flag 11 est effacé et le programme continue en ligne 13 en restaurant le rideau.

Michael Markov (301)

Traduction : Janick Taillandier (246)

Programmes WA et RA :

```
01*LBL "WA"
SF 11  GTO 01

04*LBL "RA"
CF 11

06*LBL 01
XEQ 00  SF 25  FS? 11  WRTA  FC?C 11  READA
FC?C 25  BEEP

15*LBL 00
RCL c  X<> M  -3  ATOXX  64  X<=Y?  CHS  +  -3
YTOAX  R^  R^  X<> M  STO c  END
```

Programmes WA2 et RA :

```
01*LBL "WA2"
SF 11  GTO 01

04*LBL "RA"
CF 11

06*LBL 01
XEQ 00  FS? 11  XEQ 02  FC?C 11  READA

12*LBL 00
RCL c  X<> M  -3  ATOXX  64  X<=Y?  CHS  +  -3
YTOAX  R^  R^  X<> M  STO c  RTN

28*LBL 02
SF 25  WRTA  FS?C 25  RTN  43  XEQ "01"  WRTA  END
```

HP75

J.Y. Hervé KEYLEX pour HP-75 *32*

KEYLEX

Ce fichier Lex de 195 octets fournit six nouvelles fonctions programmables développant l'utilisation du clavier du HP-75. Il rend possibles de nombreuses interactions avec la machine, et facilite en particulier les branchements par menus.

UNLOCK

Cette fonction met le clavier dans sa configuration normale (caractères en minuscules).

SHLOCK

Cette fonction bloque le clavier en configuration majuscules, comme avec la séquence manuelle [SHIFT][LOCK].

CTLOCK

Cette fonction active le pavé numérique de la machine (même effet que la séquence manuelle [CTRL][LOCK]).

WKEY$

Cette fonction est identique à l'instruction KEY$, à ceci près qu'elle attend qu'une touche ait été pressée pour s'exécuter. Elle correspond à la séquence programmée :

```
100 A$ = KEY$ @ IF A$ = "" THEN GOTO 100
```

GETKEY (A$)

Cette fonction attend qu'une touche, ou qu'une séquence de touches, correspondant à l'un des caractères indiqués dans la chaîne A$, soit pressée. Elle fournit en retour la position du caractère à l'intérieur de ladite chaîne.

Exemple : `A = GETKEY ("ON")`

Si la touche [N] est pressée, on aura A=2.

GETMSG (M$,A$)

Cette fonction est exécutée de la même façon que GETKEY, et fournit en plus à l'affichage le message M$.

Exemple d'utilisation dans un menu :

```
100 A=GETMSG ("Imprimer? Oui/Non","NO")
110 ON A GOTO 300,200
200 <routine d'impression>
300 <poursuite du programme>
```

Deux remarques pour terminer : tout d'abord, on notera que le contrôle de cohérence de la réponse est notablement allégé (puisque la fonction n'admet que l'un des caractères contenus dans la liste de réponses possibles A$), et enfin, le cas particulier où A$=CHR$(0) : dans ce cas, l'appui sur [ATTN] stoppe le programme.

Jean Yves Hervé (450)

```
KEYLEX  213 octets

Ligne ----------------Code---------------- Chk

0001 F192C3008D4C911147A34b45594C455820206A00 2E
0002 0A00260016004A004D00620069006F0076008900 19
0003 BB004E004E004E004D004D004D00FFFF554E4C4F CD
0004 43CB53484C4F43CB43544C4F43CB4745544D53C7 E0
0005 4745544B45D9574B4559A4FFFF61179E6E21A16D E6
0006 A8B406E54CE4CE9A0C6C06E30AE59EA16092B25E 7A
0007 839EA160A801F0F6A160A802F0F02A2D640AE36C 5A
0008 E364E56CE5CE8AE3CE1F5FF002182DCE111FE598 C0
0009 CE5408F9FACE4B08F9045E93F016CEA307640AE3 05
0010 E55E93648AF4E45E896326E002C0F6F3640AE3CE C1
0011 E4FC9E002E98CE5507CE60FC9E                3D
0012 5D19
```

ASSEMBLEUR

J. Baudier	L'assembleur du HP-71 (acte VI)	*34*

BASIC

F. Duret-Lamouroux	Modélistes, à vos tours !	*39*
F. Duret-Lamouroux	Programme "PASVIS"	*40*

L'ASSEMBLEUR DU HP-71
(ACTE VI)

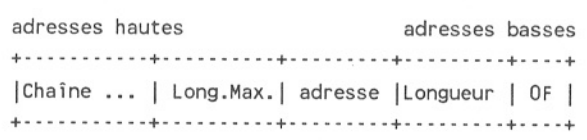
Lors des derniers actes de notre initiation à l'assembleur, nous avons appris à écrire des fonctions qui acceptent des paramètres numériques en entrée et renvoient des nombres.

Pour cette fois, je vous propose de voir comment il est possible d'écrire, dans un premier temps, des fonctions acceptant des chaînes de caractères en entrée et renvoyant une valeur numérique, puis dans un second d'autres qui renvoient de l'alphanumérique.

Dans l'acte V, nous avons vu ce qu'est la Math Stack et comment elle fonctionne en ce qui concerne les paramètres numériques. Parmi les autres objets que l'on peut trouver sur cette pile existent les chaînes de caractères. On peut, en fait, avoir quatre types d'objets sur la Math Stack. Ces types sont : les nombres réels, les nombres complexes, les chaînes de caractères et les descripteurs de tableaux. Nous avons déjà vu que les nombres réels sont mis sur la Math Stack de la façon suivante :

En ce qui concerne les trois autres types d'objets, le HP-71 place devant eux sur la Math Stack un en-tête qui sert à les identifier. Ainsi pour un nombre complexe on aura :

Les deux quartets placés sur le haut de la Math Stack (0E) que l'on appelle *signature*, indiquent que l'objet est un nombre complexe. Pour les chaînes de caractères la signature est F0. L'en-tête contient en plus de ces deux quartets, des informations parmi lesquelles on trouve la longueur de la chaîne en quartets :



On a donc ainsi, en partant des adresses hautes et en remontant vers les basses, la suite des caractères formant la chaîne, 9 quartets que l'on ignore pour le moment, la fameuse longueur de la chaîne et la signature. En ce qui concerne les descripteurs de tableaux, je vous propose de remettre leur étude à plus tard ou de consulter les *I.D.S.* car nous n'en auront pas besoin cette fois.

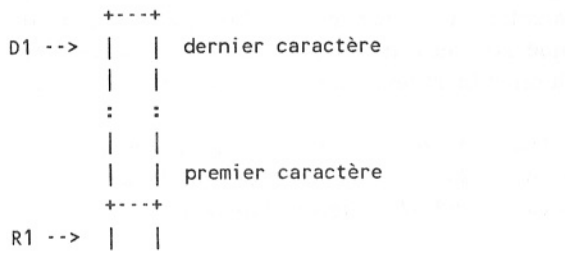
Pour éclaircir les choses, prenons simplement un exemple, avec l'appel de LEN("POPOL") qui doit renvoyer 5 normalement. Le système dépose la chaîne sur la Math Stack et y ajoute l'en-tête. Nous nous retrouvons donc avec une pile qui ressemble à peu près à ceci :

Comme pour traiter des paramètres numériques, il existe dans la Rom du HP-71 une routine (POP1S, POP one String) qui permet, non pas de charger la chaîne dans un registre (on n'aurait généralement pas assez de place), mais simplement de mettre D1 à la fin de la chaîne (X sur le schéma ci-dessus) et la longueur lue dans l'en-tête dans A[A]. On est donc prêt, après appel à POP1S, à traiter la chaîne.

Jusqu'à présent, pour sortir des fonctions et renvoyer un résultat, nous avons utilisé la routine FNRTN1 qui se charge de prendre le nombre contenu dans C et de le mettre sur la Math Stack. En effet, cette pile sert également d'intermédiaire entre les routines d'exécution des fonctions et le système.

Pour renvoyer un argument, on est donc amené à l'écrire sur la Math Stack. Lorsque l'on fait cela, il faut faire très attention à ne pas remonter au delà de AVMEMS (voir le schéma sur l'emplacement de la Math Stack en Ram dans l'acte V). Si l'on étudie par exemple la routine FNRTN1 on s'aperçoit que ceci est bien fait (cf *I.D.S.* volume III, à l'adresse #0F216).

Lorsque l'on désire écrire une fonction qui renvoie une chaîne de caractères, si aucun problème de place mémoire ne survient, on se retrouve avec le résultat final placé sur la Math Stack. On se débrouille pour avoir D1 pointant sur le haut de la chaîne (dernier caractère) et R1 sur la position mémoire suivant le premier caractère comme sur le schéma suivant :



On appelle ensuite, pour renvoyer notre résultat, la routine ADHEAD (ajout de l'en-tête) qui fabrique l'en-tête et l'ajoute sur le sommet de la pile. La longueur de la chaîne en quartets est alors la différence R1 - D1.

Supposons que nous voulions écrire une fonction qui renvoie une chaîne dans laquelle tous les caractères sont remplacés par leurs successeurs dans l'ordre ASCII. Appelons cette fonction CODE$ (l'id sera comme d'habitude #5C et le token 14) et essayons de l'écrire.

Un programme équivalent en Basic pourrait être :

```
10 SUB CODE(A$)
20   FOR I=1 TO LEN(A$)
30     A$[I,I]=CHR$(NUM(A$[I])+1)
40   NEXT I
50   DISP A$
60 END SUB
```

On voit donc que l'on a besoin de répéter une action. En entrée, on utilisera donc POP1S et en sortie ADHEAD (bien que l'on n'en aie pas besoin puisque l'on ne modifie pas l'emplacement ni la longueur de la chaîne).

Entre l'appel de ces deux routines, c'est à nous de nous débrouiller. Voici donc le Lex que je vous propose. Je vous laisse le soin de compléter le source avec les directives LEX, ENTRY, CHAR, KEY, TOKEN et autres.

Remarque : pour que ADHEAD renvoie le résultat vers le système une fois son travail terminé, il faut le lui indiquer en mettant le bit 0 du registre ST à zéro (instruction ST=0). Voir à ce sujet les *I.D.S.* II, paragraphe 14.14 ou les *I.D.S.* III à l'adresse #181B7.

```
POP1S  EQU    #0BD38   "dépile" une chaîne
ADHEAD EQU    #181B7   ajoute l'en-tête

       NIBHEX 411      le paramètre est unique et
         *             est de type alphanumérique
CODEe  GOSBVL POP1S    renvoie A[A] et D1
       CD1EX           )
       R0=C            ) on sauvegarde D1 dans R0
       CD1EX           )

* ici on a : 1. D1 ^ le dernier caractère
*           2. mode hexa
*           3. A[A] contient la longueur
*              de la chaîne en quartets
*
* on va faire maintenant :
*
* Tant que A <> 0
*   répéter | lire un caractère
*           | lui ajouter un
*           | le remettre sur la Math Stack
*           | pointer le caractère suivant
*           | décrémenter A[A] de deux
*
* Mettre D1 dans R1
*  (R1 ^ alors juste avant le début de la chaîne)
* Mettre R0 dans D1
* Mettre ST(0) à zéro (pour ADHEAD)
* Appeler ADHEAD (fin du Lex)
*

boucle ?A=0   A        A[A] est-il nul ?
       GOYES  fin      si oui : fin

       C=DAT1 B        met un caractère dans C[B]
       C=C+1  B        ajoute un
       DAT1=C B        remet le caractère
       D1=D1+ 2        ^ le caractère suivant
       A=A-1  A        )
       A=A-1  A        ) A[A] = A[A] - 2
       GOTO   boucle   on recommence ...

fin    C=R0            si on a fini :
       CD1EX           ) on repositionne
       R1=C            ) les pointeurs D1 et R0.
       ST=0   0        pour ne pas revenir ...
       GOVLNG ADHEAD   et bye bye
```

Une fois ce programme assemblé, on peut l'essayer en vérifiant que CODE$('HAL') (cf *2001, L'Odyssée de l'espace*) renvoie bien ce qu'il faut.

Les spécialistes de l'écriture de fonctions de gestion de chaînes de caractères auront sans aucun doute remarqué que l'on peut écrire une version de ce Lex sans avoir à appeler ADHEAD pour sortir. En effet, la chaîne en sortie ayant la même longueur que celle d'entrée, on n'a pas besoin de regénérer le header. On peut donc remplacer la fin du Lex par :

```
fin    C=R0
       CD1EX          D1 ^ le dernier caractère
       D1=D1- 16      D1 ^ le header

EXPR   EQU    #0F23C
       GOVLNG EXPR
```

Où EXPR est le point d'entrée de retour de fonction (du même genre que FNRTN1) avec le résultat déjà placé sur la Math Stack.

Essayons maintenant d'écrire une autre fonction qui, cette fois-ci, ne prend aucun paramètre et renvoie une chaîne de caractères. Je vous propose donc d'étudier la fonction LOCK$ qui renvoie le mot de passe du HP-71. Ce mot de passe est rangé en mémoire à partir de #2F7B2 et prend 8 octets (voir à ce sujet un court article de moi-même dans le JPC 28 d'octobre 1985 page 19).

Notre Lex devra donc lire les caractères placés à cette adresse et les mettre sur la Math Stack avant d'appeler ADHEAD. Comme je l'ai dit plus haut, lorsque l'on réalise ceci, il faut prendre garde à ne pas faire déborder la pile en remontant au dessus de AVMEMS (AVailable MEMory Start). Nous avons déjà vu que lors de l'entrée dans un Lex on a D1 ^ Math Stack. On a aussi D[A] = AVMEMS. On pourra donc comparer D1 avec D[A] pour éviter les débordements. Il existe dans la Rom du HP-71 une routine qui nous convient parfaitement : il s'agit de STKCHR qui se charge de vérifier l'évolution de la pile et d'y mettre un octet. Ses paramètres d'entrée sont :
- D1 ^ Math Stack,
- D[A] = AVMEMS et
- C[B] = octet à mettre sur la pile.

En sortie on a D1 décrémenté de deux ou l'erreur Insufficient Memory en cas de débordement de pile.

Le mot de passe, éventuellement complété par des 0, prend donc huit octets. L'algorithme utilisé par ce Lex est donc le suivant :

```
sauvegarder D0 dans R0 ;
sauver D1 dans R1 ; (R1 ^ Math Stack)
mettre l'adresse du mot de passe dans D0 ;

boucle : lire un caractère pointé par D0 ;
         s'il est nul alors aller en fin ;
         mettre le caractère sur la pile ;
                 (qui est pointée par D1)
         décrémenter D1 de deux ;
         incrémenter D0 de deux ;
         si l'on n'a pas lu 8 caractères
         alors aller en boucle ;

fin    : remettre R0 dans D0 ;
         appeler ADHEAD ;
```

Avec cet algorithme, on se rend compte que l'on est amené à faire deux tests : un pour savoir si le caractère que l'on traite est nul et l'autre pour savoir si c'est le huitième. Dans les deux cas on stoppe. On va donc avoir besoin de compter le nombre de caractères lus. Pour cela, on peut utiliser par exemple le registre P que l'on initialisera avec 8 et décrémentera à chaque lecture. Lorsque P vaudra 0 c'est que l'on aura lu 8 octets. Le listing de ce Lex dont la seule fonction a pour id 15 est donc :

```
LOCKWD EQU    #2F7B2   adresse du mot de passe
STKCHR EQU    #18504   met un octet sur la pile
ADHEAD EQU    #181B7   ajoute l'en-tête

       NIBHEX 00       pas de paramètre
LOCKe  CD0EX           )
       R0=C            ) sauvegarde de D0 dans R0

       CD1EX           )
       R1=C            )
       D1=C            ) R1 = D1 ^ Math Stack

       D0=(5) LOCKWD   D0 ^ le mot de passe
       P=     8        initialisation de P

boucle C=DAT0 B        lecture d'un caractère
       ?C=0   B        est-il nul ?
       GOYES  fin      si oui on arrête
       GOSBVL STKCHR   sinon on le met sur la pile
       D0=D0+ 2        on pointe le prochain
       P=P-1
       ?P#    0        )
       GOYES  boucle   ) si P <> 0 on recommence

fin    C=R0            )
       D0=C            ) on remet D0
       P=     0        pour ADHEAD (cf. I.D.S.)
       ST=0   0        pas de retour pour ADHEAD
       GOVLNG ADHEAD

       END
```

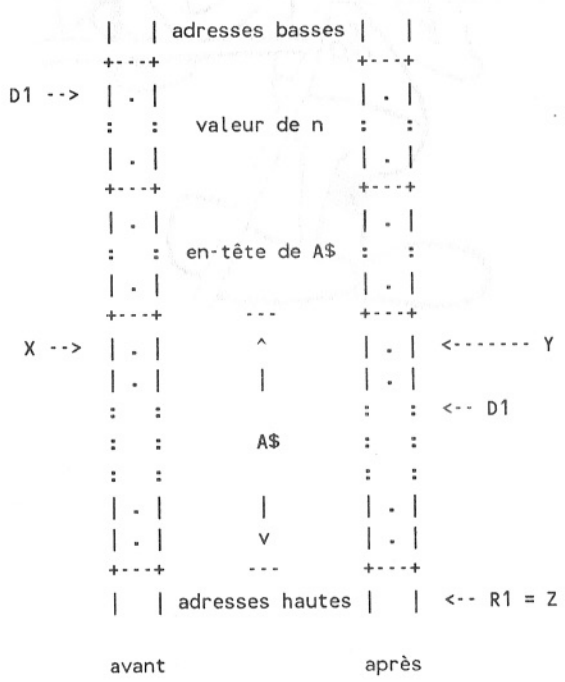
Avec le troisième Lex que je vous propose maintenant, nous allons obtenir la « fameuse » fonction `LEFT$` du Basic Microsoft qui renvoie les *n* premiers caractères de `A$` lors de l'appel de `LEFT$(A$,n)`. Il est à noter que cette fonction ne manque pas à notre Basic puisque l'on peut la remplacer avantageusement par `A$[1,n]`.

En entrée de ce Lex, nous avons sur la pile une valeur numérique et une chaîne de caractères. La première partie du code consistera donc à récupérer ces objets. On utilise POP1N puis FLTDH pour lire *n* et le convertir en hexadécimal dans A[A]. Cette routine FLTDH réalise la conversion virgule flottante vers hexadécimal, c'est à dire que A[W] est converti en A[A]. C'est en fait la fonction contraire de HDFLT vue lors de l'acte précédent avec le Lex CONTLEX.

Au passage, on vérifie que cette valeur n'est pas complexe, ni négative ou hors limite. Dans ces différent cas, il y a génération du message `Invalid Arg`.

Une fois toutes ces vérifications faites, on multiplie A par 2 (instruction `A=A+A A`) pour avoir une valeur représentant un nombre de quartets (et non d'octets) puis on range cette valeur dans R0. On peut alors appeler POP1S qui met la longueur de la chaîne dans A[A]. A partir de maintenant, il suffit de calculer les bonnes valeurs de R1 et de D1 pour obtenir le résultat souhaité.

Les deux schémas suivants montrent l'état de la Math Stack en entrée et en sortie du Lex. Juste après le POP1S, D1 se trouve en X.



Pour obtenir R1, il faut calculer D1 + LEN(A$), c'est à dire R1=D1+A[A]. On en profite également pour sauvegarder cette valeur de LEN(A$) (qui est en quartets) car on s'en servira plus tard.

Une fois R1 en place, il faut positionner D1 quelque part entre Y et Z en fonction de la valeur de *n*. En fait on doit avoir R1-D1 = *n* (en quartets). Cependant, il faut vérifier la cohérence des arguments c'est à dire si *n* est plus petit ou égal à LEN(A$) ou pas. En effet, si *n* est plus grand on renvoie A$ en entier. Par exemple `LEFT$("JPC",192)` doit renvoyer `"JPC"`. Sinon on calcule R1 et l'on s'en va par ADHEAD. Voici donc ce Lex :

```
       LEX    'LEFTLEX'
       ID     #5C
       MSG    0
       POLL   0

POP1N  EQU    #0BD1C   dépile un nombre
FLTDH  EQU    #1B223   conversion en hexa
POP1S  EQU    #0BD38   "dépile un chaîne"
ADHEAD EQU    #181B7   ajout de l'en-tête
ARGERR EQU    #0BF19   ERR:Invalid Arg

       ENTRY  LEFTe
       CHAR   #F

       KEY    'LEFT$'
       TOKEN  15

       ENDTXT

       NIBHEX 8422     Syntaxe : LEFT$(A$,n)
LEFTe  GOSBVL POP1N    A[W] = n
       GOC    argerr   si complexe : erreur
       D1=D1+  16      pointe A$
       GOSBVL FLTDH    A[W] --> A[A] en hexa
       GONC   argerr   si out of range ou NaN
       A=A+A  A        multiplication par 2
       R0=A            sauvegarde

       GOSBVL POP1S    A[A] = LEN(A$),D1 ^ chaine
* ici : R0 = n
*       D1 ^ fin de A$
*       A[A] = LEN(A$)
* on fait R1 = D1 + A[A]

       CD1EX
       B=A    A        sauvegarde de LEN(A$)
       A=A+C  A        A ^ début de A$
       R1=A            R1 aussi
       CD1EX
* ici : R0 = n
*       D1 ^ fin de A$
*       R1 ^ début de A$ (A[A] aussi)
*       B[A] = LEN(A$)
```

```
* on vérifie alors si n >= LEN(A$) et dans ce
* cas on renvoie la chaîne entière.

        C=R0            C[A] = n
        ?B<C    A       si LEN(A$) < n
        GOYES   fin     alors on renvoie tout

* sinon on fait D1 = R1 - n
* c'est à dire D1 = A[A] - C[A]

suite   A=A-C   A       ) D1 = A[A] ^ dernier
        D1=A            ) caractère à renvoyer

fin     ST=0    0       pour ADHEAD
        GOVLNG ADHEAD

argerr GOVLNG ARGERR    renvoie ERR:Invalid Arg

        END
```

Dans l'attente du prochain article, je vous propose de vous pencher sur l'écriture de deux nouvelles fonctions de la même famille que `LEFT$`, il s'agit de `RIGHT$(A$,n)`, qui renvoie les *n* derniers caractères de A$ et de `MID$(A$,p,l)`, qui renvoie au plus *l* caractères de A$ à partir du $p^{ième}$ compris. J'apporterai dans l'acte VII une possibilité de solution pour ces deux fonctions.

Je vous souhaite du succès dans l'écriture de ces deux fonctions et vous laisse consulter la documentation suivante qui vous permettra d'en savoir encore plus sur les fonctions alphanumériques.

Dans les *I.D.S.* :

Volume I : Chapitre 7 : Statement Parse, Decompile, and Execution :
- 7.4.3 pour connaître les différents objets que l'on peut trouver sur la Math Stack et savoir comment ils sont représentés.
- 7.4.4 sur POP1N.

Volume I : Chapitre 9 : Utilities :
- 9.3.1 sur les routines permettant de récupérer et de mettre des objets sur la Math Stack.

Volume I : Chapitre 3 : Memory Structure :
- 3.3.16 sur l'emplacement du mot de passe en RAM.

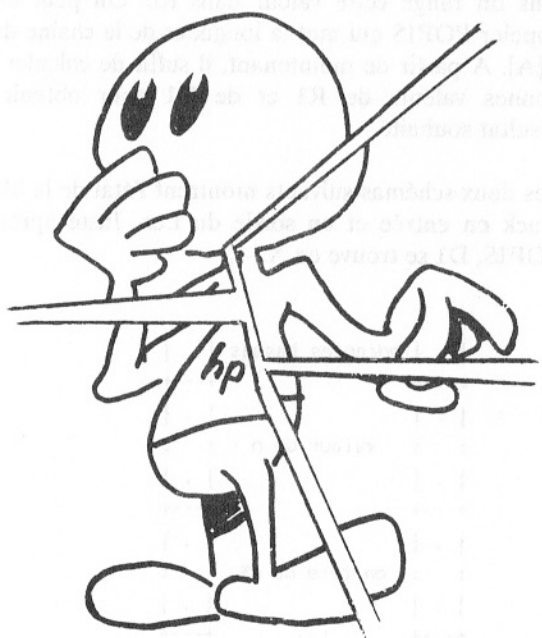
Dans JPC :

JPC 28 (octobre 85) :
- Sur le mot de passe avec les langages Basic et Forth de moi-même, page 19.

Les points d'entrée suivants dans le volume II des *I.D.S.* :

- POP1N (#0BD1C) chapitre 14.2.
- FLTDH (#1B223) chapitre 5.1.
- ARGERR (#0BF19) chapitre 24.2.
- POP1S (#0BD38) chapitre 14.4.
- ADHEAD (#181B7) chapitre 14.14.
- EXPR (#0F23C) chapitre 8.3.
- STKCHR (#18504) chapitre 15.39.

Jacques Baudier (192)



MODELISTES, A VOS TOURS !

Le programme PASVIS pourra rendre service à tous ceux d'entre vous qui utilisent un tour et confectionnent des vis ou des écrous aux normes parfois bizarres, en particulier pour le modélisme.

Ce programme est inspiré d'un programme écrit pour la HP-41 par Terry Eksted et publié par la User's Library sous le numéro 1048C.

Pour l'utilisation, il suffit de s'en remettre aux questions posées.

En ce qui concerne la valeur à donner à la première passe il semble que 0,40 à 0,60 mm au diamètre (soit une profondeur de coupe de 0,20 à 0,30 mm) convienne pour la plupart des tours d'amateurs. Au cas où la première passe devrait être sensiblement réduite et où le nombre de passes deviendrait alors supérieur à ...30, il conviendrait de modifier les lignes 170 et 290 en remplaçant 30 par un nombre suffisant, sinon la dernière passe serait démesurément forte. Les modélistes sont heureusement des gens patients !

Notes sur le fonctionnement du programme :

1 - But :

Lors de la réalisation d'un pas de vis au tour, calculer la profondeur des passes successives de façon à ce que la quantité de métal enlevée à chaque passe soit constante.

2 - Formule de base :

S=P^2*TAN(30)

où:

- S est la section du métal découpé, et
- P est la profondeur de la passe au rayon (toutefois, le programme prévoit que les profondeurs soient données au diamètre).

3 - On en déduit :

- si D1 est le diamètre nominal,
- si D3 est le diamètre après la passe de rang N, et
- si P1 est la profondeur de la première passe (au diamètre);

D3=D1-(P1*SQR(N))

4 - Par ailleurs, le diamètre D2 du noyau de la vis est donné par la relation :

D2=D1-(1,2264*Q)

où Q est le pas de la vis.

La dernière passe doit donc être telle que D3=D2.

5 - Le diamètre de perçage de l'écrou est donné par la formule :

D5=D1-(1,0825*Q)

Pour les diamètres supérieurs à 4 mm, le programme ajoute 0,1 mm à la valeur calculée.

6 - Le diamètre D4 après chaque passe est donné par :

D4=D5+(P1*SQR(N))

7 - Le diamètre intérieur à fond de filet est :

D6=D1+(0,866*Q/4)

La dernière passe doit donc être telle que D4=D6.

8 - Remarques :

Le nombre de passes est déterminé, pour un diamètre donné, par la profondeur de la première passe. Pour réduire le nombre de passes, il y a donc intérêt à choisir une valeur aussi forte que possible pour la première passe, compte-tenu de la machine, de l'outil et de la matière travaillée.

Le programme limite à 30 le nombre de passes ; au cas où ceci serait insuffisant dans le cas de très gros pas, il conviendrait de modifier le programme en conséquence.

Le nombre de passes nécessaire peut être calculé par la formule :

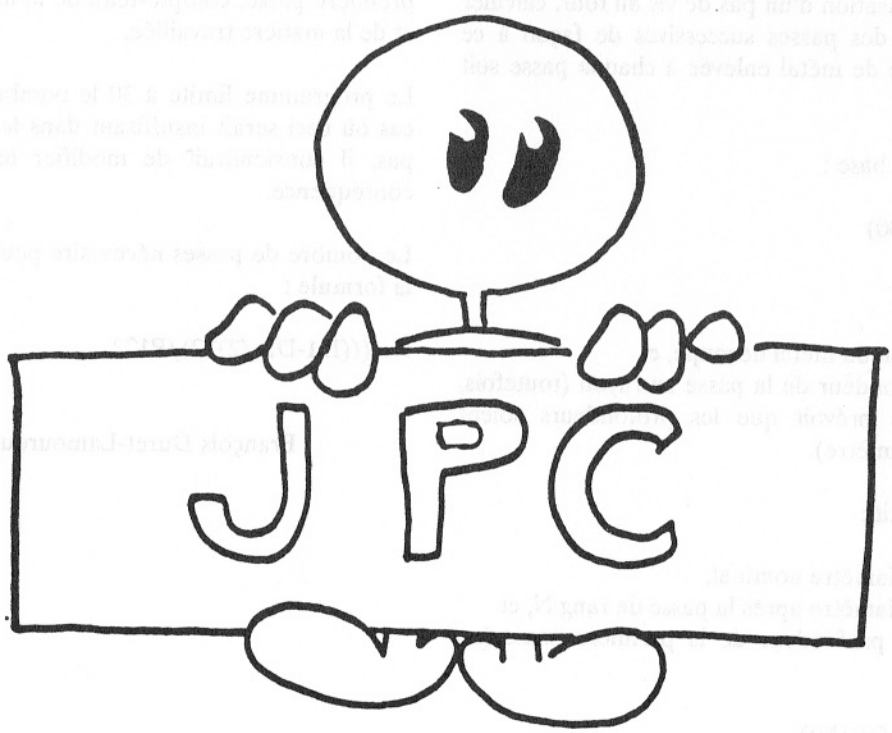
N=(((D1-D2)/2)^2)/P1^2

François Duret-Lamouroux (329)

Programme "PASVIS" (pour les amateurs de tours)

- Ce programme permet de calculer les diamètres successifs à usiner pour enlever à chaque passe une quantité constante de matière lors du filetage d'une vis ou d'un écrou. Cette quantité est fonction de la profondeur choisie pour la première passe qui dépend elle-même de la machine et de la matière usinée.
Le programme fournit en outre le diamètre de l'alésage nécessaire pour la confection de l'écrou.

```
0080 INPUT "DIAMETRE NOMINAL?:","   mm";D1$ @ D1=VAL(D1$)
0090 INPUT "PAS?:","    mm";Q$ @ Q=VAL(Q$)
0100 D2=D1-(1.2264*Q)
0110 PRINT "DIAMETRE NOMINAL=";D1;TAB(5);"PAS=";Q;TAB(5);"DIAMETRE DU NOYAU=";D2
0120 INPUT "PROFONDEUR PREMIERE PASSE(diamètre)?:";P1 @ PRINT
0130 PRINT "VIS- Diamètres à afficher:" @ PRINT
0140 FOR A=1 TO INF
0150   D3=D1-(P1*SQR(A))
0160   IF D3<=D2 OR A>30 THEN PRINT USING 180;A;D2 @ GOTO 200
0170   PRINT USING 180;A;D3
0180   IMAGE "PASSE",X,DD,":",X,DD.DD
0190 NEXT A
0200 PRINT @ PRINT "ECROU" @ PRINT
0210 D6=D1+.866*Q/4
0220 D5=D1-(1.0825*Q) @ FIX 1
0230 IF D1>=4 THEN D5=D5+.1
0240 PRINT "DIAMETRE D'ALESAGE:";D5;"mm" @ FIX 2 @ PRINT
0250 PRINT "Diamètres à afficher:"
0260 FOR A=1 TO INF
0270   D4=D5+P1*SQR(A)
0280   IF D4>=D6 OR A>30 THEN PRINT USING 180;A;D6 @ STOP
0290   PRINT USING 180;A;D4
0300 NEXT A
```



Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur du JPC, et son siège social est au 33, boulevard saint Martin, 75003 Paris.

La maquette de ce numéro a été préparée et réalisée par Pierre David, Jean-Jacques Dhénin et Janick Taillandier, grâce à un système comprenant un HP71B, deux lecteurs de disquettes HP9114A, trois HP9807A, un HP9153, deux HP9154, deux HP9134 et une imprimante LaserJet.

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

ENGLISH SUMMARY

JPC 51 - FEBRUARY 1988

We are happy to send you our Journal with its new presentation. The cover color for 1988 is brown.

Our General Meeting has elected a new board. The president is now Pierre David, the treasurer is Janick Taillandier, and the secretaries are Jean-Jacques Dhénin, Olivier Arbey, Jacques Baudier and Eric Gengoux. Pierre, who was vice-president, has been acting in fact as president for one year.

This JPC issue begins with a summary of what was told during this meeting.

This month, we have a large new products column. The infra-red module for the HP-41 allows you to use the HP-82240 printer (for the HP-18/28...). Next, a major announcement by Hewlett-Packard of four calculators is the HP-28/18 series. Since the article was written, we have received additional informations from Bill Wickes (via the Unix network) about the main differences between 28C and 28S :

- 32 Kbytes of RAM,
- directory system for USER menu,
- custom menu in which you can combine user variables and built-in commands,
- save, retrieve and combine LCD pictures/plots,
- plotting continues despite math errors/exceptions,
- `A(I)` returns the I^{th} element of an array, within an algebraic, allowing indexed variables. `A(I,J)` works too,
- programs decompile and print with automatic indentation to show control structures,
- press `[ON]` `[L]` together to dump LCD image to printer at any time, and
- singular matrix inversion is tied to the divide-by-zero exception flag.

CMT is also introducing a new HP-41 compatible computer, a new HP-71 64 Kb Ram module, and will introduce this year an 8×40 characters HP-IL LCD display.

We are very proud to present you the first article giving a very large view of the internal design of the HP-28C (version 1BB). This is a major work by Paul Courbis. He also gives two assembly language programs to help you explore the machine. We hope to read more informations about the machine soon...

Next, we have the last article of the Michael Markov's series about using HP-IL mass storage for the HP-41.

A new HP-75 Lex file appears on page 32. Jean-Yves Hervé gives us 6 programmable functions to access keyboard capabilities.

The HP-71 column is on page 34 with Jacques Baudier's sixth article about HP-71 assembly language programming. Unfortunately, this series may be disturbed, because Jacques's machine was stolen.

Until next month,

Happy Programming and JPC reading !

PPC Paris
B.P. 604
75028 Paris Cedex 01

CONFERENCE INTERNATIONALE AUX ETATS-UNIS

La firme américaine Corvallis MicroTechnology (modules Ram, Eprom, etc.) organise une conférence internationale au début du mois d'août 1988 à Corvallis, aux Etats Unis, près de l'usine Hewlett-Packard.

Si vous souhaitez y assister, contactez le Club. Nous pourrons vous faire parvenir toutes les informations (dates, prix, logement sur place) que CMT nous a envoyées.